Efficient Inference of Vision Instruction-Following Models with Elastic Cache
Supplementary Material

Zuyan Liu[1], Benlin Liu[2], Jiahui Wang[1], Yuhao Dong[1,5], Guangyi Chen[3,4], Yongming Rao[1,5], Ranjay Krishna[2,6], and Jiwen Lu[1]†

[1] Tsinghua University, [2] University of Washington, [3] Carnegie Mellon University, [4] Mohamed bin Zayed University of Artificial Intelligence, [5] Tencent, and [6] Allen Institute for AI

A Implementation Details

In this section, we will provide more details about our implementations of instantiating *Elastic Cache*.

A.1 Metrics

In the evaluation phase of our experiments, we employed two distinct metrics to gauge the performance of the accelerated inference process. Firstly, Perplexity (PPL) was utilized as a measure of predictive accuracy, quantifying the model's ability to forecast the next token in a sequence given the ground-truth texts. This metric is particularly critical in assessing the immediate, token-level precision of the model, providing insight into its understanding of the language structure on a granular level. Secondly, we incorporated the ROUGE score, as defined by Lin [1], to evaluate the overall coherence and fidelity of the generated texts in comparison with the ground-truth references. Together, these metrics furnish a robust framework for assessing the dual aspects of linguistic accuracy and contextual relevance in the generated text, thereby delivering a holistic picture of the model's performance post-acceleration.

In our study, the Perplexity (PPL) score serves as a critical metric for evaluating the language model's performance, and we adhere to the methodology delineated by Xiao et al. (2023) [2] for its computation. The perplexity is essentially a measure of how well a probability distribution or probability model predicts a sample. To calculate it, we first determine the Cross-Entropy loss for each predicted word $\hat{w}_i$ in relation to the corresponding ground-truth word w_i, given the sequence of true preceding words $w_1 w_2 \cdots w_{i-1}$. This loss quantifies the discrepancy between the predicted probability distribution and the actual distribution of the words. Once we have computed the Cross-Entropy losses for all N positions in the text, these losses are aggregated and the exponential of the average loss is taken to yield the overall PPL score for the entire text. Mathematically, the process of calculating perplexity can be encapsulated as follows:

$$\begin{aligned} \text{PPL} &= e^l \\ l &= \frac{1}{N}\sum_{i=1}^{N} \text{Cross-Entropy}(\hat{w}_i | w_1 w_2 \cdots w_{i-1}, w_i) \end{aligned} \tag{1}$$

Here, $p(w_i|w_1w_2\cdots w_{i-1})$ represents the model's predicted probability for the ground-truth word w_i, given the sequence of preceding words. The negative logarithm of this probability is the Cross-Entropy loss for each word, and the average of these losses across the text is exponentiated to compute the PPL. This metric is particularly telling as it encapsulates both the model's fluency and its ability to predict subsequent words in a sequence, with lower scores indicating a model that is better at predicting the sample.

In the manuscript, we adopt the ROUGE score as a principal evaluation metric to quantify the similarity between sentences produced by the model and the target sentences. Specifically, we utilize the ROUGE-L score, which is predicated on identifying the longest common subsequence (LCS) between the output of our model and the reference text. It is important to note that the LCS is not required to be contiguous; rather, it is a sequence that appears in both the generated text and the reference, albeit potentially with other intervening words. We compute the ROUGE-L precision, which reflects the proportion of the LCS that is present in the model-generated output. Concurrently, we assess the ROUGE-L recall, representing the proportion of the LCS found within the reference text. Subsequently, we synthesize these insights into a single metric by calculating the F1-score as follows:

$$\text{F1(ROUGE-L)} = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{(\text{precision} + \text{recall})} \tag{2}$$

A.2 Pseudo Code

In order to elucidate our proposed *Elastic Cache* mechanism, we have included a Pytorch-style pseudocode representation in the manuscript. This pseudocode, presented in Algorithm 1, serves to bridge the gap between conceptual understanding and practical implementation, providing readers with a clear, step-by-step guide to the algorithm's operational framework. It is crafted to mimic the syntactical and structural conventions familiar to users of the Pytorch library, thereby ensuring that the logic and flow of the *Elastic Cache* are both accessible and immediately applicable to practitioners in the field. More details can be found in the `code` folder. By doing so, we aim to facilitate the reproducibility of our results and enable other researchers to seamlessly integrate or build upon our caching strategy within their own computational models.

B Chatting

We showcase the capabilities of our *Elastic Cache* framework through a practical demonstration of interactive chat generation, as depicted in Figure 1. The

Algorithm 1 *Elastic Cache* PyTorch-like Style Pseudocode.

```
Define
Cache size N, Fixed position P, Ratio r, Sum of score S
Input
Past KV-Caches kv, Number of tokens n, Attention scores attn
if kv is None then
    Return None
end if
seq_len = kv[0][0].size(2)
gen_len = attn[0].size(-2)
attn = attn.mean(dim=1)
if gen_len > 1 then
    S[:, seq_len] = attn.sum(dim=-1)
end if
del_num=int(seq_len - n * (1 - r))
if del_num ≤ 0 then
    return kv
else if del_num ≥ 1 then
    kv_new = []
    fix_idx=seq_len - del_num + P
    for i, k, v in enumerate(kv) do
        keep_idx = where(argsort(S[i,:seq_len])>del_num)
        throw_idx = where(argsort(S[i,:seq_len])≤del_num)
        merge_idx = Nearest(throw_idx, keep_idx)
        k_throw=k.gather(dim=-2,index=throw_idx)
        v_throw=v.gather(dim=-2,index=throw_idx)
        k=k.scatter_reduce(dim=-2,index=merge_idx)
        v=v.scatter_reduce(dim=-2,index=merge_idx)
        k_new=k.gather(dim=-2,index=keep_idx)
        v_new=v.gather(dim=-2,index=keep_idx)
        kv_new.append([k_new,v_new])
    end for
    return kv_new
else
    kv_new = []
    for i, k, v in enumerate(kv) do
        k_new=cat([k[:fix_idx],k[fix_idx+1:]], dim=-2)
        v_new=cat([v[:fix_idx],v[fix_idx+1:]], dim=-2)
        kv_new.append([k_new,v_new])
    end for
    return kv_new
end if
```

language model, powered by the *Elastic Cache*, engages in a dialogue with users, generating responses that are not only contextually relevant but also rich in content and diversity.

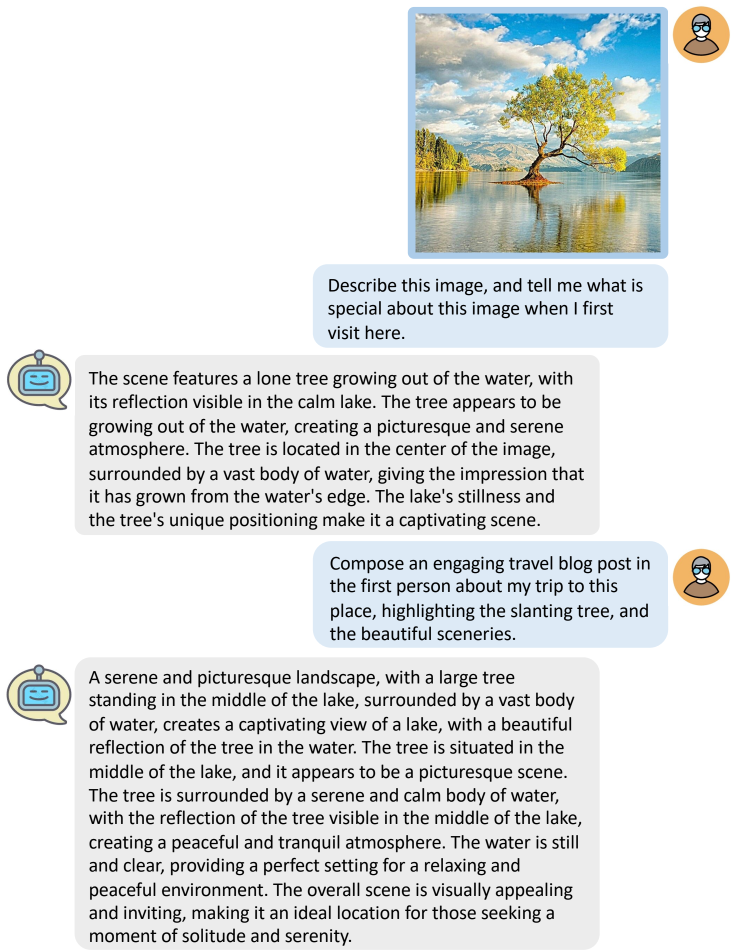

Fig. 1: Chat generation demo. We set the KV-Cache Budget as 0.5 and generate a response under the dialogue setting. The LLM generates meaningful and abundant outputs.

C More Generation Results

Our comparative analysis of cache strategies for image-text generation is meticulously documented in Tables Tab. 1, Tab. 2, and Tab. 3, where we explore the efficacy of these strategies under various Key-Value (KV) Cache budget con-

Table 1: *Elastic Cache* can correctly answer user questions and provide more detailed descriptions and explanations. As a comparison, the focus of the Local and H2O cache pruning methods is incorrect, and they may even be unable to answer questions.

User	Imagine the woman's mood in this image.
KV-Cache Budget=0.5	
Local	The woman is captured mid-air, enjoying the beach volleyball.
H2O	The woman is jumping up and down as she runs along the beach.
Elastic Cache	The woman is wearing a white tank top and shorts, and she is jumping on the beach. She is smiling and appears to be **having a great time**.
KV-Cache Budget=0.2	
Local	Thesrellsssssssssssssssst.
H2O	The woman is smiling and appears happy and enjoying herself on the beach.
Elastic Cache	The woman in the image is captured in the photo is likely to be **happy and excited**. She is smiling and **enjoying the moment**. She has a smile on her face, ... The woman's posture suggests that she is enjoying the moment and **having a good time**.

straints. The tables present a detailed examination of how each strategy performs when the available cache is limited, providing a comprehensive understanding of their capabilities and limitations.

In the scenario depicted in Tab. 1, the challenge escalates with a nuanced question, *The woman's mood*, posed to the model. Here, both the Distance and Frequency strategies buckle under the cache restrictions, unable to deliver accurate responses. Our *Elastic Cache* strategy, however, stands out by not only correctly discerning the mood of the woman but also providing a nuanced description, even within the confines of a 0.2 budget, thereby underscoring the robustness of our method in dealing with complex interpretive tasks.

In the more complex scenes of Tab. 2 and Tab. 3, where the model is tasked to *Describe this image*, the limitations of the Distance and Frequency strategies become more pronounced, with both strategies succumbing to repetitive text generation. In contrast, the *Elastic Cache* strategy maintains its composure and continues to deliver accurate depictions of the spatial relationships and interactions within the room and on the baseball field, respectively, even when operating

Table 2: *Elastic Cache* is capable of maintaining focus on key elements in the image, such as bookshelves, sofas, and chairs, even when the budget is set to 0.2, and it provides reasonable answers. In contrast, the Local and H2O methods exhibit text repetition and may even fail to answer questions when the budget is increased to 0.5.

User	Describe this image.
KV-Cache Budget=0.5	
Local	The living room is a cozy, modern and well-decorated living room with a fireplace. The living room is a modern and well-decorated living room with a fireplace. The room features a fireplace with a fireplace. ...
H2O	The room is filled with furniture, including a couch, a chair, and a coffee table. ... The room has a green and white color scheme, with a green accent wall, and a green accent wall.
Elastic Cache	The living room is filled with **furniture and decorative items**, creating a cozy and inviting atmosphere. ... There is **a large bookcase filled with numerous books**, showcasing a variety of books. The overall design of the room is modern and inviting, with **a large window** allowing natural light to fill the space.
KV-Cache Budget=0.2	
Local	Therex.
H2O	The image shows a living room with a large bookshelf, which has a large bookcase.
Elastic Cache	The image features a cozy living room with **a large bookcase filled with various books**, creating a warm and inviting atmosphere. The room is furnished with **a comfortable couch and a chair**. **A potted plant** is placed in the corner, adding a touch of greenery to the space.

on a meager 0.2 budget. This demonstrates the strategy's adaptability and its potential for handling scenarios with a higher degree of complexity.

In summary, our experimental findings across various budgets and scenarios firmly establish the *Elastic Cache* strategy as the superior choice for image-text generation in resource-limited conditions. Remarkably, it consistently yields reasonable and coherent text outputs, even at the extremely constrained budget of 0.2, thereby attesting to its high applicability and robustness for practical use. This investigation contributes novel insights and methodologies to the domain of image-text generation and offers a valuable reference for addressing challenges in resource-constrained environments.

Table 3: *Elastic Cache* is able to identify the main focal points of an image with a lower budget, providing accurate responses regarding the relationships between individuals in the picture and making reasonable inferences about the context of a competition.

User	Describe this image.
KV-Cache Budget=0.5	
Local	The image shows a baseball player in a white uniform, swinging a bat, ready to hit the ball. The catcher is wearing a white uniform, and is standing in the dirt, holding a bat, preparing to hit the ball. ... The catcher is wearing a black shirt, holding a bat, getting ready to hit the ball. ...
H2O	The batter is holding a baseball bat, ready to swing at the pitch. The catcher and umpire are behind him.
Elastic Cache	The scene takes place on a baseball field, where a baseball game is in progress. The batter is **in the middle of** the action, holding a baseball bat and preparing to swing at an incoming pitch. The catcher and umpire **are positioned behind him**, ready to **catch the ball** if the batter misses. ... The scene captures the intensity and focus of a professional baseball game.
KV-Cache Budget=0.2	
Local	The scene.
H2O	The crowd of people are watching the game.
Elastic Cache	The image shows a baseball game in progress, with a batter at home plate, holding a baseball bat, and getting ready to swing at a pitch. The scene captivating moment in the game. The batter is **in the middle of the action,** and the crowd is watching intently. The batter is ready to swing. The umpire is **standing behind the catcher**, and the catcher is prepared to **catch the ball**.The scene captures the intensity and excitement of a baseball game.

References

1. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out. pp. 74–81 (2004) 1
2. Xiao, G., Tian, Y., Chen, B., Han, S., Lewis, M.: Efficient streaming language models with attention sinks. arXiv preprint arXiv:2309.17453 (2023) 1