ONBOARD PAYLOAD-DATA DIMENSIONALITY REDUCTION

M. Penalver[(1)], F. Del Frate[(1)], M. E. Paoletti[(2)], J.M. Haut[(2)], J. Plaza[(2)], A. Plaza[(2)]

(1) University of Rome "Tor Vergata"
(2) University of Extremadura

ABSTRACT

The finer spatial, spectral and radiometric resolutions of current and planned sensors are rendering increasingly-high data rates which, coupled with limited on-board storage, downlink bandwidth and receiving ground station availability, make high-throughput, high-performance data-reduction techniques essential in forthcoming missions. On this paper we describe an algorithm well suited to high-dimensional data as those produced by multispectral and hyperspectral sensors, both highly relevant in a broad range of Earth Observation activities with the latter becoming increasingly available and delivering the highest data rates. The performance of parallel implementations of the algorithm on multi-core and GPU architectures is also evaluated.

Index Terms— spectral decorrelation, dimensionality reduction, compression, multispectral, hyperspectral, neural network, multi-core, GPU, parallel computing

1. INTRODUCTION

The noisiness of high spatial- and/or spectral-resolution sensors limits considerably the compression ratio offered by lossless techniques [1]. Lossy compression, on the other hand, allows a higher scene acquisition rate, making satellite operation more profitable while still allowing for data analysis sufficiently rich for feature extraction. It is therefore foreseeable that in the near future EO, science and deep-space missions will require lossy compression for their multispectral and hyperspectral instruments.

The algorithms so far developed to handle 3-D images, e.g. JPEG2000, 3D Tarp, 3D-SPIHT, 3D-SPECK, 3D-ICER, or extensions to the Consultative Committee for Space Data System Image Data Compression specification (CCSDS 122.0) [2]-[13], while offering good compression performance, have a prohibitive complexity and memory footprint for space applications.

After the publication by the CCSDS of the Lossless Multispectral & Hyperspectral Image Compression standard (CCSDS-123.0) defining a lossless technique suitable for onboard compression of multispectral and hyperspectral images, two additional efforts developing lossy methods are on the works: Low-Complexity Near-Lossless Multispectral & Hyperspectral Image Compression (CCSDS-123.1) [14], which bases on CCSDS-123.0 to provide near-lossless compression, and Spectral Pre-Processing Transform for Multispectral & Hyperspectral Image Compression (CCSDS-122.1) [15], which extends CCSDS 122.0 by defining a pre-processing spectral-decorrelation transform. In contrast with CCSDS-123.1, however, CCSDS-122.1 cannot guarantee a user-defined maximum reconstruction error, and the higher complexity of this transform-based approach makes it less effective at high bit rates [14].

Principal component analysis (PCA) has been widely used for spectral decorrelation and dimensionality reduction of hyperspectral data. However, the high computational complexity of the covariance estimation makes challenging its use on board. On the other hand, using the amount of signal energy as criterion to determine a number of spectral components that preserves sufficient information might leave out subtle relevant objects [16].

A nonlinear generalization of the standard PCA (NLPCA) performed by an autoassociative neural network [17] avoids both these problems, efficiently rendering principal components that share the information content uniformly. In addition, NPLCA can detect and remove both linear and nonlinear correlations, and is less affected by sensor error than PCA and other linear transformations like Maximum Noise Fraction or Independent Component Analysis [18]. NLPCA has already been successfully applied to a wide variety of datasets (e.g. [19]), and the fully-automated process described on this paper was successfully implemented on a platform developed for payload-data processing and communication [20].

We subsequently report a first evaluation of the performance gains that an implementation of this algorithm on parallel computing architectures (multi-core and GPU), arriving also to spacecraft development [21], could bring about.

2. ALGORITHM DESCRIPTION

The neural network model used is a multilayer perceptron [22] with three hidden layers, the middle one of which has smaller dimension than both the input and output layers (*Fig. 1*). The network is trained using the error-correction learning rule implemented through the backpropagation technique [22], in order to render at the output layer an approximation of the input vector. The activation values of the middle-layer nodes provide then a compact representation of the input. Therefore, the first part of the network, including the middle layer now as

output units, corresponds to the compressor algorithm, whereas the second part, which considers the middle units as inputs, embodies the decompressor to be transferred to ground.

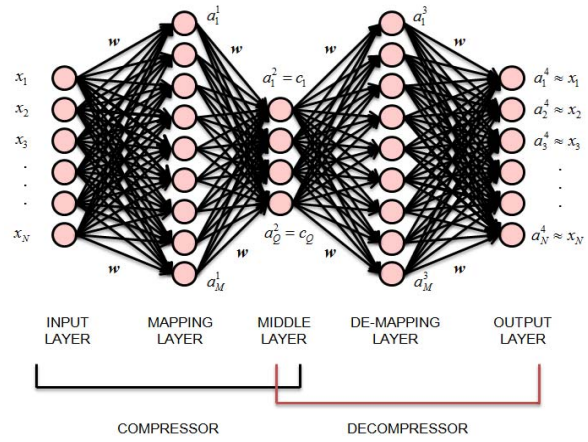

Fig. 1. Neural network for spectral compression

Since a satellite is to fly over very distinct and dynamic landscapes, the imagery produced by a payload instrument may vary largely. For this reason, the selection of the specific network topology and its training is deferred to the instrument's validation period once already in orbit, when real and comprehensive data become available for the network to extract their underlying properties.

Whereas the number of nodes in the middle layer is determined by the desired compression ratio, there is no definitive method for deciding a priori the dimensions of the mapping/demapping layers. The network must include enough units in those layers to be able to sufficiently approximate the identity function of the input data given the number of middle nodes. Too many mapping/demapping units, however, will enable the network to fit the specific traits of the training examples, which will impact its generalization capability to untrained data.

In our approach, we start with as many mapping/demapping nodes as the number of middle units in order to guarantee the capability of the compressor and decompressor networks to respectively represent and reconstruct the complexity underlying the input data at the given compression level. We then try increasing numbers of mapping/demapping nodes by training the resulting networks from initial random weights with random samples from specific areas throughout the satellite's orbit and evaluating the accuracy of the resulting implementations of the identity function.

The training of a network is considered completed when the slope of the straight line fitting the magnitude of the test reconstruction error across the last Q epochs [23] is above a threshold ($\hat{\beta}_{training}$). This situation is indicative that no further weight optimization is taking place resulting in a test error that either remains stable or has started to increase as the network begins to over-fit the training data capturing its particular noisy traits.

Following the principle of the pocket perceptron learning algorithm [24], those that produced the lowest test error during the training session are considered the optimal weights.

The test error will typically decrease as we increase the number of hidden nodes until a minimum is reached after which additional nodes will produce no noticeable error reduction or will cause it to increase as the network begins to over-fit the training data. Hence, the search for the proper topology will end when either the optional target test error ($\overline{SSE}_{test}^{\min}$) is reached, or the slope of the straight line fitting the magnitude of the lowest test error (produced by the optimal weights) across the last R sessions [23] over networks with subsequently larger number of hidden nodes is found to be above a threshold ($\hat{\beta}_{topo\log y}$).

Among all the trained topologies, that whose optimal weights produce the lowest test error will be considered the optimal one for the target compression level.

A high test error by the optimal topology will be indicative that the training and/or test sets are not sufficiently representative of the potential input data. Although improbable, it might occur that the sets, chosen randomly from the areas of interest, have left out relevant types of training examples and/or include mostly irrelevant test values.

With the test error above $\overline{SSE}_{test}^{\min}$, new training and test sets will be collected from the same areas, and reconstruction error will be calculated over the new test set to rule out the possibility that the previous one included too many uncommon samples that are not a priority for the network to replicate accurately. If the test error stays above threshold, it is likely that the previous training set did not include sample types that present properties relevant to describe the input space's underlying structure. In that case, and to clear the fitting performed previously to potentially rare, unimportant data, training will be restarted with the network topology and weights prior to the latest training session. The entire process will be repeated as necessary until $\overline{SSE}_{test}^{\min}$ is reached, although in most cases a single iteration will suffice.

A set of parameters (*table 1*) need to be determined empirically while still on ground using simulated sensor data and taking into account the characteristics of the spacecraft's orbit. These parameters may be updated later on if suggested by additional tests performed on ground when actual data become available during the in-orbit validation campaign.

The network topology found to be optimal might be modified later on following a user request for a different compression ratio.

Moreover, the network's accuracy will continue to be checked at random times in order to maintain a minimal information loss at the current compression level.

Although the initial training areas are to be carefully picked from the spacecraft's orbit as to include a comprehensive selection of relevant instrument values, the dynamic nature of the explored surface might render after the initial training phase additional representative readings potentially contributing to the robustness of the network, bringing about the need for further training with the new unfitted data. A potential increase in the complexity of the acquired-data space might also require additional mapping/demapping nodes to expand the network's representation and reconstruction capabilities.

Name	Description
Training acquisition plan	Geolocation of the areas from where training samples should be collected.
$\overline{SSE}_{test}^{\min}$	Target test error in order to consider the network configuration (topology and weights) as optimal.
η	Learning rate
Q	Number of consecutive epochs to calculate the test-error trend during the search for optimal weights.
$\hat{\beta}_{training}$	Target slope of the test-error line over Q consecutive epochs in order to consider a training session completed.
S	Number of mapping/demapping nodes to be added when searching for the optimal topology.
R	Number of consecutive sessions to calculate the lowest-test-error trend during the search for an optimal topology.
$\hat{\beta}_{topo\log y}$	Target slope of the lowest-test-error line over R consecutive sessions in order to consider the training completed even if $\overline{SSE}_{test}^{\min}$ was not reached.

Table 1. Parameters to be determined empirically on ground

3. RESULTS

The algorithm was implemented in Python 2.7 and using the numerical library Numpy 1.11, and the data-flow-graph library Tensorflow 0.12 with the Adam optimizer [25]. COTS hardware was targeted: Intel® Core™ i7-870 CPU, 16 GB RAM, and GeForce® GTX 650Ti GPU with 2GB RAM.

The implementation was tested with data from the Airborne Hyperspectral Scanner imaging radiometer [26] including 75 bands.

A network with 9 nodes in the middle layer, corresponding to a compression ratio above 8, was trained with a total of 1,440,000 samples, organized in batches of 10, from a set of 9 images, and a target reconstruction error of 4E-2. The focus of the study was on the performance evolution with a growing number of mapping/demapping nodes.

Fig. 2 depicts the execution times and speedups as reported in *table 2*, obtained using 1, 2, 4 and 8 CPU threads, and a GPU. Each value corresponds to the average over 30 executions of the training process using the same sample sets and arriving to the same optimal topology.

Due to the massively parallel nature of the algorithm's bulk, the GPU's execution time is considerably lower, with greater speedups for larger networks. For the CPU, instead, there seems to be a saturation number of threads – that increases with the number of mapping/ demapping nodes – above which no further speedup is achieved for a particular topology. To further explore this behavior, a pertinent future exercise will be to train large networks on a many-core system like the Intel® Xeon Phi™.

4. ACKNOWLEDGEMENT

The authors wish to thank GEO-K S.r.l. for the fruitful collaboration.

The parallel implementation of the algorithm was supported by Ministerio de Educación (Resolución de 26/12/2014 y de 19/11/2015) and by Junta de Extremadura (Decreto 297/2014, Ref. GR15005).

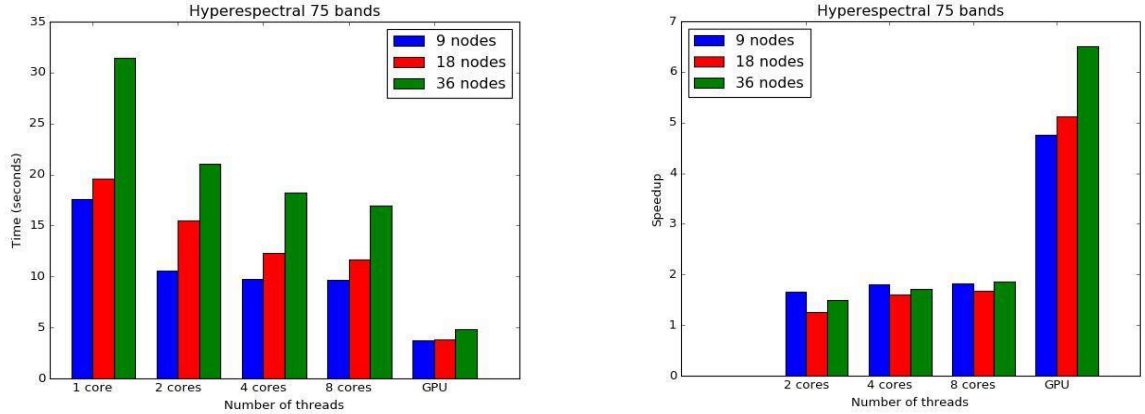

Fig. 2. Execution times (***left***) and speedups (***right***) using a CPU and a GPU

Topology	Multicore CPU (number of cores)				GPU
	1	2	4	8	
75-9-9-9-75	17.61 (0.476) [1.0]	10.62 (0.352) [1.66]	9.79 (0.158) [1.80]	9.71 (0.146) [1.82]	3.70 (0.024) [4.76]
75-18-9-18-75	19.57 (0.719) [1.0]	15.51 (0.571) [1.26]	12.27(0.221) [1.60]	11.69 (0.157) [1.67]	3.82 (0.033) [5.12]
75-36-9-36-75	31.40 (1.51) [1.0]	21.07 (1.761) [1.49]	18.27 (0.298) [1.72]	16.96 (0.227) [1.85]	4.82 (0.207) [6.51]

Table 2. Execution times (in seconds) obtained using a CPU and a GPU. The standard deviation among the 30 trials is included between parentheses for each value. The speedups are indicated between square brackets

5. REFERENCES

[1] B. Aiazzi, L. Alparone, and S. Baronti, "Near-lossless compression of 3-D optical data," IEEE TGRS 39(11), 2547–2557, 2001.

[2] B. Penna, T. Tillo, E. Magli, G. Olmo, "Embedded lossy to lossless compression of hyperspectral images using JPEG 2000," IGARSS Proc. 1(4), 25–29, 2005.

[3] J. T. Rucker, J. E. Fowler and N. H. Younan, "JPEG2000 coding strategies for hyperspectral data," IGARSS. Proc.1, 2005.

[4] JANI, RINA, and DRGR KULKARNI. "Performance analysis of role of wavelet in CCSDS image data compression algorithm."

[5] T. Xiaoli, C. Sungdae, W.A. Pearlman, "3D set partitioning coding methods in hyperspectral image compression," Proc. IEEE ICIP, 2003.

[6] J. Zhang et al. "Improvements to 3D-TARP Coding for the Compression of Hyperspectral Imagery," IGARSS Proc. 2, 2008.

[7] T., Xiaoli, W. A. Pearlman and J. W. Modestino, "Hyperspectral image compression using three-dimensional wavelet coding: a lossy-to-lossless solution," IEEE TGRS, 2004.

[8] J. Wu et al. "Hyperspectral image compression using distributed source coding and 3D SPECK," Sixth International Symposium on Multispectral Image Processing and Pattern Recognition. International Society for Optics and Photonics, 2009

[9] E. Christophe, "Hyperspectral data compression tradeoff," Optical Remote Sensing, 9–29, Springer Berlin Heidelberg, 2011.

[10] M. Enrico, G. Olmo and E. Quacchio. "Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC," IEEE GRSL, 21–25, 2004.

[11] J. E. Fowler and J. T. Rucker, "Three-dimensional wavelet-based compression of hyperspectral imagery," Hyperspectral Data Exploitation: Theory and Applications, 379–407, 2007

[12] Y. Hou, "Lossy-to-Lossless Compression of Hyperspectral Image Using the 3D Set Partitioned Embedded ZeroBlock Coding Algorithm." Journal of Software Engineering and Applications 2(2), 86–95, 2009.

[13] Y. Hu, W. A. Pearlman and X. Li, "Progressive Significance Map and Its Application to Error-Resilient Image Transmission," IEEE TIP 21(7), 3229–3238, 2012.

[14] CCSDS Multispectral and Hyperspectral Data Compression Working Group. Online: http://cwe.ccsds.org/fm/Lists/Projects/DispForm.aspx?ID=517

[15] CCSDS Multispectral and Hyperspectral Data Compression Working Group. Online: http://cwe.ccsds.org/fm/Lists/Projects/DispForm.aspx?ID=386

[16] C. Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," IEEE TGRS 42(3), 608–619, 2004.

[17] M. Kramer, "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," AIChE, 37, 233-243, 1991.

[18] G. Licciardi, F. Del Frate, "Pixel unmixing in hyperspectral data by means of neural networks," IEEE TGRS, 49(11), 4163-4172, 2011.

[19] P. Reddy Marpu, G. Licciardi, P. Gamba, F. Del Frate, "Comparison of advanced Neural Network Architectures For Hyprspectral Data Classification," IEEE GRSS WS on Hyperspectral Image and S. Processing, Reykjavik, 2010.

[20] SpacePDP: On-board payload data processing. Online: http://www.planetek.it/eng/projects/spacepdp_on_board_payload_data_processing

[21] R. Trautner, "ESA's roadmap for next generation payload data processors," Proc. DASIA Conf. Vol. 1, 2011.

[22] C. M. Bishop, "Neural Networks for Pattern Recognition," Oxford University Press, Inc., NY, 1995.

[23] J. F. Kenney and E. S. Keeping, "Linear Regression and Correlation." Ch. 15 in Mathematics of Statistics, Pt. 1, 3rd ed. Princeton, 252–285, 1962.

[24] S.I. Gallant, "Three constructive algorithms for network learning", Proc. 8th Ann Conf of Cognitive Science Soc., Amherst, 652–660, 1986.

[25] D. Kingma, and J. Ba, "Adam: A method for stochastic optimization." arXiv preprint 1412(6980), 2014.

[26] Online: http://www.uv.es/leo/sen2flex/ahs.htm