SCALING UP A MULTISPECTRAL RESNET-50 TO 128 GPUS

Rocco Sedona[1,2], Gabriele Cavallaro[1], Jenia Jitsev[1], Alexandre Strube[1], Morris Riedel[1,2] and Matthias Book[2]

[1] Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany
[2] School of Engineering and Natural Sciences, University of Iceland, Iceland

ABSTRACT

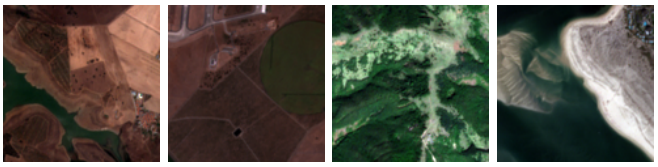
Similarly to other scientific domains, Deep Learning (DL) holds great promises to fulfil the challenging needs of Remote Sensing (RS) applications. However, the increase in volume, variety and complexity of acquisitions that are carried out on a daily basis by Earth Observation (EO) missions generates new processing and storage challenges within operational processing pipelines. The aim of this work is to show that High-Performance Computing (HPC) systems can speed up the training time of Convolutional Neural Networks (CNNs). Particular attention is put on the monitoring of the classification accuracy that usually degrades when using large batch sizes. The experimental results of this work show that the training of the model scales up to a batch size of 8,000, obtaining classification performances in terms of accuracy in line with those using smaller batch sizes.

Index Terms— Distributed deep learning, high performance computing, residual neural network, convolutional neural network, classification, sentinel-2

1. INTRODUCTION

The field of EO has rapidly evolved in the last decades due to the continuous technological advances incorporated into RS instruments. Space agencies from all around the globe fund a large number of EO programs to collect data on a daily basis. Copernicus, with its fleet of Sentinel satellites, is the world's largest single EO programme. For example, the two twin satellites Sentinel 2A and 2B deliver 23 TB/day [1] of Multispectral (MS) data [1] thanks to their high temporal resolution (i.e., 5-day revisit periodicity at the equator). EO data provide a great value for a wide range of applications, such as civil protection, traffic monitoring, climatology and commercial sectors.

DL networks have achieved large attention in a wide variety of scientific fields in the recent years due to their significant increase in performance with respect to traditional Machine Learning (ML) methods. In particular, the rise of CNN has enabled to deploy very accurate classifiers also in the RS domain. Until recently only a few annotated datasets were available. New large datasets, such as BigEarthNet [2], are of fundamental importance to train deep neural networks, but training on big data causes an increase in time required for training the models. In [3] the authors showed that with a data parallelization framework the training step of a Resnet50 [4] deep learning model can be accelerated. They reported that it was possible to scale up the training up to 24 nodes (i.e., 96 Graphics Processing Units (GPUs)) by maintaining stable accuracy performances.



(a) (b) (c) (d)

Fig. 1: Example of patches with their corresponding classes: (a) agro-forestry areas, complex cultivation patterns, non-irrigated arable land, transitional woodland/shrub, water bodies, (b) airports, olive groves, permanently irrigated land, (c) broad-leaved forest, burnt areas, transitional woodland/shrub, (d) beaches/dunes/sands, estuaries, sea and ocean, sport and leisure facilities [3].

The objective of this work is to extend the distributed training up to 32 nodes (i.e., 128 GPUs), and preserve the classification performance as the ones that can be obtained by smaller batch sizes. The experiments were run to assess the performances of the batch sizes 8,000, 16,000 and 32,000. Instead of using the Stochastic Gradient Descent (SGD) optimizer, a newer approach called Layer-wise Adaptive Rate Scaling (LARS) is adopted to reduce the risk of training divergence with large batch size [5].

2. PROBLEM FORMULATION

The problem that is tackled in this work is multilabel classification [6]. The considered dataset is BigEarthNet [2] (described further in detail in Section 4.1). It consists of patches that are annotated with a number of labels that are not mutually exclusive and can be also correlated [2]. BigEarthNet takes into account that the annotation of patches with a sin-

[1] https://sentinels.copernicus.eu/web/sentinel/news/-/article/2018-sentinel-data-access-annual-report

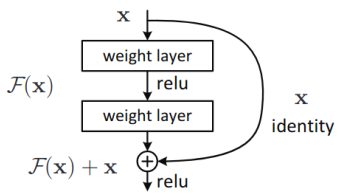


Fig. 2: ResNet-50: residual block [4].

gle class is not enough to describe the semantic content of a scene. For example, roads and agriculture classes can be present in the same patch as it can be observed in Fig. 1.

The binary cross entropy is used as loss function. Each sample is associated to a vector with the number of cells equal to the total number of classes, in this case equal to 43. With the binary cross-entropy, each label is treated separately from the others. In the last layer of the network the sigmoid function is employed to map the values of the input vector in the range between 0 and 1. This is different from the softmax function used for multiclass classification, which reprojects the values so that they sum up to 1. The sigmoid function, on the other hand, ensures that the labels are not mutually exclusive in the multilabel case, so that more than one label can be associated to each sample.

3. METHODOLOGY

3.1. ResNet-50

In this work a well established deep CNN, ResNet-50, is used. Developed in 2015, ResNet-50 is still used as benchmark for various classification tasks. ResNet-50 is a deep network, which overcomes the difficulties of training with a large number of layers (vanishing gradient problem) by using skip connections, as shown in Fig. 2. Instead of directly fitting the underlying mapping $H(x)$, the residual mapping $F(x) := H(x) - x$ is learned [4]. In ResNet the skip connections is implemented as identity mappings, resulting in the formulation $F(x) + x$ shown in Fig. 2. In [4] the authors state that learning the residual mapping helps in extending the depth of the network without incurring into the vanishing gradient problem.

3.2. HPC and distributed training

The increased amount of data that is fed into the neural networks results in longer training time. A possible approach to tackle this problem is to distribute the training process on multiple GPUs available on HPC systems. Two families of paradigms exist to do that, the model distribution and the data distribution [7]. In this work the focus is on the latter approach. In data parallelism a replica of the model is loaded on each GPU first. Then each model is run and its parameters are estimated independently. At the end of each iteration the models are synchronized by exchanging the parameters between the different workers. Horovod is used to implement the data parallel approach. It is a decentralized framework, based on MPI and NCCL libraries, where actors exchange parameters without the need of a parameter server [8]. In order to perform the parameter exchange (shown in Fig. 3) Horovod makes use of the ring-allreduce algorithm implemented by MPI. The amount of data transferred between each GPU can be computed as $D = 2(N-1)\frac{K}{N}$, where N is the number of GPUs and K is the total number of values in array being summed across the different GPUs [9]. From the equation it follows that the ring-allreduce operation enables a constant communication cost independently of the number of GPUs.

3.3. LARS optimizer

Training with a large number of GPUs in a data parallel fashion results in an increase of the effective batch size, that can cause instability during the training process [10]. A gradual warm-up was proposed by [11] to obtain stable testing performances with batch sizes up to 8,000 samples using the SGD optimizer. When dealing with effective batch sizes larger than that threshold and computing the Learning Rate (LR) with a linear policy, the simple adoption of warm-up does not avoid training divergence leading to degradation of model accuracy. Instead of using the more aggressive linear policy to set the LR, different schemes such as the square root scaling were proposed [12]. Moreover, it was observed that a LR that is large with respect to the ratio between the L2-norm of weights and gradients can cause instability in the training process [5]. While SGD uses the same LR for all layers, LARS optimizer aims at overcoming this difficulty by adapting the LR for each layer. The local LR is computed for each layer as: $\lambda^l = \eta \times \frac{\|w^l\|}{\nabla L(w^l)}$, where w^l is the set of weights at layer l, and L is the loss function.

4. EXPERIMENTAL RESULTS

4.1. Dataset

BigEarthNet [2] is an archive containing 590326 patches extracted from 125 Sentinel-2 tiles (Level 2A) acquired from June 2017 to May 2018 [2]. Each patch is annotated with a number of labels between 1 and 12, making it a multilabel classification task. The 43 labels are derived from the CORINE Land Cover (CLS), consisting in annotations from 10 European countries updated in 2018. The patches have 12 spectral bands: (a) the 3 RGB bands and band 8 at 10m resolution (120×120 pixels), (b) bands 5, 6, 7, 8a, 11, 12 at 20m resolution (60x60 pixels) and (c) band 1 and 9 at 60m resolution (20×20 pixels). Band 10 has not been considered since

[2] http://bigearth.net/

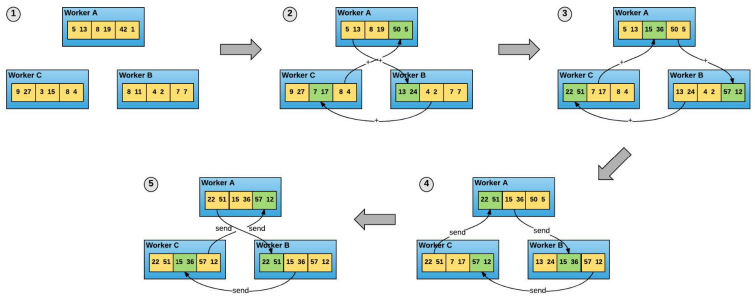


Fig. 3: Ring-allreduce algorithm for exchange of the gradient between the workers [8].

it does not provide valuable information on land cover and is used for different purposes such as cirrus detection [13]. In this work patches covered with a significant amount of snow or clouds were excluded using the list of patches provided by the creators of BigEarthNet [14]. Here, in order to make training in a parallel fashion easier, the patches of the BigEarthNet archive were converted into a Hierarchical Data Format 5 (HDF5) file, which offers routines for parallel read and write operations.

4.2. Experimental Setup

The experiments were carried out on Jülich Research on Exascale Cluster Architectures (JURECA) [15] supercomputer, an HPC system installed at the Jülich Supercomputing Centre. In particular, the partition on which experiments were run has 75 nodes each equipped with four NVIDIA K80 GPUs (with 24GB of memory each). In this work we have run the experiments using 32 nodes, i.e., 128 GPUs.

The Python libraries used for training and testing the models are: `TensorFlow 1.13.1`, `Keras 2.2.4`, `Horovod 0.16.2`, `Mpi4py 3.0.1` and `Scikit-learn 0.20.3`.

The original Sentinel-2 bands at lower resolution were upsampled to the maximum resolution of 10 m using a simple bilinear interpolation. The Geospatial Data Abstraction Library `GDAL 2.3.2` was used to extract the patches from the Level-2A tiles.

ResNet-50 was fed with the input containing 12 upsampled bands. Two data augmentation techniques were applied, randomly flipping and rotating the patches. In addition to that, L2 regularization was employed in all convolutional layers and a dropout was placed before the last layer of the model to reduce the risk of overfitting. LARS optimizer is adopted with Nesterov momentum [5]. The initial LR is computed using a linear policy as $\eta = 0.1\frac{kn}{256}$ [11], where k is the number of workers (i.e., GPUs) and n is the batch size for each worker (set here to 64 for the 8,000 effective batch size case, to 128 for the 16,000 case and to 256 for the 32,000 case). A scheduler with deterministic annealing was implemented. The LR at each epoch was computed following a multi-step decay scheme. The original LR was multiplied by 0.1 after 30 epochs, by 0.01 after 60 epochs and by 0.001 after 80 epochs. In order to avoid instability problems during the computation of the gradient at the beginning of training with a high LR , a warm-up of 5 epochs was used for all the experiments.

4.3. Evaluation

The metric used to evaluate the performances of the model on the test subset is the F1 score. In Table 1 results are reported for the different configurations (number of GPUs and batch size). If batch size equal to 512 and 8,000 are considered, it can be observed that the F1 score declines from 0.78 to 0.74. However, in [3] it was already reported that with an effective batch size of 4,000 the F1 score was lower (0.74) than the cases where a smaller batch size was used (0.78). In the present study performances are not declining drastically up to a batch size of 8,000. However, it can be observed that doubling the effective batch size from 8,000 to 16,000 leads to poorer results. F1 score drops from 0.74 in the former case to 0.64 in the latter case. F1 score results diverge significantly when considering a batch size of 32,000.

Training time are reported in Table 2. Horovod allows to cut significantly the time elapsed for training. It can be observed that training time shrinks from 49,400 s for a full run with effective batch size of 512 to 3,400 s with effective batch size equal to 8,000, i.e., about 14 times faster, resulting in a slightly less than linear scaling (for the linear scaling, it should be 16 times faster). It should be noted that although for the cases in which batch sizes are set to 8,000, 16,000 and 32,000 the same number of GPUs equal to 128 is used, the training time differs. This is caused by the fact that there are fewer read operations from HDF5 files when a larger batch size is considered.

5. CONCLUSIONS

This paper shows that it is possible to apply distributed DL methods on a large RS dataset by reducing the training time of

Table 1: Classification results for the multispectral model using different effective batch sizes: F1 score.

batch size	n. GPUs	warm-up	initial LR	F1
512	8	5	0.2	0.78
8,000	128	5	3.2	0.74
16,000	128	5	6.4	0.64 (diverge)
32,000	128	5	12.8	0.43 (diverge)

Table 2: Total training time for the multispectral model using different effective batch sizes.

batch size	n. GPUs	training time [s]
512	8	49,400
8,000	128	3,400
16,000	128	2,800
32,000	128	2,500

the networks, maintaining at the same time a stable F1 score up to a batch size of 8,000 samples. Results undertake a declining path with larger batch sizes, diverging significantly with larger batch sizes. Future work may focus on the implementation of different LR policies (f.e. the root scaling with polynomial decay [5]) that have shown to allow training with larger batch sizes. The adoption of more advanced approaches such as Layer-wise Adaptive Moments optimizer for Batch training (LAMB) could be studied as well, in order to extend the training up to a batch size equal to 16,000 or 32,000 without a drop in testing accuracies [16]. Scalability in terms of training time should also be evaluated with the deployment of TFRecord, a data format optimized specifically for TensorFlow. A repository with the code implementation will become available soon [3].

6. REFERENCES

[1] J. Aschbacher, "ESA's Earth Observation Strategy and Copernicus," in *Satellite Earth Observations and Their Impact on Society and Policy*, 2017.

[2] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, "Bigearthnet: A Large-Scale Benchmark Archive for Remote Sensing Image Understanding," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019, pp. 5901–5904.

[3] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and J. A. Benediktsson, "Remote Sensing Big Data Classification with High Performance Distributed Deep Learning," *Remote Sensing*, vol. 11, no. 24, p. 3056, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[5] Y. You, I. Gitman, and B. Ginsburg, "Large Batch Training of Convolutional Networks," 2017.

[6] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-label Classification," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 254–269.

[7] T. Ben-Nun and T. Hoefler, "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis," *ACM Computing Surveys*, 2019.

[8] A. Sergeev and M. D. Balso, "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow." *arXiv:1802.05799*, 2018.

[9] A. Gibiansky, "Bringing HPC Techniques to Deep Learning," http://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/, accessed: 2019-09-23.

[10] A. Krizhevsky, "One Weird Trick for Parallelizing Convolutional Neural Networks," 2014.

[11] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour," *ArXiv*, vol. abs/1706.02677, 2017.

[12] E. Hoffer, I. Hubara, and D. Soudry, "Train Longer, Generalize Better: Closing the Generalization Gap in Large Batch Training of Neural Networks," in *Advances in Neural Information Processing Systems*, 2017.

[13] "Sentinel2 B10: High Atmospheric Absorption Band." [Online]. Available: https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-1c/cloud-masks

[14] "Scripts to Remove Cloudy and Snowy Patches." [Online]. Available: https://gitlab.tubit.tu-berlin.de/rsim/bigearthnet-tools

[15] Jülich Supercomputing Centre, "JURECA: Modular Supercomputer at Jülich Supercomputing Centre," *Journal of large-scale research facilities*, vol. 4, no. A132, 2018.

[16] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh, "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes," 2019.

[3] https://gitlab.version.fz-juelich.de/CST_DL/projects/remote_sensing/igarss2020_distributed_resnet50