FPGA-based implementation of a CNN architecture for the on-board processing of very high resolution remote sensing images

Román Neris, Adrián Rodríguez, Raúl Guerra, Sebastián López, Roberto Sarmiento
University of Las Palmas de Gran Canaria
Institute for Applied Microelectronic (IUMA), Gran Canaria, Spain
Email: rneris@iuma.ulpgc.es, armolina@iuma.ulpgc.es, rguerra@iuma.ulpgc.es, seblopez@iuma.ulpgc.es, roberto@iuma.ulpgc.es

***Abstract*—Over the last years, Convolutional Neural Networks (CNNs) have been widely used in remote sensing applications, such as marine surveillance, traffic management or road networks detection. However, since CNNs have extremely high computational, bandwith and memory requirements, the hardware implementation of a CNN on space-grade devices like FPGAs for the on-board processing of the acquired images has brought many challenges, since the computational capabilities of the on-board hardware devices are limited. Hence, implementations have to be carefully planned. In this paper, the authors present their work towards the implementation of an efficient CNN onto a space-grade FPGA in order to achieve the on-board processing of very-high resolution remotely sensed images as soon as the data are provided by the sensor. All this work has been conducted within the EU-funded VIDEO project. As it will be presented in this paper, the work includes the introduction of a methodology based on the project constraints, the evaluation of different state-of-the-art CNN architectures by means of a new efficiency measurement also proposed in this work, the introduction of a new efficient CNN architecture, and finally, its optimized hardware implementation by means of high-level synthesis tools. The results obtained following the proposed methodology demonstrate that the uncovered architecture is able to detect targets of interest in RGB images with a much higher efficiency than state-of-the-art solutions, while requiring a much smaller amount of computing and memory resources.**

***Index Terms*—Machine learning, deep learning, convolutional neural networks, target detection, remote sensing, FPGA.**

I. Introduction

NOWADAYS, there is a rapid surge of interest in deep learning for remote sensing. These algorithms have rapidly become a hot-topic in the machine learning field, and are currently part of the remote sensing big data analysis paradigm [1]. In particular, there has been a lot of progress made in the deep learning models, particularly Convolutional Neural Networks (CNNs), which have significantly improved the performance of remote sensing image processing tasks [2]. Nevertheless, the large number of model parameters and the high computational cost of CNN algorithms make their implementation on resource-limited devices like FPGAs a difficult task. Issues like resource utilization, minimal precision arithmetic using fixed-point representations, hardware-friendly descriptions or real-time processing requirements are to be faced. Obviously, these challenges are magnified when these implementations are considered for space-grade FPGAs, since space presents an additional set of constraints such as power consumption, restricted computational capabilities and limited storage availability [3].

Hence, in order to select the right architecture, a wide evaluation of existing CNN models has to be carried out. Parameters such as the number of trainable parameters (to be stored in memory), the image size and resolution, or the number of Floating Point Operations (FLOPs) are especially relevant for the high level implementation of such architectures, as these parameters will guide the designer in the decision of the final architecture to be implemented.

The work presented in this manuscript has been conducted within the Video Imaging Demonstrator for Earth Observation (VIDEO) project [4], in which the aforementioned restrictions are of the utmost relevance. As it is shown in Section II, this project presents specific constraints that have to be taken into account, such as the large size of the images captured by the sensor, which impacts the amount of memory needed, or the encoding method used by the sensor, which will determinate the availability of the data. Within this context, the main contributions of this work are summarized as follows. First, the scanning algorithm used to process the entire image as independent blocks as soon as they are sensed has been evaluated according to different parameters, such as the sliding window size and stride. Secondly, different CNN architectures have been analysed measuring both the detection performance and the computational cost, selecting the proposed-by-the-authors MobileNetv1Lite architecture as the most suitable option for this work according to a new figure of merit also uncovered in this manuscript. This architecture is presented in more detail in Section III-D. Thirdly, different approaches have been followed to adapt the data types and to optimize the CNN operations for the FPGA computing capabilities. Finally, two implementations of the MobileNetv1 architecture have been synthesized making use of Vitis HLS, with a data type precision of 32-bit floating point and 16-bit fixed point and targeting a Xilinx Kintex Ultrascale XCKU040-2FFVA1156E.

The rest of the manuscript is structured as follows: Section II presents the VIDEO project, Section III briefly describes the main features of CNNs and proposes a selection of the architectures that will be evaluated for this project. Also,

different experiments and results obtained from the different high level implementations are discussed. Section IV shows the results of the FPGA implementation and finally, Section V summarises the conclusions extracted from this work as well as further research directions.

II. VIDEO Project Overview

The EU-funded VIDEO project has received funding from the European Union's Horizon 2020 research and innovation program. The goal of this project is to develop the next instrument generation for Earth observation. It's a novel architecture based on state-of-the-art technologies for mirrors, structures (additive manufacturing), and detection (new generation detector and processing chain). The VIDEO instrument will have the capability to perform high-resolution video monitoring on an extremely wide scene. During the project, this instrument will have the capacity to detect and track the selected target in a wide video scene. Additionally, the system will have a compression stage able to compress the sub-block of the image where the target has been detected in order to minimize the downloaded data to the ground. Accordingly, all the decisions made during this work are oriented to satisfy the requirements and restrictions imposed by the project.

Within this context, this work is focused on providing solutions for the detection of targets using high resolution RGB images and video data acquired from Earth Observation satellites. In particular, the proposed solution is based on the use of a CNN executed on a FPGA device on-board a satellite to carry out the detection of human-made targets, such as ships or planes. To reach this goal, this work includes different development stages that go from the network architecture development and training to the final FPGA implementation.

In the following subsections we present the target application workflow, the proposed high-level design and implementation methodology and, finally, an overview of how the system is expected to work and how that affects the CNN selection.

A. Target application workflow and constraints

CNNs are quite flexible in the sense that one architecture can be trained for different use-case applications. In the case of human-made objects, such as ships or airplanes, one single architecture could be trained using different data sets depending on the use-case and then the weights uploaded to the on-board device without having to modify the implemented architecture. This is possible thanks to the fact that convolutional layers located at the early stages of the CNN start looking for low-level features, such as edges and curves and then building up to more abstract concepts through the convolutional layers placed at the end of the network. The CNN uses low-level features obtained at the initial levels to generate high-level features such as paws or eyes to identify the object. Therefore, early stages of the network could be trained with a single or a mixed data set leaving the last stages of the architecture to be trained with the real use-case data set.

All these advantages are used in the VIDEO project. The selected architecture is trained on ground using as much data as possible and then the weights loaded via the uplink. As soon as the satellite starts receiving new data, new images will be sent to ground and added to the data set so that new and more realistic data starts helping to fine-tune the performance of the CNN. Figure 1 illustrates this workflow.

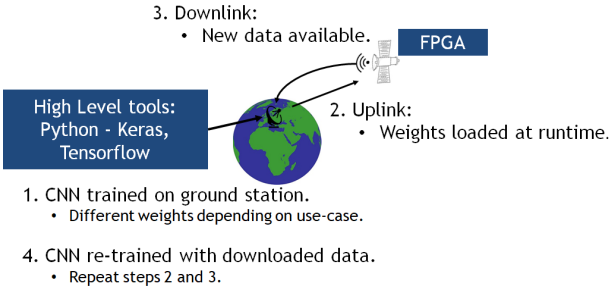

Fig. 1. Target application workflow.

B. Proposed methodology

Generally speaking, when a target detection or image classification application using CNNs is being developed, a large data set and its corresponding ground truth is required. Additionally, it is important that the data set used for the development of the algorithms is representative of the real case scenario. Sometimes this is not really possible since the device that is supposed to capture the data is not available before the development of the algorithm, so public and commercial data sets have to be used for the training stage of the CNN.

It is also important to consider that for several years, Register-Transfer Level (RTL) has been the most important and used method to describe hardware systems. However, High-Level Synthesis (HLS) provides the designer with tools to describe these systems on a behavioural level, omitting several implementation details and reducing the design effort by raising the abstraction level. HLS also helps to accelerate the verification process, as the design can often be verified using software verification tools that are faster and simpler to use than RTL simulation tools. Finally, whereas RTL design requires previous knowledge about hardware description languages such as VHDL or Verilog, HLS tools usually use languages like C/C++ [5].

Consequently, a methodology that allows the development, training and validation of the CNN from a high-level perspective needs to be planned. Figure 2 shows a flowchart to illustrate the methodology proposed in this paper. Such methodology can be divided in two main periods. The first one corresponds to the development period, where the network architecture is first designed, trained and validated using existing data sets. Once the appropriate architecture is selected, it is then implemented on the FPGA device. This period corresponds to Stages 1 and 2 shown in the flow diagram displayed in Figure 2, and it is executed just once prior to the satellite launching date. The second period covers the entire satellite mission duration and corresponds to Stages 3, 4 and 5 shown in the flow diagram displayed in Figure 2. During this period, the device is constantly acquiring new data. This data is processed using the implemented architecture

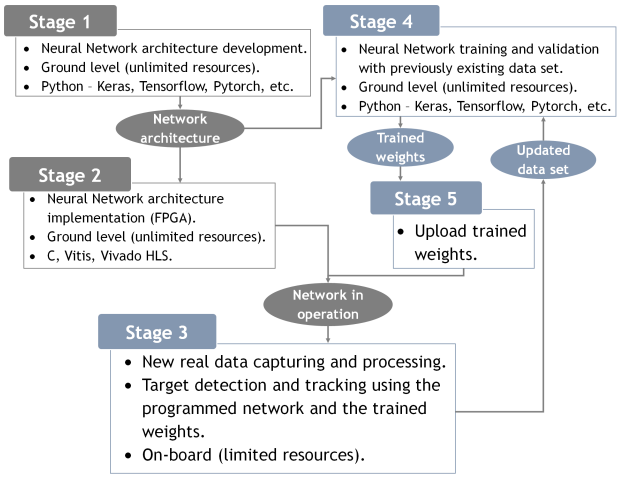

Fig. 2. Proposed methodology.

with the pre-loaded weights on-board. In parallel, part of the acquired data is sent to the ground station and can be used to increase the data set used for the training and validation stages. This way, fine-tuning can be applied [6] and the network can be re-trained without having to change its architecture. That allows the CNN to increase its detection performance as more representative data is added to the existing data set. Then, these new weights can be uploaded to the on-board electronics. Additionally, this methodology makes also possible the selection of a new detection target, for example the application could switch from ship to airplane detection, simply by training the network on ground with a different data set, a mixture of data sets or applying transfer learning [7] from another pre-trained architecture. Again, those new weights can be uploaded via the uplink as it is shown in Figure 1. The results presented in this work correspond to stages 1 and 2 of Figure 2.

C. System functionality overview

This subsection provides an overview of how the VIDEO project detection system works. The sensor to be used in this project provides the system with a considerably large image in terms of resolution (48 megapixels), so it is not a feasible idea to store the entire image in memory and start feeding the CNN with the complete image so that the CNN can process it in one shot. Also, the image encoding method used by the sensor is Band-Interleaved by Line (BIL), so waiting for the whole image to arrive and not taking advantage of the fact that the CNN could start processing smaller patches of the image as soon as the needed lines have arrived, does not represent the optimal implementation either. Therefore, to avoid the first scenario and to take advantage of the second one, a different approach is proposed in this work.

Each video frame is independently processed as an RGB image by the network. As previously discussed, the encoding method in which the video will be received is BIL, so the CNN could start processing the first line of smaller patches without having to wait until the full frame is received following a pipeline strategy. Each image will be processed applying a

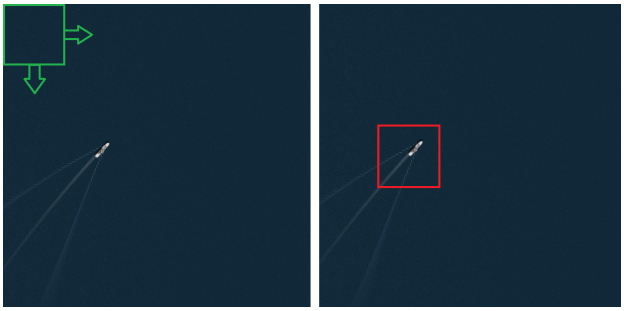

Fig. 3. Overview of the target detection process

sliding window that will move from left to right and from top to bottom with a certain stride and overlap according to the network input layer size and parameters. This overlap will avoid missing targets when they are not totally contained within the current window. It is important to mention that this overlap needs to be carefully chosen. Figure 3 presents a graphical explanation of this process.

The network result will indicate if there is at least one target present within the current window area or not. The window will have to be small enough to locate the targets within the image but also, it will have to be large enough to be able to contain most of a target so it can be correctly detected by the CNN. Finally, as soon as the result of the current window is positive, this is, one or more targets are located within the window, the process will stop firing a trigger signal that will let the system know that a target has been detected. In that case, the detection mode will stop. This approach endorses the use of the methodology previously presented, since in many cases it will not be necessary to process the whole 48 megapixels image captured by the sensor. Figure 4 illustrates this methodology.

Newer CNN architectures such as YOLO [8] are based on single shot object detection methodologies. These architectures work really well and are extremely fast. However, as presented in [9], there are some unique aspects of satellite imagery that present a whole set of different challenges. These aspects such as the enormous size of the input image and the small spatial extent of the targets within the image, rotation invariance since targets can have any orientation, make single shot object detection methodologies not suitable for object detection on satellite imagery.

D. System constraints

In order to select the right window size and stride so that a complete ship or airplane can be located within the image,

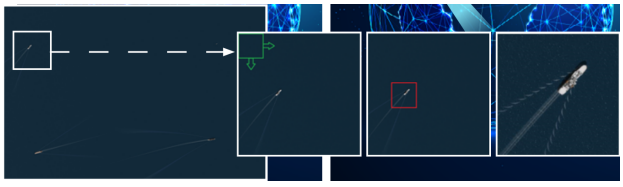

Fig. 4. Switching from detection mode to tracking mode.

some calculations are needed. The sensor has a Field of View (FOV) of 2.5 degrees and the image captured by the sensor will have a size of 48 megapixels, this is 6000 x 8000 pixels. Also, the satellite will be orbiting at around 500 km. So in order to calculate the pixel size Equations 1 and 2 must be applied.

$$\mathrm{d} = \mathrm{h} \cdot \tan \frac{\mathrm{FOV}}{2} \tag{1}$$

In Equation 1, *d* is the real distance in meters from the middle of the image to one end and *h* is the satellite orbit height in meters. Once *d* gets calculated, the pixel size can be calculated using Equation 2.

$$\mathrm{pixelSize} = \frac{2 \cdot \mathrm{d}}{\mathrm{nPixels}} \tag{2}$$

In Equation 2, *nPixels* is the distance in pixels of one side of the image (in this work's case either 6000 or 8000).

With all the above, and taking the specific values presented earlier, the pixel size in the vertical dimension of the image acquired by the sensor of the VIDEO project would be equivalent to 3.636 m and the pixel size in the horizontal dimension would be equivalent to 2.727 m. If the *Seawise Giant* [10] is taken as the longest ship ever built, with a length of 458.45 m, it would cover a minimum of 169 pixels according to the previous calculations. Accordingly, it can be concluded that a window size larger than 200 pixels should be enough. Therefore, two different window sizes have been used for the experiments: 512x512 pixels and 256x256 pixels. This should give enough margin for the target detection. In order to cover scenarios where the target has one half within the current window and the other half in the next one, or the target has a tiny portion in one window and the rest in the next one, a stride of 249 pixels for the 512x512 pixels input size case and a stride of 125 pixels for the 256x256 pixels input size case have been selected. Given that the longest ship would need 169 pixels, this makes sure that targets are never missed, since the overlap between images is big enough.

III. CNN ARCHITECTURES

A. Convolutional Neural Networks

For the past few years, Convolutional Neural Networks (CNNs) have been one of the most extensively used type of neural networks. In a general sense, CNNs are primarily used in the field of pattern recognition within images [11], since they are able to process multiple arrays of data, for example RGB images, which are a composition of 2D arrays of pixels in the three colour channels [12]. That makes them a good choice for image processing and in particular, for multiband remote-sensing image processing [13].

Diving into details, CNNs are mainly designed using four layers: convolutional layers, non-linear activation layers, pooling layers and fully-connected layers. Convolutional layers convolve the input pixels with a set of filters, also called kernels, to create a feature map that summarizes the presence of detected features in the input. The output of the convolutional layers is then processed using an element-wise non-linear transform by the non-linear activation layers. Pooling layers aggregate neighbour pixels using a permutation invariant function, normally a max or mean operation. Finally, in the fully-connected layer each node is directly connected to every node in the previous and next layers.

CNNs benefit a lot from the fact that many natural signals are compositional hierarchies, this is, the aggregation of lower-level features produce higher-level ones. Taking an image as an example, local combinations of edges form motifs, motifs assemble into parts and parts finally assemble into objects [12]. This is one of the main reasons why CNNs are a well-suited choice for object detection.

B. Analysis of CNN architectures detection efficiency

In this work five different architectures have been evaluated. The idea behind the selection of the architectures is to have an option as simple and light as possible, able to rapidly process small windows of a much larger image. The starting point of this work was the evaluation of the ResNet50, AlexNet and VGG19 architectures, as these are CNNs that are proven to give a good accuracy so they can be used as a golden reference. Then, we started to reduce the complexity of the architectures trying to avoid a reduction in the performance of the CNN. Obviously, these initial architectures turned out to be too heavy for the target FPGA device, so we started using lighter versions of the ResNet architecture and, as it will be described next, some modifications were made to the original AlexNet network to reduce the number of trainable parameters. Additionally, some newer and lighter architectures - MobileNet and DenseNet - were also introduced for the experiments. All these architectures are generally described in the following lines.

1) AlexNet: AlexNet was released in 2012 by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinto. It consists of eight trainable layers: the first five are convolutional layers and the last three are fully connected layers. AlexNet uses the ReLU activation function, which showed improved training performance over *tanh* and *sigmoid* [14]. The main goal of this work is target detection, so the final fully connected layer is connected to a 2-way softmax layer. Figure 5 shows the architecture of this CNN.

2) VGG Network: The VGG Network was developed by the Visual Geometry Group of the University of Oxford. It consists of multiple connected convolutional layers and fully-connected layers. The size of the convolutional kernel is 3x3 with a stride of 1 [15]. In this work the 19-layer version of the architecture was used thinking that this very heavy architecture would get great accuracy results.

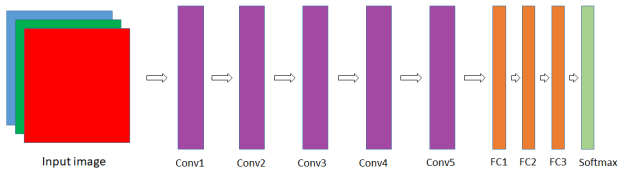

Fig. 5. AlexNet architecture.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2022.3169330, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

3) ResNet: As technology evolves networks are becoming deeper. However, really deep networks are not easy to train, since there are issues like the vanishing gradient problem [16]. As a result of this issue, as the network goes deeper its performance starts degrading rapidly. In order to resolve this, a new architecture was presented in [17] which explored the addition of an "identity shortcut connection", skipping one or more layers. These shortcut connections basically perform identity mapping, i.e. their outputs are added to the outputs of the stacked layers. These connections do not add extra parameters or computational complexity. In this work three different variants of the ResNet architecture were investigated: ResNet18, ResNet34 and ResNet50. As explained at the beginning of this section, ResNet50 often gives really good accuracy results so this architecture is one of our golden references. However, it is way too heavy, so lighter versions of this architecture were also tested.

4) MobileNet: MobileNets were originally designed by Google for mobile and embedded vision applications [18]. One of the main features of the MobileNet architecture is that it uses depthwise separable convolutions. This significantly decreases the number of trainable parameters when compared to networks with regular convolutions with the same depth. MobileNetv2 introduced the concept of linear bottlenecks and inverted residuals [19]. Therefore, this results in lightweight deep neural networks which is why this type of architecture is really attractive for the kind of scenario this work is targeting. The architecture of the MobileNetv1 CNN can be seen in Figure 6.

5) DenseNet: The DenseNet architecture [20] was designed to ensure the maximum flow of information between the layers in the network. All layers with matching feature-map sizes are directly connected with each other. In order to keep the feed-forward nature, each of the layers gets additional inputs from all the preceding layers and passes on its own feature-maps to all the subsequent layers.

C. Data sets

One of the main applications of the VIDEO project is ship and airplane detection using CNNs. This is why two public ship and airplane data sets have been used to train the architectures presented in Section III-B and Section III-D. These data sets are described in the next section.

1) Ship data set: The *Ship* data set was created using the MASATI data set [21] and a subset of the HRSC2016 data set [22]. The MASATI data set provides colour images in dynamic marine environments, and it can be used to evaluate ship detection methods. The HRSC2016 (High Resolution Ship Collections 2016) is a data set used for scientific research and contains a large number of images collected from Google Earth with different ship types. The *Ship* data set consists of 8,558 RGB images, each of them with a size of 512x512 pixels. Each image may contain one or multiple targets in different conditions (location, weather, illumination, etc.). The labels are distributed in two categories: "ship" and "no-ship". The distribution of the training, validation and test sets is 80%, 10% and 10% respectively and the inclusion of the images within each category has been executed in a random manner. Figure 7 shows some of the images in this data set.

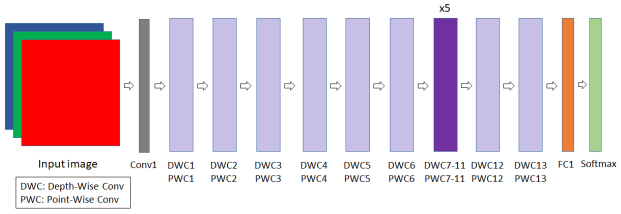

Fig. 6. MobileNet architecture.

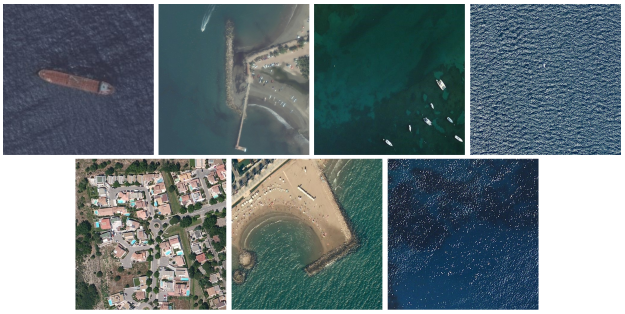

Fig. 7. MASATI data set.

Fig. 8. Airplane data set.

2) Airplane data set: The *Airplane* data set was created using a combination of images from the UC Merced Land Use data set [23] and the Aerial Image Data set (AID) [24]. The images from the UC Merced Land Use data set were manually extracted from large images from the USGS National Map Urban Area Imagery collection. This is a 21-class land use image data set meant for research purposes, containing 100 images with airplanes. The AID data set is a large-scale aerial image data set, obtained by collecting sample images from Google Earth imagery. This data set is made up of 30 aerial scene types, being "airport" one of the scene types included, making it a good choice for our work. Each image has a size of 256x256 pixels. In this work, this data set contains 2,695 RGB images divided as follows: 80% for the training set, 10% images for the validation set and finally 10% images for the test set. The labels are: "airplane" and "no-airplane". Again, the inclusion of the images within each category has been executed randomly. Figure 8 shows some of the images present in this data set.

3) Mixed data set: The mixed data set has been built using a mixture of images from the *Ship* data set and the *Airplane* data set. Since one of the main applications of the VIDEO project is ship and airplane detection, we considered that having a mixed data set made up of a mixture of the *Ship* and *Airplane* data sets would be useful to test the methodology

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2022.3169330, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

discussed in sub-section II-A. This data set has 8,235 images with the same sets distribution as explained before, this is, 80% for the training set, 10% for the validation set and 10% for the test set.

D. Modified architectures

As discussed in Section I, the large number of trainable parameters and the considerable complexity of the involved calculations, make the implementation of a CNN architecture on an FPGA quite challenging. In order to reduce these requirements, two of the architectures that provide better results in the literature and are lighter in terms of FLOPs have been modified: AlexNet and MobileNetv1.

The original AlexNet architecture presents three dense layers at the end of the processing chain. This type of layer generates a huge amount of trainable parameters that adds an extra overhead in the memory footprint. Therefore, we propose a new architecture called AlexNetLite that removes two of the final three dense layers, achieving a reduction of 98.32% in the number of trainable parameters.

A similar reasoning has been followed with the original MobileNetv1 architecture. In this case, a reduction in the number of filters by a factor of 4 plus the activation of the *shallow* option which removes 5 stages of the network has been applied. This modified architecture, named MobileNetv1Lite, gives a reduction of 96.15% in the number of trainable parameters.

Both architectures are introduced in Figure 9 and Figure 10. Additionally Table I presents a detailed description of the MobileNetv1Lite architecture.

E. Metrics

A reasonable performance metric for CNN evaluation is the ratio between the amount of correctly classified samples and the total number of samples. This measure is named accuracy and can be defined as follows:

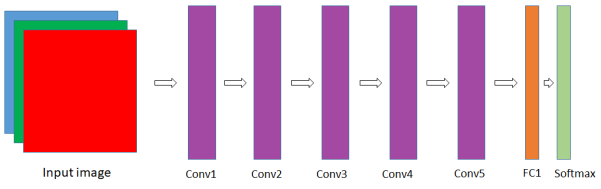

Fig. 9. AlexNetLite.

Fig. 10. MobileNetv1Lite

TABLE I
MOBILENETV1LITE ARCHITECTURE

Stage	Layer	Kernel size	Strides	Filters
1	conv2D	3x3	2,2	8
2	dwconv2D	3x3	1,1	N/A
2	pwconv2D	1x1	1,1	16
3	dwconv2D	3x3	2,2	N/A
3	pwconv2D	1x1	1,1	32
4	dwconv2D	3x3	1,1	N/A
4	pwconv2D	1x1	1,1	32
5	dwconv2D	3x3	2,2	N/A
5	pwconv2D	1x1	1,1	64
6	dwconv2D	3x3	1,1	N/A
6	pwconv2D	1x1	1,1	64
7	dwconv2D	3x3	2,2	N/A
7	pwconv2D	1x1	1,1	128
8	dwconv2D	3x3	2,2	N/A
8	pwconv2D	1x1	1,1	256
9	dwconv2D	3x3	1,1	N/A
9	pwconv2D	1x1	1,1	256
10	globalAvgPool	N/A	N/A	N/A
10	dense	N/A	N/A	N/A

$$\mathrm{Acc} = \frac{\mathrm{TruePos} + \mathrm{TrueNeg}}{\mathrm{TruePos} + \mathrm{FalsePos} + \mathrm{TrueNeg} + \mathrm{FalseNeg}} \tag{3}$$

However, if the data set is unbalanced, accuracy is not a reliable measure as it can give an overoptimistic estimation on the majority class [25]. In this situation, F1Score gives a much more reliable measure. F1Score can be calculated as:

$$\mathrm{F1Score} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{4}$$

where *Precision* is the amount of True Positives divided by the number of True Positives + False Positives and *Recall* is the number of True Positives divided by the number of True Positives + False Negatives.

As discussed in subsection III-C, the data sets used in this work are unbalanced and thus F1Score is a better metric to quantify the performance of the models.

At this point, it is important to define which metrics will be considered to measure the suitability of the different network architectures to carry out an efficient hardware implementation that matches the VIDEO project goals and restrictions. In this sense, we consider that a trade-off between the F1Score and *WFs* obtained by each network provides a good measure of the suitability of the architecture for this particular problem, where *WFs* is a function that gets the FLOPs per window based on the sliding window width (sw_w) and height (sw_h). Hence, in order to facilitate the selection of the right architecture, we define the efficiency factor of the CNN detection process as follows:

$$\mathrm{efficiency} = \frac{\mathrm{F1Score}}{\mathrm{WFs}(\mathrm{sw_w}, \mathrm{sw_h})} \tag{5}$$

TABLE II
SUMMARY OF TRAINABLE PARAMETERS AND FLOPS - SHIP DATA SET (512X512X3)

Architecture	Trainable params (Millions)	GFLOPs
ResNet50	23.79	20.34
ResNet34	2.55	2.85
ResNet18	1.42	1.74
AlexNet	230.17	6.17
AlexNetLite	3.85	5.71
MobileNetv1	3.21	3.04
MobileNetv1Lite	0.12	0.05
MobileNetv2	2.24	0.33
DenseNet	1.52	3.39
VGG19	573.70	103.26

Once the window size and stride are selected, the total number of FLOPs needed to process the entire high resolution image can be defined as:

$$\mathrm{TFs} = \mathrm{WFs}(\mathrm{sw_w}, \mathrm{sw_h}) \cdot \mathrm{WpR}(\mathrm{S}, \mathrm{i_w}) \cdot \mathrm{WpC}(\mathrm{S}, \mathrm{i_h}), \quad (6)$$

where windows per row (WpR) is a function dependent on the applied stride (S) and the width of the input image (i_w); and windows per column (WpC) is a function dependent on the applied stride (S) and the height of the input image (i_h).

F. Experiments and results

Each of the considered network architectures has been implemented using Python and Keras [26]. Once the high level implementation was done, each network was trained. As presented in Section III-C, all the results obtained from the different implementations were collected using the *Ship*, *Airplane* and *Mixed* data sets. Each architecture was trained running 25 epochs.

1) Ship and Airplane data sets results: Table II and Table III show the number of trainable parameters and FLOPs corresponding to each architecture for both the *Ship* and the *Airplane* data sets. It is important to highlight that the input size of each data set is different as explained in Section III-C1 and Section III-C2. Also, it is worth mentioning that the input size used for both data sets accomplishes the calculations made in Section II-D.

TABLE III
SUMMARY OF TRAINABLE PARAMETERS AND FLOPS - AIRPLANE DATA SET (256X256X3)

Architecture	Trainable params (Millions)	GFLOPs
ResNet50	23.60	5.08
ResNet34	2.55	0.71
ResNet18	1.42	0.43
AlexNet	62.40	1.41
AlexNetLite	3.76	1.29
MobileNetv1	3.21	0.76
MobileNetv1Lite	0.12	0.012
MobileNetv2	2.24	0.43
DenseNet	1.49	0.81
VGG19	171.04	25.82

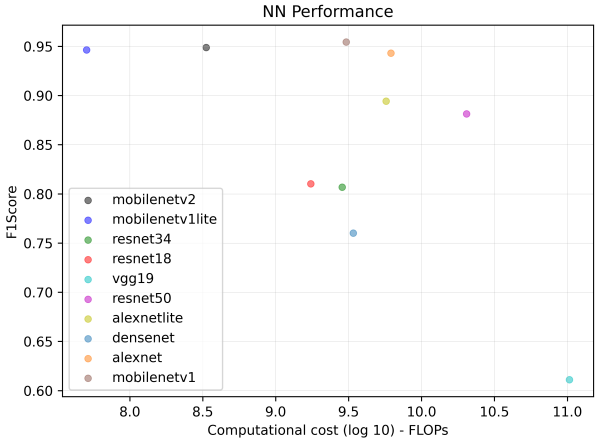

Fig. 11. Ship data set - F1Score vs. FLOPs.

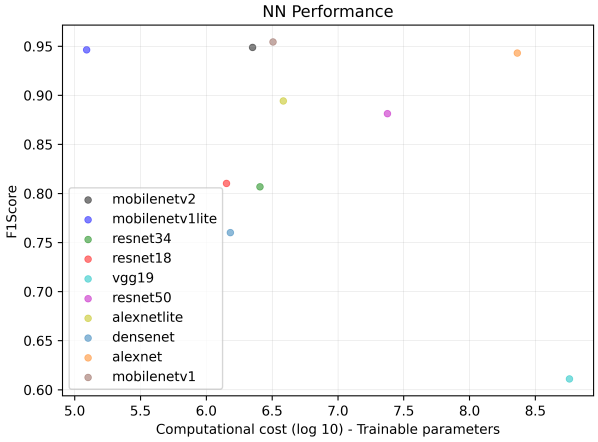

Fig. 12. Ship data set - F1Score vs. Trainable parameters.

The obtained results can be seen in Figures 11-14, where the F1Score is compared with the number of trainable parameters (equivalent to the number of trained weights to be stored in memory) and the number of FLOPs. Having a look at Figures 11 and 12, which correspond to the *Ship* data set results, it can be concluded that all the flavours of the MobileNet architecture get the best F1Score. However, MobileNetv1Lite not only gets the third best F1Score but it is also the lightest architecture which is a critical requirement for the FPGA implementation. Taking a look now at Figures 13 and 14, this is the *Airplane* data set results, MobileNetv1Lite gets the third best result being also the lightest architecture.

Additionally, Equation 5 can be applied to get the efficiency numbers for each CNN. These results can be seen in Table IV. Having a look at these results, it is very clear, particularly in the case of the *Airplane* data set, that MobileNetv1Lite gets the best efficiency result, being MobileNetv2 the second option

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2022.3169330, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

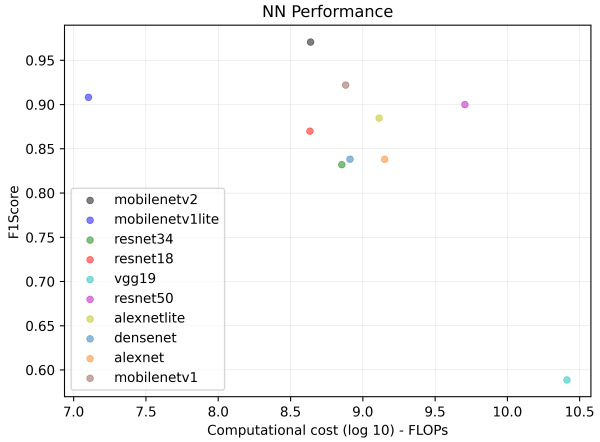

Fig. 13. Airplane data set - F1Score vs. FLOPs.

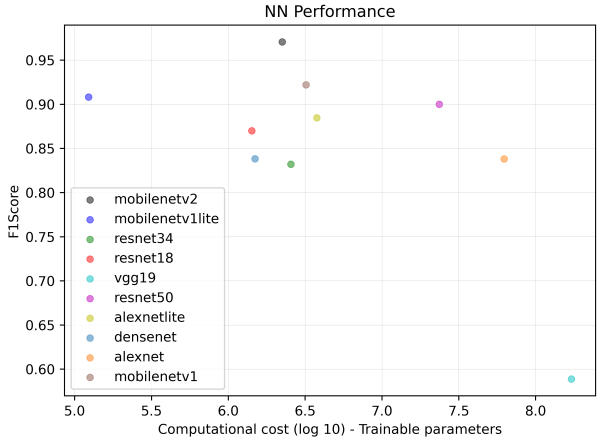

Fig. 14. Airplane data set - F1Score vs. Trainable parameters.

but with a much lower efficiency factor. MobileNetv1Lite gets a 557% efficiency increase in the *Ship* data set and a 3,469% increase in the *Airplane* data set compared with MobileNetv2.

According to these results, it is concluded that this architecture is the best candidate for the FPGA implementation of the project, since ship detection is the main use case of the VIDEO project. Also, architecturally speaking, MobileNetv2 is slightly more complex in terms of number of layers needed, which may cause an increase in the resources consumption when implemented in hardware. Finally, as the data path is longer in the case of MobileNetv2, the critical path may affect the speed of the implementation. This is why MobileNetv1 was considered as a good candidate for the optimization as it has a better trade-off between implementation simplicity and accuracy. However, MobileNetv2 will be definitely considered for future implementations.

Finally, after applying Equation 6 to the MobileNetv1Lite architecture, it can be estimated that, for an input size of 512x512 and a stride of 249 pixels, which means 30 horizontal strides and 22 vertical strides, *TFs*, as already defined in Equation 6, is 33.37 GFLOPs. For an input size of 256x256 and a stride of 125 pixels, which means 62 horizontal strides and 46 vertical strides, *TFs* is 34.22 GFLOPs. Therefore, 512x512 seems like a reasonable size for ship detection given that it needs 2.48% less FLOPs compared to the results obtained with a size of 256x256. Also, that size gives enough margin considering the numbers presented in Section II-D. In the case of airplane detection, 256x256 seems like a fair size too, since generally airplanes are smaller than ships and therefore the window size can be smaller.

TABLE IV
ARCHITECTURE EFFICIENCY

Architecture	Ship data set	Airplane data set
ResNet50	0.043	0.171
ResNet34	0.282	1.139
ResNet18	0.465	1.797
AlexNet	0.152	0.609
AlexNetLite	0.156	0.643
MobileNetv1	0.313	1.217
MobileNetv1Lite	18.716	76.526
MobileNetv2	2.845	2.144
DenseNet	0.223	0.995
VGG19	0.006	0.023

2) Addressing the bandwidth limitation: One of the main constraints of the VIDEO project application is the bandwidth of the up-link and the down-link. Also, the target of the application can change over time. In order to solve this, we propose a transfer learning methodology to reduce the required bandwidth in case new weights need to be uploaded onto the on-board electronics. For this reason, the architecture that provided the best results in terms of efficiency, MobileNetv1Lite (please refer to Table IV for more details), was trained with the *Mixed* data set.

After a first training was done, the weights corresponding to the trainable layers of stages 1-7 were stored. After that, the model was loaded with those weights and re-trained using the *Ship* and *Airplane* data sets, but only the layers of stages 8-10 were allowed to re-train. The obtained results are shown in Table V.

As it can be seen, there is a slight decrease in the efficiency factor achieved by this approach for both the *Ship* and the *Airplane* data sets, 10.043% and 5.342% respectively, but still the efficiency factor is much higher compared to the ones obtained by the other architectures.

The main advantage this approach offers is that the on-

TABLE V
MIXED DATA SET RESULTS

	Ship data set	Airplane data set
F1Score	0.951	0.917
FLOPs	0.05	0.012
Efficiency	18.809	72.438

board device can be loaded with the *Mixed* data set weights corresponding to the trainable layers of stages 1-7 of the architecture and then, depending on the target of the application, only the weights corresponding to the trainable layers of stages 8-10 of the architecture have to be uploaded via the up-link, saving a considerable amount of bandwidth and reconfiguration time.

IV. FPGA IMPLEMENTATION

In the previous section the obtained results for each architecture were presented, reaching the conclusion that MobileNetv1Lite is the best candidate for the FPGA implementation, based on the size of the network in terms of trainable parameters and the efficiency factor presented in Equation 5.

The increasing computational complexity has also brought an increase in the performance of the state-of-the-art CNNs. However, the extra cost in terms of computational resources and memory requirements makes these networks non-viable for applications that involve real time processing or their deployment on embedded hardware with limited resources such as FPGA devices [27]. That makes the research of solutions that reduce this complexity and requirements a critical step in the FPGA implementation. Some of these strategies can be performed before deploying the design on the board. In this work, two main optimizations have been implemented and are presented next. In particular, the target FPGA device that is used in the VIDEO project is the Xilinx Kintex Ultrascale XCKU040-2FFVA1156E, a commercial equivalent of the space-grade Xilinx Kintex Ultrascale XQRKU060 with less capacity, which ensures the feasibility of the design since if the available resources of the targeted FPGA are enough, then the implementation on the XQRKU060 device should be fine as well in terms of capacity. This device is integrated in the Xilinx Kintex UltraScale FPGA KCU105 Evaluation Kit board. Also, we make use of the Xilinx Vivado Design Suite, which includes Vitis HLS and Vivado for the hardware implementation.

A. Data type conversion

The initial Keras implementation of the different architectures made use of 32-bit single-precision floating point. Although this precision gave really good results, we propose a floating point to fixed point conversion for the CNN hardware implementation. As a first step, the weights from the Keras model were obtained using 16-bit single precision floating point. The Keras API provides the user with tools so that this conversion can be done in a few steps. Next, the input and weights were read by the C model using the *half* data type included in *hls_half.h*. A floating point to fixed point conversion was then performed using the *ap_fixed* data type provided by Vitis HLS. A reduction in both the memory footprint and the computational cost was achieved with this conversion, as it is presented in Section IV-C.

B. Batch Normalization layer optimization

The Batch Normalization layer is used to reduce *Covariate Shift*. That is the change in distribution of activation of a component. It also helps to reduce the effects of exploding and vanishing gradients [28]. This layer optimization gets applied for inference only. As it is explained next, all the trainable parameters are obtained at high level during the training phase. Considering the input has *C* channels, the output of this layer gets calculated by applying the following formula:

$$\mathrm{y} = \mathrm{gamma_c} \cdot \hat{\mathrm{x}} + \mathrm{beta_c}, \tag{7}$$

where $\hat{x}$ is the normalized input value and $gamma_c$ and $beta_c$ are trainable parameters. $\hat{x}$ is calculated as follows:

$$\hat{\mathrm{x}} = \frac{\mathrm{x} - \mathrm{mean(x_c)}}{\sqrt{\mathrm{var(x_c)} + \epsilon}}, \tag{8}$$

where x is the layer input, $mean(x_c)$ is the mean value of the input within a batch and $var(x_c)$ is its variance also within a batch. As it can be observed, there are many operations that are costly in terms of hardware resources and latency, particularly the square root and the division. This work proposes a change in the way the output of this layer is calculated based on the execution of most of these operations in the high level Keras implementation. In order to do this, new formulas have been developed. These are presented in Equation 9 and Equation 11.

$$\hat{\mathrm{x}} = \frac{\mathrm{x} - \mathrm{mean(x_c)}}{\sqrt{\mathrm{var(x_c)} + \epsilon}} = (\mathrm{x} - \mathrm{mean(x_c)}) \cdot \mathrm{var'_c}, \tag{9}$$

where *var'* is defined as follows:

$$\mathrm{var'_c} = \frac{1}{\sqrt{\mathrm{var(x_c)} + \epsilon}} \tag{10}$$

If $\hat{x}$ is substituted in Equation 7, a new formula for the output with just one multiplication and one subtraction can be obtained. This is shown in Equation 11.

$$\mathrm{y} = \mathrm{x} \cdot \mathrm{gv_c} - \mathrm{gbmv_c}, \tag{11}$$

where gv_c is $gamma_c \cdot var'_c$ and $gbmv_c$ is $(gamma_c \cdot mean(x_c) \cdot var'_c - beta_c)$.

Hence, the Python code now provides the hardware implementation with values for gv_c and $gbmv_c$, saving a considerable amount of hardware resources and also reducing the latency. Additionally, given that most of the calculations are executed at high level using 32-bit floating point, there is no reduction in the data precision when executing this. Finally, with this approach the number of parameters to be used by this layer gets reduced by a factor of two. This also causes a positive impact in the memory requirements since now it is only required to store gv_c and $gbmv_c$ in memory, instead of having to store $gamma_c$, $beta_c$, $mean(x_c)$ and $var(x_c)$ (needed for Equations 7 and 8).

C. Implementation results

After selecting the most suitable architecture for the project based on the results shown in Section III-F, the next step is the hardware implementation. As discussed in Section IV-A, the precision of the data type for this implementation must be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTARS.2022.3169330, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing

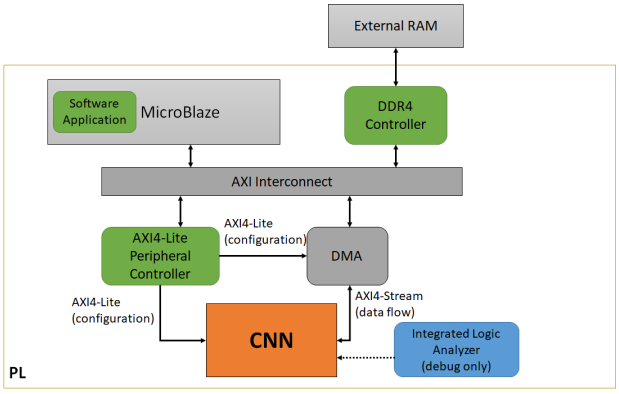

Fig. 15. FPGA implementation setup.

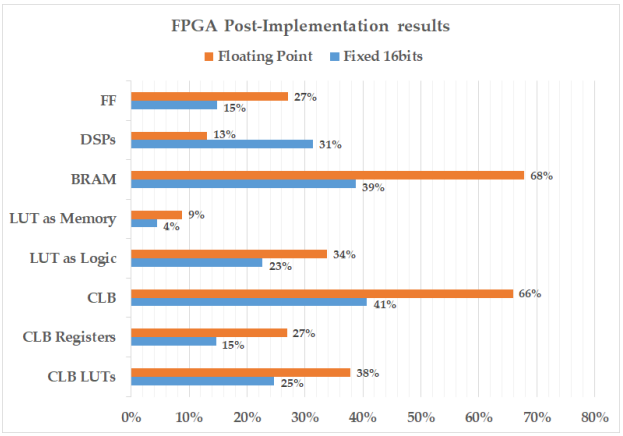

Fig. 16. FPGA post-implementation results.

changed from floating point to fixed point. In order to evaluate the improvements that can be achieved with this modification, two different FPGA implementations are presented next.

The fixed point precision format used in these experiments was *ap_fixed<16,6,AP_RND,AP_WRAP>*, where 16 is the word length in bits, 6 is the number of bits used to represent the integer value, *AP_RND* is the quantization mode (rounding to plus infinity) and *AP_WRAP* is the number of saturation bits in wrap modes (wrap around) [29]. Figure 15 shows the hardware implementation setup used.

As it can be seen in Table VI, the frequency achieved in both cases is 200 MHz. Also, resources consumption of the 32-bit floating point implementation is higher than the 16-bit fixed point implementation, which is expected. Diving into details, the percentage of CLBs, LUTs, BRAMs, DSPs and FFs needed in the case of the 16-bit fixed point implementation is 41%, 25%, 39%, 31% and 15% respectively, versus a 66%, 38%, 68%, 13% and 27% in the case of the 32-bit floating point implementation. The number of DSPs is bigger for the 16-bit fixed point implementation as this precision type makes a better usage of these resources.

Figure 16 show the post-implementation results in a more graphical way.

V. Conclusions and Further Research.

In this manuscript, eight different state-of-the-art CNNs have been evaluated for their hardware implementation on an FPGA. The selected CNN will have to perform object detection on HRRS images, therefore, several application constraints such as the image size or the easiness of the hardware implementation of the architecture have to be considered. In order to evaluate all the different candidates, memory footprint (in the form of number of parameters to be stored) and FLOPs were calculated, comparing the results with the F1Score of each model, as well as uncovering an efficiency factor that allows the right selection of the architecture. After getting the results and selecting the proposed-by-the-authors MobileNetv1Lite architecture as the best fit for the hardware implementation, a new experiment using transfer learning was run. This shows how the selected CNN can be trained for different applications by simply re-training the final layers of the network using pre-trained weights for the previous layers. Also, two hardware implementations using different data type precision have been done, showing how the conversion from a 32-bit floating point to a 16-bit fixed point implementation improves the resources utilization.

Since we are working towards a space-grade FPGA implementation and considering all the different issues that radiation in space could cause to the hardware device, we think that an ablation study needs to be done in order to measure the contribution of the different components to the overall system. Also, and thanks to the flexibility of the methodology presented, new modified architectures will be selected and implemented, giving priority to MobileNetv2, since this architecture gave really good results and its easiness of implementation makes it a good candidate. The automation of the implementation from high level to hardware is another key goal that fits within the scope of this team's interests, so works have already started in that direction.

TABLE VI
FPGA Implementation Results

Precision	CLB	LUTs	BRAM	DSP	FF	Freq
Float 32bits	19,964	91,809	407	252	131,265	200 MHz
Fixed 16bits	12,338	59,862	233	603	71,917	200 MHz

Acknowledgments

This work has been conducted within the Video Imaging Demonstrator for Earth Observation (VIDEO) project, that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 870485. This publications reflects only the authors' view. The Agency is not responsible for any use that may be made of the information it contains. This work has also been supported by the Spanish Government and European Union (FEDER funds) as part of support program in the context of TALENT-HExPERIA (HypErsPEctRal Imaging for Artificial intelligence applications) project, under contract PID2020-116417RB-C42

REFERENCES

[1] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.

[2] 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.

[3] M. Wirthlin, "High-reliability fpga-based systems: Space, high-energy physics, and beyond," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 379–389, 2015.

[4] Video Imaging Demonstrator for Earth Observation (VIDEO), [Online] .Available:https://video-h2020.eu/, (Accessed October 2021).

[5] S. Lahti, P. Sjövall, J. Vanne, and T. D. Hämäläinen, "Are we there yet? a study on the state of high-level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 898–911, 2019.

[6] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[7] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer International Publishing, 2018, pp. 270–279.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] A. V. Etten, "You only look twice: Rapid multi-scale object detection in satellite imagery," 2018.

[10] P. Stott, "The use of benchmarks in the popular reporting of commercial shipping: Is the titanic an appropriate measure to convey the size of a modern ship?" *The Mariner's Mirror*, vol. 100, no. 1, pp. 76–83, 2014. [Online]. Available: https://doi.org/10.1080/00253359.2014.866378

[11] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 152, pp. 166–177, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271619301108

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[16] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[21] A. P. Antonio-Javier Gallego and P. Gil, "Automatic ship classification from optical aerial images with convolutional neural networks," *Remote Sensing*, vol. 10, no. 4, 2018.

[22] Z. Liu., L. Yuan., L. Weng., and Y. Yang., "A high resolution optical satellite image dataset for ship recognition and some new baselines," in *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods - ICPRAM,*, INSTICC. SciTePress, 2017, pp. 324–331.

[23] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," 01 2010, pp. 270–279.

[24] 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, p. 3965–3981, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/TGRS.2017.2685945

[25] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, 2020.

[26] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[27] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2849–2858. [Online]. Available: https://proceedings.mlr.press/v48/linb16.html

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html

[29] Xilinx Vivado Design Suite User Guide, [Online].Available:https://www.xilinx.com/support.htmlsupport.html#documentation, pp. 76–77, (Accessed October 2021).

Romén Neris was born in Tenerife, Spain, in 1983. He received the degree in telecommunication engineering from the University of Las Palmas de Gran Canaria in 2012, and the M.Sc in telecommunication technologies in 2012. After working in the UK as Hardware Emulation Engineer for more than 6 years, he joined the Integrated Systems Design Division at the Institute for Applied Microelectronics (IUMA) in 2020, where he started his Ph.D. thesis in the field of deep learning and neural network implementations on FPGAs, in parallel with his job as deep learning researcher in the VIDEO project. His current research interests include neural network architectures, hardware/software co-design strategies, design verification, hardware emulation and automation methodologies.

Adrián Rodríguez was born in Las Palmas de Gran Canaria, Spain, in 1998. He received the Industrial Electronics and Automatic Engeneering degree from the University of Las Palmas de Gran Canaria in 2020. Later, he received the master's degree in Applied Electronics and Telecommunications imparted from the Institute for Applied Microelectronics, IUMA, in 2021. Currently, he is doing a PhD related with image and video processing in UAV.

Raúl Guerra was born in Las Palmas de Gran Canaria, Spain, in 1988. He received the Electrical Engineering degree from the University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain, in 2012, the master's degree in telecommunications technologies from the Institute of Applied Microelectronics, University of Las Palmas de Gran Canaria, and the Ph.D. degree in telecommunications technologies from the University of Las Palmas de Gran Canaria, in 2017. He was funded by the University of Las Palmas de Gran Canaria to do his Ph.D. research in the Integrated System Design Division. In 2016, he was a Researcher with the Configurable Computing Lab, Virginia Tech University. His research interests include the parallelization of algorithms for multispectral and hyperspectral images processing and hardware implementation.

Sebastian López (M'08–SM'15) was born in Las Palmas de Gran Canaria, Spain, in 1978. He received the degree in electronic engineering from the University of La Laguna in 2001, and the Ph.D. degree in electronic engineering from the University of Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain, in 2006. He is currently an Associate Professor with the University of Las Palmas de Gran Canaria, where he is currently involved in research activities with the Integrated Systems Design Division, Institute for Applied Microelectronics. He has coauthored more than 120 papers in international journals and conferences. His research interests include real-time hyperspectral imaging, reconfigurable architectures, high-performance computing systems, and image and video processing. Dr. López was a recipient of regional and national awards during the electronic engineering degree. Furthermore, he has acted as one of the program chairs of the IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), in 2014, and the SPIE Conference of High Performance Computing in Remote Sensing, from 2015 to 2018. He was an Associate Editor for the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, from 2008 to 2013. He is currently an Associate Editor for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, Remote Sensing (MDPI), and the Mathematical Problems in Engineering journal. In addition, he has been the Guest Editor of different special issues in JCR journals related with the research interests. He is also an active Reviewer for different JCR journals and a Program Committee Member of a variety of reputed international conferences.

Roberto Sarmiento received the Ph.D. degree in electronic engineering from the University of Las Palmas de Gran Canaria, in 1991. He is currently a Full Professor in Electronic Engineering with the Faculty of Telecommunication Engineering, University of Las Palmas de Gran Canaria. He contributed to the birth of this center in 1989. He was the Dean of the Faculty from 1994 to 1998 and the Vice-Chancellor for academic affairs and staff from 1998 to 2003. In 1993, he was a Visiting Professor with the University of Adelaide, Adelaide, SA, Australia, and later with the Edith Cowan University, Joondalup, WA, Australia. He was the Co-Founder of the Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria, and the Director of the Integrated Systems Design Division of this Institute. Since 1990, he has published more than 40 journal papers and book chapters and more than 120 conference papers. He has participated in more than 35 projects and research programmes funded by public and private organizations, from which he has been leader researcher in 16 of them. He has conducted several agreements with companies for the design of high performance integrated circuits, being the most remarkable the collaboration with Vitesse Semiconductor Corporation, Camarillo, CA, USA, and Thales Alenia Space, Spain. His research interests include multimedia processing and video coding standard systems, reconfigurable architectures, and real-time processing and compression of hyperspectral imaging. Dr. Sarmiento has been awarded with four six-years research periods by the National Agency for the Research Activity Evaluation in Spain.