Qeios

Peer Review

Review of: "Remote Inference Over Dynamic Links via Adaptive Rate Deep Task-Oriented Vector Quantization"

Iris A.M. Huijben[1]

1. Maastricht University, Maastricht, Netherlands

The authors propose a variant of the VQ-VAE that is able to perform VQ with a time-varying bit budget and progressive decoding for low latency. Overall, these are interesting goals for real-world applications, and the ideas are intriguing. The paper is fairly well written; albeit, I will provide some minor points below to improve clarity. The methodological side of the paper seems quite solid, although some related work and comparison to SOTA methods in the experimental section are missing. My remarks and questions are outlined below:

General

1. The authors describe three types of their model: ARTOvEQ, mixed resolution ARTOvEQ, and progressive ARTOvEQ. It is not clear to me whether all three are proposed as separate models or whether the progressive ARTOvEQ is the final proposal. Does the latter include the mixed resolution aspect as well or not?
Related: In the legend of Fig. 10, I would change ARTOVeQ to variable-rate ARTOVeQ to prevent confusion. And is the progressive ARTOVeQ variant in Fig. 10 including or excluding variable-rate?

2. Some relevant related work is missing or unexplained. First, it might be worthwhile to explain the LBG algorithm in some preliminaries section, as it seems to have inspired your idea for training the progressive model. Second, I would also explain RVQ-VAE, as it seems to be the only multi-rate baseline you present. Third, a whole body of literature exists around multi-codebook quantization, among which product quantization (PQ) [1] is very popular thanks to its simplicity. Your approach, in which you split the vector into M sub-vectors, resembles PQ, but with a shared codebook per sub-space. Also, other

 doi.org/10.32388/DOZVI8

multi-codebook quantization methods, like residual quantization [2], are related to your idea of progressive coding, and recent advances [3] were shown to be able to do multi-rate quantization as well.

[1] Jégou et al., 2010. Product quantization for nearest neighbor search.

[2]. Chen et al., 2010. Approximate nearest neighbor search by residual vector quantization.

[3] Huijben et al., 2024. Residual Quantization with Implicit Neural Codebooks.

Related to the latter remark, the authors refer to their method as a single multi-resolution codebook, but given that it relates so much to product quantization, I wonder why the authors don't refer to it as multi-codebook quantization? And since multiple codebooks are already used anyway, why not train different ones for different sub-vectors then (like in PQ)?

3. I was a bit confused about some of the notations. In Figure 3, x_e is the output of the encoder, while in Figure 4, x_e is the input for the encoder. Also, in Figure 4, the notation x^(i)_e is used, while in paragraph C.1 Architecture, the authors use x^e_t.
Also, the super- and subscript for vectors x have different meanings than those in e. It might help the reader to more explicitly introduce the meaning of all sub- and superscripts.

4. How do you determine how you distribute the bits per sub-vector? At some point, you mention that you assign more bits to the first vectors and fewer to later ones. But what is the rationale behind that? Why do you expect that the first few sub-vectors contain more important information than the later ones?

5. How sensitive is the algorithm to the setting of eta_l in eq. 7? How poorly does the model perform if you set eta_l to infinity? I.e., the previously trained codewords cannot change at all. Given that eta has an l subscript in eq. 7, does it mean the variable is set differently dependent on l? And yes, how is it tuned by the authors?

Structure of the paper

1. I'm not sure if it is requested by the journal, but I found it a bit odd to read a discussion as part of the method section, before numerical experiments were presented.
2. It could help the reader if the headings of the subsections were numbered in a unique way. Now there are, for example, three headings called 2. Training, while they belong to different sections. I had to scroll back while reading to realize in which main part I was reading again. So why not call it

B.2. Training, C.2 Training, etc.? It will help the reader to more easily digest the structure of the paper.

3. The authors claim that variable-rate ARTOVeQ is competitive with single-rate VQ-VAE, but no reference to a table or figure is made in this paragraph to back up this statement; only later is figure 6 mentioned. I would first start with explaining the graphs and only then write down the conclusion.

Experimental results

1. Initialization of the codebooks is done using the LBG algorithm in the original ARTOvEQ model. This is not done in the conventional VQ-VAE. What is the effect of using this initialization, and how does it affect your comparison to VQ-VAE? If you want to showcase the effect of multi-resolution quantization in isolation, it might be good to either use the same initialization scheme for VQ-VAE or also show results without that initialization scheme.

2. Did the authors make a proper training/validation/test split of the datasets? And on which set were hyperparameters tuned? In general, information is lacking on how the datasets were handled in the experiments.

3. In the Single-rate LBG baseline, is LBG trained on the training set or directly on the test set?

4. Can the authors explain the extremely worse performance of the RVQ-VAE baseline? Are they certain it is not a bug in their implementation of RVQ-VAE or a lack of proper parameter tuning, which ARTOVeQ probably did undergo?

5. I find it apparent that the unquantized model performs the same for d=2 and d=4, and VQ-VAE performs even better for d=2, which seems odd. Related to this, the authors write: '*Some performance degradation is observed when transitioning from* $\boldsymbol{d = 2}$ *to* $\boldsymbol{d = 4}$ *as the number of bits per codeword remains constant*'

So this means that the number of code words remained the same, irrespective of the embedding dimension d. I don't understand why this should explain a decrease in performance because larger d implies that the encoding operation encompasses less compression, compared to using smaller d, while keeping the same compression rate at the quantizer stage. So the total compression is lower when using larger d, so that does not explain why results are worse for d=4.

6. In figures 10 and 11, the quantized models never reach the unquantized model performance, also not if

the number of bits keeps increasing; the performance seems to plateau. Can the authors explain this?

7. In Table 1, what is the bit rate of ARTOVeQ and Progressive ARTOVeQ? Now it's hard to see to which single-rate VQ-VAE it should be compared against.

8. In general, I think the comparison to existing multi-rate quantizers is limited; only RVQ-VAE (which is not SOTA) is compared against in figure 6. If the authors decide not to compare against newer methods, then at least it should be explained which fundamental difference exists between ARTOVeQ and such approaches that makes it impossible to compare to those.

9. Given that the authors present their method as useful for low-latency decoding, it would be of interest to compare the decoding times of the different algorithms.

Some minor typos

p.20 L3: 'is remains constant'. → 'remains constant'

p.20 L4: 'quantizaiton'. → 'quantization'

p.26, 2nd paragraph: 'datasets sare' → 'datasets are'

Subheading IV.A.: Experiential setup → Experimental

Declarations

Potential competing interests: No potential competing interests to declare.