中国人民大學
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

Transformer-VQ: Linear-Time Transformers via Vector Quantization

Lucas D. Lingle
Independent Researcher

Shen Yuan

2023-12-07

Outline

- Introduction
- What is VQ?
- Linear-time encoder(decoder) attention
- Localized positional biases
- Experiments
- Conclusion

- Introduction
- What is VQ？
- Linear-time encoder(decoder) attention
- Localized positional biases
- Experiments
- Conclusion

Efficient Transformer

高瓴人工智能学院
Gaoling School of Artificial Intelligence

Model	Attention$(\boldsymbol{V};\boldsymbol{Q},\boldsymbol{K})$	Complexity	Attention Structure
Transformer [1]	$\text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V}$	$\mathcal{O}(DN^2)$	Dense + Row-wisely normalized
SparseTrans [4]	$\text{Local2D-Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V}$	$\mathcal{O}(DN^{1.5})$	Sparse + Row-wisely normalized
Longformer [17]	$\text{Local1D-Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V}$	$\mathcal{O}(DNL)$	Sparse + Row-wisely normalized
Reformer [18]	$\text{LSH-Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V}$	$\mathcal{O}(DN\log N)$	Sparse + Row-wisely normalized
CosFormer [19]	$(\boldsymbol{Q}_{\cos}\boldsymbol{K}_{\cos}^\top+\boldsymbol{Q}_{\sin}\boldsymbol{K}_{\sin}^\top)\boldsymbol{V}$	$\mathcal{O}(\min\{DE_{QK},NE_Q\})$	Sparse
Performer [20]	$\phi_r(\boldsymbol{Q})\phi_r(\boldsymbol{K})^\top\boldsymbol{V}$	$\mathcal{O}(DNr)$	Low-rank
Linformer [21]	$\text{Softmax}\left(\frac{\boldsymbol{Q}\psi_r(\boldsymbol{K})^\top}{\sqrt{D}}\right)\psi_r(\boldsymbol{V})$	$\mathcal{O}(DNr)$	Low-rank + Row-wisely normalized
Sinkformer [22]	$\text{Sinkhorn}_K\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V}$	$\mathcal{O}(KDN^2)$	Dense + Doubly stochastic

Problems

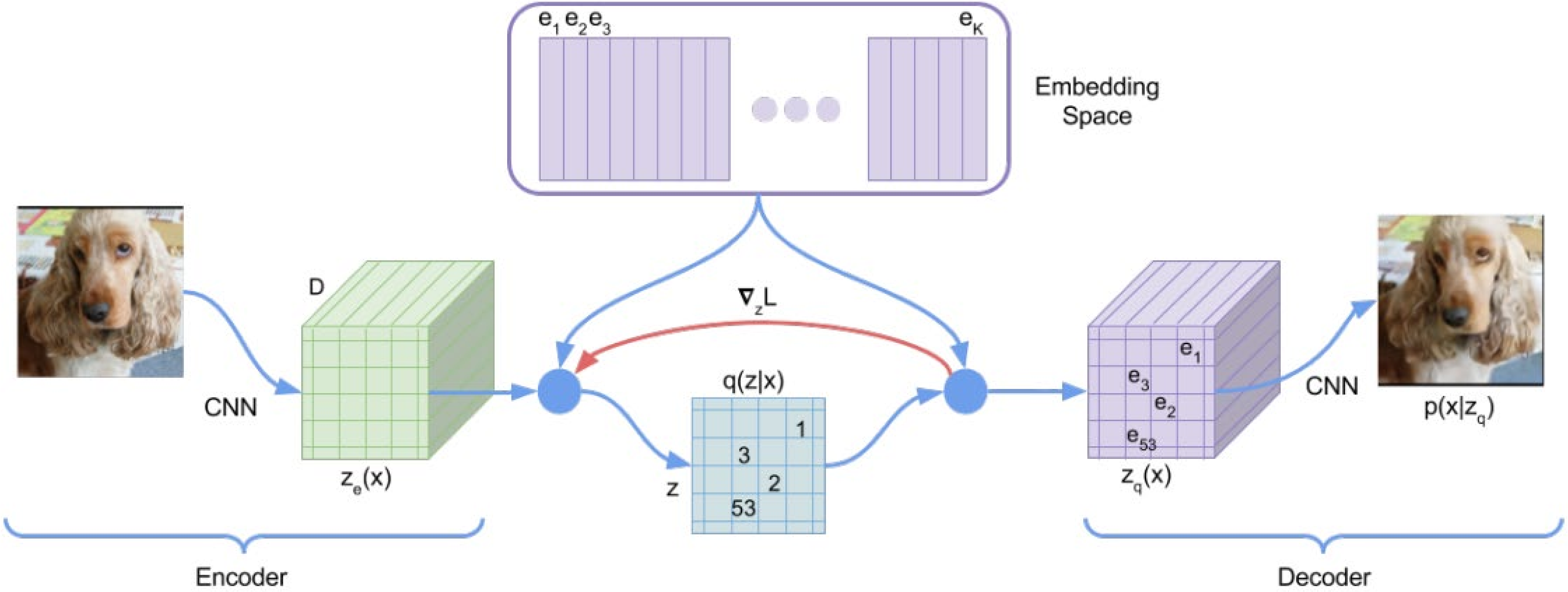
- Many of the improvements in Efficient Transformers speed **come at the expense of efficiency.**
- The complexity reduction of many Efficient transformers is only **theoretical**, which is **not obvious in practice**.
- Some Efficient Transformers is difficult to be used for training **Causal LM**, so it is no longer useful when LLM is popular.
- **Flash Attention** shows that even the standard Transformer still has a lot of room to speed up.

Contributions

- This paper presented Transformer-VQ, a transformer **decoder** with dense self-attention computable in **linear time** with respect to sequence length.
- This is made possible through a combination of **Vector Quantized(VQ)** keys, **localized positional biases**, and a truncation-free yet fixed-size **cache mechanism**.

- Introduction
- What is VQ？
- Linear-time encoder(decoder) attention
- Localized positional biases
- Experiments
- Conclusion

VQ-VAE



Van Den Oord A, Vinyals O. Neural discrete representation learning[J]. Advances in neural information processing systems, 2017, 30.

Vector Quantization(VQ)

高瓴人工智能学院
Gaoling School of Artificial Intelligence

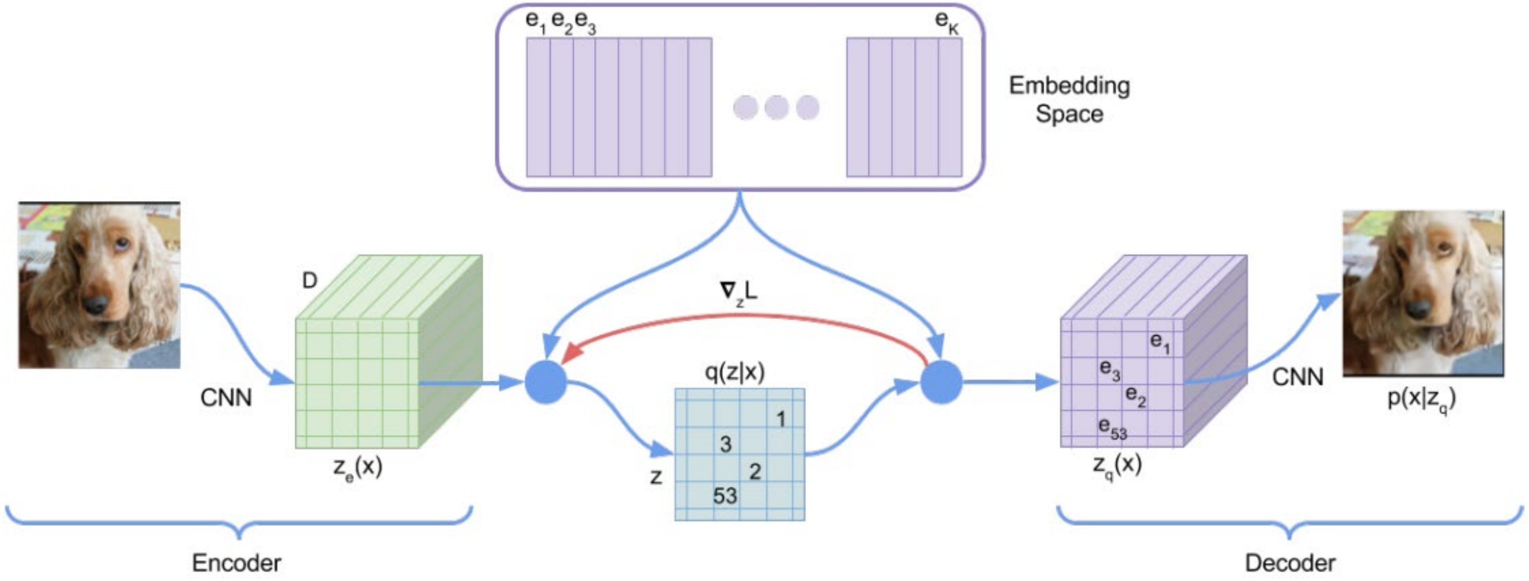
- Original autoencoder:

$$z = encoder(x)$$

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

- VQ-VAE:

$$z = encoder(x)$$



Codebook(Embedding matrix): $C = [e_1, e_2, \ldots, e_K]$

Nearest neighbor search: $k = \text{argmin}_j \|z - e_j\|_2 \quad z \rightarrow e_k$

We can write the codebook vector e_k as z_q

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

Straight-Through Estimator(STE)

高瓴人工智能学院 Gaoling School of Artificial Intelligence

- The loss of original AE: $\|x - decoder(z)\|_2^2$
- The loss of VQ-VAE:

  Maybe we should use $\|x - decoder(z_q)\|_2^2$?

- Straight-Through Estimator:

$$\|x - decoder(z + sg[z_q - z])\|_2^2 \qquad sg := \text{“stop gradient”}$$

forward propagation: $decoder(z + z_q - z) = decoder(z_q)$

back propagation: $decoder(z + sg[z_q - z]) = decoder(z)$

- Introduction
- What is VQ？
- Linear-time encoder(decoder) attention
- Localized positional biases
- Experiments
- Conclusion

Transformer-VQ

- Assume that $Q, K \in \mathbb{R}^{n\times d_k}$ $V \in \mathbb{R}^{n\times d_v}$, the standard attention is

$$softmax(QK^\top)V \qquad (1)$$

For brevity, we omit the scale factor.

- Transformer-VQ:

$$softmax\left(Q\hat{K}^\top\right)V, \quad \hat{K} = VQ(K, C) \qquad (2)$$

where $C \in \mathbb{R}^{c\times d_k}$ is the codebook.

Linear-time encoder attention

- Assume a one-hot matrix $\Delta \in \{0,1\}^{n \times c}$, such that $\hat{K} = \Delta C$

- Then we can get

$$\exp\left(Q\hat{K}^\top\right)V = \exp(QC^\top\Delta^\top)V = \exp(QC^\top)\Delta^\top V = \exp(QC^\top)(\Delta^\top V) \quad (4)$$

$$Q, K \in \mathbb{R}^{n \times d_k} \quad V \in \mathbb{R}^{n \times d_v} \quad C \in \mathbb{R}^{c \times d_k}$$

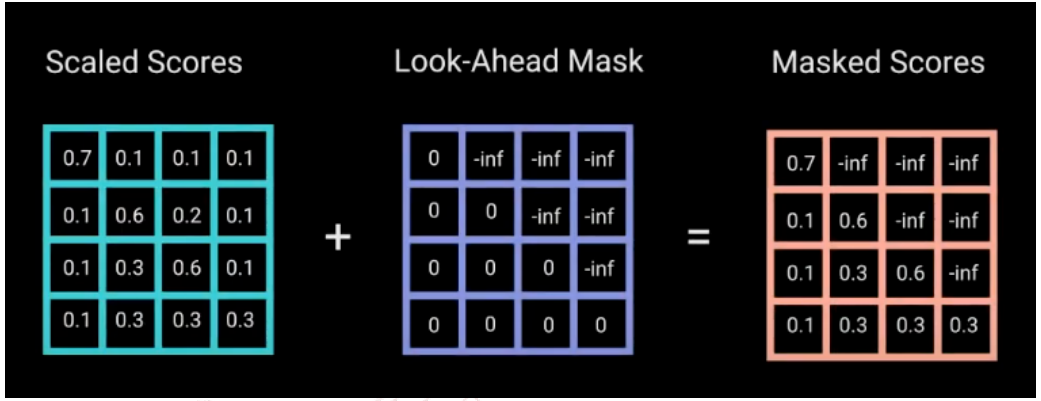
$$\mathcal{O}(n^2 d_k + n^2 d_v) = \mathcal{O}(n^2)$$

$$\mathcal{O}(ncd_k + ncd_v + ncd_v) = \mathcal{O}(n)$$

cache mechanism

Linear-time decoder attention

- To prevent the decoder from looking at future tokens, we will apply a look ahead mask like:



The output of decoder attention:

$$\begin{aligned} O_i &= \sum_{j\le i} \exp\left(Q_i \hat{K}_j^\top\right) V_j = \sum_{j\le i} \exp\left(Q_i C^\top \Delta_j^\top\right) V_j \\ &= \sum_{j\le i} \exp\left(Q_i C^\top\right) \Delta_j^\top V_j = \exp\left(Q_i C^\top\right) \sum_{j\le i} \Delta_j^\top V_j \end{aligned} \quad (5)$$

- in RNN form:

$$O_i = \exp\left(Q_i C^\top\right) U_i, \quad U_i = U_{i-1} + \Delta_j^\top V_j \quad (6)$$

where $U_i = \sum_{j\le i} \Delta_j^\top V_j \in \mathbb{R}^{c\times d_v}$

Speed up by “Block by block”

- In the inference stage, this step by step recursive calculation is acceptable.
- But in the training stage, step by step may be slower, we can change to **block by block** to speed up:
- Let $n = lm$, l denotes the block size, m denotes the block number:

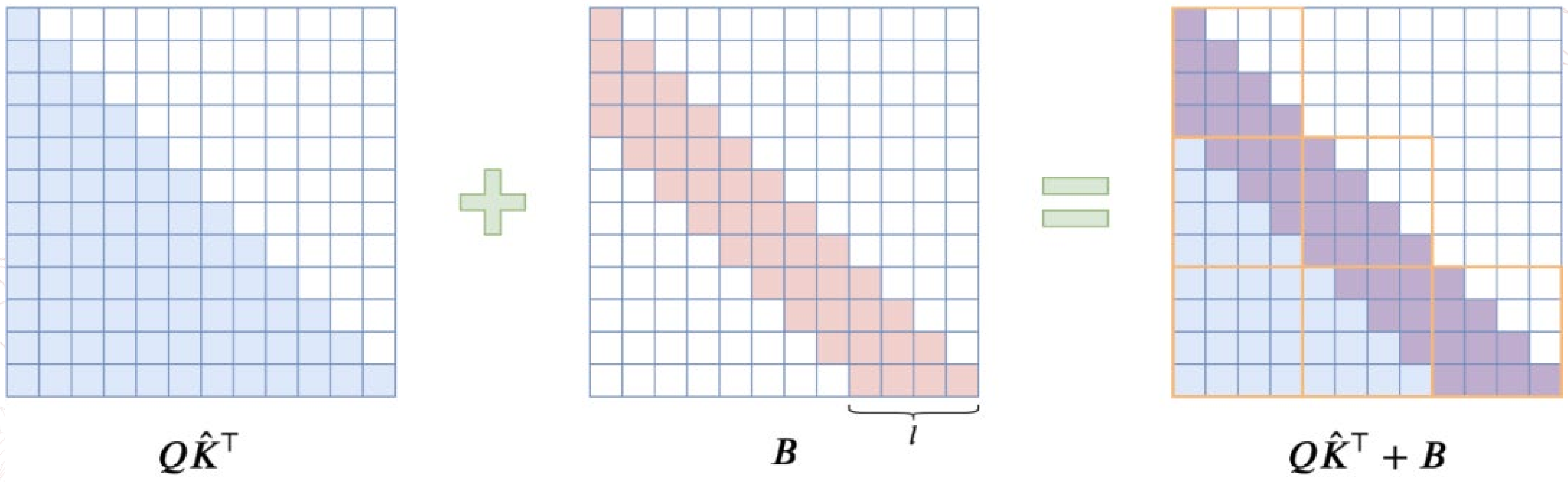
$$
\begin{aligned}
O_{[i]} &= \exp\left(Q_{[i]}\hat{K}_{[i]}^{\top} + M\right)V_{[i]} + \sum_{j<i}\exp\left(Q_{[i]}\hat{K}_{[j]}^{\top}\right)V_{[j]} \\
&= \exp\left(Q_{[i]}\hat{K}_{[i]}^{\top} + M\right)V_{[i]} + \sum_{j<i}\exp\left(Q_{[i]}C^{\top}\Delta_{[j]}^{\top}\right)V_{[j]} \qquad (7) \\
&= \exp\left(Q_{[i]}\hat{K}_{[i]}^{\top} + M\right)V_{[i]} + \exp(Q_{[i]}C^{\top})\sum_{j<i}\Delta_{[j]}^{\top}V_{[j]}
\end{aligned}
$$

where $M \in \{-\infty, 0\}^{l\times l}$ is the lower triangular matrix

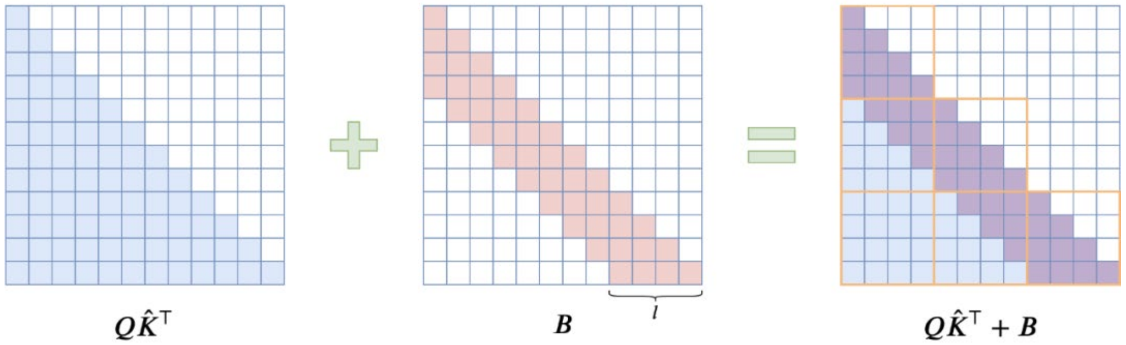
$$
O_{[i]} = \exp\left(Q_{[i]}\hat{K}_{[i]}^{\top} + M\right)V_{[i]} + \exp(Q_{[i]}C^{\top})U_{i-1}, \quad U_i = U_{i-1} + \Delta_{[i]}^{\top}V_{[i]} \quad (8)
$$

Localized positional biases

- For language models, **nearby tokens** are often more **important** than **distant tokens**.
- Thus, the author adds weights to the nearby tokens:



Localized positional biases



$$B_{i,j} = \begin{cases} 0, & i < j \text{ or } i - j \geq l \\ i - j, & \text{otherwise} \end{cases}$$

$$\begin{aligned} O_{[i]} &= \exp\left(Q_{[i]}\hat{K}_{[i]}^\top + B_{[i,i]}\right)V_{[i]} + \exp\left(Q_{[i]}\hat{K}_{[i-1]}^\top + B_{[i,i-1]}\right)V_{[i-1]} + \sum_{j<i-1} \exp\left(Q_{[i]}\hat{K}_{[j]}^\top\right)V_{[j]} \\ &= \exp\left(Q_{[i]}\hat{K}_{[i]}^\top + B_{[i,i]}\right)V_{[i]} + \exp\left(Q_{[i]}\hat{K}_{[i-1]}^\top + B_{[i,i-1]}\right)V_{[i-1]} + \sum_{j<i-1} \exp\left(Q_{[i]}C^\top\Delta_{[j]}^\top\right)V_{[j]} \qquad (9) \\ &= \exp\left(Q_{[i]}\hat{K}_{[i]}^\top + B_{[i,i]}\right)V_{[i]} + \exp\left(Q_{[i]}\hat{K}_{[i-1]}^\top + B_{[i,i-1]}\right)V_{[i-1]} + \exp(Q_{[i]}C^\top) \sum_{j<i-1} \Delta_{[j]}^\top V_{[j]} \end{aligned}$$

Localized positional biases

- Finally, the output of decoder attention:

$$
\begin{aligned}
O_{[i]} &= \exp\left(Q_{[i]}\hat{K}_{[i]}^{\top} + B_{[i,i]}\right)V_{[i]} + \exp\left(Q_{[i]}\hat{K}_{[i-1]}^{\top} + B_{[i,i-1]}\right)V_{[i-1]} + \exp\left(Q_{[i]}C^{\top}\right)U_{i-2} \\
U_i &= U_{i-1} + \Delta_{[i]}^{\top}V_{[i]}
\end{aligned} \tag{10}
$$

- Introduction
- What is VQ？
- Linear-time encoder(decoder) attention
- Localized positional biases
- Experiments
- Conclusion

Ablation study

高瓴人工智能学院
Gaoling School of Artificial Intelligence

- Larger codebook sizes may allow more flexible attention patterns and could improve the fidelity of the gradients, both of which are likely to benefit model quality at the expense of additional wall time.

- Dataset: Enwik8

Table 1: Codebook size ablations.

Setting	Val. BPB	Rel Time
$S = 256$	1.010	0.927
$S = 512$	1.005	1.0
$S = 1024$	1.000	1.109

- **BPB** and **BPC** are metrics used in compression and language modelling related to compression ratio. Bits-per-Byte measures **how many bits** a compression program needs to know **to guess the next symbol** on average. For example, if the compression program is perfect, then the next symbol is obvious to it, and it needs 0 bits, so BPB = 0.

Experiments

Table 3: Test bits-per-byte on Enwik8.

Model	**BPB**
Ma et al. (2023) - Mega	1.02
Dai et al. (2019) - XL	0.99
Child et al. (2019) - Sparse	0.99
Beltagy et al. (2020) - Longform.	0.99
Roy et al. (2021) - Routing	0.99
Sukhbaatar et al. (2019a) - Adapt. Sp.	0.98
Sukhbaatar et al. (2019b) - All-Attn.	0.98
Nawrot et al. (2021) - Hourglass	0.98
Rae et al. (2020) - Compress.	0.97
Zhu et al. (2021) - Long-Short	0.97
Fan et al. (2020b) - Feedback	0.96
Lei (2021) - SRU++	0.95
Sukhbaatar et al. (2021) - Expire Sp.	0.95
Lutati et al. (2023) - Focus Attn.	**0.94**
Transformer-VQ	0.99

Experiments

Table 4: Test word-level perplexity on PG-19.

Model	WLP
Yu et al. (2023) - MegaByte	36.4
Rae et al. (2020) - XL	36.3
Rae et al. (2020) - Compressive	33.6
Roy et al. (2021) - Routing	33.2
Hawthorne et al. (2022) - Perceiver AR	28.9
Hutchins et al. (2022) - Block-Recur.	**26.5**
Transformer-VQ	26.6

Table 5: Validation bits-per-byte on ImageNet64.

Model	BPB
Ren et al. (2021) - Combiner	3.42
Kingma et al. (2021) - VDM	3.40
Hawthorne et al. (2022) - Perceiver AR	3.40
Yu et al. (2023) - MegaByte	3.40
Grcic et al. (2021) - DenseFlow	3.35
Lipman et al. (2023) - Flow Matching	3.31
Hazami et al. (2022) - Efficient VDVAE	3.30
Transformer-VQ	**3.16**

Experiments

Figure 2: Minibatch of generated samples from our unconditional ImageNet64 model; nucleus 0.999.

- Introduction
- What is VQ？
- Linear-time encoder(decoder) attention
- Localized positional biases
- Experiments
- Conclusion

Conclusion

- VQ operation reduces the time complexity of attention computation to linear complexity.
- **Localized positional biases** is the key to differentiating Transformer-VQ from other Kernelized Attention.

中國人民大學
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

Thank You for listening!

Shen Yuan

2023-12-07