Remote Sensing Image Classification Using CNNs With Balanced Gradient for Distributed Heterogeneous Computing

Sergio Moreno-Álvarez, Mercedes E. Paoletti, *Senior Member, IEEE*, Gabriele Cavallaro, *Member, IEEE*, Juan A. Rico, and Juan M. Haut, *Senior Member, IEEE*

***Abstract*— Land-cover classification methods are based on the processing of large image volumes to accurately extract representative features. Particularly, convolutional models provide notable characterization properties for image classification tasks. Distributed learning mechanisms on high-performance computing platforms have been proposed to speed up the processing, while achieving an efficient feature extraction. High-performance computing platforms are commonly composed of a combination of central processing units (CPUs) and graphics processing units (GPUs) with different computational capabilities. As a result, current homogeneous workload distribution techniques for deep learning (DL) become obsolete due to their inefficient use of computational resources. To address this, new computational balancing proposals, such as heterogeneous data parallelism, have been implemented. Nevertheless, these techniques should be improved to handle the peculiarities of working with heterogeneous data workloads in the training of distributed DL models. The objective of handling heterogeneous workloads for current platforms motivates the development of this work. This letter proposes an innovative heterogeneous gradient calculation applied to land-cover classification tasks through convolutional models, considering the data amount assigned to each device in the platform while maintaining the acceleration. Extensive experimentation has been conducted on multiple datasets, considering different deep models on heterogeneous platforms to demonstrate the performance of the proposed methodology.**

***Index Terms*— Distributed computing, heterogeneous computing, image classification, remote sensing.**

Manuscript received March 2, 2022; revised April 22, 2022; accepted May 4, 2022. Date of publication May 10, 2022; date of current version May 24, 2022. This work was supported in part by the Spanish "Ministerio de Ciencia e Innovación" under Project PID2019-110315RB-I00/AEI/10.13039/501100011033 (APRISA), in part by the European Regional Development Fund "A way to achieve Europe" (ERDF), in part by the "Junta de Extremadura" under Grant IB20040, in part by the "Consejería de Economía, Ciencia y Agencia Digital (Junta de Extremadura)," in part by the "Fondo Europeo de Desarrollo Regional de la Unión Europea" under Grant GR21099 and Grant GR21040, in part by the Project HPC-EUROPA3 with code "HPC171QKDE" under Grant INFRAIA-2016-1-730897, in part by the EC Research Innovation Action through the H2020 Programme under Grant 730897, and in part by the DEEP-EST Project Computing Resources financed by the European Unions Horizon 2020 Research and Innovation Programme under Agreement 754304. *(Corresponding author: Sergio Moreno-Álvarez.)*

Sergio Moreno-Álvarez and Juan A. Rico are with the Department of Computer Systems Engineering and Telematics, University of Extremadura, 06006 Badajoz, Spain (e-mail: smoreno@unex.es).

Mercedes E. Paoletti is with the Department of Computer Architecture, Complutense University of Madrid, 28040 Madrid, Spain.

Gabriele Cavallaro is with the Jülich Supercomputing Centre, Forschungszentrum Jülich, 52428 Jülich, Germany.

Juan M. Haut is with the Hyperspectral Computing (Hypercomp) Laboratory, Department of Technology of Computers and Communications, University of Extremadura, 10003 Cáceres, Spain.

Digital Object Identifier 10.1109/LGRS.2022.3173052

I. Introduction

REMOTE sensing (RS) scenes provide rich information about the surface of the Earth. Multiple techniques have been proposed to take advantage of this information. Deep learning (DL) techniques, such as convolutional neural networks (CNNs), have enhanced feature extraction by using large amounts of data and by taking advantage of spatial and structural information. Such information has been used in multiple applications, including land-cover classification for urban planning and disaster monitoring.

Classification methods have been studied in the literature [1] to obtain accurate mapping results. Particularly, land-cover classification through neural networks focuses on the extraction of spectral and pixelwise information [2]. This information is refined by a feature extraction process from a large number of samples with large spatial and spectral dimensions, such as hyperspectral images (HSIs) [3]. At this point, CNNs take advantage of the feature extraction, identifying discriminative features of the input data.

Due to the high computational requirements for running convolution-based models, partially derived from the increase in the number of trainable parameters [4] during recent years, there is a need to speed up this process. At this point, high-performance computing (HPC) platforms are proposed as a solution to accelerate these models. These platforms are composed of resources with great computing capabilities and, hence, with a high processing speed. Hence, the data are distributed between the platform available resources, and the model is trained using multiple replicas of the CNN at the same time. This technique is known as data-parallelism.

In data-parallelism applications [5], each data partition $\mathbf{X}_p$ assigned to a process p (also called replica) is obtained by generating slices of the total dataset $\mathbf{X} \in \mathbb{R}^{B\times H\times W\times D}$, where B represents the batch size (number of samples processed in each training iteration), and H, W, and D are the height, width, and depth (number of bands) of the inputs, respectively. In each batch iteration, replicas perform the training step using the current weights $\mathbf{W}^t$ at time step t. After the forward iteration, a classification error is computed through a loss function $\mathcal{L}$, which is used to reproduce the error of each weight (gradients $\nabla^{\mathbf{t}}$). This calculation aims to later optimize the loss function $\mathcal{L}$, seeking the optimal weights ($\mathbf{W}^{\text{opt}}$ values) that minimize the error, as shown in the following equation:

$$\mathbf{W}_p^{\text{opt}} = \underset{\mathbf{W}_p}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{X}_p\to\mathbf{Y}_p}\left[\mathcal{L}(\mathbf{W}_p, \mathbf{X}_p)\right] \tag{1}$$

where $\mathbb{E}$ is the expectation of the loss function $\mathcal{L}$ for the set of weights $\mathbf{W}_p$, $\mathbf{X}_p$ is the input data, and $\mathbf{Y}_p$ is the set of labels. It is noteworthy that gradients are obtained using the derivative ∂ of the weights through model layers $l \in [1, L]$, as shown in the following equation, for a specific time step t and each sample $x_p \in \mathbf{X}_p$:

$$\nabla w_p^t(l) = \frac{\partial \mathcal{L}(w_p(l), x_p)}{\partial w_p(l)}. \tag{2}$$

The weights are then updated. Several optimizers that achieve accurate results have been proposed in the literature. For instance, AdaGrad [6] focuses on sparse parameters with learning rate variations $\mathbf{W}_p^{t+1} = \mathbf{W}_p^t - ((\gamma \cdot \nabla W_p^t)/(\sqrt{D_p^t} + \epsilon))$, where D is the sum over the parameter matrix diagonal of the gradients squares, γ the learning rate, and ϵ a smoothing term, while Adam [7] proposes an adaptation of the first $M_{(1)}$ and second $M_{(2)}$ order moments $\mathbf{W}_p^{t+1} = W_p^t - ((\gamma \cdot M_{(1)}^t)/(\sqrt{M_{(2)}^t} + \epsilon))$.

These training calculations are complex, and hence, the usage of HPC is required to conduct them in the shortest amount of time. In the literature, most of the efforts aimed at land-cover classification using DL focus on the increase of the model parameters and the input data. As a result, the speedup is set as an objective of these approaches. Accordingly, to take full advantage of HPC platforms, it is necessary to use all the available resources efficiently. Thus, the distribution of the workload between replicas deployed on the heterogeneous computing resources of the platform should be done taking into account the speed of each device [8], [9]. As a consequence, the gradients computed by each replica are obtained from disjoint and nonhomogeneous data slices. In order to obtain the global gradients from all replicas, a communication step is performed. Global gradients are obtained as the average between all replicas, as shown in the following equation, which is clearly unfair since replicas have been trained on different amounts of data:

$$\nabla W_{\text{total}}^k = \frac{\sum_{p=0}^{P} \nabla' W_p^k}{P}. \tag{3}$$

Equation (3) represents the baseline procedure for gradient calculation and entails many limitations. Indeed, assuming homogeneous data splits, the same amount of data is assigned to each replica, leading the slowest replicas to negatively affect the training time. As a result, this strategy introduces an incoherence at the weight update step, where global gradients are used to update replicas' weights. This phenomenon is known as staleness and refers to the delay among the parameters of the replicas due to the time difference at which replicas communicate. Hence, the staleness is directly related to the difference between the performances of the replicas. Therefore, the weight optimization is performed with the available gradients at the optimization step, while these gradients could be delayed from previous iterations. In order to palliate these drawbacks, this work proposes an innovative heterogeneous gradient calculation. Moreover, the gradient baseline calculation is applied for heterogeneous data workloads, as shown in Fig. 1(A.2) for comparative purposes. Conversely, it has the additional problem of unfair weight updating due to the different data amounts between replicas.

In this regard, two more parallelization techniques have been studied in the literature to surmount the previous limitation. In particular, model-parallelism technique splits the model in multiple parts when is too big to fit in the memory of one device. In addition, hybrid parallelism combines both data, and model parallelism schemes take advantage of both approaches. Model and hybrid parallelisms are strategies for more specific situations. In this regard, the data-parallelism scheme usually obtains better results and acceleration.

This work focuses on optimizing the training methodology to provide accurate classification results under a data-parallelism scheme. The contribution of this work is a novel weighted gradient calculation to improve the model accuracy considering the resource speeds found in heterogeneous HPC platforms while using the maximum available resources.

II. Methodology Details

The proposed heterogeneous gradient optimization algorithm can be used for the processing of complex models and datasets in widespread HPC platforms. The method takes advantage of different computational resources, such as different graphics processing unit (GPU) models or central processing units (CPUs) + GPUs schemes. The methodology is composed of three main steps. The former determines the speed of the resources that characterize the computing capabilities of the platform. Indeed, a vector s_p containing the speed of each replica is obtained $S = \{s_0, s_1, \ldots, s_P\}$ from a pretraining step using a CNN training microbenchmark with a very rough estimate of the computation, where $\sum_{p=0}^{P} s_p = 1$. The second step splits the total data into slices that are assigned to replicas. This partitioning is performed considering the previous speed vector s_p and generates a heterogeneous data slice $\mathbf{X_p}$ for each resource. As a result, the batch size of each replica B_p is adjusted to $B_p = (X_p/N)$ to perform the same number of iterations N in all replicas. This adjustment is conducted since the input data size of each replica is proportional to its batch size. Thus, each replica is trained during N iterations using a different number of training samples. After the forward propagation on each replica, a shuffle step is performed, and a new data slice $\mathbf{X_p}$ is assigned to replicas to prevent them from training at the same time with equal data, ensuring that the complete set of data is processed at the end of the iterations of an epoch. Consequently, in the communication phase, the number of messages sent by the replicas is the same. This phase takes place after each batch size iteration, i.e., after the gradient combination step. In the literature, two strategies are used for the communication step. The asynchronous sharing of gradients avoids the use of synchronization points in which replicas wait for slower ones, resulting in an acceleration in the training times. Nonetheless, it has the disadvantage of the staleness problem, and it is quite hard to known that the iteration of the execution each replica is conducting. On the contrary, synchronous communication schemes establish synchronization points at the end of each training iteration. This ensures coherence in the global weight updating, but it slows down the training. In this work, an AllReduce message passing interface (MPI)-based collectives are used under an asynchronous procedure for the gradient sharing step, assuming a minimum delay between replicas given the previously performed heterogeneous workload balancing. In

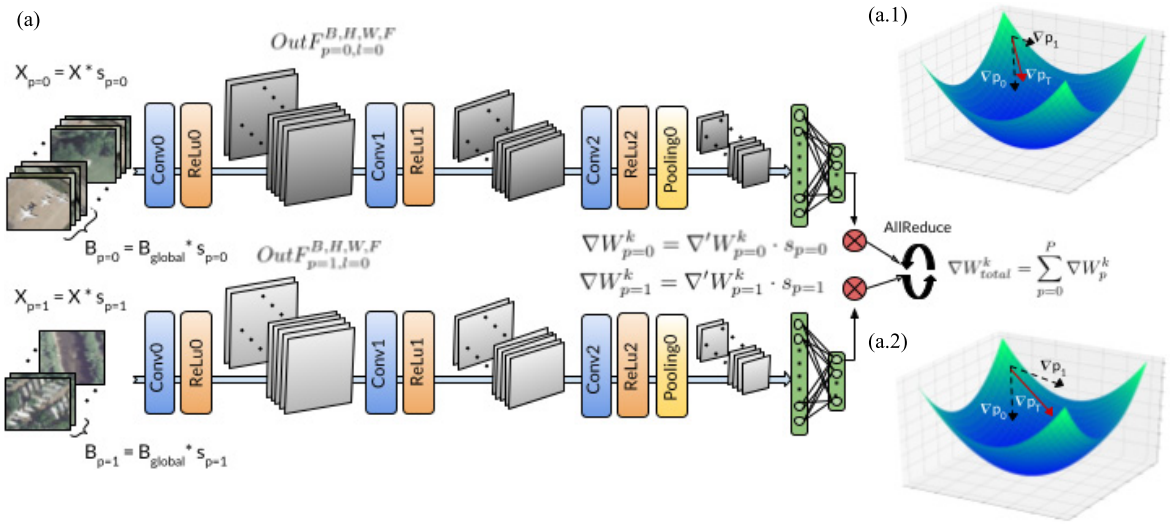

Fig. 1. CNN example using the proposed methodology for the heterogeneous gradient calculation with UCMERCED dataset as input. In the main Fig. A, two replicas are shown with different capabilities and, hence, different amounts of data and batch size. As a consequence of the data-parallelism scheme, the model is replicated for both replicas. $\mathrm{OutF}_{p,l}^{B,H,W,F}$ represents the output features and its size for a specific replica and layer. Gradients are combined in the AllReduce step. (A.1) and (A.2) Gradient optimization step for the proposed and baseline methodologies, respectively. Replica gradients are denoted as ∇p, ∇p_T denotes the total gradient values after the combination, and ∇' represents the current gradient values in a previous step before being weighted.

summary, each process sends and receives gradients from different data chunks to perform an average operation between them. This communication scheme is described in the Horovod framework [10] under a ring procedure.

Finally, the third step implements the following equation to adjust the impact of the replica gradients using the processes speeds; thus, replicas contribute to the global gradients computation depending on the amount of remote sensing images processed:

$$\nabla W_{\text{total}}^{k} = \sum_{p=0}^{P} \nabla' W_{p}^{k} \cdot s_{p}. \tag{4}$$

Equation (4) has some benefits with respect to the homogeneous (3) usually implemented in the literature. The required computation for the heterogeneous calculation does not increase compared with the homogeneous approach. Meanwhile, replicas with a reduced vision of the input RS images and the classes that compose it do not negatively affect those replicas with more data. This avoids gradients to have the same importance in replicas with less data, as in the baseline case. Therefore, low-speed replicas help to find the most suitable weights while speeding up the computation.

Finally, the parameter updating step modifies the replica parameter values, as shown in the following equation:

$$W_{p}^{k+1} = W^{k} + \gamma\, W_{\text{total}}^{k}. \tag{5}$$

The whole procedure of the proposed methodology is depicted in Fig. 1.

III. Experimental Settings

In this section, a detailed description of the used platform, datasets, models, and obtained results is provided. In particular, the settings are chosen to provide global and accurate proofs of the proposed methodology for a wide range of different RS data in a computational heterogeneous system.

A. Heterogeneous Platform Attributes

The experimentation was conducted in the DEEP module from the extreme scale booster (DP-ESB). This module is part of the modular supercomputer architecture (MSA), which is financed by the DEEP-EST European project. A total of four nodes from 75 nodes were selected, where eight processes (replicas) were deployed using the respective CPU and GPU. In particular, "Cascade Lake" Silver 4215 CPUs with eight physical cores and NVIDIA Tesla V100 GPUs with 640 Tensor Cores were used. Nodes from the DEEP module are connected through an Infiniband with a speed of 100 Gb/s.

B. Training Specification

Considered networks are: 1) ResNet18-50 [11]; 2) VGG16 [12]; and 3) DenseNet121 [13]. These were selected to provide a wide range of popular RS classification models and have been trained over three different RS datasets.

1) UCMERCED [14] contains images of size $256 \times 256 \times 3$ with a total (100%) of 2100 images classified in 21 land-cover classes. Images have a resolution of 1 ft per pixel.
2) NWPU-RESISC45 [15] is also composed of scenes of size $256 \times 256 \times 3$. The total amount of images (100%) is 31 500 distributed over 45 classes. The resolution of these varies from 0.2 to 30 m per pixel.
3) The aerial image dataset (AID) [16] holds a total of 10 000 images (100%) of size $600 \times 600 \times 3$, classified over 30 different classes. Each scene has a specific resolution of 0.2 to 30 m per pixel.

C. Results Assessment

Conducted experiments have been performed for 25%, 50%, and 80% of the UCMERCED dataset, 10% and 20% of the

TABLE I

AVERAGE TRAINING RESULTS (%) FOR THE EXPERIMENTAL SETTINGS [NETWORKS, DATASETS, AND DATA SIZE (DS %)]. THE PROPOSED METHOD IS DENOTED AS "PROPOSED," WHILE "BASELINE" IS THE HOMOGENEOUS PROCEDURE FOR THE DATA-PARALLELISM SCHEME OF THE LITERATURE

	UCMERCED			NWPU-RESISC45			AID		
Network	DS	Baseline	Proposed	DS	Baseline	Proposed	DS	Baseline	Proposed
ResNet18	25%	93.05±0.70	**93.53±0.37**	10%	84.82±1.49	**86.17±0.40**	20%	91.92±1.13	**91.95±0.68**
	50%	96.69±0.21	**97.02±0.24**	20%	92.29±1.10	**92.55±1.02**	50%	96.40±0.48	**97.04±0.22**
	80%	98.80±0.50	**98.97±0.65**						
ResNet50	25%	92.84±0.13	**94.03±0.65**	10%	79.91±0.45	**84.58±0.77**	20%	89.93±0.94	**91.38±0.70**
	50%	96.50±0.24	**97.13±0.37**	20%	90.60±1.24	**91.79±0.16**	50%	96.50±0.64	**97.02±0.13**
	80%	98.66±0.69	**98.83±0.40**						
VGG16	25%	90.11±0.55	**92.23±0.06**	10%	78.45±0.63	**83.19±0.90**	20%	86.39±0.64	**89.41±0.56**
	50%	94.96±0.81	**95.91±0.17**	20%	86.04±0.47	**89.35±0.70**	50%	93.42±1.32	**94.27±0.93**
	80%	98.24±0.62	**98.44±0.71**						
DenseNet121	25%	94.48±0.84	**95.39±0.11**	10%	87.23±1.08	**88.35±0.63**	20%	93.71±0.79	**93.81±0.49**
	50%	97.69±0.41	**98.13±0.34**	20%	93.54±0.95	**94.03±0.35**	50%	98.10±0.10	**98.47±0.09**
	80%	98.97±0.51	**99.14±0.34**						

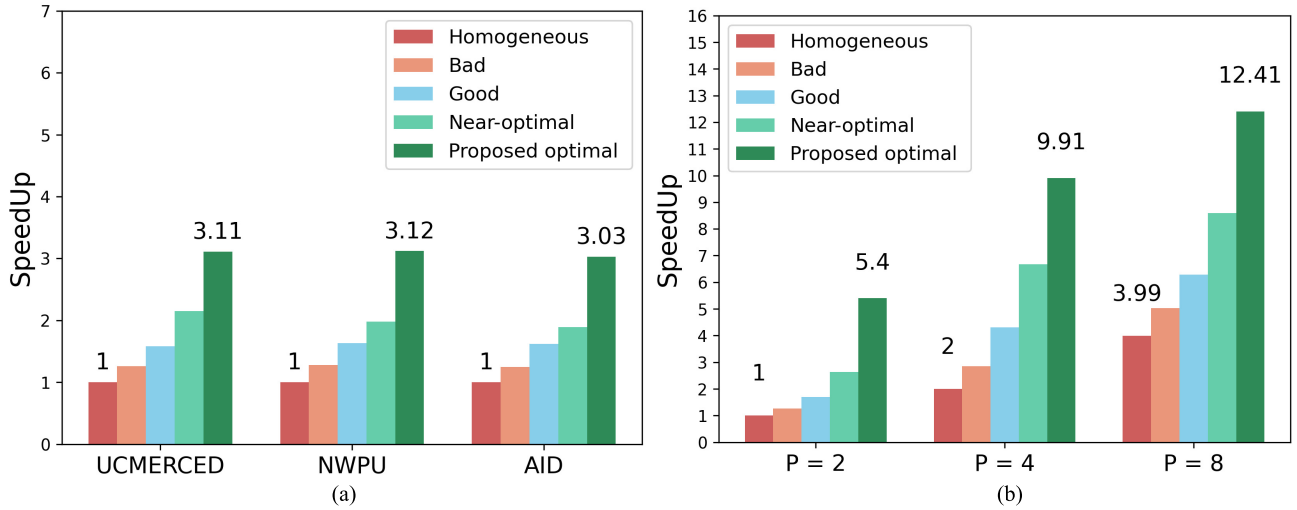

Fig. 2. Obtained speedups for two different studies. Five different schemes were used depending on the precision of the speed measurement to represent each replica. Measurement schemes were categorized from total homogeneous case to the proposed heterogeneous optimal, going through several less precise intermediate measurements (bad, good, and near-optimal). The first diagram (a) shows the speedup obtained for each scheme in the UCMERCED, NWPU, and AID datasets, while diagram (b) represents the scaling caused by the increase of the number of processes P involved in the training.

NWPU-RESISC45 dataset, and 20% and 50% of the AID dataset. This data split was done to evaluate the effectiveness of the proposed method under different scenarios. Therefore, the number of parameters varies with the network model, while the data size varies with the percentage of images and its dimensions. Results are shown in Table I for five Monte Carlo runs of each experiment.

It is remarkable that the overall results demonstrate that our proposal produces significant improvements with respect to the baseline methodology. The adjustment of the gradients according to the data amounts assigned to the replicas rectifies, more fairly and accurately, its direction to avoid local minima.

Focusing on the UCMERCED dataset, the ResNet18 model obtains accuracy gains of +0.48, +0.33, and +0.17 for the different amounts of data considered, respectively. This and the VGG16 model, which obtain +2.12, +0.95, and +0.20, respectively, are the networks that obtain the poorest accuracy, as feature extraction is not deep. On the other hand, ResNet50 performs better than ResNet18 and VGG16 models with an average accuracy of 92.23% and 95.91% for the 25% and 50% data sizes, respectively; while using 80% of the data, the three models have a similar behavior. Nonetheless, DenseNet121 achieves the highest accuracy considering all data sizes with a maximum accuracy of 99.14% using the 80% of the data. The baseline model also obtains the maximum score of all the models with a value of 98.97%. It is noteworthy that the accuracy gets higher and closer between both strategies as the amounts of data increase. There are multiple points that benefit this fact, such as the depth of the models. Nevertheless, in these cases where accuracy is high (around 99%), and a small improvement in accuracy is hard to obtain and is equally or even more important than the improvements observed at lower values.

Furthermore, NWPU-RESISC45 was also considered. The obtained results are the following. First, for the ResNet18 model, the obtained gains are +1.34 and +0.26 for 10% and 20% of the data, respectively. Note that ResNet18 obtains better classification results than ResNet50 with an average accuracy of 92.55% compared to 91.79% using the 20% training data. On the other hand, VGG16 provides inferior results

in comparison to other models. Despite this, the improvements in VGG16 are also observable with gains of +4.74 and +3.31. As with the UCMERCED dataset, the best results are obtained with the DenseNet121 model, which produces an average accuracy of 88.35% and 94.03% for both data sizes, with gains of +1.12 and +0.49, respectively. Moreover, it can be seen how the standard deviation is reduced in the proposed method with the increment of data size, achieving less randomness in the training parameters and more accurate feature extraction.

Finally, the AID dataset is used in the last experimental test, considering 20% and 50% of the data. The ResNet18 produces gains of +0.03 and +0.64 for the respective sizes. As previously, the highest accuracy is reached in the DenseNet121 for both sizes with a final accuracy of 93.81 and 98.47, respectively. Finally, the results observed in the UCMERCED and NWPU datasets, i.e., a minor standard deviation on the proposed model and a decrease in the accuracy gap between the two methods as the data increase, are also observed. The experimentation provides the same conclusions for all datasets and sizes, ensuring a consistent operation of the methodology.

In addition, we conduct a speedup experimentation. In order to not harming the algorithm functioning, the maximum number of processes is set to eight since more processes would led to small data splits for the studied methodologies, which is proven to be inefficient. On the other hand, reported frameworks from the literature [10] implement data-parallelism techniques under a homogeneous workload partitioning. This fact increases the training time in heterogeneous platforms. In this work, a heterogeneous workload partitioning is used, achieving a notable training time reduction. As shown in Fig. 2(a), an evaluation of different speed-based schemes is performed. The speedup gaining increases with proximity to the optimal heterogeneous workload balancing. Thus, speedup value is directly influenced by the computational difference between the resources. This is observable in Fig. 2(a) with a maximum speedup of three for all cases. Finally, Fig. 2(b) considers the platform scaling factor by adding a number of processes to the training execution.

IV. Conclusion

Land-cover images obtained from remote sensing observation technologies provide rich information about the Earth's surface. DL applications face two main issues to process such information, i.e., its computation requirements and algorithm accuracy. In this regard, methods reported in the literature implement the data-parallelism scheme to accelerate computation. Nevertheless, on heterogeneous platforms, this scheme negatively affects the model accuracy since all replicas contribute equally to the global gradient calculation. In addition, it produces the staleness problem, which increases the classification error produced by replica differences. In this letter, a novel methodology to address these problems is proposed. Unlike other reported methods, the proposed approach aims to perform a balanced computation on devices with different computational capabilities and, hence, exploits the most of the resources in the HPC platform. Furthermore, as the main contribution of this work, the accuracy of the model is notably improved by using a weighted gradient calculations, considering the heterogeneity of the resources. Indeed, replica gradients contribute more fairly to the global gradient in the communication, which avoids local minima. The efficiency of DL algorithms for the classification task overwhelmingly increases while producing a notable speedup. Our experimental discussion evaluates this by using different amount of computation and data of completely opposite characteristics, showing the advantages of our methodology for remote sensing classification applications.

References

[1] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS J. Photogramm. Remote Sens.*, vol. 158, pp. 279–317, Dec. 2019.

[2] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 778–782, May 2017.

[3] F. Luo, Z. Zou, J. Liu, and Z. Lin, "Dimensionality reduction and classification of hyperspectral image via multistructure unified discriminative embedding," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–16, 2022.

[4] L. Mou and X. X. Zhu, "RiFCN: Recurrent network in fully convolutional network for semantic segmentation of high resolution remote sensing images," 2018, *arXiv:1805.02091*.

[5] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proc. 28th Int. Conf. Int. Conf. Mach. Learn.*, 2011, pp. 265–272.

[6] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015, *arXiv:1412.6980*.

[8] S. Moreno-Álvarez, J. M. Haut, M. E. Paoletti, J. A. Rico-Gallego, J. C. Díaz-Martín, and J. Plaza, "Training deep neural networks: A static load balancing approach," *J. Supercomput.*, vol. 76, no. 12, pp. 9739–9754, Dec. 2020.

[9] C. Chen, Q. Weng, W. Wang, B. Li, and B. Li, "Fast distributed deep learning via worker-adaptive batch sizing," in *Proc. ACM Symp. Cloud Comput.*, New York, NY, USA, Oct. 2018, p. 521.

[10] A. Sergeev and M. D. Balso, "Horovod: Fast and easy distributed deep learning in tensorflow," 2018, *arXiv:1802.05799*.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, May 2015, pp. 1–14.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[14] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS)*, 2010, pp. 270–279.

[15] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proc. IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017, doi: 10.1109/JPROC.2017.2675998.

[16] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? The inria aerial image labeling benchmark," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2017, pp. 3226–3229.