Authors’ Response Letters

Adaptive Compression for Online Computer Vision: an Edge Reinforcement Learning Approach

(Manuscript ID: TOMM-2020-0161)

Response to the Editor

First of all, we would like to express our sincere gratitude to the editor and the reviewers for advising minor revision and providing valuable comments. Guided by the comments, we have tried our best to address the problems, and the amendments are highlighted in blue in the revised manuscript. In what follows, we include a detailed response to the reviewer and discuss how we have addressed the comments.

We hope that the revision has satisfactorily addressed all of the concerns raised by the review team. We look forward to hearing back from you soon.

With best wishes,

Sincerely,Zhaoliang He, Hongshan Li, Zhi Wang, Shutao Xia, and Wenwu Zhu.

Response to Reviewer 1

**Comment 1** *Because the proposed algorithm aims to be used in edge-application scenario, this reviewer is highly interested in the memory space occupied by the fully loaded algorithm (e.g. model caching library and memory buffer).*

**Response:** Thanks for your useful comment. We feel sorry that we did not provide enough information about memory overhead. We add Section 4.9.1 Memory Overhead to illustrate this issue. The memory overhead is divided into two parts: the static occupied memory space and dynamic memory usage. Firstly, the static occupied memory space is the memory of the model’s size. We use Keras to construct and save MobileNetV2 and the RL agent. We check the file size of the saved model as the model’s size. A RL agent model’s size is 1 MB, and the feature extractor MobileNetV2’s size is 8.9 MB. The model caching library stores the most recent five RL agents. The overall memory of the model’s size is at most 13.9 MB. Secondly, the dynamic memory usage is memory usage during AdaCompress runs. The overall memory usage mainly includes the memory buffer of transitions and the memory usage caused by the RL agent and feature extractor. When caching transitions of 1000 images in the memory buffer, it occupies about 500 KB. We use Psutil to measure the peak memory usage during AdaCompress runs. The peak memory usage is about 367 MB. Thirdly, in real-world edge-application scenarios, common edge infrastructures have enough memory to deploy this algorithm, such as Raspberry Pi 4 Model B’s memory is 2/4/8 GB, and Huawei Atlas 500 edge station’s memory is 4/8 GB.

**Comment 2** *Statistical analyses are suggested to be amended* *especially when experimentally revealing the relationship between sceneries and compression quality.*

**Response:** Thanks for your positive remind. In Section 3.6.1 Compression Quality Level Choice Variation, the statistical analyses about compression quality level choice are not accurate enough. To better present the choice variation, we divide figure 5 (compression quality level selection for different cloud services) into three figures based on cloud services. Similarly, we divide figure 6 (compression quality level selection for different ``sceneries’’) into two figures based on scenery image inputs. Also, we compute the mean and standard deviation of chosen compression quality levels, and carefully amend the analyses. Our experiment results show that the distribution of compression quality levels is quite different for different cloud services and different ``sceneries’’. The agent can adaptively make a proper compression strategy in different complex environments, indicating our solution's generality and practicality.

**Comment 3** *The latency caused by RL agent reloading and retraining is suggested to be highlighted.*

**Response:** Thanks for your valuable suggestion. We add Section 4.9.2 Training and Re-loading Latency to illustrate the latency. The latency of re-loading a RL agent model is about 1.85s. To erase this latency, we use another process to re-load the model in our experiments. The mechanism still uses the old RL agent to choose a compression quality level before the re-loading process has finished. The re-training time cost is similar to the training time cost. The number of training images is around 900. The information about the training time cost refers to Comment 1 of Reviewer 2, please.

Response to Reviewer 2

**Comment 1** *Although the agent can be highly efficient to run on today’s edge infrastructures, the authors need to tell the time cost in the training.*

**Response:** Thanks for your constructive comment. It is a significant missing description of our manuscript. We add Section 4.9 Overall Memory and Latency Overhead to illustrate the memory and time cost. The experiment results show the number of training images is respectively 898 and 910 on ImageNet and FLIR. The time cost mainly depends on the frequency of sending images and the inference speed of the cloud service. If the end-user sends images continuously, and cloud service's inference speed is about 10 images per second, the time costs are respectively about 197.3s and 200.8s on ImageNet and FLIR. In the training phase, we need to upload reference images and compressed images. We consider the time cost of uploading reference images and obtaining the feedbacks as the benchmark latency. If we minus this benchmark latency, the real training time cost is not over 100s. More information about training details, such as the condition of convergence, refers to Section 4.9.2 Training and Re-loading Latency, please.

**Comment 2** *In Section 4.8, the authors used 1000 images of ImageNet to give a form of the latency between image uploaded and inference result feedback. It’s better to add another 1000 images of FLIR to get a more convinced result.*

**Response:** Thanks for your valuable comment. In Section 4.8 End-to-End Latency Simulation, we add 1000 images of FLIR in this experiment, and the results are similar to the experiment results on ImageNet. Compared with the benchmark solution, our solution effectively reduces the end-to-end latency on both ImageNet and FLIR dataset.

**Comment 3** *The purpose of this work is to reduce the upload traffic load of deep learning applications. However, it needs to upload reference images and compressed images together to the cloud-end in the training phase, leading some upload size overhead at the beginning.*

**Response:** Thanks for your positive comment. For training a RL agent, our solution indeed causes more upload size overhead at the beginning. Figure 8 (page 14) shows upload image size decreases as the training procedure runs. In the inference phase, our solution’s upload size overhead is only 1/2 of the benchmark’s. Section 4.9.2 Training and Re-loading Latency shows the number of training images is around 900. It is worth to spend such upload size overhead at the beginning when we need to infer thousands of images.

**Comment 4** *Some writing problems, such as: Page 3, when describing the contributions of the paper, the authors used items style of slides.*

**Response:** Thanks for your careful check. We are sorry for our carelessness. We amend the item’s style and carefully proofread the manuscript to clear out the writing problems. We also polish the manuscript, such as making Algorithm 1 (page 8) better-looking, adding hyperparameters in Table 1 (page 14).