L3: Accelerator-Friendly Lossless Image Format for High-Resolution, High-Throughput DNN Training

Jonghyun Bae, Woohyeon Baek, Tae Jun Ham, and Jae W. Lee

Seoul National University, Seoul, Korea
{jonghbae, baneling100}@snu.ac.kr, ham.taejun@gmail.com, jaewlee@snu.ac.kr

A Appendix

Table 1. Comparison of L3 to PNG and JPEG

	Algorithm		Lossless?	GPU-support?
	Filter	Compression		
PNG	Five types (None, Sub, Up, Avg, Paeth)	Deflate	◯	×
JPEG	DCT, Quantization	Run-length + Huffman coding	×	△
L3	Custom Paeth	Base-delta coding	◯	◯

A.1 Comparison of L3 design to PNG and JPEG design

Table 1 summarizes the design of PNG, JPEG, and L3 based on their filtering and compression schemes. PNG represents the most popular *lossless* image format, which is the *least* GPU-friendly. The PNG encoder selects the best of the five data filters for a scanline (None, Sub, Up, Average, Paeth) and applies it to both the current and the next scanlines. The compression stage employs the Deflate algorithm, which is inherently sequential, hence making it very challenging to execute efficiently on data-parallel accelerators like GPU.

JPEG provides high decoding throughput but with *loss* of information. The JPEG encoder filters the pixels of the same color to the top-left corner in a macroblock by discrete cosine transformation (DCT), followed by data quantization. This method drops color areas that are not sensitive to human eyes, which causes information loss. Both run-length coding and Huffman coding are utilized to compress pixels. While some of those operations (e.g., DCT, quantization) can be offloaded to GPU, others still need to run on the CPU or a custom accelerator which is not widely available on commodity GPUs.

Table 2. DALI data loader configurations

Function	Argument	Value
`Pipeline`	`num_threads`	12
	`exec_pipelined`	True
	`prefetch_queue_depth`	2
	`exec_async`	True
`fn.readers.file`	`prefetch_queue_depth`	2
	`random_shuffle`	True
`fn.decoders.image`	`device_memory_padding`	256MB each
	`host_memory_padding`	
	`device_memory_padding_jpeg2k`	
	`host_memory_padding_jpeg2k`	
	`preallocate_height_hint`	Same as the resolution of dataset
	`preallocate_width_hint`	

Fig. 1. Lossless image data preparation throughput using prior works on GPU-implemented lossless data compression

L3 provides high throughput and high metric performance (with no loss of information) on commodity data-parallel accelerators, thus delivering high end-to-end throughput close to the ideal even if there is no format-specific custom hardware. This work will also raise the bar on hardware seekers for lossless image formats by serving as a solid baseline to prevent them from seeking custom hardware support prematurely.

A.2 Detailed Experimental Setting

We implement a data loader using the NVIDIA DALI library, and all image formats, including L3, are loaded and decoded through the DALI data loader. Table 2 summarizes the configuration of the DALI data loader.

A.3 Throughput Comparison with Other Lossless Formats for GPU

Figure 1 reports the data preparation throughput of prior lossless data compression methods covered in the related work section. In fact, we have evaluated all of the following open-source GPU implementations: (1) CULZSS [2], (2) CUDPP [3], and (3) CUHD [4]. However, we eventually reported the results

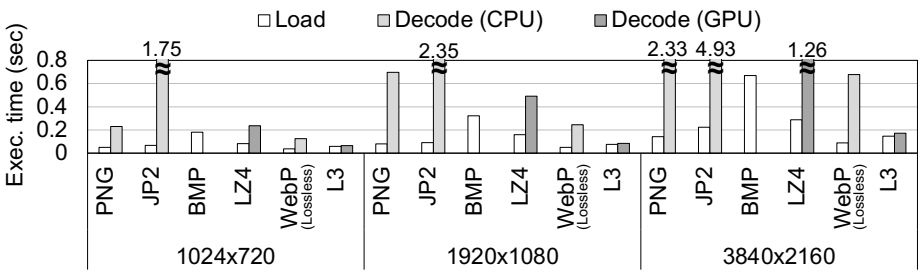

Fig. 2. `Load` and `Decode` execution time breakdown

Table 3. Detailed descriptions about models, datasets, and mini-batch sizes for FP32 on V100, and mixed precision on A100

Dataset (Resolution) # of train/val/test	Model (Backbone)	Batch size	
		FP32 on V100	Mixed on A100
Cityscapes (1920×1080) 2975/500/1525	DDRNet23-slim	16	64
	DeepLabv3+ (MobileNetv2)	20	96
	MaskFormer (ResNet50)	16	64
	PointRend (SemanticFPN)	12	64
KITTI (1024×720) 3519/3462/500	EgoNet	12	48
	PointPillars	16	80
	YOLOv5	64	256

from LZ4 on GPU only (nvcomp [1] implementation from NVIDIA) as they all perform worse than LZ4.

A.4 Separating Load/Decode Performance

Figure 2 shows an execution time breakdown of data preparation for three image resolutions. Note that the `Load` and `Decode` are fully pipelined. Thus, the longer of the two determines the data preparation throughput. Both PNG and L3 have comparable `Load` time due to similar compression ratios, whereas the decoding time is significantly lower in L3.

A.5 Performance Comparison on Low-End GPUs

L3 also achieves performance improvement on other low-end GPUs. The TOPS requirement of L3 for data preparation is relatively small compared to the compute capabilities of mainstream GPUs. For example, L3 requires up to 1.15 TOPS for FHD datasets, which is less than 10% of the peak TOPS of NVIDIA Turing-based RTX 2080 Ti (13.4 TOPS) and Pascal-based GTX Titan Xp (12.1 TOPS).

Table 3 summarizes the model, dataset, and batch size used for training on NVIDIA V100 GPU. Figure 3(a) compares the data preparation throughput

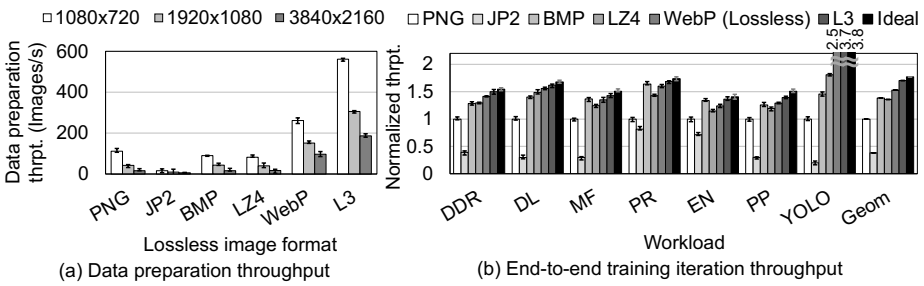

Fig. 3. Throughput Comparison with lossless-encoded datasets on NVIDIA V100 (a) Data preparation (`Load+Decode`) throughput (b) Normalized end-to-end training iteration throughput

(`Load+Decode`) of L3 and other lossless decoding methods with various image resolutions. For experiments, we use a `p3.2xlarge` AWS EC2 instance with NVIDIA V100 GPU with 16GB HBM and Intel Xeon E5-2686 v4 CPU with 8 cores. We utilize the Cityscapes dataset as input. L3 shows throughput gains by 5.0×, 7.3×, and 11.7× on HD, FHD, and UHD-resolution images, respectively, compared to PNG. L3 achieves higher data preparation throughput than all the other formats, regardless of image resolution or system.

Figure 3(b) presents the normalized training throughput (iterations/sec) of different lossless image formats on various object detection and semantic segmentation models. The training throughput is normalized to the ideal case where the system has zero overhead for data preparation (`Load+Decode`). Overall, L3 achieves the best end-to-end training throughput by hitting an average of 1.70× speedup over PNG.

A.6 Mixed-Precision DNN Training Throughput

Mixed-precision training is becoming popular to improve the training throughput at a negligible loss of test set accuracy compared to full-precision floating-point training. To support this, NVIDIA GPUs have specialized computation units called *Tensor Cores* for half- and mixed-precision operations. In this setting, the DNN training pipeline spends even less time on `Compute` to become more bottlenecked by the data preparation time. Table 3 summarizes the model, dataset, and batch size used for mixed-precision training on NVIDIA A100 GPU.

Throughput Comparison with Lossless Decoders. Figure 4(a) shows the normalized mixed-precision training throughput of the seven models on the lossless image datasets. As stated above, the efficiency of data preparation in half-precision training has a greater impact on the end-to-end training throughput than in full-precision training. As a result, DNN training with the other lossless formats yields a much lower throughput than the ideal case (i.e., with no data preparation time). L3 achieves a 2.26× geomean throughput, which is significantly higher than the other lossless image formats because L3 significantly reduces the decoding time with its GPU-accelerated decoding.

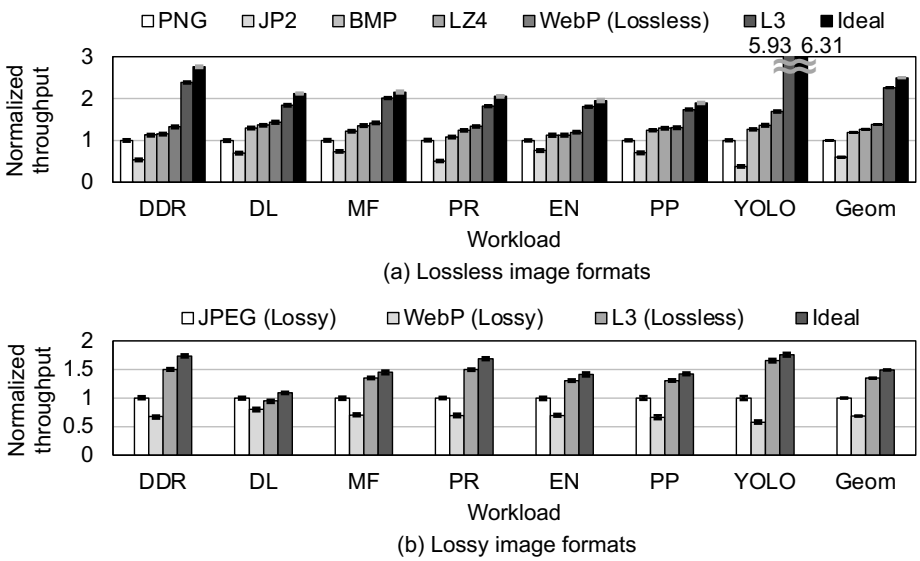

Fig. 4. Normalized mixed-precision training throughput. (a) Lossless-encoded datasets (b) Lossy-encoded datasets

Throughput Comparison with Lossy Decoders. Figure 4(b) compares the mixed-precision DNN training throughput with the WebP (Lossy), JPEG (Lossy), and the L3 (Lossless)-encoded dataset. We use the same hyperparameters and quality factors of each lossy image format of the main paper except the batch size. The figure shows that the training throughput of L3 is higher than that of both JPEG and WebP (Lossy) by a factor of 1.34× and 1.97×, respectively.

A.7 Performance Improvement by Algorithm and Hardware Accelerator

Figure 5 shows the data preparation throughput of the PNG- and L3-encoded datasets on CPU with image-level parallelism (i.e., parallelized across the batch dimension). The figure reports the average data preparation throughput with 2,048 FHD resolution images from Cityscapes dataset on an AWS EC2 instance of Intel Xeon Platinum 8275CL CPU with 96 cores. L3 (CPU) processes the L3 decoder on the CPU without optimization, and L3 (AVX) operates the entire custom Paeth and the add operation of base-delta decoding with advanced vector extensions (AVX) SIMD instructions for x86 ISA. The performance improvement by the lightweight algorithm is represented by the performance difference between PNG, L3 (CPU), and L3 (AVX). Furthermore, the throughput improvement by whether the hardware accelerator is used or not is shown by the comparison on L3 (AVX) and L3 (GPU).

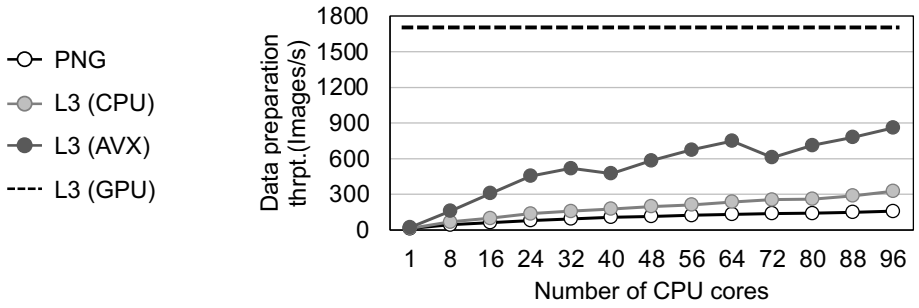

Fig. 5. Data preparation throughput of PNG- and L3-encoded datasets on various number of CPU cores

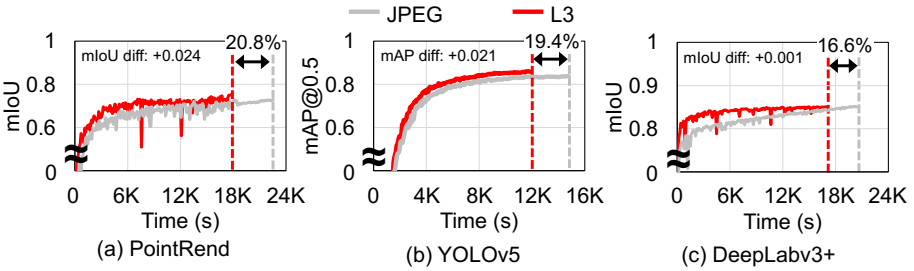

Fig. 6. Training time and accuracy of L3- and JPEG-encoded datasets. The red-dotted and gray-dotted vertical lines present the time of training termination of the L3 and JPEG dataset. (a) PointRend (b) YOLOv5 (c) DeepLabv3+

Overall, both PNG, L3 (CPU), and L3 (AVX) show a scalable throughput improvement, L3 (CPU) and L3 (AVX) show performance improvement by a factor of 2.1×, and 5.4× on 96 cores compared to PNG. L3 (GPU) enables the GPU to process multiple patches in parallel to exploit the massive parallelism of GPU, thereby achieving throughput improvement by 5.2× and 2.0× than L3 (CPU) and L3 (AVX), respectively.

A.8 Accuracy Convergence on Lossless/Lossy Image Format

Figure 6 shows the overall model accuracy over time when training with L3- and JPEG-encoded datasets. The representative models are trained until the loss of the validation set does not decrease for 10 consecutive epochs. The red-dotted and gray-dotted vertical lines mark when the training was finished for L3 and JPEG, respectively.

Overall, the training with lossless L3 results in superior accuracy and shorter training time than training with lossy JPEG. While training with other lossless formats (e.g., PNG, WebP) also results in the exact same accuracy and the number of iterations, their per-iteration throughput is much lower than that of L3. Thus, their end-to-end training time is much longer than JPEG. In contrast,

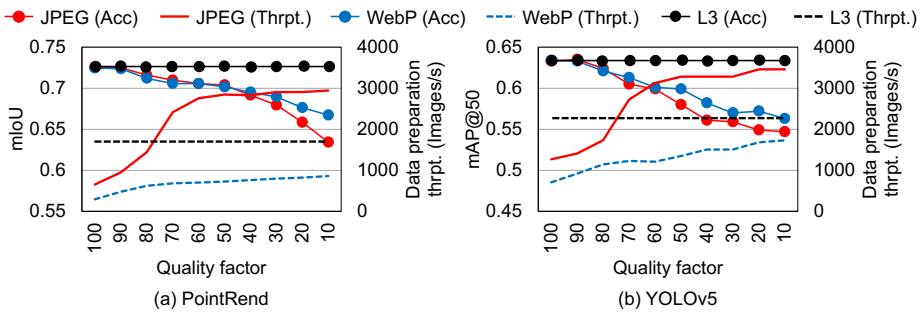

Fig. 7. Test-set accuracy and data preparation throughput with JPEG (Lossy), WebP (Lossy), and L3 (Lossless)-encoded datasets on various quality factor (a) PointRend (b) YOLOv5

Table 4. Data compression ratio of the origin and custom Paeth filter (The compressed size / the decompressed size). Lower is better.

	Cityscapes	KITTI	Random	Black
Raw (MB)	5.93	2.24	5.93	5.93
Origin	0.42×	0.63×	1.02×	0.12×
Custom	0.44×	0.64×	1.02×	0.12×

L3 demonstrates an advantage over lossy JPEG in terms of both accuracy and overall training time.

A.9 Accuracy on Various Q-factors of Lossy Image Format

A lower quality factor affects the test-set accuracy. Figure 7 shows the model accuracy of PointRend and YOLOv5 trained with JPEG (Lossy), WebP (Lossy), and L3 (Lossless)-encoded datasets with varying quality factors. The accuracy is evaluated with the test-set. The results show that a high quality factor must be used to achieve a comparable accuracy of the lossless image format, but this causes a significant reduction of data preparation throughput.

A.10 Impact of Dependency Patterns in the Paeth Filter

Table 4 shows that the change in the dependency pattern between the original and custom Paeth filter yields an insignificant difference (2% at the worst) in compression ratio. In summary, our custom Paeth unlocks ample data parallelism on GPU without degrading the compression ratio.

A.11 Encoding Performance

Since DNN training features a write-once-read-many (WORM) access pattern, dataset encoding is just a one-time cost. Thus, optimizing encoder performance

was not a priority in this work. However, because of the lightweight algorithm, the L3 encoder still achieves higher encoding throughput than PNG by 9.3%, 12.2%, and 14.1% for HD, FHD, and UHD resolution, respectively.

References

1. NVIDIA: nvcomp: A library for fast lossless compression/decompression on the GPU. https://github.com/NVIDIA/nvcomp (2021)
2. Ozsoy, A., Swany, M.: CULZSS: LZSS lossless data compression on CUDA. In: Proceedings of the 2011 IEEE International Conference on Cluster Computing. pp. 403–411 (September 2011)
3. Patel, R.A., Zhang, Y., Mak, J., Davidson, A., Owens, J.D.: Parallel lossless data compression on the GPU. In: Proceedings of the 2012 Innovative Parallel Computing. pp. 1–9 (May 2012)
4. Weißenberger, A., Schmidt, B.: Massively parallel huffman decoding on GPUs. In: Proceedings of the 47th International Conference on Parallel Processing. Association for Computing Machinery (August 2018)