GPU-Friendly Neural Networks for Remote Sensing Scene Classification

Juan M. Haut, *Senior Member, IEEE,* Adrian Alcolea, Mercedes E. Paoletti, *Member, IEEE,* Javier Plaza, *Senior Member, IEEE,* Javier Resano and Antonio Plaza, *Fellow, IEEE*

***Abstract*—onvolutional neural networks (CNNs) have proven to be very efficient for the analysis of remote sensing (RS) images.onvolutional neural networks (CNNs) have proven to be very efficient for the analysis of remote sensing (RS) images.C Due to the inherent complexity of extracting features from these images, along with the increasing amount of data to be processed (and the diversity of applications), there is a clear tendency to develop and employ increasingly deep and complex CNNs. In this regard, graphics processing units (GPUs) are frequently used to optimize their execution, both for the training and inference stages, optimizing the performance of neural models through their many-core architecture. Hence, the efficient use of the GPU resources should be at the core of optimizations. This letter analyzes the possibilities of using a new family of CNNs, denoted as TResNets, to provide an efficient solution to the RS scene classification problem. Moreover, the considered models have been combined with mixed precision to enhance their training performance. Our experimental results, conducted over three publicly available RS datasets, show that the proposed networks achieve better accuracy and more efficient use of GPU resources than other state-of-the-art networks. Source code: https://github.com/mhaut/GPUfriendlyRS**

***Index Terms*—Remote sensing (RS), classification, convolutional neural networks (CNNs), graphics processing units (GPUs).**

I. Introduction

EARTH observation has experienced a fast development in the last years due to the great improvement of remote sensing systems and computing technologies. Remote sensing (RS) images offer a major opportunity to increase our knowledge of the surface of the Earth, allowing for the collection of useful terrestrial information through a variety of observation systems from both aircraft and space platforms. The large and rich amount of RS data (collected in different formats, with different spatial, spectral and temporal resolutions) is highly attractive for the development of a wide range of applications in a diverse set of disciplines [1]. As a result, both the quantity and quality of RS images have been substantially increased, opening the door for automatic and deep processing methods.

Since the first studies about the applications of RS images [2], a great amount of work has been conducted to take advantage of machine (ML) and deep learning (DL) techniques for the analysis of RS data. Cheng *et al.* [3] and Paoletti *et al.* [4] provide a comprehensive review of the most recent achievements. For instance, traditional ML-based techniques such as k-means and gradient boosting have been used to perform RS scene classification [5], [6], while deep models and CNNs have demonstrated to be highly accurate techniques due to their generalization power and their ability to extract discriminative features automatically.

For instance, Hu *et al.* [7] studied how to extract relevant features, using successfully pre-trained CNNs, from high-resolution RS images. In a different approach, Zou *et al.* [8] proposed the use of unsupervised DL-based techniques to perform feature selection, reformulating the problem as a feature reconstruction one. The authors in [9] introduce a novel method based on two stages: first, they use fine-tuned techniques (for a pre-trained network) to perform land-use classification, then they implement an automatic object detection algorithm that divides the high-resolution image into overlapping small patches, and classifies them. Cheng *et al.* [10] developed a novel representation called bag of convolutional features (BoCFs), in which the words are directly generated from the features extracted by a CNN. After training the CNN, they used the k-means to cluster the features, generating descriptors that can be quantized into a bag of visual words. To deal with the limited availability of labeled training data, Scott *et al.* [11] used both fine-tuning (from previously trained models) and data augmentation to train deep CNNs for classification purposes. Li *et al.* [12] use the information hidden in different layers of a pre-trained CNN model to improve feature discrimination techniques. They build mid-level features from the hidden layers of the CNN. Then, they perform dimensionality reduction. Another feature fusion strategy is proposed in [13]. They combine a very deep CNN (VGG-Net) to perform feature extraction with a discriminant correlation analysis (DCA) to fuse the extracted features, and then use a support vector machine (SVM) as the final classifier. Cui *et al.* [14] proposed a hybrid model, combining a convolutional auto-encoder (CAE) with a CNN, to perform object-oriented scene classification allowing to simplify the the CNN and to reduce its parameters. Cheng *et al.* [15] use a metric learning regularization term on the features of the CNN. The idea is to make the model more discriminative so, in the new feature spaces, the images from the same class are mapped closely to each other while the images of different classes are mapped as far as possible. The CNN-CapsNet in [16] combines both CNNs and capsule

Supported by Junta de Extremadura (GR18060), PID2019-105660RB-C21 and TIN2016-76635-C2-1-R (AEI/ERDF, UE) and gaZ: T58_17R. *(Corresponding author: Juan M. Haut)*

M. E. Paoletti, J. M. Haut, J. Plaza and A. Plaza are with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, 10003 Cáceres, Spain.

A. Alcolea and J. Resano are with the Computer Architecture Group of the University of Zaragoza, C/María de Luna, 1, 50018 Zaragoza, Spain.

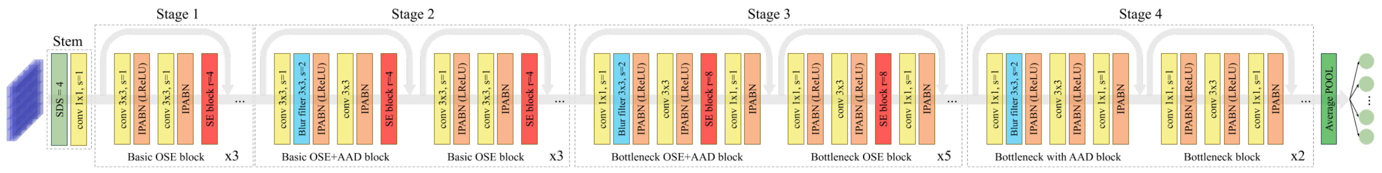

Fig. 1. Graphical representation of proposed architecture for RS image classification. The S2D block is developed to reduce the input spatial resolution without losing spatial information. Then, four feature extraction stages are implemented. Stages 1 and 2 employ the basic block as the main building block, including an optimized SE layer. Stages 3 and 4 employ the bottleneck block, introducing SE layers within stage 3. Downscaling layers have been replaced by AAD units.

networks to avoid the use of fully connected layers as a classifier, keeping the spatial information in the last layers. Siamese CNNs have also been successfully adapted to the RS image classification problem in [17]. These networks are composed of two identical CNNs sharing parameters that are simultaneously fed with a pair of images. In addition, several final layers are trained to recognize whether this pair of images belong to the same category or not. In summary, CNNs have been successfully applied to extract feature representations from RS data. However, their training and inference stages involve thousands of millions of floating point operations for each image to be processed. This can easily exceed the computational capacity of most conventional processors.

With the aim of overcoming this limitation, graphics processing units (GPUs) have demonstrated to be a suitable architecture to deal with these computations. As a result, there is a growing trend within the RS community to adapt traditional algorithms and develop new processing methods using GPU devices, in order to exploit their high parallelization power through their many-core architecture [18]. For this reason it is necessary to search for new solutions focused on a better exploitation of GPU resources.

Recently, the authors in [19] proposed TResNet, a new family of GPU-dedicated neural networks specifically designed to take advantage of the GPU architecture. Their main idea relies on the modification of successful deep models, such as the ResNet50, by implementing several refinements to achieve a better accuracy-throughput trade-off within the GPU. Thus, an optimized stem unit has been developed on the bottom of the model in order to avoid the loss of spatial information while reducing the input's resolution. Also, basic and bottleneck residual blocks are combined along the architecture to enhance the speed-accuracy trade-off. Within each block, several mechanism are introduced to improve the feature extraction performance. For instance, the computationally-expensive stride downsampling is replaced by anti-alias based components, while the memory consumer BatchNorm is replaced by an Inplace-ABN module to double the batch size. Finally, the dedicated squeeze-and-excitation layers are introduced within residual blocks to improve the extraction of relevant features.

In this letter, we analyze three TResNet models (denoted as M, L and XL, according to their size) and adjust them to perform efficient classification of RS scenes. Our experiments have been conducted over three well-known datasets, comparing the considered methods with other state-of-the-art networks. An i9-9940X processor and a NVIDIA Titan RTX GPU have been considered for experiments, while every network has been trained with and without mixed precision. The obtained results demonstrate that TResNets reach the best results in terms of accuracy and use of GPU resources. It should therefore become an essential reference for RS scene classification.

II. Methodology

We introduce several very deep networks [19] for RS scene classification whose architectures/operations have been optimized through a series of modifications to make them more GPU-friendly. They have been developed taking into account the original ResNet50 model, to which some improvements have been conducted to optimize not only their inference step, but also their training procedure. In this sense, the developed models include five refinements: i) space-to-depth (S2D) stem, ii) anti-alias downsampling (AAD), iii) in-place activated batchnorm (IPABN), iv) new block-type selection (NBS) and v) optimized squeeze-and-excitation (OSE) layers. Each one is detailed below, while Fig. 1 presents the general architecture of the developed networks.

Traditional deep models implement a stem unit composed by initial convolution-activation-pooling layers as bottom block to reduce the input volume resolution. Although this strategy can increase the GPU throughput by reducing the feature size, a significant amount of information is lost due to fast downscaling. To overcome this limitation (searching for a good speed-accuracy trade-off), a dedicated S2D unit is employed to rearrange the spatial information along the spectral dimension [20]. In this context, instead of summarizing the spatial information through $k \times k$ kernels with stride s, this is distributed as the new volume's depth. Then, a point-wise convolution layer elongates or shortens the spectral dimension to the desired depth. This prevents the loss of spatial information, involving a lighter 1×1 kernel computation.

Furthermore, AAD [21] is implemented to replace the downscaling conv3×3 layers with stride $s = 2$ by an efficient downscaling block. This combines a conv$3{\times}3$ layer with $s = 1$ and a $3{\times}3$ blur filter with $s = 2$, increasing the model accuracy and consistency with a significantly lower computational cost in comparison with standard 3×3 kernels with $s = 2$.

Also, the BN followed by activation functions have been replaced by IPABN layers [22], in order to implement them as a single in-place operation. In this sense, IPABN avoids the data copy, so its memory consumption is considerably lower than the original BN. Moreover, IPABN is implemented with leaky ReLU (LReLU) as activation function, which avoids the dying ReLU problem in deep models and provides better accuracy than the standard ReLU.

The original ResNet50 implements bottleneck residual blocks, where each one combines two point-wise layers with one 3×3 layer. In this sense, bottleneck blocks exhibit higher accuracy in comparison with basic blocks (composed by two 3×3 layers), but they also have higher power and compute consumption. In order to reduce the computational burden without losing accuracy, the proposed models for RS data classification combine both blocks, placing basic blocks at the beginning of the network, while the bottleneck has been placed at the end of the model to balance the computational load across the architecture. Moreover, residual connections have been implemented as in-place operations to reduce the number of memory accesses.

Finally, in the first three stages, a squeeze-and-excitation (SE) block [23] is included within each bottleneck and basic residual block, considering a reduction factor of 8 and 4, respectively (OSE) in order to lighten the computational burden of the original SE.

During the training and inference steps of the proposed models, the NVIDIA Apex library [24] has been used to optimize the operations of the models when performing RS scene classification.

III. Experiments

We trained the three TResNet models in [19] and compared their results with vgg16, inception_v3, ResNet50 and ResNet101. Training and inference have been executed in an i9-9940X processor with a NVIDIA Titan RTX GPU, using 24GB of memory and 4608 cores. We also used the NVIDIA Apex library [24] for mixed precision, to increase the performance of the models, comparing the results of the baseline and the optimized executions.

A. Datasets

We selected three publicly available datasets to evaluate the considered models. Their main characteristics are summarized below:

- **UC Merced (UCMERCED)** [25]: This collection of RGB Earth surface images is composed of 2100 aerial shots classified into 21 different categories. Each one of the RGB orthoimages has a 256×256-pixel size, and its spatial resolution is one foot per pixel.
- **Aerial Image Dataset (AID)** [26]: The 10000 RGB images of this dataset are distributed into 30 classes, and its size is 600×600 pixels, whose resolution varies from $8m$ to $0.5m$.
- **NWPU-RESISC45 (NWPU)** [27]: This is a collection of 31500 images that are uniformly distributed within 45 categories. They are RGB images of size 256×256-pixels, and their spatial resolution varies from $30m$ to $0.2m$.

B. Experimental results

For the experiments we used two different modes of the NVIDIA Apex library (for pure and mixed precision). The *FP32* refers to the baseline without including any 16 bits floating point (FP16) values, while *mixed* refers to the use of NVIDIA Apex mixed precision modes to combine FP16 together with 32 bit values, which usually implies a reduction of the accuracy as a trade-off for computational efficiency.

Each model has been trained 10 MonteCarlo iterations of 120 epochs using Adam optimizer with learning rate of $1\mathrm{e}{-3}$ and batching size of 100. For data augmentation we proceed as follows: resize to 256×256, aleatory select a 224×224 section, horizontal and vertical rotations both with 50% of probability, randomly change the brightness, contrast and saturation with 0.4 factor, normalize and then apply the technique on [28] with 50% of probability. For testing we resize to 256×256 and select the 224×224 central section. Tables I to III show the average accuracy values with the standard deviation in brackets.

Depending on the characteristics of each dataset, we selected a different amount of images as training data. For each dataset we explored different percentages, and selected some representative values. For small datasets (such as UCMERCED) we selected three different scenarios with 20%, 50% and 80% of the data as the training set. As it can be seen in Table I, TResNet clearly provides the best results when few images are available for the training set. This is a remarkable result, since the lack of labeled images is a major concern in the RS field. For the other two scenarios, the results are very similar for all the analyzed CNNs. The results for the AID dataset are shown in Table II. In this case, since the dataset is larger, we selected 20% and 50% of the data for training. For this dataset, TResNet achieved the best results for all the tested configurations. Finally, for the NWPU dataset (which includes more images than the previously considered ones), we selected two scenarios with 10% and 20%. In Table III, we can see that, as in the previous case, the TResNet CNNs achieve the best results.

TABLE I
UCMERCED results with 20%, 50% and 80% of training data

	Tr=20%		Tr=50%		Tr=80%	
Method	FP32	Mixed	FP32	Mixed	FP32	Mixed
ResNet50	87.47(2.0)	90.62(2.1)	95.60(1.2)	96.93(1.5)	96.46(1.5)	96.76(1.4)
ResNet101	87.83(2.4)	91.73(2.4)	95.71(1.8)	97.17(1.2)	96.70(1.4)	97.50(1.3)
TResNet_M	**91.31**(1.7)	90.09(1.6)	**96.55**(1.5)	96.82(1.2)	**97.62**(1.0)	97.83(1.1)
TResNet_L	89.82(1.7)	91.67(1.6)	96.54(1.4)	97.11(1.2)	96.85(1.1)	97.89(1.0)
TResNet_XL	89.20(2.8)	**93.30**(1.8)	95.77(1.6)	**97.50**(0.9)	97.05(1.1)	**98.12**(1.0)
VGG16	85.95(2.4)	64.40(2.5)	94.37(1.7)	82.92(2.1)	95.15(1.7)	87.44(1.7)
Inception_V3	89.11(2.2)	80.83(2.2)	95.77(1.7)	96.10(1.5)	96.73(0.9)	96.82(1.3)

TABLE II
AID results with 20% and 50% of training data

	Tr=20%		Tr=50%	
Method	FP32	Mixed	FP32	Mixed
ResNet50	90.58(1.5)	90.99(1.1)	94.45(0.6)	94.14(1.0)
ResNet101	90.74(1.0)	91.91(0.9)	94.11(0.9)	94.75(0.8)
TResNet_M	**92.59**(0.8)	91.01(1.1)	**95.10**(0.7)	94.96(0.6)
TResNet_L	92.06(1.0)	92.45(0.9)	95.01(0.9)	95.32(0.4)
TResNet_XL	92.31(0.6)	**92.91**(1.0)	95.07(0.6)	**95.60**(0.7)
VGG16	89.28(0.8)	75.50(0.7)	92.95(0.6)	83.41(1.3)
Inception_V3	92.05(1.4)	89.70(0.8)	94.29(0.7)	93.86(0.8)

TABLE III
NWPU RESULTS WITH 10% AND 20% OF TRAINING DATA

	Tr=10%		Tr=20%	
Method	FP32	Mixed	FP32	Mixed
ResNet50	86.40(0.6)	84.06(0.7)	90.13(0.4)	88.17(0.6)
ResNet101	86.47(0.4)	85.91(0.6)	89.91(0.6)	89.73(0.6)
TResNet_M	**88.05**(0.8)	85.00(0.8)	**91.19**(0.4)	89.29(0.5)
TResNet_L	87.68(0.7)	87.38(0.6)	90.94(0.8)	91.10(0.5)
TResNet_XL	87.74(0.7)	**87.74**(0.5)	91.12(0.5)	**91.77**(0.5)
VGG16	82.79(0.7)	56.76(1.0)	87.26(0.7)	62.73(0.8)
Inception_V3	86.90(0.6)	80.75(0.7)	90.57(0.5)	86.10(0.9)

An interesting special case to analyze is the comparison between TResNet_M and ResNet50 models. TResNet_M has been specifically designed to provide a similar GPU throughput with regards to ResNet50. The objective is to carry out a fair comparison between them [19]. As it can be seen in the corresponding tables, TResNet_M achieves better accuracy than ResNet50 in 13 of the 14 analyzed scenarios.

A remarkable result is that, with the exception of VGG-16, mixed precision options allow to reduce the size and the computations of the models without reducing the accuracy. As it can be seen in Figure 5, it accelerates the inference by a factor of 2. In fact, it often improves accuracy. The reason is that large models can easily fall into overfitting, and, if that is the case, mixed precision can reduce it and improve the results.

Accuracy is a critical metric, but it is not the only one that is important. We have also analyzed other relevant information, such as the size of the models, and the computations needed. Figs. 2 to 4 depict the scenario for FP32 precision (*FP32*) and the highest number of images used as training set for each dataset. The x-axis represents the number of operations, and the y-axis represents the accuracy, while the size of each circle is proportional to the number of parameters of the model.

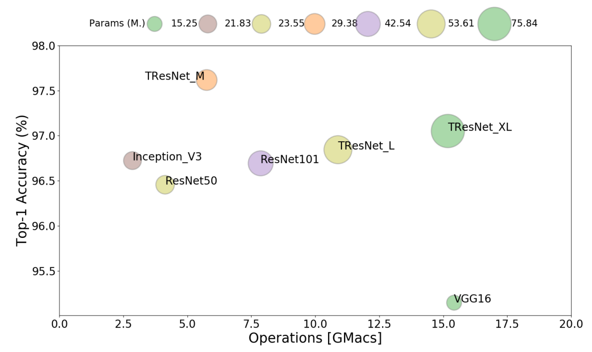

Fig. 2. Network characteristics for UCMERCED dataset. The x-axis shows the number of GMAC operations per image during inference. The y-axis shows the achieved accuracy (Tr=80% and FP32). The size of each circle represents the number of parameters, which is also specified at the top of the graph.

For UCMERCED and AID TResNet_M provides an interesting trade-off, since it achieves the best accuracy with smaller size and less MAC operations than TResNet_L, TResNet_XL and ResNet101, and with less than half of the operations that VGG16 requires. For NWPU, TResNet_XL achieves the best accuracy results, but it is also the largest model (and also the one that needs to calculate more MAC operations).

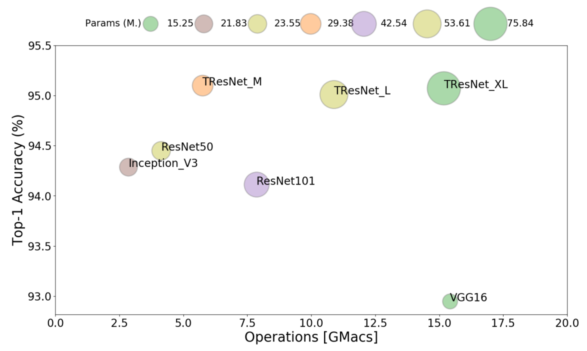

Fig. 3. Network characteristics for AID dataset. The x-axis shows the number of GMAC operations per image during inference. The y-axis the achieved accuracy (Tr=50% and FP32). The size of each circle represents the number of parameters, which is also specified on top of the graph.

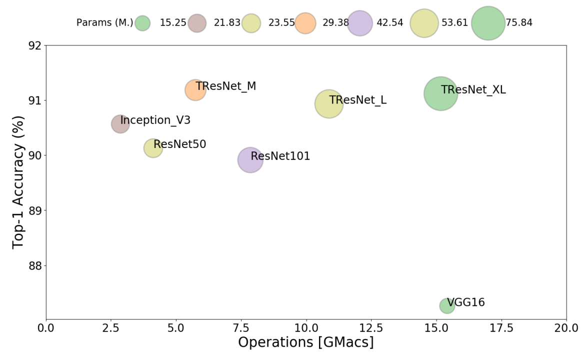

Fig. 4. Network characteristics for NWPU dataset. The x-axis shows the number of GMAC operations per image during inference. The y-axis the achieved accuracy (Tr=20% and FP32). The size of each circle represents the number of parameters, which is also specified on top of the graph.

Another important metric is the throughput achieved by every model during inference. Fig. 5 shows the throughput in terms of images per second for both scenarios, *FP32* and *mixed*. Again, TResNet_M provides a very interesting trade-off between accuracy and throughput. It is remarkable that it achieves better throughput than smaller models that carry out less computations, such as inception_V3 or ResNet50. This is not an intuitive result, but the truth is that GPU performance is often not bounded by the number of operations, but by communications and memory bandwidth. Because this network has been designed with these parameters in mind, it achieves throughput results which are far better than what could be expected according to its size and the number of required computations. Regarding TResNet_L and TResNet_XL, they clearly provide worse throughputs than TResNet_M, although for some datasets they can provide slightly better results. Therefore, they only need to be taken into account if accuracy is the only objective, and even in that case it is likely that TResNet_M may achieve the best results. If throughput is indeed important, we recommend TResNet_M since it provides very good results in all the considered scenarios and it is three times faster than TResNet_XL.

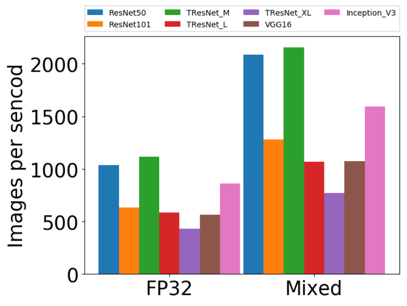

Fig. 5. Number of images per second analyzed on inference by each model (for FP32 and mixed precision configurations).

IV. Conclusions

In this letter, we have analyzed the performance/accuracy trade-offs of the recent family of TResNet networks on RS scene classification. The main contribution of TResnets is that they have been specifically designed for GPUs (i.e., with the GPU architecture in mind). We have tested TResnets in three widely used RS datasets. Thanks to their GPU-friendly design, these neural networks are capable of fully taking advantage of the GPU resources, achieving better performance than smaller models that require considerably fewer operations. Moreover, they also provide better accuracy results than other well-known, state-of-the-art networks, in most scenarios. We have also identified that, in most scenarios, TResNet_M, (the smaller version of the TResNet family) provides the best results, both in terms of throughput and accuracy. Hence, it appears to be the most suitable one for RS data processing. This was not the case in [19], where TResNet_XL provided better accuracies in other application domains. The main reason is that the considered datasets are smaller than those used in [19]. Another interesting result is that for these data sets mixed precision can be used to reduce the size and the computations of the models without reducing their accuracy. In fact, it can even improve it. Hence, it is important to explore this option. After analyzing the obtained results, we believe that TResNets (specially TResNet_M) may become an essential reference for RS classification purposes. Further work will be focused on analyzing other (possibly larger) datasets and different kinds of RS data.

References

[1] S. Liang, *Comprehensive Remote Sensing*. Elsevier, 2017.
[2] A. F. Goetz, B. N. Rock, and L. C. Rowan, "Remote sensing for exploration; an overview," *Economic Geology*, vol. 78, no. 4, pp. 573–590, 1983.
[3] G. Cheng, X. Xie, J. Han, L. Guo, and G. Xia, "Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 3735–3756, 2020.
[4] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "Deep learning classifiers for hyperspectral imaging: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 158, pp. 279–317, 2019.
[5] J. M. Haut, M. Paoletti, J. Plaza, and A. Plaza, "Cloud implementation of the k-means algorithm for hyperspectral image analysis," *The Journal of Supercomputing*, vol. 73, no. 1, pp. 514–529, 2017.
[6] F. Zhang, B. Du, and L. Zhang, "Scene classification via a gradient boosting random convolutional network framework," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–10, 10 2015.
[7] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, p. 14680–14707, Nov 2015. [Online]. Available: http://dx.doi.org/10.3390/rs71114680
[8] Q. Zou, L. Ni, T. Zhang, and Q. Wang, "Deep learning based feature selection for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, pp. 2321–2325, 2015.
[9] I. Ševo and A. Avramović, "Convolutional neural network based automatic object detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 5, pp. 740–744, 2016.
[10] G. Cheng, Z. Li, X. Yao, L. Guo, and Z. Wei, "Remote sensing image scene classification using bag of convolutional features," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1735–1739, 2017.
[11] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis, "Training deep convolutional neural networks for land–cover classification of high-resolution imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 4, pp. 549–553, 2017.
[12] E. Li, J. Xia, P. Du, C. Lin, and A. Samat, "Integrating multilayer features of convolutional neural networks for remote sensing scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 10, pp. 5653–5665, 2017.
[13] S. Chaib, H. Liu, Y. Gu, and H. Yao, "Deep feature fusion for vhr remote sensing scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4775–4784, 2017.
[14] W. Cui and Q. Zhou, "Application of a hybrid model based on a convolutional auto-encoder and convolutional neural network in object-oriented remote sensing classification," *Algorithms*, vol. 11, no. 1, p. 9, Jan 2018. [Online]. Available: http://dx.doi.org/10.3390/a11010009
[15] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 5, pp. 2811–2821, 2018.
[16] W. Zhang, P. Tang, and L. Zhao, "Remote sensing image scene classification using cnn-capsnet," *Remote Sensing*, vol. 11, no. 5, p. 494, Feb 2019. [Online]. Available: http://dx.doi.org/10.3390/rs11050494
[17] X. Liu, Y. Zhou, J. Zhao, R. Yao, B. Liu, and Y. Zheng, "Siamese convolutional neural networks for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 8, pp. 1200–1204, 2019.
[18] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Scalable recurrent neural network for hyperspectral image classification," *The Journal of Supercomputing*, pp. 1–17, 2020.
[19] T. Ridnik, H. Lawen, A. Noy, and I. Friedman, "Tresnet: High performance gpu-dedicated architecture," *arXiv preprint arXiv:2003.13630*, 2020.
[20] M. Sandler, J. Baccash, A. Zhmoginov, and A. Howard, "Non-discriminative data or weak model? on the relative importance of data and model resolution," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 1036–1044.
[21] J. Lee, T. Won, and K. Hong, "Compounding the performance improvements of assembled techniques in a convolutional neural network," 2020.
[22] S. Rota Bulò, L. Porzi, and P. Kontschieder, "In-place activated batchnorm for memory-optimized training of dnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
[23] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.
[24] N. Corporation, "A pytorch extension: Tools for easy mixed precision and distributed training in pytorch," https://https://github.com/NVIDIA/apex, 2019, [Online; accessed June 03, 2020].
[25] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 270–279.
[26] G.-S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu, "Aid: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3965–3981, 2017.
[27] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017.
[28] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Neighboring region dropout for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 6, pp. 1032–1036, 2020.