Supplementary Material for PILC: Practical Image Lossless Compression with an End-to-end GPU Oriented Neural Framework

Ning Kang [1*] Shanzhao Qiu [2*] Shifeng Zhang [1] Zhenguo Li [1†] Shutao Xia [2†]
[1] Huawei Noah's Ark Lab [2] Tsinghua University
kang.ning2@huawei.com, qiusz20@mails.tsinghua.edu.cn

A. Coder Design

With the same notation as ours, the original rANS can be written as:

$$\text{Encoding: } S \leftarrow 2^M \times \lfloor S/P_x \rfloor + C_x + S \bmod P_x \quad (1)$$

$$\text{Decoding: } S \leftarrow P_x \times \lfloor S/2^M \rfloor + S \bmod 2^M - C_x \quad (2)$$

Where in decoding, same as the modified one, to get x, binary search is also needed to be applied on $S \bmod 2^M$.

Let n be the message length, if applying Eq. (1) and Eq. (2) directly, the time complexity is $O(n^2)$. So in real applications, S is usually truncated to a range, such as $[2^{32}, 2^{64})$, reducing the time complexity to $O(n)$ for encoding and $O(n \log X)$ for decoding, with minor reduction in compression ratio. However, it is still not efficient enough for real-time applications due to:

- P_x and C_x need to be calculated online for each x.
- One division operation and one modular operation are required for encoding.
- Binary search is needed for decoding.

In this work, S is truncated to $[2^M, 2^{M+1})$, which boosts the efficiency, but affects the compression ratio more severely, and more memory is needed. In next subsections, we prove that the BPD loss and memory are both upper-bounded by an acceptable value.

A.1. BPD loss for ANS-AI

Lemma 1. *BPD for the ANS-AI is at most* $2 - \log_2 e$ *worse than the original rANS.*

Proof. As ANS-AI and the modified rANS are numerically equivalent, it suffices to prove the effectiveness of the latter. Without loss of generality, assume quantization for pdf values does not introduce bias in density estimation, which means $p(x = i) = P_i/2^M$ for all i, otherwise both original and modified ones are affected by the same way. Also each symbol x is independent of each other. Unless specified otherwise, all log in this material means logarithm function with base 2. Then from Shannon's source coding theorem:

$$\begin{aligned} \texttt{BPD(rANS)} &\geq \frac{\sum_i P_i \times \log 2^M/P_i}{2^M} \\ &= M - \frac{\sum_i P_i \times \log P_i}{2^M} \end{aligned} \quad (3)$$

For the ANS-AI, BPD value equals to the expected number of bits to push to steam during encoding. The value S before each encoding is uniform in $[2^M, 2^{M+1})$ since P_i is proportional to $p(x = i)$. Therefore:

$$\begin{aligned} \texttt{BPD(ANS\_AI)} &= \sum_{j \in [2^M, 2^{M+1})} \frac{\sum_i P_i \times \lfloor \log(j/P_i) \rfloor / 2^M}{2^M} \\ &\leq \frac{\sum_{j \in [2^M, 2^{M+1})} \sum_i P_i \times \log(j/P_i)}{2^{2M}} \\ &= \frac{\sum_{j \in [2^M, 2^{M+1})} \log j}{2^M} - \frac{\sum_i P_i \times \log P_i}{2^M} \\ &\leq \frac{\int_{2^M}^{2^{M+1}} \log j}{2^M} - \frac{\sum_i P_i \times \log P_i}{2^M} \\ &= M + 2 - \log e - \frac{\sum_i P_i \times \log P_i}{2^M} \end{aligned} \quad (4)$$

Bring Eq. (3) and Eq. (4) together, we get

$$\texttt{BPD(ANS\_AI)} - \texttt{BPD(rANS)} \leq 2 - \log \texttt{e} \quad (5)$$

□

A.2. Memory consumption for ANS-AI

Lemma 2. *In our frame work, when* $M \leq 12$*,* δ *can be represented as an 16 bit unsigned integer.*

Proof. Let k be the integer such that $P_x \times 2^k \in [2^M, 2^{M+1})$. In our framework, it always satisfies that

*Equal contribution.
†Correspondence to: Zhenguo Li (li.zhenguo@huawei.com) and Shutao Xia (xiast@sz.tsinghua.edu.cn).

Table 1. Throughput & decomposition of time for the whole compression/ decompression process on CIFAR10 of CPU Coder and GPU Coder. Throughput is calculated with respect to the size of the original data. Time is measured as μs per image.

	CPU Coder			GPU Coder		
	Phase	Throughput (MB/s)	Time (μs)	Phase	Throughput (MB/s)	Time (μs)
Compress	RAM to GPU	9186	0.33	RAM to GPU	9246	0.33
	Model Inference	276	11.12	Model Inference	276	11.11
	GPU to RAM	1158	2.65	Coder Encode	675	4.55
	Coder Encode	873	3.52	GPU to RAM	2985	1.03
	Total	**174**	**17.62**	**Total**	**180**	**17.02**
Decompress	Coder Decode	5932	0.52	RAM to GPU	11101	0.28
	Latent to GPU	96254	0.03	Coder Decode	11091	0.28
	VQ-VAE Decode	720	4.27	VQ-VAE Decode	721	4.26
	Distribution to RAM	2426	1.27	Coder Decode	672	4.57
	Coder Decode	624	4.93	AR Decode	869	3.53
	Residual to GPU	8867	0.35	GPU to RAM	2521	1.20
	AR Decode	849	3.62	-	-	-
	GPU to RAM	2515	1.22	-	-	-
	Total	**186**	**16.21**	**Total**	**217**	**14.12**

$P_x \in [1, 2^{M-1})$, thus $k \in [2, M]$. Therefore:

$$\lfloor \frac{\delta[d,x]+S}{2^M} \rfloor = \begin{cases} k & S \in [P_x \times 2^k, 2^{M+1}) \\ k-1 & S \in [2^M, P_x \times 2^k) \end{cases} \quad (6)$$

Then it suffices to prove that when $\delta[d,x] = k \times 2^M - P_x \times 2^k$, both unsigned 16 bit constraint and Eq. (6) are satisfied.

1. $\delta[d,x] < k \times 2^M - 2^M < (M-1) \times 2^M \leq 11 \times 2^{12} < 2^{16}$, and $\delta[d,x] \geq 2 \times 2^M - P_x \times 2^k > 0$. Therefore $\delta[d,x]$ is in range of unsigned 16 bit integer.

2. If $S \in [P_x \times 2^k, 2^{M+1})$, then $\delta[d,x] + S \geq \delta[d,x] + P_x \times 2^k = k \times 2^M$, and $\delta[d,x] + S < k \times 2^M - P_x \times 2^k + 2^{M+1} \leq k \times 2^M - 2^M + 2^{M+1} = (k+1) \times 2^M$. Therefore, in this case, $\lfloor \frac{\delta[d,x]+S}{2^M} \rfloor = k$.

3. If $S \in [2^M, P_x \times 2^k)$, then $\delta[d,x] + S < \delta[d,x] + P_x \times 2^k = k \times 2^M$, and $\delta[d,x] + S >= k \times 2^M - P_x \times 2^k + 2^M > k \times 2^M - 2^{M+1} + 2^M = (k-1) \times 2^M$. Therefore, in this case, $\lfloor \frac{\delta[d,x]+S}{2^M} \rfloor = k - 1$.

□

Corollary 2.1. *In general, when $M \leq 11$, δ can be represented as a 16 bit signed integer.*

If one is using distributions different from us, then the range of P_x may changed to $[1, 2^M)$, which means the probability of one symbol may be at least 0.5, then the condition is changed to the one shown in Corollary 2.1. The proof is almost the same with Lemma 2.

	Threads	Throughput rANS (MB/s)	Throughput ANS-AI (MB/s)
Encode	1	5.1	81.7
	4	10.8	239.0
	8	15.9	433.9
	16	21.6	598.8
Decode	1	0.8	122.0
	4	2.8	467.9
	8	5.5	925.9
	16	7.4	1190.0

Table 2. Throughput comparison of rANS and ANS-AI. To generate the test data, first 8 logistic distributions are fixed. For each symbol, we first select a distribution uniformly randomly, and then sample data according to the distribution.

Corollary 2.2. *When $M \leq 12$, ANS-AI encoding requires $4 \times D \times X$ bytes of memory, and decoding requires $D \times 2^{M+2}$ bytes.*

A.3. Speed comparison of rANS and ANS-AI

We compare the speed of rANS coder and ours in the same machine as stated in experiment section of the main body. For fair comparison, both coders are implemented using C++ with OpenMP, and run on CPU. As is shown in Tab. 2, ANS-AI is 10 to 100 times faster than rANS.

Table 3. The effect of different codebook size, dimension and model capacity. Theoretical BPD and throughput on CIFAR10 are reported.

Mid Channel Dim	Blocks	Codebook Size	Codebook Dim	BPD	Throughput (MB/s)
32	**4**	**256**	**32**	**4.17**	**277**
32	4	128	32	4.19	306
32	4	256	16	4.18	293
32	4	256	64	4.18	249
32	4	512	32	4.19	232
32	4	512	64	4.19	210
32	8	256	32	4.16	177
64	4	256	32	4.14	132
64	8	256	32	4.10	74
64	16	256	32	4.06	41

Table 4. Ablation study on the receptive field of AR model and the parallel mechanism. Theoretical BPD and decompress throughput on CIFAR10 (red channel) are reported. Receptive field means the current point is predicted by how many previous points.

Receptive Field	BPD	Throughput (MB/s)
3 (with parallel)	5.7714	**382.5**
3	5.7714	48.5
4	5.7737	47.5
5	5.7708	46.0
6	5.7701	44.8
7	5.7449	44.0

B. GPU Coder vs CPU Coder

We use the same algorithm we designed to create a CPU Coder. As shown in Tab. 1, we compare the composition throughput between CPU Coder and GPU Coder. The CPU Coder is set to 16 threads. We duplicate the CIFAR10 validation set for ten times for experiment.

Compress. In terms of compression time, the throughput of CPU and GPU coders is comparable because both require two RAM-GPU transfers. We notice that GPU to RAM time is different between the two Coder. Because the residual data, latent data, and distribution parameters must all be transferred from GPU to RAM for CPU Coder, but only compressed data must be transferred from GPU to RAM for GPU Coder, which takes less time.

Decompress. The CPU Coder requires two additional RAM-GPU transfers during decompression because the Coder decoding is done on the CPU. The CPU Coder decodes latent vector indexes before sending them to the GPU. Then we need to transmit the distribution parameters to RAM after VQ-VAE decodes them. The residual is then decoded by the Coder and passed back to the GPU. The Three-Way Auto-regressive recovers original data from residual data. The GPU Coder, on the other hand, only requires two RAM-GPU transfers, and all Coder decoding is done on the GPU, which saves a lot of time.

Table 5. Different settings of how to predict residual distribution. *i.e.*, different inputs of VQ-VAE. We use original image, reconstruction image, residual, and original image concatenated with residual to predict the residual.

Given	BPD
Original Image	**4.17**
Reconstruction Image	4.23
Residual	4.19
Origanl Image concat Residual	4.17

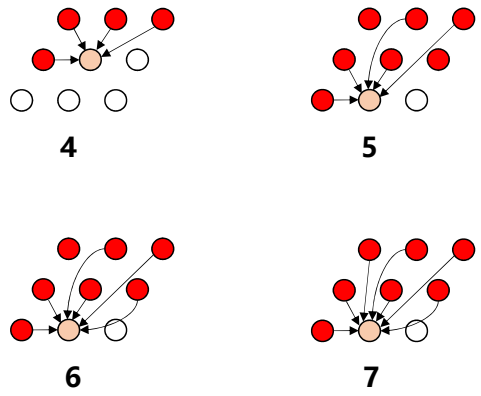

Figure 1. The receptive field rules of the auto-regressive model. Receptive field from 4 to 7.

C. Model architecture

We do further experiments to see how different codebook sizes and dimensions affect the results. A large codebook size or codebook dimension favors the BPD but reduces performance, as seen in Tab. 3. A tiny codebook size or codebook dimension does indeed produce substantially faster throughput, but it would contains relatively little information when applying our model to high resolution images. As a result, on codebook size/dimension, there is a trade-off between BPD and throughput. To balance the trade-off, we set the codebook size to 256 and the codebook dimension to 32 in our framework.

We investigate the impact of various model capacities. We modify the number of residual block and the mid channel dimension to change the model capacity. As shown in Tab. 3, the larger model achieves lower BPD, but the throughput of the model drops even faster, which deviates from the 'practical' aim. As a result, we adjust our model to 4 residual blocks and 32 mid channel dimensions.

D. Receptive field rules of AR model

Figure 1 demonstrates the receptive field rules of the auto-regressive model on a single channel which we use in the ablation study. As the figure suggests, these receptive field settings cannot be parallelized using what we implement in Three-Way Auto-regressive.

E. Parallel vs No Parallel

We add the experiment result of Three-Way Auto-regressive without parallel. As shown in Tab. 4, Three-Way Auto-regressive using our designed parallel mechanism achieves the fastest throughput with competitive compression ratio.

F. Why predict residual given original image?

We investigate the different input of VQ-VAE model. We use the original image, reconstruction image, residual, and original image concatenated with residual to predict the residual. As illustrated in Tab. 5, directly using residual to predict residual is not the best solution. As the original image has a lot of spatial information, we choose to model the residual given original image.