VALIDATED EFFICIENT IMAGE COMPRESSION FOR QUANTITATIVE AND AI APPLICATIONS

Bruno Sanguinetti, Christoph Clausen, Michael Desert, and Evgeniya Balyshева
Dotphoton AG, Zeughausgasse 17, 6300 Zug, Switzerland

ABSTRACT

Artificial intelligence will allow vast quantities of earth observation data to be analysed [MA18]. Image compression can both provide a performance boost and make systems more economical. However, most high-performance compression methods have been tuned to the human eye, rather than developed for machine vision, generating artefacts that although invisible limit the performance of advanced algorithms. In this paper, we discuss requirements for compression tuned for machine vision, demonstrate an implementation achieving a compression ratio in the range 5:1–10:1 at a rate > 200 MB/s/core in software and 400 MB/s on a VHDL FPGA simulation having a 5k-LUT footprint. We also show that adding a machine-learning component to our compressor increases the compression ratio by 10% and allows for easy portability of an otherwise complex algorithm on enterogenous architectures: Intel CPU, Intel VPU (Movidius Miryad X), and Nvidia and Intel GPUs. The compression has fixed metrological quality (1.2 dB SNR loss) and does not produce bias or systematic errors. We validate compression both through acquiring images of a laboratory-based test target, as well as on drone images emulating the ground sampling distance and point-spread function of high-resolution satellites.

Key words: compression; machine learning; AI; earth observation; ESA.

1. INTRODUCTION

Recent developments in processing technology, such as GPUs and special-purpose tensor processors, together with theoretical advances, allow for complex machine vision tasks (segmentation[HK20], classification[VCM20], localization, detection, ...), as well as image processing (de-noising[TFZ’’], upsampling, spectral analysis, sharpening, ...) to be performed efficiently and with relatively little programming effort, using a mature ecosystem of machine learning libraries and techniques. The often parallel nature of this type of processing allows it to easily scale from low power “edge” devices (for example the Intel Movidius Myriad X[Oh17]) all the way to supercomputers[ Wan20]. Machine learning can be seen as programming methodology where part of the solution is defined as the algorithm (e.g. the neural network structure) and another part of the solution is defined by the training data. The quality and quantity of this training data, as well as appropriate selection and pre-processing is of great importance in realizing reliable machine learning. The training data must span the entire space of possible inputs, so that each potentially different type of pre-processing (e.g. compression parameters) should be represented in the training dataset, increasing the volume of required training data. To mitigate this requirement, it is preferable to preserve raw image data and to pre-process it as and when needed for Machine Learning training or inference. This gives the opportunity to normalize data arising from different systems (e.g. past, present and future) and pre-process it through the same algorithm before using it. This is of particular importance as different pre-processing techniques are optimized for different applications, i.e. if images are to be viewed by the human eye, or analysed to extract global image properties or properties of individual objects within the image. For example, the human eye is sensitive to local changes of the image only, whereas neural networks may correlate very distant pixels. An illustration of the envisaged storage and processing structure is shown in Figure 1.

It would be often important to preserve raw data, however, this is often not possible due to its requirements in terms of storage and bandwidth, a problem exacerbated at the large number of images used in AI applications.

Figures 2 and 3 illustrate the importance of high data quality when applying AI algorithms. Figure 2 is from the company gamma.earth who provides generative up-sampling algorithms for earth observation applications (drone and satellite). The first row of images (a,b,c) was compressed with a visually lossless algorithm, whereas the second row (x,y,z) was not compressed. In the unprocessed images (a, x), the difference between the compressed one (a) and un compressed (x) indeed invisible to the human eye. However, as soon as a 4x upsampling algorithm is applied (b, y), prominent artefacts appear in the image generated from visually lossless compressed data (b), whereas the upsampling works well on the image generated from uncompressed data (y).

Figure 3, illustrates the excellent de-noising performance
that may be achieved when training AI algorithms with calibrated raw data. The figure shows three views of details of a microscopy image. The noisy image is the original, the Ground Truth image is obtained by taking an average of 1000 frames, and has a signal-to-noise ratio 22 dB higher than the noisy image. The “AI” image is processed through our AI denoiser, and has an SNR (w.r.t. ground truth) 9 dB higher than the original image. When training the same network on uncalibrated, visually lossless compressed data, we were unable to obtain an SNR increase with respect to the noisy image.

2. REQUIREMENTS

To satisfy the needs in terms of data quality and quantity for AI processing, we assume that the requirements are:

1. Handle large volumes of data
   (a) High compression ratio. Achieve the intrinsic information content of the image. We measured this to typically be in the range 1bpp–3bpp. So that a 2bpp target on a 16bpp sensor would result in a target 8:1 compression ratio.
   (b) High speed. At 2bpp, a 1Gbps link requires the compressor to handle 500Mpix/s (potentially using multiple cores).
   (c) Low per-core resource and power usage in FPGA, to scale to high throughput sensors.

2. Maintain metrological quality
   (a) No artefacts, visible or invisible

<table>
<thead>
<tr>
<th>Lossy</th>
<th>Visually lossless</th>
<th>Limited-error</th>
<th>Metrological</th>
<th>Lossless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss invisible</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bounded error</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>No artefacts / bias</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>in-distinguishable from raw</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit-accurate</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1. Table showing, from left to right, compression methods of increasing image quality, from lossy compression, targeted at large-scale image distribution for consumer products, to lossless compression which is bit-accurate.

(b) No bias,
(c) Works with raw data

3. Interoperability

(a) Can be transcoded across different lossless file formats in a bit-perfect manner (e.g. tiff, hdf5)
(b) No adverse interaction with lossy file formats (e.g. DCT-based, wavelet-based)
(c) Can easily be integrated in existing systems and benefit from existing codec components (e.g. entropy coders), as well as transparently integrate in existing data pipelines.

Table 1 describes the properties of different types of image compression, from lossy to lossless. In this work, we present a compression method that maintains the metrological properties of the image and guarantees that no bias or artefacts are introduced, and that the compressed image statistically equivalent to a genuine raw image.

We achieve a compression ratio in the range 5.1–10.1 and a throughput capable of sustaining the data rate requirements of modern cameras, together with power requirements compatible with nanosatellites.

3. WORKING PRINCIPLE

An overview of the compression algorithm is shown in Figure 4. The algorithm is split into three distinct components: image preparation (1), compression (2), decompression (3).

Image preparation — Fist, the image preparation routine uses sensor calibration data to correct systematic errors, such as PRNU. This increases image quality and
Figure 2. Illustration of issues arising from visually lossless compression when used in AI applications. Image (a) is a visually lossless compression of image (x). Indeed, the human eye does not see any differences between images (a) and (x). However, when a 4x AI-based upsampling algorithm is applied, clear differences become apparent: image (b) was generated with image (a) as input, and displays clear artefacts, whereas image (y), generated from image (x) remains clean. This data was kindly contributed by Dr. Yosef Akhtman from the company gamma.earth.

Figure 3. Illustration of de-noising performance achieved by training a convolutional neural network on calibrated raw data. The left “noisy image” is a crop of a microscope image. The centre “ground truth” image is obtained by averaging 1000 consecutive frames, and displays a much higher SNR ratio, with respect to the noisy image. The right “AI” image was obtained with our in-house AI denoiser trained on raw calibrated data. It achieves an SNR increase of 9 dB with respect to the noisy image, where the reference is taken to be the ground truth image. Data courtesy of Dr. Enrico Pomarico of HEPIA.
improves compression. Next, a calibrated sensor noise model is used to replace random noise from the image with pseudorandom noise. This step results in a small reduction of SNR (1.2 dB), but no other artefacts. Next, the noise model and random seed are then steganographically embedded within the image data. The output of image preparation is a “prepared raw” which carries all statistical and metrological properties of the original raw, besides a potential improvement in SNR due to the corrections, and a 1.2 dB SNR reduction due to noise replacement. This image also contains steganographically embedded metadata.

Compression — Image sensor model parameters and pseudorandom seed are extracted from their steganographic embedding and used by the predictor to achieve accurate prediction. Prediction residuals are then encoded using Huffman coding. Image model parameters and pseudorandom seeds are saved in the compressed bitstream.

Decompression — The prepared image is reconstructed by inverting the steps used during compression.

The separation of compression into these 3 distinct parts is useful, as image preparation can be performed on camera, so that any downstream system may profit from efficient lossless compression, from having corrected images and calibration metadata. The metadata necessary to perform efficient lossless compression (seed and sensor model) are embedded within the data itself, which means that in its uncompressed state, image data may traverse any existing data pipeline or communication channel and remain highly losslessly compressible. As the compression step is lossless, it means that transcoding across different versions of our codec, or indeed any other lossless codec is easy.

4. IMPLEMENTATION PERFORMANCE

We implemented the system on 3 different architectures: Software, FPGA and GPU/VPU. The FPGA implementation is intended to be used in flight, whereas the software and GPU/VPU implementations are intended to be used on earth.

4.1. Software

The software implementation is written in C++. It is extensively tested (96% code coverage) and acts as a reference for the FPGA and GPU/VPU implementations. It may be used as a command-line utility, as a library compressing raw data blocks, or as a codec for existing file formats (e.g. TIFF). It is multithreaded, however, the performance results below are shown for a single thread.

The benchmarks were run on a virtual machine running Ubuntu 16.04 LTS on a computer equipped with a processor of the type Intel Xeon) CPU E3-1230 v6 and 16 GB of RAM. The virtual machine has access to two processor cores, but the benchmark is utilizing a single core, only.

The benchmark results are presented in Fig 5. The compression rates (left panel) are typically around 140 Mpix/s (280 MB/s). Similarly, decompression rates are presented in the middle panel of Fig 5. Also here, the rates are mostly independent of the specific ISO. The decompression rate is about one third of the compression rate, as more optimization effort has been put towards the encoder w.r.t. the decoder. The right panel of Fig 5 shows the compression ratio. The ratio was calculated with respect to the input file size, which stores the data in an uncompressed format that uses 16 bits per pixel. The output size in bits per pixel (BPP) is also given in the figure. For a given image, the compression ratio increases with ISO, because at higher iso, the intrinsic information content of the image is reduced due to the lower exposure time, which gives the compression algorithm an opportunity to further decrease file size without affecting quality. For all 11 images in the set, the compression ratio exceeded the bound of a factor of 5. However, as the algorithm guarantees the retention of image information, the compression ratio does depend on image content, so no guarantees can be given that the ratio will always be larger than 5. Because of this heavy dependence of compression to image content, particular care was taken in collecting the test image dataset, as described is Section 6.

No specific processor hardware features, such as SIMD instructions, were explicitly used.
4.2. FPGA

The software reference implementation uses floating-point operations, which were translated to fixed-point operations in the FPGA, so that in rare cases, the least significant bit of a calculation result differs between the FPGA and reference software implementation.

We implemented the pipeline at several fixed-point precisions, and here present results for implementations with 8-bit and 16-bit precision after the decimal point. The result of each of these implementations are within the image quality specification. The 8-bit implementation privileges performance and resource utilization, whereas the 16-bit implementation privileges similarity to the software floating-point implementation.

The synthesis and implementation were performed with Vivado 2018.2. Synthesis was done with respect to the KCU105 evaluation board, containing a KU040 device. As the architecture is identical to the target KU060, results are taken to be also valid for the latter.

4.2.1. 8-bit fixed-point decimal precision

Synthesis results for the 8-bit implementation are shown in Figure 2. The operating frequency is 200 MHz and power consumption is 680 mW, corresponding to 3.2 nJ/pixel. A proportion of 0.94% of pixels are different with respect to the reference implementation.

Table 2. 8-bit synthesis utilization results on KU040

<table>
<thead>
<tr>
<th>Resource</th>
<th>Utilization</th>
<th>Available</th>
<th>Utilization %</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>3593</td>
<td>242400</td>
<td>1.48</td>
</tr>
<tr>
<td>LUTRAM</td>
<td>975</td>
<td>112800</td>
<td>0.86</td>
</tr>
<tr>
<td>FF</td>
<td>4462</td>
<td>484800</td>
<td>0.92</td>
</tr>
<tr>
<td>BRAM</td>
<td>1</td>
<td>600</td>
<td>0.17</td>
</tr>
<tr>
<td>DSP</td>
<td>1</td>
<td>1920</td>
<td>0.05</td>
</tr>
<tr>
<td>IO</td>
<td>56</td>
<td>520</td>
<td>10.77</td>
</tr>
<tr>
<td>BUFG</td>
<td>2</td>
<td>480</td>
<td>0.42</td>
</tr>
</tbody>
</table>

4.2.2. 16-bit fixed-point decimal precision

Synthesis results for the 16-bit implementation are shown in Figure 3. The operating frequency is 180 MHz and power consumption is 763 mW, corresponding to 4.2 nJ/pixel. In this implementation, 235 pixels out of 5 Mpixels are different with respect to the software implementation, corresponding to 0.003%.

5. GPU/VPU

In our implementation, the number of operations required to prepare and transfer the data between CPU and GPU/VPU and back has a significant impact on the algorithm's performance, making an identical implementation on GPU/VPU unattractive. On the other hand, once data is transferred, a large number of operations may be executed in parallel. This allowed us to revisit the possibility of integrating an advanced predictor, as to more closely approach the intrinsic information content of the image. Adapting and testing predictors/coders such as MANIAC[SW16] (used in FLIF) to our pipeline showed that the compression ratio could be increased by 10%. These techniques provide state-of-the-art compression ratio, at a great cost in terms of performance (1 Mpixel/s), and at an increased complexity in terms of testing. We therefore opted for developing and training a neural network to be used as the predictor component. This strategy allows for describing a complex algorithm in terms a few matrix multiplications, and to profit from the massively parallel architectures provided by GPUs and other tensor-specialized hardware, such as VPPUs.

We developed a neural network model in Tensorflow on an Nvidia GeForce RTX2070 MaxQ and deployed it with Intel's OpenVINO framework to an Intel i7-9750H (2.6 GHz) CPU and also to an Intel NCS2 VPU device (Movidius Myriad X).

The compression ratio increased by approximately 10% for all images in the dataset, matching the compression performance of FLIF and achieving up to 10Mpix/s throughput. As only the predictor was affected, images remained bit-for-bit identical to those of the reference implementation. Results are shown in Figure 6.

6. TESTING AND VALIDATION

Testing was performed in four phases, simulation, unit/system testing, laboratory “black-box” statistical testing and application testing.

First, we artificially generated images using a sensor model adapted from the EMVA1288 standard[emv16]. This allowed us to verify that, at least under the assumptions of the model, for illuminations spanning the dynamic range of the sensor:

1. the difference between the original pixel value and compressed value is never greater than one standard deviation according to the sensor model.
Table 4. Performance summary of the predictor neural network running on different hardware and at different power levels. Tot power refers to the total power used by the computer (including screen) during prediction. Base power refers to the power used by the entire computer when idle. Actual power is the Tot power – Base power. Time is the execution time for prediction only. Pixels is the number of pixels in a batch. Pixels/s is the number of pixels that the predictor processes per second. µJ/pix is the energy used per pixel. The performance results for the VPU are not reported here because of a strong discrepancy between the theoretical and actual performance of the device, which we trust will be solved together with Intel.

<table>
<thead>
<tr>
<th>Device</th>
<th>Tot power (W)</th>
<th>Base power (W)</th>
<th>Actual power</th>
<th>Time (s)</th>
<th>Pixels</th>
<th>Pixels/s</th>
<th>µJ/pix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>70</td>
<td>20</td>
<td>50</td>
<td>0.86</td>
<td>5059600</td>
<td>3883256</td>
<td>8</td>
</tr>
<tr>
<td>CPU</td>
<td>25</td>
<td>17</td>
<td>8</td>
<td>1.5</td>
<td>5059600</td>
<td>3373067</td>
<td>2</td>
</tr>
<tr>
<td>CPU</td>
<td>50</td>
<td>17</td>
<td>33</td>
<td>0.88</td>
<td>5059600</td>
<td>5749545</td>
<td>6</td>
</tr>
<tr>
<td>GPU</td>
<td>21</td>
<td>17</td>
<td>4</td>
<td>1.5</td>
<td>5059600</td>
<td>3373067</td>
<td>1</td>
</tr>
<tr>
<td>GPU</td>
<td>55</td>
<td>17</td>
<td>38</td>
<td>0.49</td>
<td>5059600</td>
<td>10325714</td>
<td>4</td>
</tr>
<tr>
<td>VPU</td>
<td>1.2</td>
<td>n/a</td>
<td>1.2</td>
<td>n/a</td>
<td>1000000</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 6. Summary of compression ratios using a neural-network predictor, for the images in the test set.

2. the loss in SNR is smaller than the required 2 dB

3. compressed pixel values are unbiased.

Results are shown in Figure 7

Next, we performed unit/system testing to a high level of coverage (96%), to verify that the implementation is consistent with the theoretical model.

Then, we implemented “block-box” laboratory statistical testing. For each camera setting, we acquired 120 images of a test scene (Fig. 8 containing many elements that are challenging to compress, such as high-contrast detail and low and high contrast gradients. For each pixel in this test scene, we verified that compression errors remained within the specification criteria of maximum error (fig. 9) and SNR reduction (fig. 10) described above.

Finally comes application testing. To emulate a high-resolution satellite system (28cm ground-sampling distance, 2.5 pixel point-spread function diameter), we used a DJI Mavic 2 Pro Drone, equipped with a Hasselblad L1D-20c camera. This system has 2.4 µm pixels, in

Figure 7. Results of numerical testing of the compression algorithm on simulated data. a) shows that a negligible number of samples is beyond a single standard deviation. This number is consistent with statistical fluctuations apart from close to sensor saturation where the model may still have to be improved. b) shows that SNR loss due to compression is always smaller than 1.2 dB. c) shows that compression is completely unbiased both in the mean and median.

Figure 8. Test scene used to verify compression quality. The scene contains a number of elements engineered to present a challenge to compression. We acquire a number of identical images of the scene and verify the overlap of the individual pixel histograms between the original and compressed datasets.
Bayer filter array. We selected only half of the green pixels for tests, resulting in a pixel pitch of 4.8 μm. We assume a peak sensitivity in the middle of typical panchromatic range, i.e. at 625 nm.

Images were taken with the drone hovering for maximum stability. We verified this to be better than a single pixel by calculating the correlation of subsequent images. The objective has a focal length of 10.3 mm. We operated this objective at a f-number of \( N = 8 \), to emulate the PSF circle diameter relative to the pixel pitch and GTD. Operating at \( N = 8 \) also minimizes vignetting, aberrations, and increases depth of focus. We verified GSD and PSF by measuring the edge-spread function of a ground-based target.

Twelve different scenes, shown in Figure 11 were acquired representing urban, suburban, rural, forest, water and clouds in each of clear and diffuse lighting conditions. For each scene, four different combinations of gain and exposure settings (ISO) were taken (ISO 100, 200, 400, 800). These images were used to evaluate the compression ratio.

To further verify suitability of the algorithm for AI applications, we collaborated with six different institutions using advanced machine learning techniques in the field of medical diagnostics and biological research. We compared the statistics of results obtained directly from processing the raw data and those obtained processing compressed data. Results obtained for 2D segmentation tasks are shown in Figure 12. For each of the tasks, results obtained with this algorithm (labelled DP) are unbiased and statistics overlap with the statistics due to the natural measurement uncertainty. The 10:1 jpeg compression (commonly used for AI tasks), however, exhibits both increased statistical variation (up to 10 standard deviations) as well as bias (also up to 10 standard deviations). An article presenting these results is soon to be published.

7. CONCLUSION

In this paper, we have shown that preserving raw images is important to unlock the full potential of AI and quantitative applications. We demonstrated a fast and efficient compression method that preserves the metrological properties of the image, and is adapted to compress raw data. We detail the performance of software, FPGA and GPU/VPU implementations.

8. ACKNOWLEDGEMENTS

We are particularly grateful to Phillip Geier, Andres Upegui, Enrico Pomarico and Jerome Extermann from HEPIA, and Roberto Camarero from ESA for their invaluable expertise and advice in the domains of FPGA, testing methodology and space systems. We are also very
grateful to the Swiss Space Office and ESA, who funded this project through a GSTP Element 1 grant.

REFERENCES


