National and Kapodistrian University of Athens
School of Science, Department of Physics
Section E: Section of Electronic Physics and Systems

Doctoral Thesis

Design Techniques of Parallel Accelerator Architectures for Real-Time Processing of Learning Algorithms

Elissaios Alexios Papatheofanous
201900512

3-member Advisory Committee:

Dionysios Reisis
Professor, National and Kapodistrian University of Athens (Supervisor)

Dimitrios Soudris
Professor, National Technical University of Athens

Anna Tzanakaki
Associate Professor, National and Kapodistrian University of Athens

Athens, March 2023

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Σχολή Θετικών Επιστημών, Τμήμα Φυσικής
Τομέας Ηλεκτρονικής Φυσικής και Συστημάτων

Διδακτορική Διατριβή

Τεχνικές Σχεδίασης Παράλληλων Υπολογιστικών Αρχιτεκτονικών για Επεξεργασία σε Πραγματικό Χρόνο Αλγορίθμων Μάθησης

Ελισσαίος Αλέξιος Παπαθεοφάνους
201900512

Τριμελής Συμβουλευτική Επιτροπή:

Διονύσιος Ρεΐσης
Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (Επιβλέπων)

Δημήτριος Σούντρης
Καθηγητής, Εθνικό Μετσόβιο Πολυτεχνείο

Άννα Τζανακάκη
Αναπληρώτρια Καθηγήτρια, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Αθήνα, Μάρτιος 2023

7-member Examination Committee:

Dionysios Reisis
Professor, National and Kapodistrian University of Athens (Supervisor)

Dimitrios Soudris
Professor, National Technical University of Athens

Anna Tzanakaki
Associate Professor, National and Kapodistrian University of Athens

Hector Nistazakis
Professor, National and Kapodistrian University of Athens

Markos Anastasopoulos
Associate Professor, National and Kapodistrian University of Athens

George Lentaris
Assistant Professor, University of West Attica

Konstantinos Nikitopoulos
Reader, University of Surrey, UK

Abstract

The current doctoral thesis focuses on Convolutional Neural Networks (CNNs) for computer vision applications and particularly on the deployment of the inference process of CNNs to embedded accelerators suitable for edge computing. The objective of the thesis is to address several challenges regarding the optimization techniques of CNNs towards their edge deployment as well as challenges in the field of CNN accelerator architectures design techniques. In this direction, the thesis focuses on different deep learning applications, including on-board payload data processing as well as solar irradiance forecasting, and makes distinct contributions to four different challenges in the fields of CNN optimization and CNN accelerators design.

First, the thesis contributes to the existing literature regarding image processing techniques and deep learning-based image regression for solar irradiance estimation and forecasting. It proposes an image processing method which is based on accurate sun localization in sky images and which utilizes the solar angles and the mapping functions of the lens of the sky imager camera. When the proposed method is applied to the sky images before these are processed by the image regression CNNs, the results from the extensive study that the thesis conducts, show that the method can improve the accuracy of the irradiance values that the CNNs produce in all cases by introducing only minimal computational overhead.

Next, the thesis focuses on the task of deep learning-based semantic segmentation in order to enable cloud detection from satellite imagery in on-board payload data processing applications. In particular, the thesis proposes a lightweight CNN model architecture, based on the U-Net architecture, which aims at providing an improved trade-off between model size and binary semantic segmentation performance. The proposed model utilizes several CNN techniques in order to reduce the number of parameters and operations required for the inference but at the same time maintain satisfying performance. The thesis conducts a study among CNN models for cloud detection, which are evaluated on the same test dataset as the proposed model, and thus showcases the advantages of the proposed model.

Then, the thesis targets the efficient porting of the inference process of image processing CNNs to edge-oriented embedded accelerator devices. The thesis opts for CNN acceleration based on Field-Programmable Gate Arrays (FPGAs) and contributes the adopted development flow which utilizes the Xilinx Vitis AI framework. Apart from exploring the capabilities of Vitis AI, including its advanced quantization solutions, the thesis also showcases an acceleration approach for accelerating different processes of a single computer vision task by taking advantage of the heterogeneous resources of the FPGA. The execution time and throughput results of the CNN models, for the tasks of binary semantic segmentation for cloud detection as well as image regression for irradiance estimation, on the FPGA, showcase the real-time processing capabilities of the accelerator.

Finally, the thesis contributes the design details of a bi-directional interfacing system

for high-throughput and fault-tolerant image transfers between deep learning embedded accelerators, in the context of on-board payload data processing architectures. The interfacing system is developed for interfacing an FPGA with the Intel Movidius Myriad 2 and the extensive testing campaign based on both commercial as well as prototype hardware platforms, shows that it can achieve a bit-rate of up to 2.4 Gbps duplex image data transfers.

Περίληψη

Η παρούσα διδακτορική διατριβή έχει ως βασικό αντικείμενο μελέτης τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs) για εφαρμογές υπολογιστικής όρασης (computer vision) και συγκεκριμένα εστιάζει στην εκτέλεση της διαδικασίας της εξαγωγής συμπερασμάτων των CNNs (CNN inference) σε ενσωματωμένους επιταχυντές κατάλληλους για εφαρμογές της υπολογιστικής των παρυφών (edge computing). Ο σκοπός της διατριβής είναι να αντιμετωπίσει τις τρέχουσες προκλήσεις σχετικά με τη βελτιστοποίηση των CNNs προκειμένου αυτά να υλοποιηθούν σε edge computing πλατφόρμες, καθώς και τις προκλήσεις στο πεδίο των τεχνικών σχεδίασης αρχιτεκτονικών επιταχυντών για CNNs. Προς αυτή την κατεύθυνση, η παρούσα διατριβή επικεντρώνεται σε διαφορετικές εφαρμογές βαθιάς μάθησης (deep learning), συμπεριλαμβανομένης της επεξεργασίας εικόνων σε δορυφόρους και της πρόβλεψης ηλιακής ακτινοβολίας από εικόνες. Στις παραπάνω εφαρμογές, η διατριβή συμβάλλει σε τέσσερα διακριτά προβλήματα στα πεδία της βελτιστοποίησης CNNs και της σχεδίασης επιταχυντών CNNs.

Αρχικά, η διατριβή συνεισφέρει στην υπάρχουσα βιβλιογραφία σχετικά με τεχνικές επεξεργασίας εικόνας, βασισμένες στα CNNs, για την εκτίμηση και πρόβλεψη ηλιακής ακτινοβολίας. Στα πλαίσια της διατριβής, προτείνεται μια μέθοδος επεξεργασίας εικόνας η οποία βασίζεται στον ακριβή εντοπισμό του Ήλιου σε εικόνες του ουρανού, χρησιμοποιώντας τις συντεταγμένες του Ήλιου και τις εξισώσεις του fisheye φακού της κάμερας λήψης εικόνων του ουρανού. Όταν η προτεινόμενη μέθοδος εφαρμόζεται σε φωτογραφίες του ουρανού πριν από την επεξεργασία τους από τα CNNs, τα αποτελέσματα από την εκτεταμένη μελέτη που διενεργεί η διατριβή, δείχνουν πως μπορεί να βελτιώσει την ακρίβεια των τιμών ακτινοβολίας που παράγουν τα CNNs σε όλες τις περιπτώσεις και με μικρή μόνο αύξηση στο πλήθος των υπολογισμών των CNNs.

Στη συνέχεια, η διδακτορική διατριβή επικεντρώνεται στην κατάτμηση εικόνων βασισμένη στη βαθιά μάθηση, με στόχο τον εντοπισμό σύννεφων από δορυφορικές εικόνες σε εφαρμογές επεξεργασίας δεδομένων σε δορυφόρους. Πιο συγκεκριμένα, στα πλαίσια της διατριβής προτείνεται μια αρχιτεκτονική μοντέλου CNN περιορισμένων υπολογιστικών απαιτήσεων, βασισμένη στην αρχιτεκτονική U-Net, η οποία στοχεύει σε μια βελτιωμένη αναλογία ανάμεσα στο μέγεθος του μοντέλου και στις επιδόσεις του στη δυαδική κατάτμηση της εικόνας. Το προτεινόμενο μοντέλο εκμεταλλεύεται πλήθος τεχνικών CNNs προκειμένου να μειώσει το πλήθος των παραμέτρων και πράξεων που απαιτείται για την εκτέλεση του μοντέλου, αλλά ταυτόχρονα να πετυχαίνει ικανοποιητική ακρίβεια αποτελεσμάτων. Η διατριβή διενεργεί μια μελέτη ανάμεσα σε CNN μοντέλα της βιβλιογραφίας για εντοπισμό σύννεφων που έχουν αξιολογηθεί στα ίδια δεδομένα με το προτεινόμενο μοντέλο, και έτσι αναδεικνύει τα προτερήματά του.

Επιπλέον, η διδακτορική διατριβή στοχεύει στην αποδοτική υλοποίηση του inference των CNNs επεξεργασίας εικόνας σε ενσωματωμένους επιταχυντές κατάλληλους για εφαρμογές edge computing. Για τον σκοπό αυτό, η διατριβή επιλέγει τα Field-Programmable Gate Arrays (FPGAs) για την επιτάχυνση των CNNs και συνεισφέρει τις λεπτομέρειες

της μεθοδολογίας ανάπτυξης που υιοθετήθηκε και η οποία βασίζεται στο εργαλείο Xilinx Vitis AI. Πέρα από τη μελέτη των δυνατοτήτων του Vitis AI, όπως των προχωρημένων τεχνικών κβάντισης των μοντέλων, η διατριβή παρουσιάζει επιπλέον και μια προσέγγιση επιτάχυνσης για την επιτάχυνση των επιμέρους διεργασιών μιας ολοκληρωμένης εργασίας μηχανικής όρασης η οποία εκμεταλλεύεται τους ετερογενείς πόρους του FPGA. Τα αποτελέσματα χρόνων εκτέλεσης και διεκπεραιωτικότητας (throughput) των CNNs τόσο για τη δυαδική κατάτμηση εικόνων για εντοπισμό σύννεφων όσο και για την εκτίμηση ηλιακής ακτινοβολίας από εικόνες, στο FPGA, αναδεικνύουν τις δυνατότητες επεξεργασίας σε πραγματικό χρόνο του επιταχυντή.

Τέλος, η διδακτορική διατριβή συνεισφέρει τη σχεδίαση ενός συστήματος διεπαφής, υψηλών επιδόσεων και με ανοχή στα σφάλματα, για την αμφίδρομη μεταφορά εικόνων ανάμεσα σε ενσωματωμένους επιταχυντές βαθιάς μάθησης, στα πλαίσια υπολογιστικών αρχιτεκτονικών για επεξεργασία δεδομένων σε δορυφόρους. Το σύστημα διεπαφής αναπτύχθηκε για την επικοινωνία ανάμεσα σε ένα FPGA και τον επιταχυντή Intel Movidius Myriad 2 και η εκτεταμένη διαδικασία επαλήθευσης του συστήματος, τόσο σε εμπορικά διαθέσιμες όσο και σε πρωτότυπες πλατφόρμες, έδειξε πως αυτό μπορεί να επιτύχει μέχρι και 2.4 Gbps αμφίδρομους ρυθμούς μετάδοσης δεδομένων εικόνων.

Στους γονείς μου, Παύλο και Δέσποινα,

για την αμέριστη συμπαράστασή τους.

Contents

Chapter 1

Introduction

Machine learning is currently being a transformative technology in many scientific fields. A large variety of machine learning model types, such as Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and decision trees, serve the needs of countless applications including natural language processing, recommender systems and computer vision among others. In particular, Convolutional Neural Networks (CNNs) which utilize multiple processing layers, also called Deep Convolutional Neural Networks (DCNNs), are part of the broader family of deep learning methods and are currently being extensively employed for many advanced computer vision applications.

The reason behind this wide adoption of CNNs for computer vision applications is that they have been shown to excel in particular tasks with some the most significant ones being classification, regression, semantic segmentation and object detection among others. In classification problems, it is assumed that there is one major object in the image. A CNN model processes the pixel values of the image and produces as many values as the discrete number of classes that the image may belong to, which represent the probability of the image to belong to that class. Regression problems, in the context of computer vision, usually require a CNN model to produce a single numerical value with a continuous range after processing an image. Semantic segmentation problems require from a CNN model to divide an image into distinct regions which belong to different semantic classes by producing a segmentation mask with a pixel-level prediction on a predefined set of pixel semantic classes. Finally, the more complex object detection problem arises when there are multiple objects of interest in an image and then the requirement from a CNN model is to produce both the positions of the objects, in the forms of bounding boxes, as well as the classes of the boxes contents.

1.1 Convolutional Neural Networks Fundamentals and Advancements

While the potential of CNNs on computer vision tasks was beginning to become apparent with the introduction of LeNet-5 [1] for handwritten digits recognition, at that time, there was also a lack of capable enough hardware to handle their processing requirements. The popularity of CNNs really began to explode with the introduction of AlexNet [2] where a CNN was shown to achieve significantly improved performance on the well-known ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012. The key characteristic behind the high recognition performance of AlexNet was its deep model architecture which is illustrated in Fig. 1.1. The AlexNet architecture features some of the fundamental layers and components used by state-of-the-art CNNs up to this day. These are the convolutional layers, the activation functions, the pooling layers and the fully-connected layers. Regarding the convolutional layers, each one consists of several convolution kernels, with trainable weights, which are applied to each of the input feature maps of the layer with a two-dimensional convolution operation. This way, the convolutional layers can be trained to extract visual features from the input feature maps or images, such as edges and corners. Most commonly, an element-wise nonlinear activation function is applied to the output feature maps of a convolutional layer which in the case of AlexNet is the Rectified Linear Unit (ReLU) activation function. In a CNN architecture, the output feature maps are forwarded to subsequent convolutional layers which are used to extract more high-level features. Convolutional layers are usually followed by pooling layers which perform spatial subsampling of the feature maps. Max pooling and average pooling are the two most common types of pooling encountered in CNNs. After several consecutive convolutional and pooling layers, before the final output of a CNN, one or more fully-connected layers follow. They are responsible for reducing the two-dimensional feature maps into one-dimensional feature vectors which can either be forwarded for further processing or can be considered as the output of the network. The operation that a fully-connected layer performs is essentially a matrix multiplication.

The deep CNN model architecture of AlexNet was essential for its high recognition performance. However, the most significant contribution of the authors of the AlexNet paper was the fact that they made feasible the computationally demanding process of training such a deep CNN on a very large image dataset, which traditional CPUs were not possible to handle at the time. CNN training is a supervised learning process where a labeled image dataset is used. First, a forward pass is performed to process the image samples and produce the CNN model results. The results are compared to the original label of the samples and the error is quantified with the loss function. Then, the back-propagation step is performed where the gradient of the model's weights with regards to the loss function is calculated. Finally, the optimization algorithm, e.g. the stochastic gradient descent algorithm, updates the weights using the weight gradients. All the above process is performed in an iterative way for several batches of the original dataset. In order to address

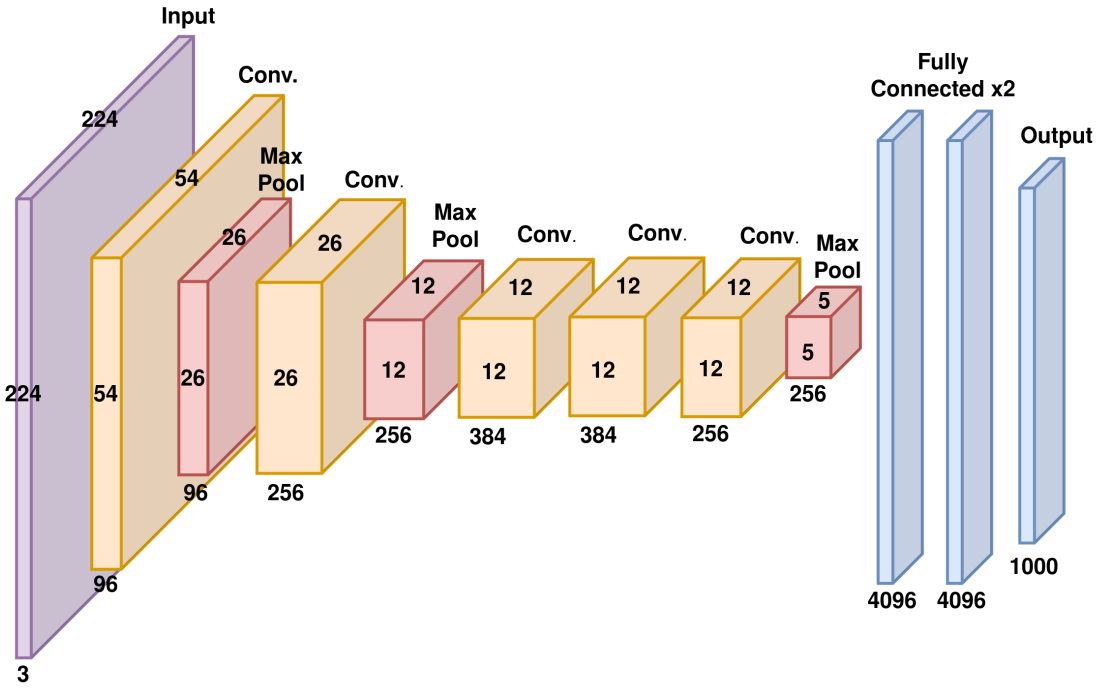

Figure 1.1: The AlexNet model architecture with the number of feature maps and their dimensions indicated at the output of each layer.

the training challenge, the authors of AlexNet accelerated the computations of the model on a Graphics Processing Unit (GPU) using the Compute Unified Device Architecture (CUDA) framework. This approach made AlexNet one of the most influential works in CNNs and up to this day, accelerating the training process of CNNs on GPUs is considered standard practice.

The advancements of deep learning acceleration frameworks for commodity GPUs as well as the increasing availability of publicly accessible and high-quality datasets for several different applications, have significantly boosted the development of deep learning models by the research community. Particular focus is placed on increasing the performance on CNNs on different tasks by increasing their representational capacity. This is achieved by including larger numbers of consecutive and more sophisticated layers in the CNN model architectures, thus making them even more computationally demanding. This trend of the growing computational demands of CNNs is showcased below where we calculate, using the PyTorch framework, the number of operations and parameters required for some of the most influential models over the years. The pioneering AlexNet model requires 0.71 GOPs to perform inference and includes 61 million parameters and by following a similar approach with an ever deeper model architecture, the VGG11 [3] model requires 7.62 GOPs and includes 133 million parameters. The ResNet-50 [4] model architecture which utilizes the concept of residual connections requires 4.09 GOPs and includes 26 million parameters while the InceptionV3 model [5] that consists of sophisticated In-

ception modules [6] requires 5.71 GOPs and includes 27 million parameters.

However, the emergence of the edge computing paradigm requires novel approaches in the fields of CNN model architectures design and CNN acceleration. The advances in sensors technology and the developments in the field of the Internet of Things (IoT) has led to a significant increase in the amounts of data generated from devices at the edge of the network. The edge computing paradigm addresses the challenge of transmitting all these data to the cloud by processing them as close to the data sources as possible. This way, data transfers over the communication network which can hinder real-time response and can raise security issues are minimized. Following this paradigm, it is now possible to deploy the inference process of image processing CNNs to the edge of the network, close to the camera sensors, and thus enable computer vision applications which require real-time decision making based on deep learning.

1.2 Convolutional Neural Networks Acceleration for Edge Computing

The concept of deploying the CNN inference to the edge has opened up an entire research area regarding the hardware acceleration as well as the efficient optimization of CNNs. The following paragraphs introduce three key aspects for the hardware acceleration and optimization of CNNs towards their employment for edge computing applications that the current thesis focuses on.

The first aspect of CNN acceleration on the edge is the employment of specialized hardware accelerators. The powerful general purpose GPUs traditionally used for CNN training are not a suitable platform due to the spatial and power consumption constraints of embedded systems. On the other hand, embedded Central Processing Units (CPUs) lack the computational capabilities to handle the demanding CNN processing. Consequently, the research interest is focused on power-efficient embedded accelerators. Several different types of such accelerators exist, based on different technologies such as Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Arrays (FPGAs), GPUs, CPUs and which are integrated on System-on-Chips (SoCs) or System-on-Modules (SoMs). Each type of edge-oriented hardware accelerator offers different trade-offs between development flexibility, processing capabilities, interfacing options, power consumption and spatial footprint. A few notable examples of such deep learning accelerators targeting edge applications include, first, the Intel Movidius Vision Processing Units (VPUs) [7], namely the Myriad 2 and its successor the Myriad X. The Myriad 2 is a SoC designed for ultra low-power operation and its architecture is based on twelve Very Long Instruction Word (VLIW) 128-bit vector processors and two LEON 32-bit Reduced Instruction Set Computer (RISC) processors combined with a wide range of interfacing capabilities for integration with camera sensors. Another deep learning accelerator for

edge applications, in the form of an ASIC, is the Edge Tensor Processing Unit (TPU) from Google. The Edge TPU consists of a two-dimensional array of processing elements with each one of them having several processing cores and dedicated memory. Additionally, the Jetson Nano [8] is a low-power embedded computing board from NVIDIA, designed for accelerating deep learning applications. It integrates an ARM-based CPU and a 128-core NVIDIA Maxwell architecture GPU which serves as the main accelerator of the system. Finally, the Xilinx Ultrascale+ SoC FPGAs [9] combined with the Xilinx Vitis AI framework provide FPGA-based CNN acceleration on the edge. The ARM-based processor of the FPGA is integrated with the configurable logic on which the Deep Learning Processor Unit (DPU) is implemented.

The second aspect of CNN acceleration on the edge is the development of CNN model architectures matching the limited computational capabilities of the embedded hardware accelerators for deep learning applications on the edge. CNN models which target execution on the cloud, aim at achieving optimal performance with increased model architecture dimensions resulting in a significant increment in the number of parameters and operations required for the inference. In contrast, the CNN model architectures which are tailored to edge computing applications usually aim at reduced model parameters and number of operations in order to achieve real-time results while maintaining satisfactory performance on the corresponding task. A notable example of such lightweight CNNs are MobileNets [10] targeting mobile and embedded vision systems. In particular, MobileNetV2 [11] utilizes depthwise separable convolutions, linear bottlenecks and inverted residuals in order to minimize the computational cost and memory footprint of the model while making only small sacrifices in accuracy. SqueezeNet [12] is another notable CNN model architecture designed with embedded systems deployment in mind. It implements several architectural design strategies along with its Fire module consisting of squeeze and expand convolutional layers. Finally, ShuffleNet [13] is designed for mobile devices with very limited computing power. Its units take advantage of the pointwise group convolution with the channel shuffle operation for efficiency in computations.

The final aspect of CNN acceleration on the edge is the employment of optimization methods on an already defined CNN model architecture in order to reduce its computational complexity without significantly sacrificing its performance. One of the most notable optimization approaches is data quantization. The approach of data quantization reduces the precision of the data type used to represent the weights and activations of the model. By reducing the bit-width of the weights and activations, the memory requirements can be reduced and the hardware for the computations can be simplified resulting in faster inference speed and lower power consumption. Quantization of 32-bit floating-point weights to 8-bit integer ones with small accuracy losses is quite common while binarization is the most extreme form of quantization. An additional optimization approach is network pruning in order to reduce the number of memory accesses and operations. With weight pruning, weights which are identified to contribute very little to the final result are eliminated from the model. Alternatively, even larger scale pruning can be employed for entire filters or layers.

1.3 Thesis Contributions and Organization

The current doctoral thesis covers all three of the aspects described in the previous subsection. In particular, it focuses on several challenges regarding the optimization of CNNs towards their edge deployment as well as challenges in the field of CNN accelerator architectures design. The first challenge is to develop image pre-processing methods which can improve the accuracy of the CNNs results by introducing minimal computational and resources overhead. The second challenge is to design lightweight CNN model architectures in order to match the limited computational capabilities and resources of edge and on-board accelerators. The third challenge is to efficiently take advantage of the resources of CNN hardware accelerators in order to result in real-time processing capabilities for the target, deep learning-based, computer vision tasks. The fourth challenge is to design interfaces for embedded CNN accelerators in order to support high-throughput data transfers.

Based on the aforementioned challenges, the current thesis makes the following distinct contributions to each one of them:

1. The thesis proposes a novel image processing method, based on the accurate sun localization in the image, to improve the performance of image regression CNN models for the application of irradiance estimation and forecasting. It conducts a study on four different deep learning models which showcases the effectiveness of the proposed method that improves the accuracy of the irradiance values that the CNN models produce in all cases and by up to 13.75% for the MobileNetV2 model.

2. The thesis studies and designs a novel CNN model architecture to provide an improved trade-off between CNN model size and performance on the task of on-board semantic segmentation for cloud detection. The proposed model architecture is a lightweight variant of the U-Net model and its significance lies in the fact that it achieves competitive performance with reduced model size and number of operations when compared to the state-of-the-art.

3. The thesis studies the porting process of CNN models to an edge-oriented FPGA, using the Xilinx Vitis AI framework, to result in efficient acceleration of the CNNs in edge and on-board payload data processing applications. During the porting process, several data quantization solutions are explored and a distinct acceleration approach is showcased for accelerating different processes of a single task on the heterogeneous resources of the FPGA. The resulting accelerator showcases real-time processing rates for the applications of image regression for irradiance estimation and semantic segmentation for on-board cloud detection.

4. The thesis designs and develops, in VHSIC Hardware Description Language (VHDL), an interfacing system for image transfers between an FPGA and the Intel Myriad 2 VPU, to enable the next generation of on-board payload data processing architectures. The interfacing system is designed to be high-throughput and is shown to

achieve up to 2.4 Gbps duplex bit-rate on the extensive test campaign performed. The interfacing system also implements a fault-tolerance mechanism in order to support the fault mitigation strategy of the entire processing system.

The current thesis is organized in seven chapters. Apart from the current introductory chapter, Chapter 2 presents and discusses related works from the literature for the four distinct challenges that the thesis makes contributions on. In Chapter 3, the thesis elaborates on the proposed image processing method for the application of irradiance estimation and forecasting. The chapter presents results from the evaluation of the proposed method on four popular CNN models and showcases its effectiveness. Chapter 4 introduces the proposed lightweight CNN model for binary semantic segmentation for cloud detection in satellite imagery. The proposed CNN model is evaluated on a well-known test dataset and the achieved results are compared to the state-of-the-art, highlighting the corresponding improvements. In Chapter 5 the development methodology for porting CNN models to an FPGA, based on the Xilinx Vitis AI framework, as well as the acceleration approach is introduced. The evaluation results regarding both the quantization as well as the acceleration of models are presented and discussed. Chapter 6 provides an in-depth explanation of the design details of the high-throughput and fault-tolerant interfacing system between the FPGA and the Myriad 2 VPU. Moreover, the steps and corresponding results of the testing campaign are presented in detail. Finally, Chapter 7 sums up the current doctoral thesis and outlines potential directions for future research.

Chapter 2

Related Work

The current thesis addresses different challenges regarding CNN models optimization techniques and CNN accelerators design techniques towards deploying CNNs to different edge computing applications. The following subsections present related results in the literature organized based on the according deep learning challenges and applications that each chapter of the thesis focuses on. In particular, Section 2.1 discusses the related results in the literature regarding image processing methods and image regression deep learning models which target the application of irradiance estimation and forecasting. Then, Section 2.2 presents related works in the field of lightweight models for semantic segmentation of images in the context of cloud detection and towards the implementation of the models in SoC accelerators for on-board payload data processing systems. In Section 2.3, notable related works from the literature, with respect to end-to-end deep learning frameworks for CNN accelerator architectures targeting FPGAs are highlighted. Finally, Section 2.4 presents the current state-of-the-art in heterogeneous and mixed-criticality on-board payload data processing architectures, which target deep learning-based computer vision applications on-board, and how fault-tolerance at several levels of the architecture is implemented.

2.1 Image Processing & Image Regression for Irradiance Forecasting

Image processing methods towards irradiance forecasting can be applied on satellite imagery [14, 15, 16]. However, images obtained from fisheye lens (180° field of view) Sky Imagers (SIs) local to the Photovoltaic (PV) parks can provide increased spatial and temporal resolution. This fact can favor the precise irradiance forecasting for specific areas such as PV parks and for very short-term forecast horizons of up to 15 minutes.

Several works in the literature propose image processing methods to extract information regarding cloud coverage from sky images for irradiance forecasting [17, 18, 19, 20]. The authors of [17] used ground-based sky images statistical features such as the Red Blue Ratio (RBR) and Red Blue Difference (RBD) for cloud cover calculation. The authors also performed cloud cover forecasting using machine learning approaches such as Support Vector Regression (SVR) and Artificial Neural Network (ANNs). In [18] the authors introduced the Hybrid Thresholding Algorithm (HYTA) for cloud detection, which utilizes Normalized Red Blue Ratio (NRBR) image metrics and adaptive thresholding methods. The authors in [19] proposed a method for forecasting irradiance using the cloud cover index. They used the RBR to calculate the cloud cover index, a Long Short-Term Memory (LSTM) model for forecasting the future cloud cover index and a numerical solar radiation model for calculating future irradiance. The work of [20] used image processing methods to identify the cloud coverage and combined it with the clear sky index and LSTM models to perform irradiance forecasting. The current thesis differentiates to the above results, as it proposes an image processing method that precedes the corresponding CNN processing and provides information to CNNs regarding the position of the sun in the image instead of the cloud coverage features.

Regarding sun localization in sky images, different image processing approaches are reported in the literature [21, 22, 23]. In [21] the authors introduced a sun localization algorithm based on pixel values and a masking algorithm for background removal based on edge detection. Numerical NRBR values were generated from the processed sky images and these values were fed to a Multilayer Perceptron (MLP) network for irradiance forecasting. The work of [22] implemented and compared three different approaches for the identification of the sun position in sky images targeting nowcasting applications. The machine learning-based approach was shown to overcome the ones that are based on solar coordinates calculation and traditional image processing. The sun tracking algorithm introduced in [23] used image pixel values to identify the position of the sun when it is visible and an interpolation method when it is not. While the aforementioned works performed sun localization based on pixel values, the method which is proposed in the current thesis utilizes the solar angles and the mapping function of the fisheye lens to accurately calculate the position of the sun in the image regardless of any sky image effects such as clouds, high-intensity glares and background objects.

The reported results that are the most closely related to the current work are the ones which perform sky image regression using deep learning to produce irradiance values [24, 25, 26, 27, 28]. The work in [24] provided an in-depth comparison of deep learning model types for short-term irradiance forecasting from sky images. The four model types, namely CNN, CNN plus LSTM, 3D-CNN and Convolutional LSTM (ConvLSTM) were all shown to achieve around 20% forecast skill on the 10 minutes ahead prediction. The authors in [25] used the ResNet CNN model for performing irradiance regression and forecasting tasks. For forecasting, by stacking different RGB channels of past images on the input of the model they could achieve up to 18% forecast skill for the 10 minutes horizon. In [26] the authors proposed replacing costly irradiance measurement instruments, e.g.,

pyranometers with a CNN-based image regression model to produce irradiance values. They particularly focused on cloudy days that have the most sudden irradiance deviations and thus show the feasibility of their approach. The work in [27] investigated the performance of CNN and LSTM models on image regression for irradiance mapping. Apart from the deterministic methods it also applied probabilistic ones to statistically evaluate the performance of the models. The authors of [28] performed irradiance nowcasting from sky images by enhancing a traditional CNN with attention modules to improve its performance. It is worth noting that, all of the above works utilized datasets with a relatively limited number of samples of up to a few tenths of thousands. In contrast, the work performed in the current thesis utilizes the entire Folsom, CA open-source dataset which contains three consecutive years of data with more than 700,000 image samples. This allows to extract the evaluation results of the CNN models from an extensive test dataset of one whole year that includes indicative samples of all sky image effects. By examining the performance on individual months we quantify how the distributions of sky image phenomena affect the irradiance forecasting problem.

2.2 Semantic Segmentation for On-Board Cloud Detection

Several works in the literature focus on lightweight CNN models for on-board payload data processing tasks [29, 30, 31]. The authors of [29] proposed a series of increasingly smaller semantic segmentation models, based on the original U-Net model, for cloud extraction in RGB remote sensing images. The sizes of their models varied from 51K parameters to only 273 and offered attractive trade-offs regarding semantic segmentation performance and memory footprint. The most compact of the models was implemented on an Intel Altera SoC FPGA used in space missions. In the work of [30], the Refined U-Net Lite is proposed which aims at identifying edge-precise cloud regions from satellite imagery with reduced number of parameters. The Refined U-Net model achieves improved performance metrics compared to the the original U-Net with but with a higher execution time. The authors provide the Refined U-Net as an open-source implementation. Similarly, the authors of [31] introduce a lightweight U-Net for cloud detection of visible and thermal infrared remote sensing images. The proposed model achieved higher performance metrics than the original U-Net with a lower execution time for the inference of a single image on a CPU. The current thesis differs from the aforementioned works as it proposes a U-Net variant with limited computational cost and memory footprint which combines a large variety of CNN techniques such as depthwise separable convolutions, dilated convolutions and residual blocks in order to result in competitive semantic segmentation performance. Additionally, in the current thesis, the proposed U-Net variant model is ported and accelerated on an edge-oriented FPGA suitable for space applications.

2.3 Deep Learning Accelerator Frameworks for FPGAs

The literature contains several reports of frameworks for generating FPGA-based accelerators for CNNs. These frameworks accept a high-level CNN model description and generate a hardware architecture on the target FPGA for performing the computations of the model. The frameworks vary in terms of their key characteristics which include the interfacing with other high-level frameworks for accepting model descriptions, the generated hardware architectures, the supported CNN operations and the achieved latency among others.

One of the most important key characteristics of the frameworks for deploying CNNs to FPGAs, is the type of underlying hardware architecture that they generate. The types of architectures can be divided in two main categories, streaming architectures and single computation engine architectures. In streaming architectures, each layer of the CNN model is mapped to a distinct hardware block. The hardware blocks which are responsible for consecutive layers are highly pipelined in order to allow concurrent execution of the operations of multiple layers when possible. A notable example of a streaming architecture framework is the end-to-end fpgaConvNet [32, 33] framework. FpgaConvNet accepts as inputs trained CNN model descriptions from well-known frameworks, such as Caffe, together with the specifications of the target FPGA and utilizes Vivado HLS in order to generate a streaming architecture. The streaming architecture can be optimized for either latency-sensitive or throughput-sensitive applications and can also be re-configured during runtime in order to execute different subgraphs of a single CNN. Another notable end-to-end framework is the FINN framework [34, 35, 36]. Similarly to fpgaConvNet, it accepts model descriptions from PyTorch and Brevitas and generates High-Level Synthesis (HLS) blocks for individual layers of a CNN and then stitches them together to form the streaming hardware architecture. A key difference from other frameworks is that FINN particularly focuses on heavily quantized (up to 4-bit representations) and even binarized CNNs for which the computational blocks for operations such as convolution and pooling significantly differ from the conventional ones. In contrast to the above two frameworks, the current thesis opts for the use of the Xilinx Vitis AI framework which is based on a single computation engine hardware architecture. This approach results in flexibility advantages since an FPGA has to be programmed only once but can execute a wide variety of CNNs by utilizing executable instructions.

An alternative approach that many other frameworks follow is the generation of single computation engine hardware architectures. In this design approach, a unified architectural template can be configured based on the resources utilization requirements or the support of specific CNN operations and generate a single processing architecture which can support the operations of a variety of CNNs by means of executable instructions. An example of such a framework is Angel-Eye [37]. The single computation engine of Angel-Eye is based on an array of processing elements and an on-chip buffer which are controlled by a controller module responsible for decoding instructions received from the CPU of the

SoC FPGA which Angel-Eye targets. Angel-Eye can receive high-level model descriptions from the Caffe framework which then quantizes automatically in a dynamic way for the execution of each layer but for a pre-defined wordlength for the entire CNN model. DNNWeaver [38] follows an approach very similar to Angel-Eye where the computational engine is comprised by an array of processing units each one containing a datapath of more low-level processing elements. A key difference from Angel-Eye, is that DNNWeaver can be optimized based on the target CNN model description, provided by Caffe, and then only constrained by the FPGA resources utilization target. In terms of arithmetic precision, DNNWeaver supports both floating-point and fixed-point with dynamic number of fractional part bits as described by dedicated parts of the executable instructions. The Vitis AI single computation engine framework that the current thesis opts for, differs from the above frameworks by allowing the developer to provide model descriptions from a wider variety of more up-to-date frameworks such as TensorFlow 2.0 and PyTorch. Regarding the arithmetic precision, Vitis AI supports 8-bit fixed-point representations of weights and activations only, with automatically determined integer and fractional parts number of bits per layer. Additionally, it offers several different quantization solutions, each one tailored to the corresponding high-level framework for which the model description is provided, in order to reduce the effects of the quantization to the accuracy of the CNN model as much as possible.

2.4 Heterogeneous and Fault-Tolerant On-Board Payload Data Processing Architectures

Regarding heterogeneous on-board payload data processing architectures, the literature includes a number of related works [39, 40, 41]. The authors in [39] propose an avionics architecture which combines the Commercial Off-The-Shelf (COTS) SoC Myriad 2 for accelerating digital signal processing and a space-grade FPGA to ensure high dependability. Additionally to the architecture, the authors proposed a methodology and framework for the Myriad 2 SoC in order to enable advanced parallel programming which takes advantage of the underlying SoC hardware. The evaluation is performed on the task of accelerating pose tracking on the Myriad 2. In [40] introduce an on-board computing architecture with the Microsemi Smart Fusion 2 SoC FPGA serving as the primary control node of the system. The SoC FPGA is integrated with a GPU-based accelerator, the Nvidia Tegra X2/X2i, for computationally demanding tasks. The authors also address thermal concerns and explore radiation mitigation techniques including error correction code (ECC) memory, software mitigation techniques and material shielding. The work of [41] presents a heterogeneous computer solution called SpaceCloud iX5100. The solution includes an AMD accelerated processing unit (APU) SoC based on a GPU + CPU combination, paired with a Microsemi SmartFusion2 SoC FPGA for interfacing with on-board instruments. The authors also outline the possibility of including additional accelerators such as the Myriad 2 VPU. In contrast to the aforementioned works, the current thesis

focuses on the design of a high-throughput interface between an FPGA and the Myriad 2 VPU in the context of an on-board architecture which features the GR716 microcontroller, a Xilinx Kintex Ultrascale FPGA and three distinct Myriad 2 VPUs for image processing and deep learning tasks.

When it comes to the fault-tolerance aspect of the heterogeneous on-board payload data processing architectures, related works in the literature combine fault-mitigation techniques at different levels of these architectures [42, 43, 44]. In [42], the High-Performance Compute Board (HPCB) prototype platform is introduced which implements fault-tolerance at platform-level. The platform includes three Myriad 2 VPU accelerators which can operate in triple-modular redundancy mode. Moreover, a fault-tolerant supervisor, the GR716 microcontroller, is included which can perform voting on the processing results of the three Myriad 2 VPUs and monitor their status over a heartbeat signal. In [43], the authors addressed the challenges associated with mixed criticality applications executed on the Xilinx Zynq Ultrascale+ MPSoC. To do so they proposed an approach based on three distinct design pillars which include secure and non-secure chip partitions isolation, secure data exchange between partitions and an external supervisor, the rad-hard PolarFire FPGA, for recovery from critical faults. The authors of [44] introduce system-level hardening techniques which are explored in the context of the development of the Leopard data processing unit. The techniques include redundancy in the processing nodes, storage of critical software images in triple modular redundant memories, over-current protections and more. While the aforementioned works explore system-level fault-tolerance, the current thesis designs and implements in hardware, a fault-tolerance technique, based on CRC, for protecting payload data transfers between an FPGA and the Myriad 2 VPU which serves as the deep learning and image processing accelerator of the on-board computing system.

Chapter 3

Image Regression for Irradiance Estimation and Forecasting

The current chapter focuses on the computer vision application of solar irradiance estimation and forecasting, based on deep learning and CNNs. In the context of the thesis, deep learning image regression CNNs are utilized for irradiance estimation and forecasting on the edge and particularly as close to the camera sensor as possible by means of an embedded CNN accelerator. This edge computing application imposes limits on both the memory footprint and computational cost of the image regression CNNs as well as the power consumption of the embedded processing device. Still, it requires the high accuracy of the irradiance values produced. In order to address the above challenges, the current thesis proposes an image processing method tailored specifically to deep learning-based image regression for irradiance estimation and forecasting. The proposed method is based on sun localization in the image and when it is applied to the images before these are being processed by the CNNs, the results show that it can consistently improve the accuracy of the irradiance values that the CNNs produce by introducing only minimal computational overhead to the CNN models.

The remainder of this chapter is structured as follows. First, Section 3.1 provides the necessary background about deep learning-based image regression for irradiance estimation and forecasting for Photovoltaic (PV) applications. Then, in Section 3.2 the proposed methodology is described in detail, including both the image dataset analysis as well as the proposed image processing method. Finally, Section 3.3 presents and discusses the results from the extensive evaluation of the proposed image processing method when it is applied on several different image regression CNN models for both irradiance estimation and forecasting.

3.1 Background

The ongoing transition from traditional coal and fossil fuels to renewable energy sources, has led to solar Photovoltaic (PV) parks having an increased share in the energy production mix of many countries. In order to serve the needs of this energy transition, the PV production plants have to be integrated in the utility electrical grids or even in mini-grid and off-grid systems such as in autonomous islands. Moreover, the development of technologies in the fields of Artificial Intelligence (AI), computer vision and edge computing, combined with the smart grid concept [45] promotes on-site data intelligence in PV parks. In this direction, AI-enabled smart PV parks can contribute towards adapting the PV power production to the dynamic requirements of the grid [46].

Currently, the attempts to efficiently and reliably integrate PV production into the energy mix come up against the challenge of controlling its high intermittency and variability [47]. A source of this variability is the short-term meteorological effects and especially the dynamic changes of the cloud coverage over the PV facilities. The cloud features such as their thickness, their distribution in the sky and their position with regards to the sun, affect significantly the incoming solar irradiance, which is the most important factor for the PV power generation. The ability to forecast the short-term irradiance locally for the PV facility plays a key role in controlling the intermittence of the PV generated electricity [48]. However, the forecasting methods that are based solely on historical irradiance data cannot reliably model the short-term effects of cloud flows. On the other hand, the image analysis techniques and the computer vision-based methods can provide information about the state of the clouds in the sky with a high spatial resolution when the images are extracted from Sky Imagers (SIs) located closely to the PV panels [49]. Furthermore, the continuous advances in the field of machine learning and particularly deep learning and CNNs have led the modern computer vision tasks and applications to employ these deep learning and CNN techniques. The combination of the CNN-based sky image processing with the irradiance data of high temporal frequency (at least 1 minute) constitutes a very promising direction towards short-term (up to 15 minutes) forecasting, also called nowcasting, of irradiance [50]. The current availability of high quality and publicly accessible image datasets [51, 52, 53] is also in favor of the image-based deep learning approaches for irradiance forecasting. Finally, deep learning-based image processing is following the trend to move from the cloud servers towards the edge devices in order to enable smart applications and the Internet of Things (IoT) [54]. Edge devices can have resources and energy constraints and this has led to the development of lightweight CNN models such as MobileNetV2 [11] and SqueezeNet [12].

Regarding the formulation of the image regression for irradiance estimation problem it can be described as follows. Given a dataset that consists of pairs of a sky image X_t and a corresponding irradiance measurement I_t at time t, an image regression CNN model can be trained to produce a single value that represents an estimation of the irradiance $\hat{I}_t$ when provided with the sky image X_t. With the above formulation, the image regression for

irradiance estimation can be expanded to image regression for irradiance forecasting in two ways. First, assuming an external module that can predict the future sky image $\hat{X}_{t+\mathcal{H}}$ after time $\mathcal{H}$ in a sequence [55, 56, 57], the image regression CNN model can then produce the corresponding estimation of the irradiance $\hat{I}_{t+\mathcal{H}}$ from this image. This approach is illustrated in Fig. 3.1 Alternatively, an image regression CNN model can be trained on sequences of images where the corresponding irradiance values have been shifted backwards in time by $\mathcal{H}$. This way, when the model is provided with a sky image X_t at time t it will produce the irradiance forecast $\hat{I}_{t+\mathcal{H}}$. This alternative approach is shown in Fig. 3.2.

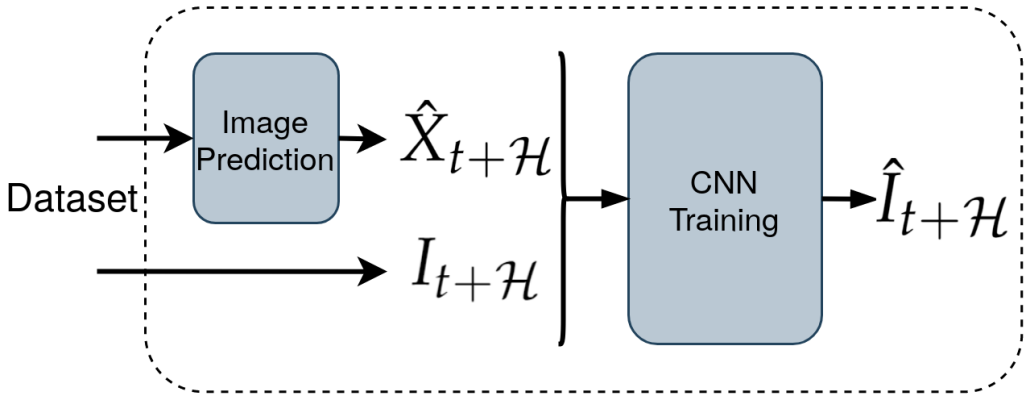

Figure 3.1: Irradiance forecasting using an image regression CNN with an external module for future sky image prediction.

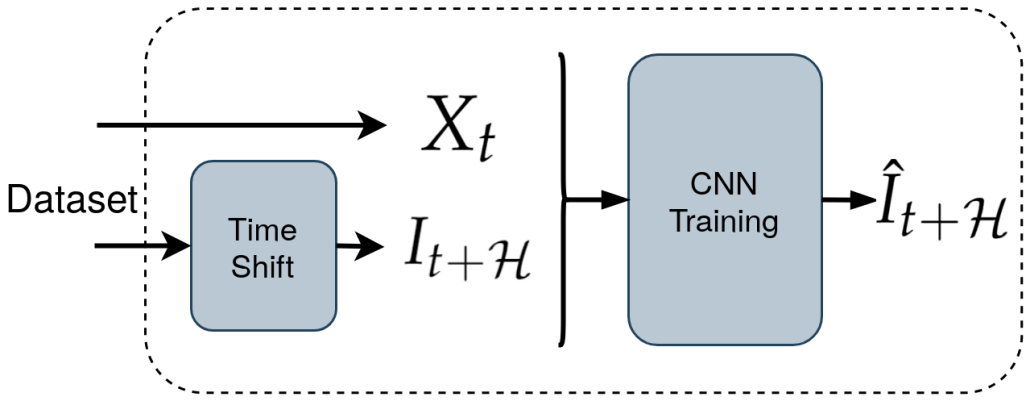

Figure 3.2: Irradiance forecasting using an image regression CNN with irradiance values shifted backwards in time by $\mathcal{H}$.

The objective of the tasks described above is to minimize the error between the actual I_t values and the $\hat{I}_t$ values produced by a CNN model after processing an image. For the evaluation of such models it is common to use the error metrics of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). In order to be able to compare the performance of a model on different subsets that can have largely different distributions

of irradiance values (e.g., a cloudy winter month vs. a clear sky summer one), we put emphasis on the Normalized Root Mean Square Error (nRMSE) which we calculate using the mean irradiance value, $\bar{I}$ of the set as

$$nRMSE = \frac{RMSE}{\bar{I}}. \tag{3.1}$$

Regarding forecasting, the Persistence Model (PM) serves as a reference forecasting model in order to benchmark other CNN models against it. It assumes that the predicted future irradiance value $\hat{I}_{t+\mathcal{H}}$ remains unchanged from the current irradiance value I_t into the forecast horizon $\mathcal{H}$ and can be expressed as

$$\hat{I}_{t+\mathcal{H}} = I_t. \tag{3.2}$$

In order to measure the improvement of any model against the persistence model, the Forecast Skill (FS) metric is introduced. It is common for the solar irradiance forecasting application to use the root mean square error of the persistence model, denoted as $RMSE_P$, and of the benchmarked model, $RMSE_M$, for the FS calculation as

$$FS = 1 - \frac{RMSE_M}{RMSE_P}. \tag{3.3}$$

3.2 Methodology

In the current section, the proposed methodology for the deep learning application of image regression for solar irradiance estimation and short-term forecasting is explained in detail. First, in Subsection 3.2.1 the dataset used for this application is described and analyzed, contributing the findings on which are the sky image features which can affect the performance of image regression CNN models. Then, Subsection 3.2.2 introduces the SunMask generation image processing method that the current thesis proposes. The proposed method aims at improving the accuracy of the irradiance results that the image regression CNNs produce, when it is applied to images before the corresponding CNN processing.

3.2.1 Folsom, CA Dataset Description & Analysis

For the development and evaluation of the proposed image processing method and CNN models, the current thesis utilizes the Folsom, CA dataset introduced in the work of [58] and publicly available in [52]. The dataset contains high-resolution 1536×1536 sky

images of three consecutive years, 2014, 2015 and 2016 taken at the Folsom, CA, USA site ($38.642°$, $-121.148°$). The images come with corresponding Global Horizontal Irradiance, Direct Normal Irradiance and Diffuse Horizontal Irradiance (GHI, DNI, DHI) measurements and have a high temporal resolution of 1 minute, making the dataset suitable for very short-term irradiance forecasting. The GHI describes the total irradiance from the sun on a horizontal surface including both the DNI and DHI terms and thus we focus on GHI for irradiance forecasting. So, the term irradiance will refer to the GHI for the remaining sections of this thesis. The first step towards adjusting the dataset to our application is to perform data cleaning. In particular, several images are missing corresponding irradiance measurements and vice versa. For this purpose, the first step taken is the rounding of the images timestamps to the closest integer minute value. Then, we match the images to the irradiance timestamps and formulate pairs that include both an image and a valid irradiance measurement. During the latter step we also remove all the samples that correspond to very low solar elevation values during the night. At this point we can perform the dataset analysis.

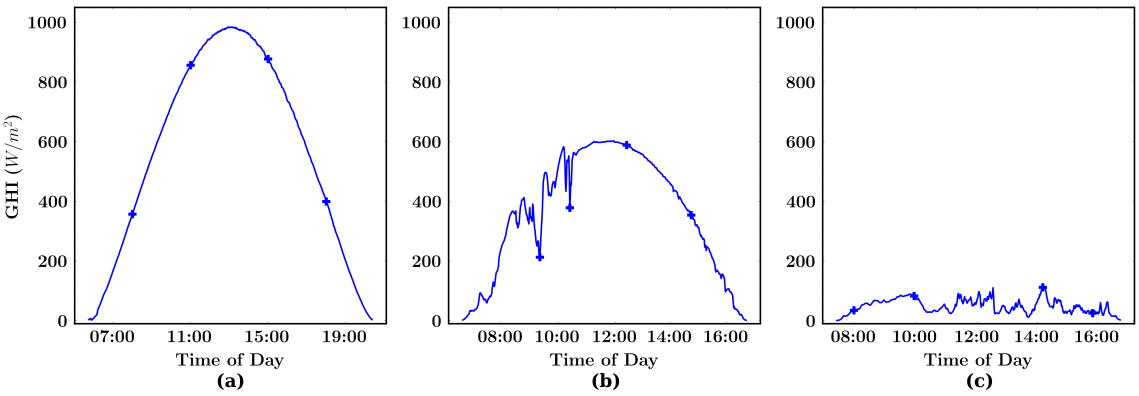

Figure 3.3: Daily GHI plots for the days **(a)** 7/6/2014 **(b)** 3/11/2014 and **(c)** 24/12/2014.

In order to distinguish the image features that affect the irradiance measurements, we first perform a daily analysis. A single day demonstrates the lowest degree of periodicity in sky image events, that is, the daily change of the sun's elevation angle. To perform the analysis, we transform the original dataset timestamps from the UTC+0 timezone to the Los Angeles timezone (UTC-7, UTC-8) that corresponds to the location of the sky imager in order to correctly identify the day and night periods. Fig. 3.3 presents daily GHI plots for three indicative days while in Fig. 3.4, Fig. 3.5 and Fig. 3.6 we show the sky images which correspond to the markers of each daily GHI plot. From the first day that includes only clear sky images, we observe that the position of the sun in the image and its distance from the image center is the feature that is directly correlated with the GHI measurement. The second day which features both clear sky and complex cloud effects in the images, indicates that when clouds obstruct the sun there can be fluctuations and sudden drops in the measured irradiance. Moreover, due to the larger distance of the sun from the center of the image during noon (lower elevation angle) the peak irradiance measurement is lower when compared to the first day. Finally, in the third day that is characterized by images

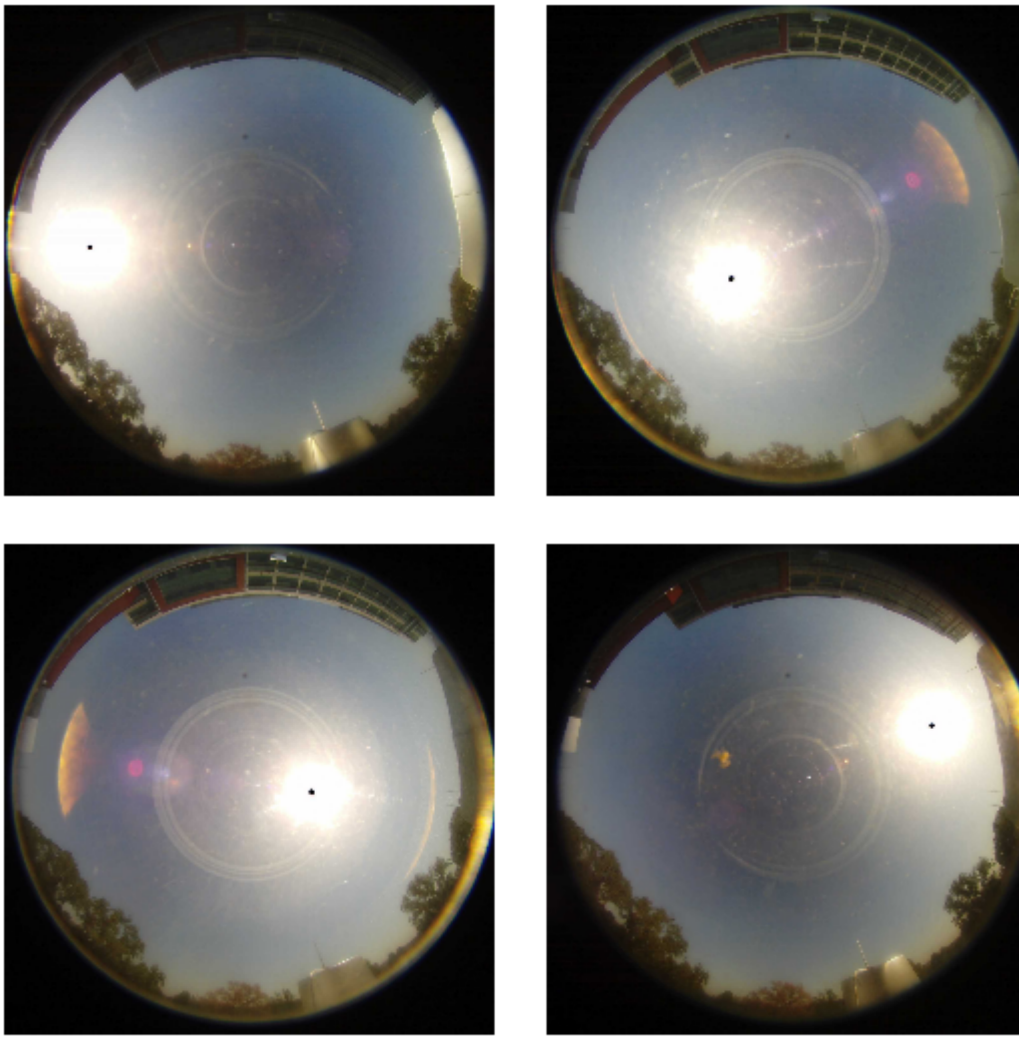

Figure 3.4: The sky images that correspond to the markers on the daily irradiance plot, from left to right and top to bottom, for the day 7/6/2014.

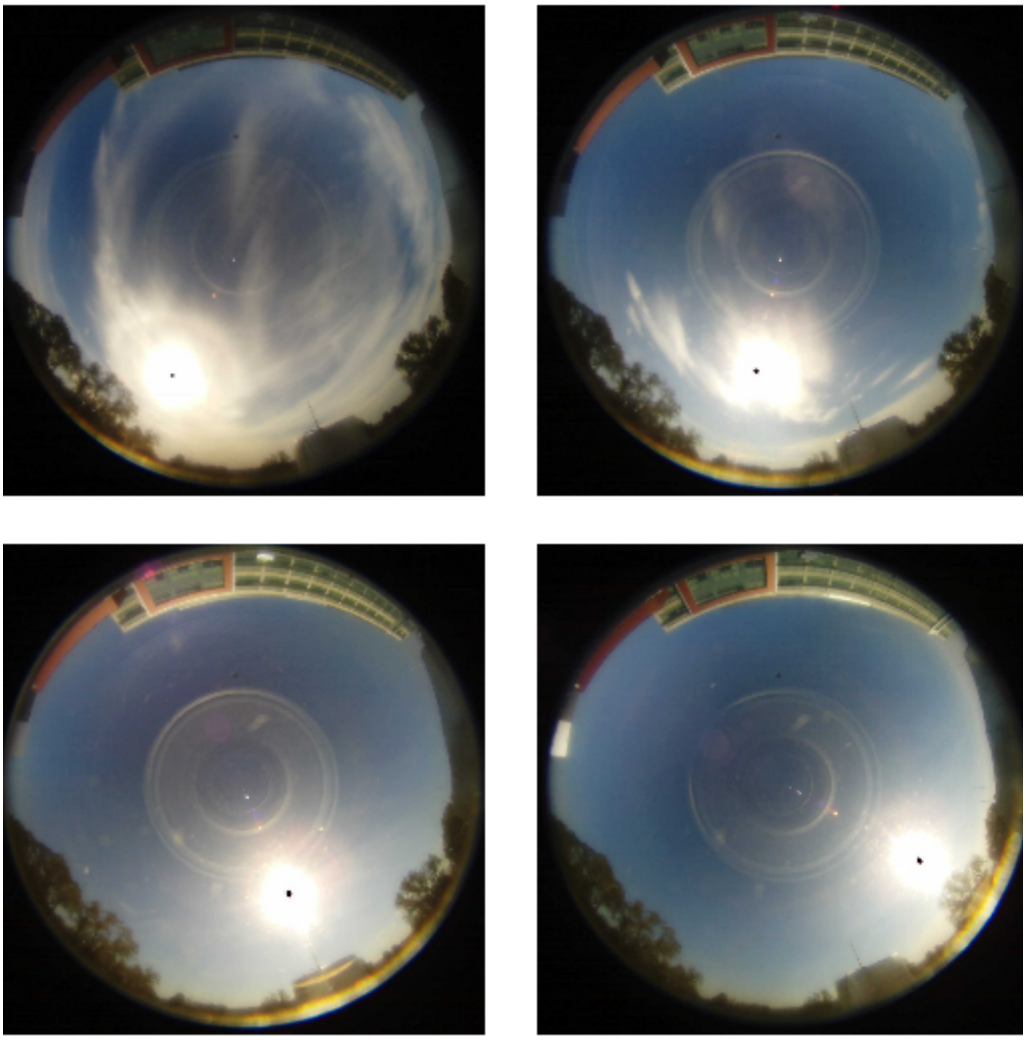

Figure 3.5: The sky images that correspond to the markers on the daily irradiance plot, from left to right and top to bottom, for the day 3/11/2014.

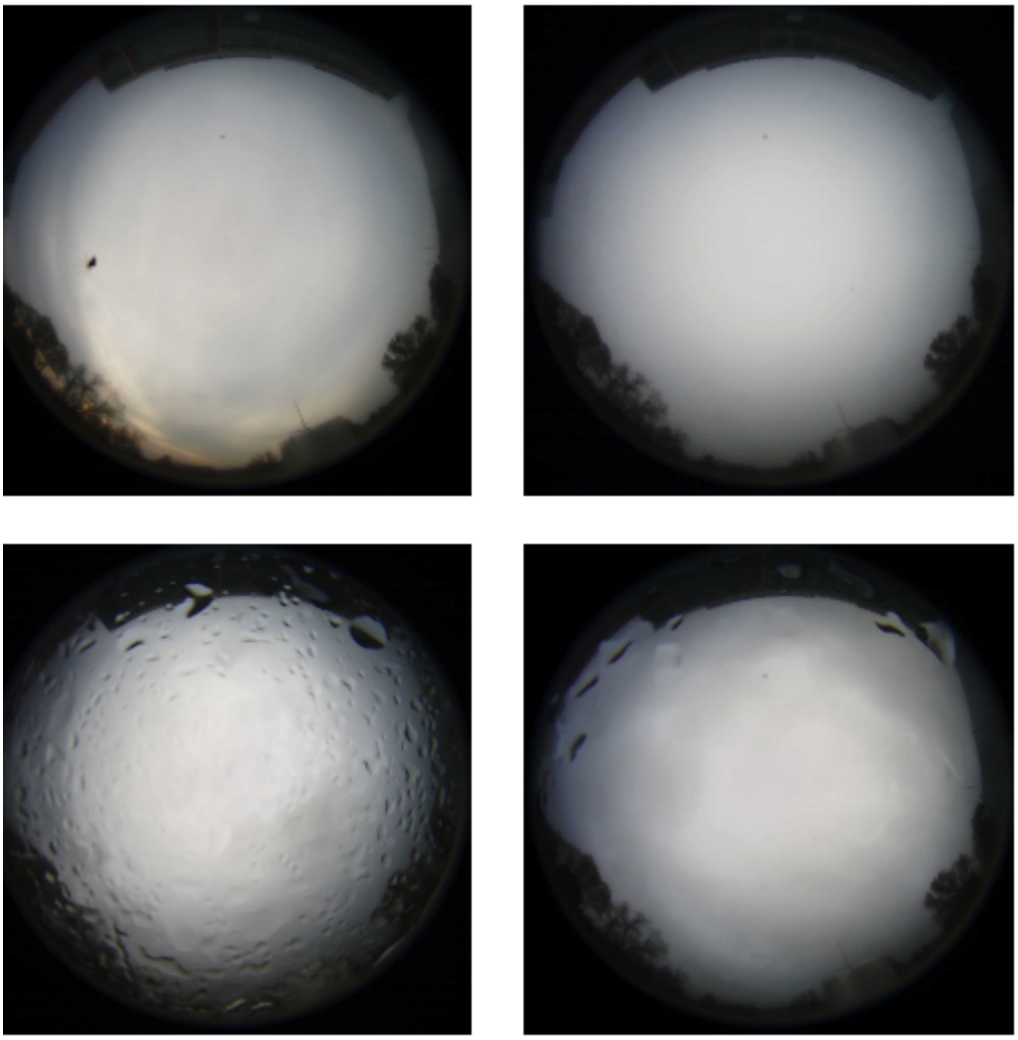

Figure 3.6: The sky images that correspond to the markers on the daily irradiance plot, from left to right and top to bottom, for the day 24/12/2014.

with only overcast sky, the measured GHI maintains consistent and very low values.

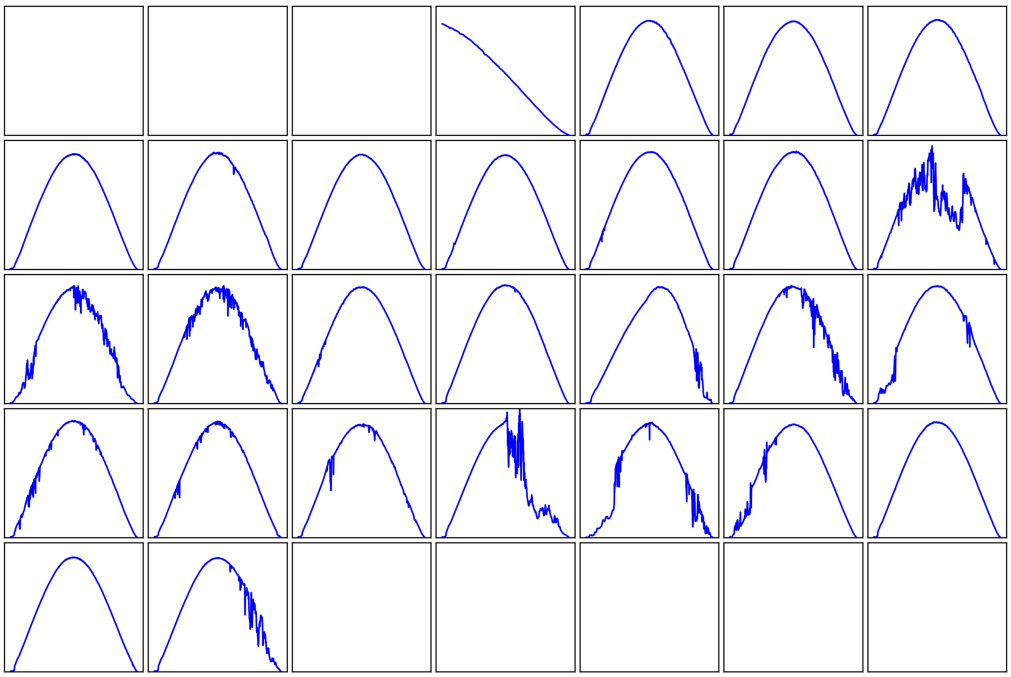

Figure 3.7: Daily GHI plot calendars of June 2014.

Based on the observations of the daily analysis, we expect that in the cases with complex cloud effects and overcast sky, the CNN models that perform image regression will produce a less accurate irradiance value, which will result in a larger nRMSE. The reason for this, is the lack of information regarding the position of the sun in the image which is the main image feature that the CNNs can model in order to produce an irradiance value. We note here that, clear days consistently occur in summer months while days with complex cloud effects occur during the rest of the year. We highlight this in Fig. 3.7 and Fig. 3.8 where we show all the daily GHI plots for June and March correspondingly, 2014 arranged in a calendar. Thus, we can quantify the effects of cloud phenomena on the performance of CNN models by evaluating them on different months that have different distributions of cloudy vs. clear sky days.

3.2.2 The SunMask Generation Image Processing Method

The current thesis proposes an image processing method to support the CNNs in providing more accurate irradiance estimations from sky images. The method is based on sun localization in the image. The intuition behind it is that the CNNs can model the position of the sun in the image and the cloud effects in the sun disk area in order to produce an irradiance value. The proposed method consists of locating the sun's center in the image and generating the SunMask, a circular mask around the center of the sun which we append

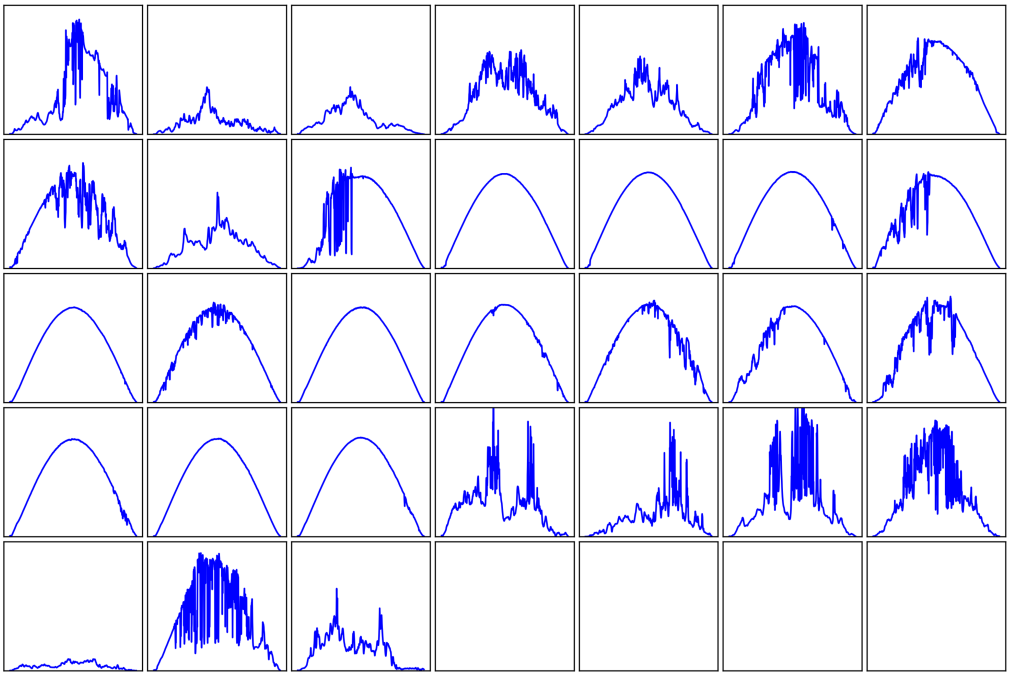

Figure 3.8: Daily GHI plot calendars of March 2014.

as an additional 4th channel to the original 3-channel RGB image. This method provides information to the CNN models about the position of the sun's center in the image that is particularly useful in images where the sun is covered by clouds. The steps of our image processing method are summarized in Fig. 3.9 and are explained in detail in the following paragraphs.

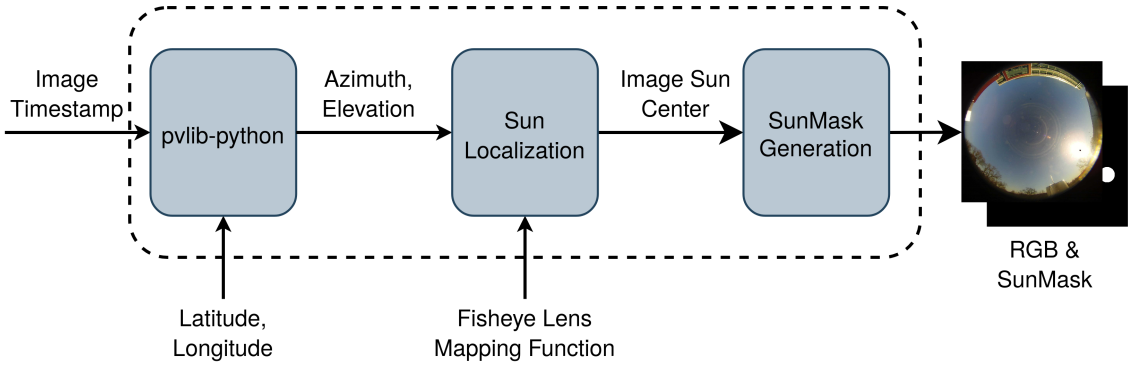

Figure 3.9: Summary of the steps of the SunMask generation image processing method.

The first step of the method, is to calculate the center of the sun disk in the image. Approaches that are based on the RGB values of the image often fail to identify the position of the sun when it is hidden by clouds. Furthermore, high intensity background objects and glare effects can also be erroneously identified as the sun. Due to the above, we opt

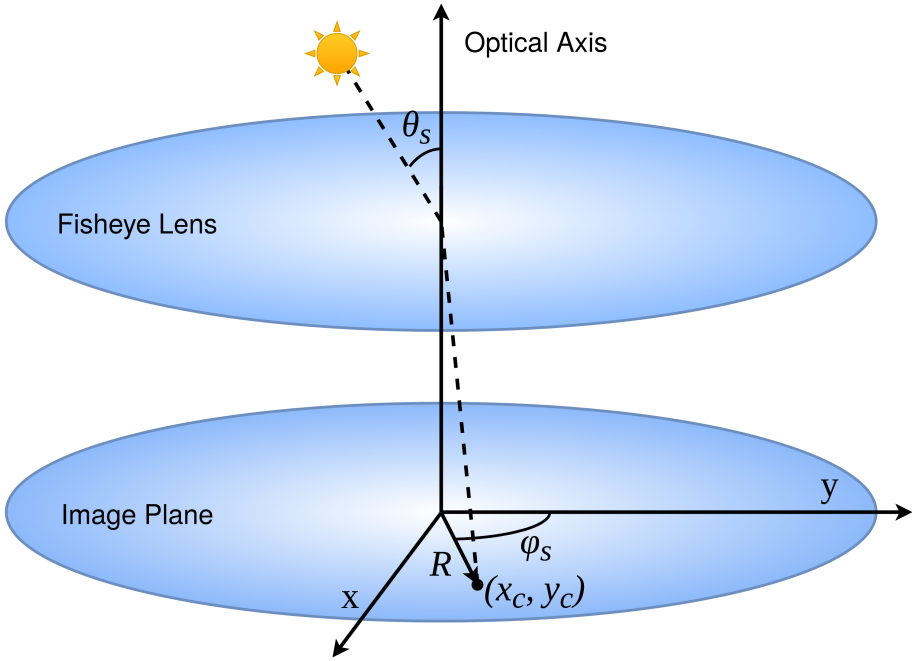

Figure 3.10: Overview of the sun localization approach on a sky image produced by fisheye lens.

for an approach based on the solar azimuth and zenith angles, ϕ_s and θ_s. The summary of our approach for sun localization is illustrated in Fig. 3.10. Given the timestamp of an image and the latitude, longitude coordinates we calculate ϕ_s and θ_s using pvlib-python [59]. Because of the fisheye distortion of the lens of the camera, simply projecting the solar coordinates on a flat surface does not result in accurate identification of the sun's center in the image. To address this, we use the mapping function of the lens to calculate the distance of the sun from the center of the image R, as a function of the focal length of the lens f, and the angle from the optical axis Φ. The mapping functions vary based on the type of projection (stereographic, equidistant, etc.) of the lens and this information, as well as the focal length, is not available for the camera of the Folsom, CA dataset used. Through trial and error, we identify that the stereographic projection with a focal length of $f = 0.48$ provides the most accurate results. Thus, we use the mapping function of the stereographic projection

$$R = 2f \tan \frac{\Phi}{2} \tag{3.4}$$

where we replace Φ with the solar zenith angle θ_s. After calculating the distance of the sun from the center of the image, we calculate the cartesian coordinates x_c, y_c using the solar azimuth angle ϕ_s with

$$x_c = R \sin(\phi_s), \tag{3.5}$$

$$y_c = R \cos(\phi_s). \tag{3.6}$$

In our case, we adjust ϕ_s with an angular correction of $165°$ to compensate for the ori-

entation of the camera. Finally, we transform the cartesian coordinates x_c, y_c to pixel coordinates x_p, y_p in the image by multiplying them with the image radius which is half the width of the image.

After calculating the position of the sun's center in the image, the last step is to generate the SunMask and append it as a 4th channel to the RGB image. The SunMask consists of a mask around the center of the sun with the pixel value 255 while the rest of the pixels have the value 0. In Fig. 3.11 we showcase the results of the proposed image processing method for indicative images where we color the SunMask with orange. We observe that the proposed sun localization method can accurately identify the position of the sun in all the images which vary in terms of solar azimuth angle, zenith angle and cloud coverage of the sun.

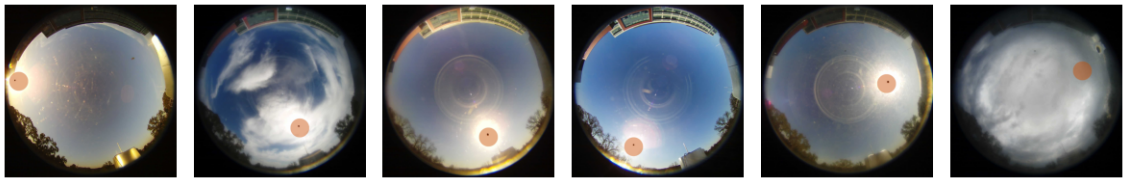

Figure 3.11: The SunMask, in orange color for visualization purposes, for a few selected image samples.

Based on the above, we apply the two steps of our image processing method to the entire Folsom, CA dataset and we provide 4-channel images during training and testing of image regression CNN models for irradiance estimation. When the image regression CNNs are configured to perform irradiance forecasting standalone, by being trained on backwards shifted GHI data, we perform one additional image processing step. Instead of providing only one 4-channel image (RGB & SunMask), denoted as X_t, we stack two additional 4-channel images, $X_{t-\mathcal{H}}$ and $X_{t-2\mathcal{H}}$ which correspond to 2 steps backwards in the forecast horizon $\mathcal{H}$. As a result, the CNNs now operate on an input of 12 channels in total. For example, if the image regression CNN is trained to perform forecasting 5 minutes ahead, we provide as input the images X_t, $X_t - 5$ and $X_t - 10$ and their corresponding SunMasks. This way, the CNN model has additional information regarding the past state of the sky.

3.3 Evaluation & Results

In the current section, the thesis presents the evaluation results of the studied CNN models and the proposed SunMask generation image processing method on the task of image regression for irradiance estimation and forecasting. We study the performance of the VGG11 [3] and ResNet-50 [4] models for which similar comparative studies have been conducted on various applications [60, 25]. We also study the performance of two

CNN models which target edge devices with limited resources, the MobileNetV2 [11] and SqueezeNet [12] models. With this selection of models we cover a wide range of model sizes and number of operations as shown in Table 3.1. The models and all training and evaluation processes are implemented in Python using PyTorch on a Linux workstation with the Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz and the NVIDIA GeForce RTX 3080 GPU.

Table 3.1: Model size and number of operations for the four models of the study.

Model	**# Parameters**	**# OPs (Mult-Adds)**
VGG11	128.77 Mil.	2.57G
ResNet-50	23.51 Mil.	1.33G
MobileNetV2	2.23 Mil.	0.10G
SqueezeNet	0.74 Mil.	0.23G

First, we train the four models of the study to perform irradiance estimation from sky images. The training dataset includes the years 2015 and 2016 (522320 samples) while the test dataset is the entire 2014 (240944 samples). During training, we use the Mean Square Error (MSE) loss function which is suitable for image regression tasks. The hyperparameters are tuned based on the RMSE result of the trained models on the entire test dataset. The tuning is performed for the ResNet-50 model and the hyperparameters are kept the same for all the training procedures in the work in order to keep the experimental environment consistent. All the models are trained for 10 epochs which were identified to be enough since all models showcased overfitting after only a few epochs. This can be attributed to the high number of training steps during each epoch due to the large size of the training dataset. Regarding the batch size and the image size these were limited to 16 and 128×128 correspondingly due to GPU memory limitations and training time required. Regarding the learning rate, it is initialized to 10^{-3} and it is automatically tuned by a scheduler that reduces the learning rate by a factor of 0.75 if the validation loss has plateaued for 5 epochs.

The training dataset is split to the training subset and validation subset for the evaluation of the models during training. Instead of a random split, we select all the samples of one random day of each consecutive month of the training dataset and add them to the validation subset until we reach the desired training-validation split ratio, e.g., 80-20%. This way, the validation subset includes indicative samples of the entire dataset in terms of the yearly periodical phenomena in sky images. Furthermore, we avoid including in the validation subset samples that are only 1 minute apart from almost identical ones in the training subset. We evaluate the performance of the models after every training epoch and select the saved model with the minimum validation loss to avoid overfit.

Table 3.2, presents the performance evaluation results of the four different models

on the original dataset as well as on the dataset with the images enhanced with the SunMask channel. The ResNet-50 model appears to result in the lowest error metrics with an RMSE of $64.83 W/m^2$. When the proposed SunMask generation method in introduced, we observe that it consistently improves the performance of all models. The MobileNetV2 model is favored most by our SunMask generation method that decreases its RMSE from $75.95 W/m^2$ to $65.51 W/m^2$, a 13.75% improvement. When it comes to the number of operations overhead that the additional SunMask channel introduces to the CNNs it is evaluated and determined to be only minimal. In particular, when compared to the number of operations of Table 3.1, there is a 0.4% increase for the VGG11 model, a 1.5% increase for the ResNet-50 model, a 1.2% increase for the MobileNetV2 model and a 7.1% increase for the SqueezeNet model.

Table 3.2: Evaluation and comparison of the four models of the study. SM indicates that the models are trained and evaluated with 4-channel images which include the SunMask.

Model	**RMSE** (W/m^2)	**nRMSE (%)**	**MAE** (W/m^2)
VGG11	65.25	15.88	38.31
VGG11$_{\text{SM}}$	**59.03**	**14.36**	**32.93**
ResNet-50	64.83	15.77	37.23
ResNet-50$_{\text{SM}}$	**60.31**	**14.67**	**36.02**
MobileNetV2	75.95	18.48	47.74
MobileNetV2$_{\text{SM}}$	**65.51**	**15.94**	**39.53**
SqueezeNet	70.18	17.08	44.65
SqueezeNet$_{\text{SM}}$	**62.93**	**15.31**	**38.56**

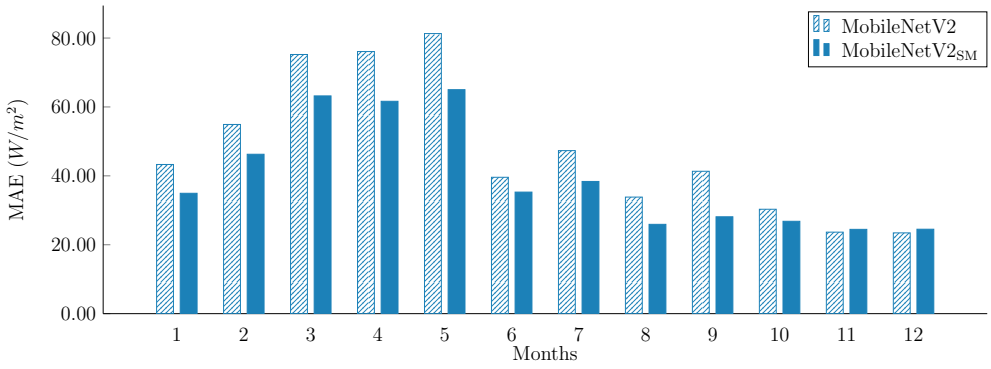

Figure 3.12: Monthly MAE barchart for MobileNetV2, with and without the SunMask.

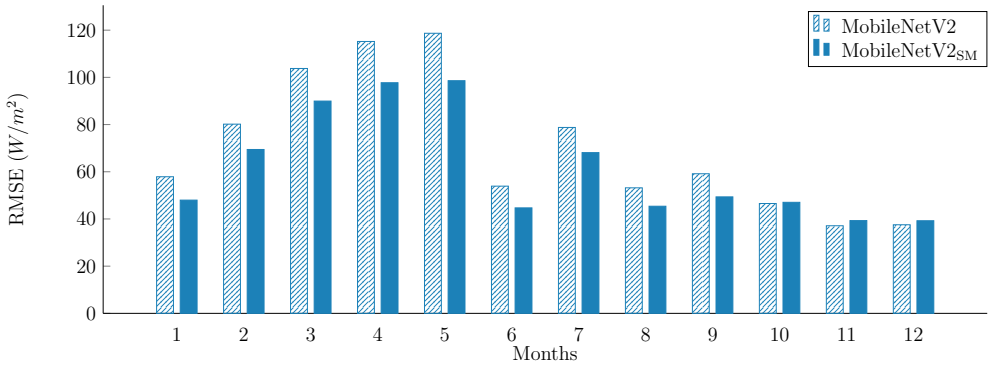

Figure 3.13: Monthly RMSE barchart for MobileNetV2, with and without the SunMask.

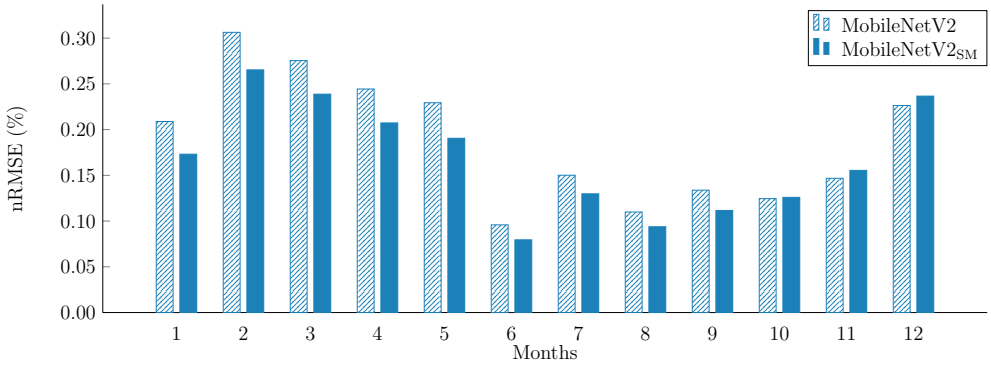

Figure 3.14: Monthly nRMSE barchart for MobileNetV2, with and without the SunMask.

In order to quantify the effect that the SunMask generation method has on different distributions of sky image phenomena, we evaluate the MobileNetV2 model, which shows the largest improvement with the SunMask, on individual months and we present the MAE, RMSE and nRMSE results in Fig. 3.12, Fig. 3.13 and Fig. 3.14 respectively. We note here, that in order to directly compare the performance of the models across different months we focus on the nRMSE metric. The reason for this is that, the distribution of the sky image features and irradiance values is different for each month. Thus, a month with a smaller RMSE can correspond to a larger nRMSE and vice versa and this is apparent in many months such as February, May and December. The nRMSE plot indicates, that in general, the model tends to perform better in months when there are less complex effects of clouds obstructing the sun in the image such as June and August. Furthermore, the improvement in the performance due to the SunMask is larger in months with complex sky image phenomena. In particular, MobileNetV2 shows the largest nRMSE improvement with the SunMask in the months February (4.1%), May (3.9%), March (3.7%), April (3.7%) and January (3.6%). The current work focuses on the dataset generated at the location of Folsom, CA, USA ($38.642°$, $-121.148°$). However, it would be worth exploring the performance of image regression CNNs for irradiance estimation and of the proposed SunMask generation method on more geographic areas. These geographic areas could include regions with significantly different sky image features distributions such as higher latitude regions where the sun is close to the horizon. Of course, this would require an extensive, publicly available and high-quality dataset for the particular area. The availability of such extensive, annotated and public datasets is currently an open issue in the field of deep learning-based irradiance forecasting and machine learning in general.

Following the results for irradiance estimation, we select the ResNet-50 model which was the one with the best performance, to perform standalone irradiance forecasting. For this purpose, we shift the irradiance values of the dataset backwards in time by the forecast horizon $\mathcal{H}$ as explained in Section 3.1. We formulate the dataset in this way for three different forecast horizons, 5, 10 and 15 minutes ahead. We train the ResNet-50 model similarly to before, with the simple RGB input and the 12-channel stacked SunMask images described in Section 3.2.2. The results of the ResNet-50 model on image regression for irradiance forecasting and the persistence model are shown in Table 3.3. We observe that the ResNet-50 model achieves a forecast skill which is incremental with regards to the forecast horizon, having slightly worse forecasting performance than the persistence model for the 5 minutes ahead forecasting. When ResNet-50 is trained to operate on the stacked SunMask images that we have proposed, the results show that it achieves consistently improved forecast skill for all forecast horizons. With this method the ResNet-50 model can now surpass the persistence model even in the very short-term forecast horizon of 5 minutes, adding 8.79% to its forecast skill.

Table 3.3: Standalone irradiance forecasting results for the ResNet-50 model on 5, 10 and 15 minute horizons. SSM indicates that the input is 3 stacked images of 4 channels each (RGB & SunMask).

Horizon	**Model**	**RMSE** (W/m^2)	**nRMSE (%)**	**FS (%)**
5-min	Persistence	72.64	17.32	-
	Resnet-50	73.18	17.44	-0.75
	Resnet-50$_{\text{SSM}}$	**66.79**	**15.93**	**8.04**
10-min	Persistence	86.77	20.26	-
	Resnet-50	78.06	18.23	10.04
	Resnet-50$_{\text{SSM}}$	**73.06**	**17.06**	**15.80**
15-min	Persistence	94.52	21.67	-
	Resnet-50	80.87	18.54	14.44
	Resnet-50$_{\text{SSM}}$	**75.78**	**17.37**	**19.83**

Chapter 4

Semantic Segmentation for Cloud Detection

The current chapter focuses on the application of deep learning-based semantic segmentation for on-board cloud detection in satellite images. Particular focus is placed on binary semantic segmentation of images generated from on-board sensors with multiple spectral bands. This kind of application is limited by the power budget of the satellite. Thus, accelerators of limited processing capabilities are employed which in turn impose constraints in the memory footprint and computational cost of the image segmentation CNN models. At the same time, it is important for the CNNs to maintain satisfactory performance on the task of cloud detection as images with significant cloud cover could be discarded. Aiming at contributing a CNN model with an attractive trade-off between computational requirements and semantic segmentation performance, the current thesis proposes a novel, lightweight variant of the U-Net semantic segmentation model architecture. The design and development of the proposed model architecture employs several state-of-the-art CNN techniques and the results show that the proposed model can achieve a competitive trade-off between performance and model size when compared to other state-of-the-art models from the literature on the same task and test dataset.

The remainder of this chapter is structured as follows. First, Section 4.1 provides the background about the application of interest. Then, Section 4.2 elaborates on the proposed methodology including both the satellite imagery dataset used as well as the details of the proposed CNN model architecture design. Finally, in Section 4.3 the results from the evaluation of the proposed CNN model architecture are presented and a comparative study with other lightweight U-Net variants from the literature is conducted.

4.1 Background

The ongoing advancements of small form factor satellites have raised the interest in Earth Observation (EO) applications. The corresponding on-board payload data processing and particularly, image processing tasks show increased computational demands due to the larger volumes of data generated by the advanced sensors and also, due to the complexity of AI and machine learning algorithms. Among these, the semantic segmentation of satellite imagery is considered as a classic computer vision task with most solutions based on CNNs. In particular, cloud related features extracted with CNNs from satellite images can provide useful information regarding climate, weather and natural disasters [61]. Furthermore, images with increased cloud coverage carry little information about the earth's surface and thus can be discarded in order to save on valuable on-board resources such as storage space, power and downlink bandwidth.

Due to the computationally demanding nature of the CNN models, the designers of on-board systems focus on more powerful processing platforms than the space-oriented general-purpose CPUs. Moreover, in order to match both the wide variety and resource demanding nature of deep learning-based computer vision tasks, they include heterogeneous SoC embedded architectures as accelerators complementary to general-purpose CPUs. Such SoC accelerators include VPUs, TPUs and SoC FPGAs. These SoCs offer attractive processing capabilities and development flexibility at reduced Size, Weight, Power and Cost (SWaP-C) [62] compared to the CPUs and GPUs available for terrestrial applications. Given though the limited and heterogeneous resources of these SoCs, the developers need to focus on compact CNN models for semantic segmentation and the utilization of diverse techniques for efficiently mapping the models onto the resources of the SoC.

4.2 Methodology

In this section, the proposed methodology for the design of the lightweight semantic segmentation CNN model architecture is presented. First, in Subsection 4.2.1, the dataset used for the development and evaluation of the model is explained in detail. Then, in Subsection 4.2.2 the proposed lightweight CNN model architecture is introduced and the CNN techniques employed for its development are analyzed.

4.2.1 95-Cloud Dataset Description

For the training and the evaluation of models on the task of semantic segmentation, we use the 95-Cloud dataset [63] which is an extension to the 38-Cloud dataset [64] and is introduced in [65]. 95-Cloud is an extensive and open source dataset for the task of detection of clouds, which is an important step in many remote sensing applications. Its purpose

is to help researchers to evaluate their deep learning-based cloud segmentation models. The dataset comes from data generated by the Landsat 8 satellite which is equipped with two sensors, the Operational Land Imager (OLI), capable of capturing data in nine spectral bands and the Thermal Infrared Sensor (TIRS), capturing data in two spectral bands. 95-Cloud includes only four spectral bands, corresponding to the Red, Green, Blue and Near-Infrared.

The dataset contains 95 scenes and their manually extracted pixel-level Ground Truths (GTs) where each scene has a size ~80 Mpixels. Four selected sample scenes from the 95-Cloud dataset including the natural color images generated from false color images for visualization purposes, and the pixel-level ground truth masks, are shown in Fig. 4.1. Due to the large spatial size of the scenes, each one is divided to non-overlapping patches of size 384×384 and corresponding GTs. As a result, the well-defined 95-Cloud training set consists of 75 scenes which correspond to 34,701 patches while the test set of 20 scenes includes 9,201 patches.

4.2.2 Semantic Segmentation Model: LD-UNet

CNNs used for semantic segmentation often adopt an encoder-decoder model architecture In these models, the encoder extracts features from the input images while downsampling them. Then, the decoder performs upsampling of the extracted features to generate the segmentation masks with the same dimensions as the original image. The U-Net [66] is one of the most well-known CNNs used for semantic segmentation that adopts the encoder-decoder model.

The current thesis starts with the baseline U-Net model and sets the following two objectives. First, to meet the limited processing capabilities, memory resources and power budget of an on-board processor. Second, to result in state-of-the-art CNN performance metrics on semantic segmentation benchmarks. Aiming at an effective contribution to both the aforementioned objectives, this work proposes the Lightweight Dilation UNet (LD-UNet) model illustrated in Fig. 4.2. LD-UNet is a significantly downsized version of the baseline U-Net, which targets the reduced computational cost and memory footprint of the model while it maintains competitive performance on the 95-Cloud test dataset.

The main features of the LD-UNet model are as follows. First, we remove all skip connections between the encoder and the decoder of the original U-Net. The skip connections increase the number of input channels to the convolutional layers of the decoder and thus, introduce both a computational overhead as well as a memory overhead for temporarily storing the encoder output feature maps until it is time to perform the corresponding decoder operations. Additionally, the skip connections contribute only little to the performance of the model due to the limited representational capabilities required for the binary semantic segmentation problem. Second, we reduce the downsampling and upsampling stages in the encoder and the decoder of the original U-Net respectively. While U-Net

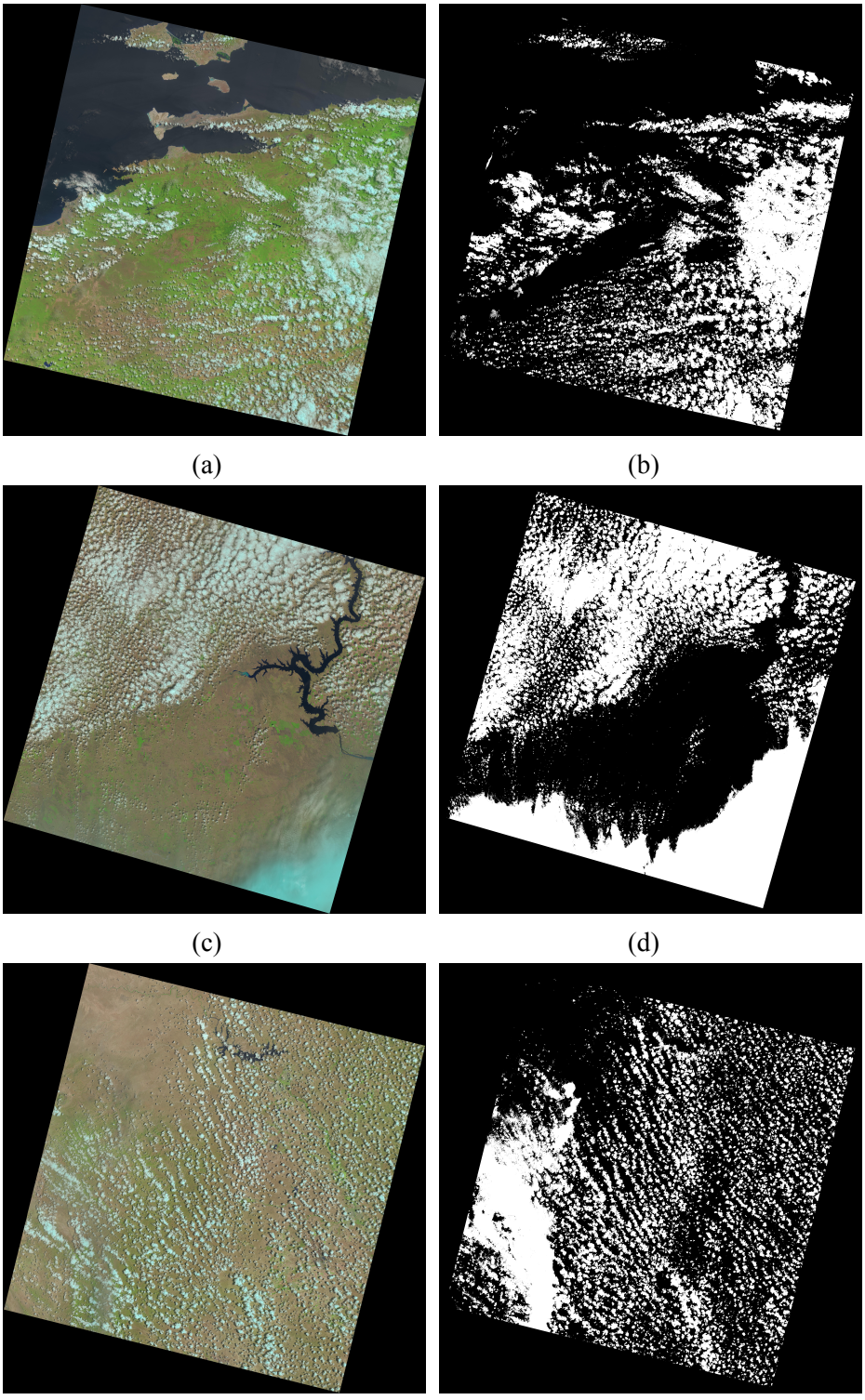

Figure 4.1: Three sample scenes from the 95-Cloud dataset with (a), (c), (e) the natural false color images and (b), (d), (f) the corresponding pixel-level ground truths.

included four upsampling and downsampling stages we reduce these to only two in LD-UNet, effectively reducing the number of convolutional layers which results in smaller model size as well as less number of operations required. Finally, for each stage, the original U-Net included a block with two consecutive convolutional layers. The current thesis proposes the design of the custom Dilation Residual Block that replaces the standard convolution blocks in U-Net. The custom Dilation Residual Block is shown in Fig. 4.3. It employs a number of techniques for reducing the number of operations required and but at the same time increasing its representational ability, which are explained in detail in the following paragraph.

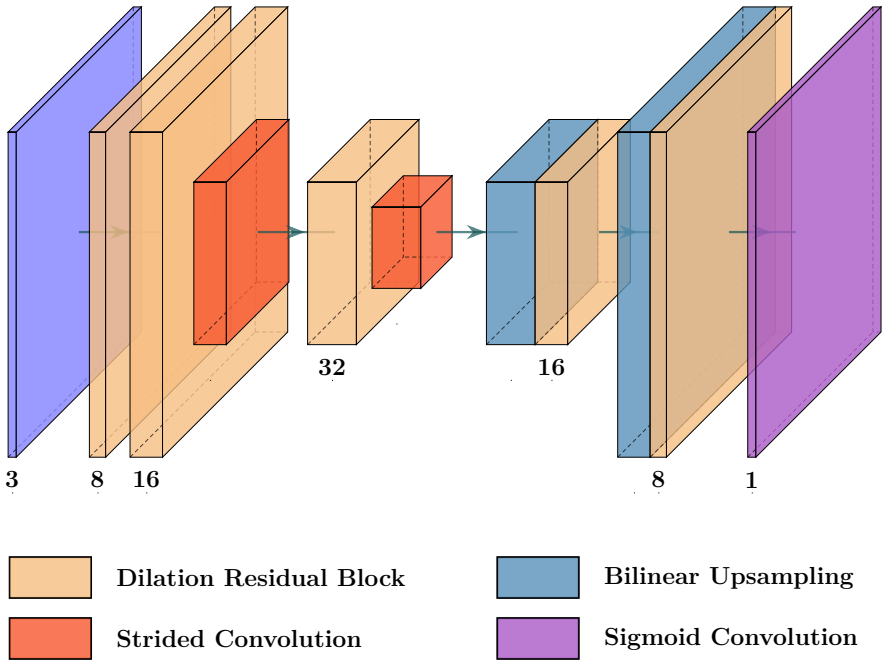

Figure 4.2: The LD-UNet model. The numbers below each layer indicate the number of output feature maps.

The Dilation Residual Block consists of two layers which perform depthwise-separable convolutions. Depthwise-separable convolutions require significantly less operations for processing the same number of input and output feature maps when compared to the standard 2D convolutions. The first depthwise-separable convolution involves a dilated depthwise convolution with a 3×3 kernel, a dilation rate of two and a stride of one. The dilated convolution is employed in order to increase the receptive field of the convolution by effectively increasing the size of the region of the input feature map which produce a single point of the output feature map. The second depthwise-separable convolution involves a depthwise convolution with a 1×1 kernel in order to produce feature maps which have the same dimensions as the ones produced by the first depthwise-separable convolution layer. This way the second depthwise-separable convolution serves as the identity shortcut of the Dilation Residual Block. Finally, the output of both depthwise-separable convolutional

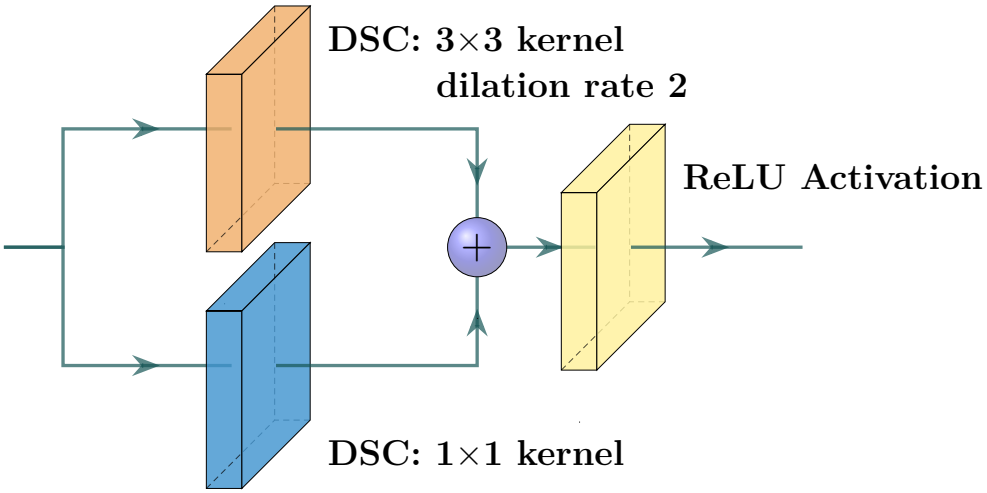

Figure 4.3: The custom Dilation Residual Block with the two Depthwise-Separable Convolutions (DSCs).

layers is added and followed by a ReLU activation layer. Further to the Dilation Residual Block, we replace the max pooling operations of the original U-Net encoder with strided convolutions. This technique introduces learnable parameters in the downsampling process of the encoder. Regarding the decoder, the transpose convolution operation is replaced by a bilinear upsampling layer followed by our custom block. The final layer of the LD-UNet model architecture is a standard 2D convolution with the sigmoid activation function.

4.3 Evaluation & Results

The LD-UNet model is trained and evaluated for the task of binary semantic segmentation on the 95-Cloud dataset which contains 95 scenes. Due to the large spatial size of each scene, the 95-Cloud dataset is divided to non-overlapping patches of size 384×384 pixels. The training set consists of 34,701 patches extracted from 75 scenes and the test set consists of 9,201 patches extracted from 20 scenes.

For the LD-UNet, only the three of the spectral bands provided, namely the RGB ones are used. The input patches are resized from 384×384 down to 128×128, a decision that significantly reduces the number of operations by a factor of up to ~ 9 per convolution operation, with little effect on the performance of the model. Due to the orientation of the 95-Cloud scenes, a large number of patches extracted contain mostly or only zero-valued (black) pixels. In order to maintain the balance of the training dataset, all patches with more than 80% zero-valued pixels are eliminated. All patches are pre-processed by normalizing their pixel values in the $[0 - 1]$ range.

The 80% of the training dataset is utilized as the training set for the LD-UNet model and the remaining 20% is utilized as the validation set for evaluating its performance during training. The split between the training and the validation set is performed in a random way. For the loss function, the Dice loss function is utilized which is suitable for performing image segmentation tasks on unbalanced datasets such as 95-Cloud. LD-UNet is trained with a batch size of 24 for 200 epochs with an adjustable learning rate. After training, the model which corresponds to the epoch with the minimum loss value on the validation set is selected in order to avoid overfit.

To perform inference on the unknown test scenes, we resize each 384×384 input test patch to 128×128 and feed it to the network. During inference only, the output of the network is binarized to 0s and 1s using the threshold value $12/255$ as in [67]. The output segmentation mask of each patch of 128×128 pixels is then upsampled to 384×384 using bilinear upsampling. All the output patches corresponding to a single ground-truth test scene are stitched together to generate the segmented test scene on which the performance metrics are calculated.

In order to extract the performance metrics of LD-UNet on the test set, first the confusion matrix of the model is formulated. To do this, first, the number of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values that the model produces, compared to the ground truth values, are calculated on the entire test set. Then, these numbers are normalized to a percentage based on the total number of values in the test set and the result is illustrated in the normalized confusion matrix of Fig. 4.4. The confusion matrix validates that the LD-UNet model can achieve satisfactory performance with very little FPs and FNs. The imbalance between the TPs and TNs can be attributed to the unbalanced nature of the dataset where there are many more non-cloudy (zero-valued) pixels that cloudy (one-valued) ones.

Based on the values of the confusion matrix, the performance metrics of LD-UNet on the test set of 95-Cloud can be calculated. The performance metrics considered in the current thesis for the evaluation of the model are the Accuracy, the Intersection over Union (IoU), the Recall and the Precision metrics. They are defined as

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.1}$$

$$IoU = \frac{TP}{TP + FN + FP}, \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

After the metrics calculation, the performance results of LD-UNet along with the results of other state-of-the-art models from the literature when evaluated on the 95-Cloud test set are shown in Table 4.1.

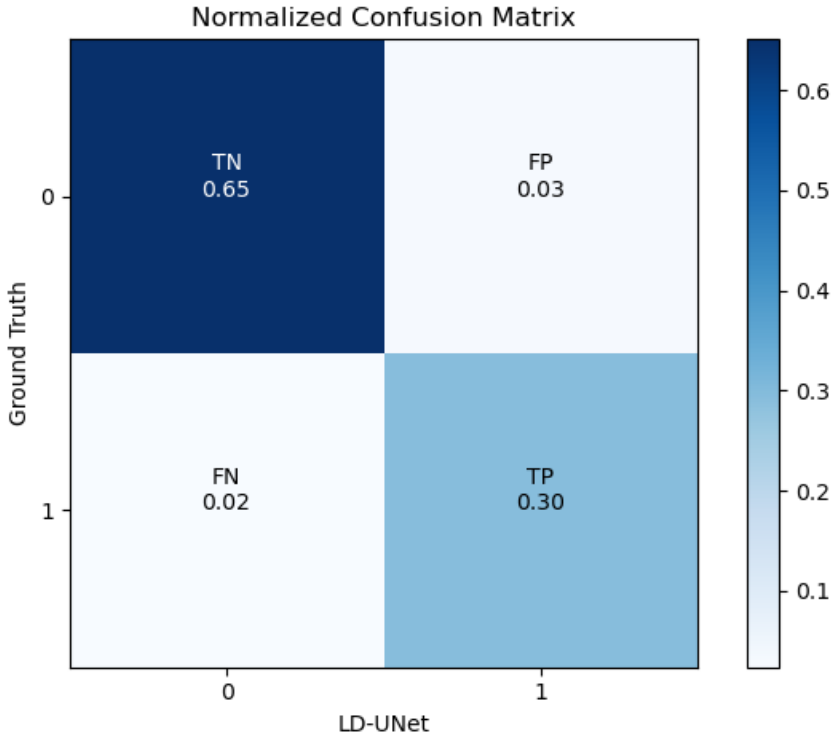

Figure 4.4: The normalized confusion matrix of the model on the entire test dataset.

Table 4.1: Model Performance Comparison on 95-Cloud Test Dataset

Model	**Params.**	**Acc.**	**IoU**	**Rec.**	**Prec.**
LD-UNet	5652	94.57	84.47	92.42	90.75
C-UNet++[29]	9129	94.85	83.89	88.48	94.18
C-FCN[29]	1438	93.91	81.47	88.39	91.23
Cloud-Net+[65]	32.9M	97.23	91.57	94.28	96.94

We observe that with only 5652 parameters and operating on downsized 128×128 patches with three spectral bands (RGB) , LD-UNet maintains competitive performance against the larger C-UNet++ which is introduced in [29] and operates on four spectral bands of 384×384 patches. LD-UNet outperforms the smaller C-FCN, also of [29], in almost all presented performance metrics achieving similar precision. Finally, the Cloud-Net+ model [65] performs better than LD-UNet and supports multiclass segmentation but at the cost of having 4 orders of magnitude more parameters.

Chapter 5

Porting CNN Models to the Programmable Engine of an FPGA

The current chapter of the doctoral thesis focuses on the efficient deployment of CNN models on an edge-oriented FPGA. Several edge computing applications nowadays rely on CNN models being executed on accelerator devices which are embedded, or in very close proximity, to the camera sensors. The deep learning-based image regression for irradiance estimation and forecasting for edge computing Photovoltaic (PV) applications as well as the semantic segmentation of satellite images for on-board cloud detection presented in the two previous chapter of the thesis are two such applications. Both of these applications require quick and end-to-end deployment for rapid prototyping and evaluation of a wide variety of CNN models regarding their performance and execution times on the edge device. In order to address this challenge, the current thesis adopts a development flow which utilizes the state-of-the-art Xilinx Vitis AI framework in order to result in efficient FPGA-based CNN accelerators for edge applications. The capabilities of the framework are extensively explored and an acceleration approach is showcased for accelerating distinct processes of a single task on the heterogeneous resources of the FPGA. The results validate the adopted development flow and the acceleration approach by showcasing real-time processing rates for the deep learning applications of the previous two chapters of the thesis.

The remainder of this chapter is structured as follows. First, Section 5.1 provides the necessary background about unified frameworks for porting CNNs to FPGAs. Then, Section 5.2 elaborates on the proposed methodology by analyzing both the development flow based on the Vitis AI framework as well as the proposed acceleration approach. Finally, Section 5.3 presents the evaluation of CNN models on the edge FPGA, based on the development flow and the acceleration approach.

5.1 Background

FPGAs are steadily rising as a promising hardware platform for the acceleration of CNNs. When compared to other platforms, FPGAs constitute an attractive alternative which offers reduced power consumption to power-hungry but very powerful GPUs and increased reconfiguration capabilities to fixed-function but power-efficient ASICs. The reconfigurable nature of FPGAs allows for the generation of different CNN hardware accelerator architectures that can meet a variety of system-level requirements including those of embedded systems for edge computing applications. Despite the potential advantages that FPGAs can offer in terms of latency, power consumption and spatial footprint, one aspect that they usually lack in is the ease of programmability. Frameworks such as PyTorch and TensorFlow provide high-level Application Programming Interfaces (APIs) to the developers for efficiently accelerating CNNs on CPUs and GPUs. Even for specialized ASICs and SoCs, frameworks such as TensorFlow Lite and Intel OpenVINO help to increase the productivity of developers for deploying CNNs to edge and mobile devices. On the other hand, deploying a CNN on an FPGA can become a strenuous task depending on the design requirements which many times dictate the development flow and level of abstraction that can be utilized for the FPGA design.

The FPGA development flow where a Register-Transfer Level (RTL) design is described with a Hardware Description Language (HDL) language, such as VHDL, is currently considered the most low-level approach for describing a CNN accelerator on an FPGA. The RTL development flow requires extensive hardware expertise as all the elements of the hardware architecture of the CNN model such as the mapping of computations to processing elements as well as the corresponding buffering techniques and memory access patterns, need to be optimized by the designer by hand. This high optimization flexibility in the RTL development flow can potentially satisfy the constraints and requirements of different platforms such as resources utilization or latency, but introduces increased development effort. An alternative to the RTL development flow is the High-Level Synthesis (HLS) approach for generating CNN accelerators on FPGAs. HLS tools, such as Xilinx Vitis HLS, provide to the developer a higher level of abstraction by allowing C/C++-based development which however still requires expertise in hardware concepts. Important high-level design decisions need to be made such as a opting for a streaming architecture vs. a single computation engine architecture. Furthermore, in order to exploit the parallelization capabilities of the FPGA's fabric, more low-level optimizations need to be made to the C/C++ description such as loop pipelining, flattening, partial-unrolling, etc.

The increased interest in efficiently and quickly deploying CNN models to FPGAs has led to automated frameworks which aim at providing even more hardware abstraction. These CNN-to-FPGA automated frameworks can provide interfaces for model descriptions with popular frameworks for CNN development such as TensorFlow and Caffe and can generate FPGA designs for different hardware architectures (streaming vs. single

computation engine architecture) based on different technologies such as generic RTL or vendor-specific HLS.

5.2 Methodology

For the application of deep learning-based image regression for irradiance estimation and forecasting, presented in Chapter 3, the thesis aims to support the concept of an edge-enabled smart PV park that can fulfill the real-time requirements of the PV control. Correspondingly, for the application of semantic segmentation for on-board cloud detection from satellite imagery, presented in Chapter 4, the thesis aims to showcase real-time processing rates in order to support the high volume of payload data generated by on-board sensors with devices that meet the limited power budget of satellites. Based on the aforementioned objectives, the CNN models studied and developed for both applications, are deployed to the edge-oriented Xilinx Zynq UltraScale+ MPSoC FPGA. This SoC family of FPGAs features heterogeneous resources including both an ARM-based Processing Subsystem (PS) and the configurable fabric, known as the Programmable Logic (PL). For porting the models to the FPGA, we utilize the Xilinx Vitis AI framework [68]. The framework provides an end-to-end set of tools that allows the developer to port and execute CNN models from popular frameworks such as PyTorch and Tensorflow, to Xilinx FPGAs. The development workflow that the current thesis follows for implementing & accelerating the CNN models on the FPGA is summarized in Fig. 5.1 and is described in the following paragraphs.

On the host PC side, the tools include the Vitis AI Quantizer. Using the Quantizer, first, we develop the Python application for quantizing our 32-bit floating-point CNN model descriptions to the corresponding 8-bit fixed-point ones, as required by the framework. The conversion of the 32-bit floating-point weights and intermediate results of the model, to 8-bit fixed-point representations reduces the computational complexity of the model and results in less required memory bandwidth, lower latency and increased power efficiency. However, the quantization process can potentially result in prediction accuracy loses. To address this issue, Vitis AI offers several quantization related solutions. The Post Training Quantization (PTQ) that implements the cross layer equalization algorithm [69] and the Fast Finetuning (FF) that implements the AdaQuant algorithm [70] are the two quantization solutions which do not perform back-propagation and thus do not require a labeled dataset. They rely on several iterations of inference on a set of unlabeled images in order to capture the statistics of the results and calibrate the activations, finetune the weights and improve the accuracy of the quantized model overall. Additionally to the above quantization solutions, Quantization Aware Training (QAT) is used to further improve the accuracy of a quantized model after the original floating-point model training. QAT is a computationally demanding process as it operates on the entire original training dataset and performs back-propagation on the quantized model, thus it is employed only

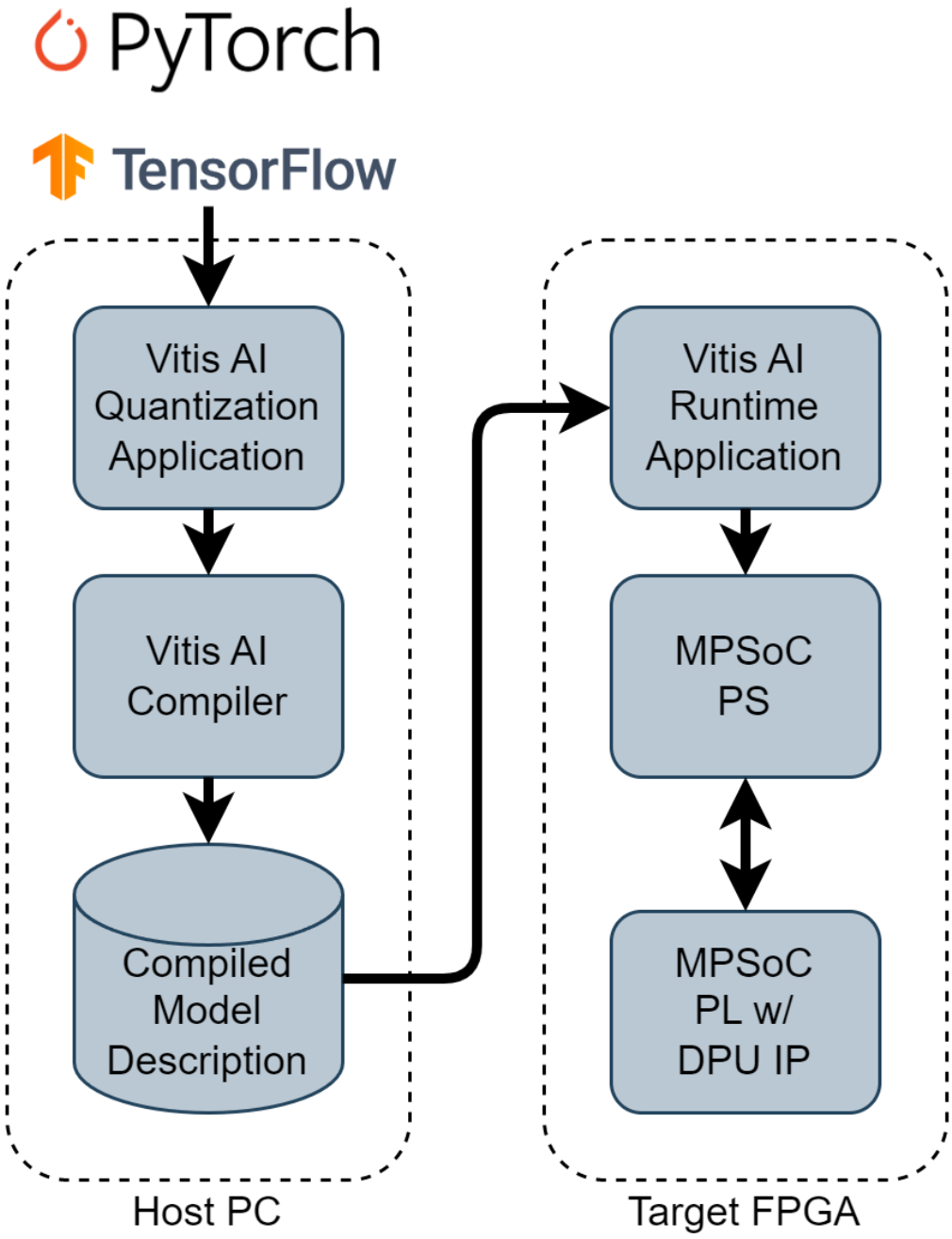

Figure 5.1: The proposed Vitis AI development flow.

for a few epochs on a pre-trained model. The current thesis, combines the PTQ, FF and QAT solutions in order to explore their capabilities and reduce the effect of quantization to the performance of the studied models as much as possible. After quantization, the models are compiled with the Vitis AI Compiler in order to produce the graph description and instructions which are going to be executed on the target FPGA during runtime.

On the target FPGA side, the framework provides the Deep Learning Processor Unit (DPU) IP Core. The DPU is a programmable computation engine which is implemented on the PL resources of the target FPGA. Its architecture and the set of instruction it supports is highly optimized for accelerating a wide range of operations that most of the popular CNNs require. In the context of the current thesis, the DPU is configured in an optimized way in order to result in improved resources utilization and increased processing throughput. In particular, the DPU is configured in order to utilize UltraRAMs, an alternative kind of on-chip memory resources, when compared to the classic block RAMs, that the Zynq UltraScale+ devices include. By enabling the utilization of UltraRAMs, the DPU design can now be implemented with reduced block RAMs, allowing the placement and routing of two processing cores of the DPU on the same design. The two processing cores of the DPU can be combined with multi-threading in order to result in increased processing throughput of CNN applications. Finally, the DPU is integrated in the FPGA design which also includes the ARM-based PS. After FPGA programming, we develop the runtime application that is being executed on the PS of the FPGA and controls the DPU during runtime. The development of the runtime application is based on the APIs that the Vitis AI Runtime library exposes to the developer that are available in both C++ and Python.

Further to CNN processing with Vitis AI, the current thesis aims to showcase the distinctive approach of taking advantage of the heterogeneous resources of the MPSoC FPGA to accelerate different processes of an entire computer vision application. This acceleration approach is employed for the application of binary semantic segmentation for cloud detection in satellite images and for the LD-UNet CNN model that was introduced in Chapter 4. In particular, an HLS kernel, taken from the Vitis Vision Library [71], for the pre-processing of images, is combined with the DPU for the CNN inference on the PL side. On the PS side, an optimized version of the CPU C++ code for performing post-processing is proposed. The HLS kernel implemented on the PL performs resizing of the input image to the dimensions that the CNN accepts as input. Furthermore, it performs scaling of the original image data to the fixed-point representation that the DPU accepts at its input layer. When pre-processing is complete, the control returns back to the PS and the images are ready to be processed by the DPU. The DPU implementation is configured for including two distinct homogeneous cores. By also including support for multithreading in the target C++ application, Vitis AI allows for a batch of images to be processed by two threads that access the two distinct DPU cores for increased throughput. Finally, when the CNN processing is complete, the post-processing follows. The final activation function of the LD-UNet model, proposed for semantic segmentation in the current thesis, is the sigmoid activation function that is not supported by the DPU. Following the sigmoid, a pixel-wise thresholding operation is performed to create the binary segmentation mask.

In order to accelerate the above two operations, we fuse them to a single C++ operation of accessing a Lookup Table (LUT) based on the DPU output value. These DPU outputs are limited to only 256 possible values due to their 8-bit fixed-point representation and thus the sigmoid and thresholding exact result can be automatically pre-calculated and stored in a LUT during the initialization of the application. This way, during runtime the sigmoid and thresholding operations requires only accessing the LUT instead of calculating the corresponding value. The entire FPGA processing architecture that is produced as a result of the integration of the 2-core DPU with the pre-processing HLS kernel is shown in Fig. 5.2.

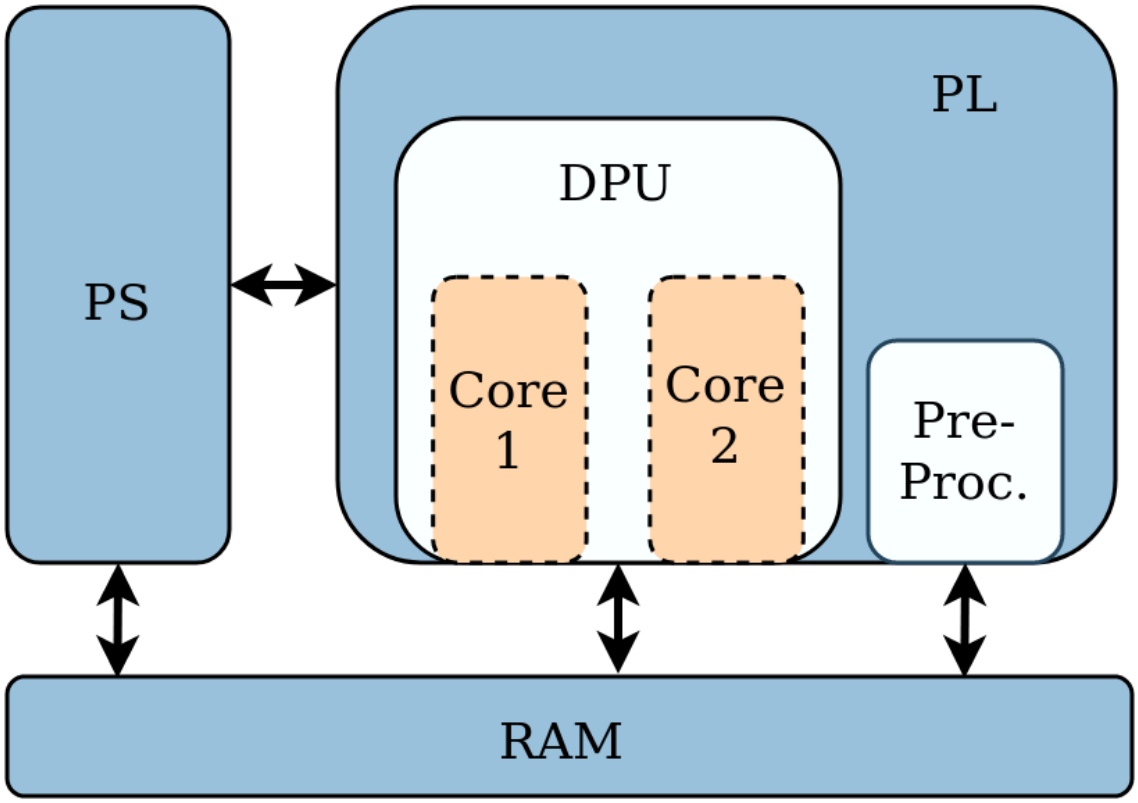

Figure 5.2: The FPGA processing architecture

5.3 Evaluation & Results

The current section of the thesis, presents the results regarding the porting process of CNNs to the edge-oriented Xilinx MPSoC FPGA. The porting process is based on the adopted development flow which is described in the previous section and it is performed for two different edge computing applications based on CNNs. The first application is the deep learning-based image regression for irradiance estimation and forecasting for edge computing photovoltaic applications presented in Chapter 3. The second application is

the semantic segmentation of satellite images for on-board cloud detection presented in Chapter 4. The steps of the porting process for each of the above two applications are described in Subsection 5.3.1 and Subsection 5.3.2 respectively.

5.3.1 Image Regression CNN Models Porting Process

The first step towards deploying the four different image regression CNN models for irradiance estimation, studied in Chapter 3, to the FPGA, is to perform quantization. The Python quantization application that we develop utilizes several different quantization functionalities of the Xilinx Vitis AI 2.0 Quantizer. The results of the quantization process for the four different models of the study are presented in Table 5.1. First, we perform Post Training Quantization using a batch of unlabeled images. We observe that the PTQ has a very significant effect of $111.23W/m^2$ increased RMSE on the performance of the original floating-point VGG11 of Table 3.2. After performing and additional Fast Finetuning step using 1000 unlabeled images, we reduce the effect to $4.26W/m^2$. For the ResNet-50 model, PTQ results in a slight increase in RMSE of $3.52W/m^2$ which is reduced down to $2.18W/m^2$ with FF. The MobileNetV2 suffers a loss in performance which cannot be corrected even after FF, resulting in a loss of $12.47W/m^2$. Finally, the quantized SqueezeNet model has a large performance degradation from its original floating-point model, an increase in RMSE of $19.76W/m^2$. With FF the increase in RMSE becomes $6.84W/m^2$. The SqueezeNet model architecture allows us to perform an additional Quantization Aware Training step instead of FF. We train the SqueezeNet model for 1 additional epoch using the QAT capabilities of the Vitis AI Quantizer. After the QAT step, the performance of SqueezeNet is restored to similar one as the original floating-point model suffering only an $2.09W/m^2$ RMSE increase.

After quantization of the CNN models, we implement the FPGA processing architecture on the Xilinx ZCU104 FPGA board using the Xilinx Vitis and Vivado 2021.2 tools. The 2-core DPU IP is operating at 300 MHz and the entire design consumes 15.585 W based on the power analysis tool of Vivado. In Table 5.2, we present the resources utilization of the PL of the FPGA. We observe that the 2-core DPU IP consumes a very significant amount of resources, especially regarding the DSPs that are responsible for performing most computations. It is worth noting that for applications where processing throughput is not critical, the developer can configure the DPU with a single processing core to reduce the resources utilization by about half for most resources.

In order to showcase the real-time capabilities of the edge FPGA on the image regression task, we benchmark the 4 different CNNs on the DPU IP Core. We evaluate their throughput in terms of Frames per Second (FPS) on both cores of the DPU by using multithreading and we present the results in Fig. 5.3. The results show that the ResNet-50 model with the highest number of parameters and operations has the lowest throughput of 158 FPS on a single core of the DPU. SqueezeNet has the highest throughput of 1028 FPS even though its original floating-point model requires more operations but with lower

Table 5.1: Performance metrics for the 4 quantized models with combinations of PTQ, FF and QAT.

Model	**Quantization Method**	**RMSE (W/m^2)**	**nRMSE (%)**	**MAE (W/m^2)**
VGG11	Floating-Point	65.25	15.88	38.31
	PTQ	176.48	42.94	111.71
	PTQ & FF	69.51	16.91	41.04
ResNet-50	Floating-Point	64.83	15.77	37.23
	PTQ	68.37	16.63	40.39
	PTQ & FF	67.01	16.30	39.18
MobileNetV2	Floating-Point	75.95	18.48	47.74
	PTQ	88.42	21.51	59.91
	PTQ & FF	88.78	21.60	60.67
SqueezeNet	Floating-Point	70.18	17.08	44.65
	PTQ	89.94	21.89	63.24
	PTQ & FF	77.02	18.74	50.30
	PTQ & QAT	72.27	17.58	45.84

Table 5.2: FPGA resources utilization for the implemented design on the ZCU104 board.

Resource	**2-Core DPU IP**	**1-Core DPU IP**
LUTs	108K (47%)	50K (22%)
FFs	204K (44%)	98K (21%)
DSPs	1394 (81%)	690 (40%)
RAMBs	203 (65%)	145 (46%)

number of parameters than MobileNetV2. When utilizing the 2 cores of the DPU all the models can achieve a little less than $\times 2$ throughput. For the ResNet-50, MobileNetV2 and SqueezeNet models, the achieved throughput rates can be considered to satisfy the real-time requirements, e.g., for a sky imager providing a video at 60 FPS, leaving space for additional algorithms to complete more PV related processes.

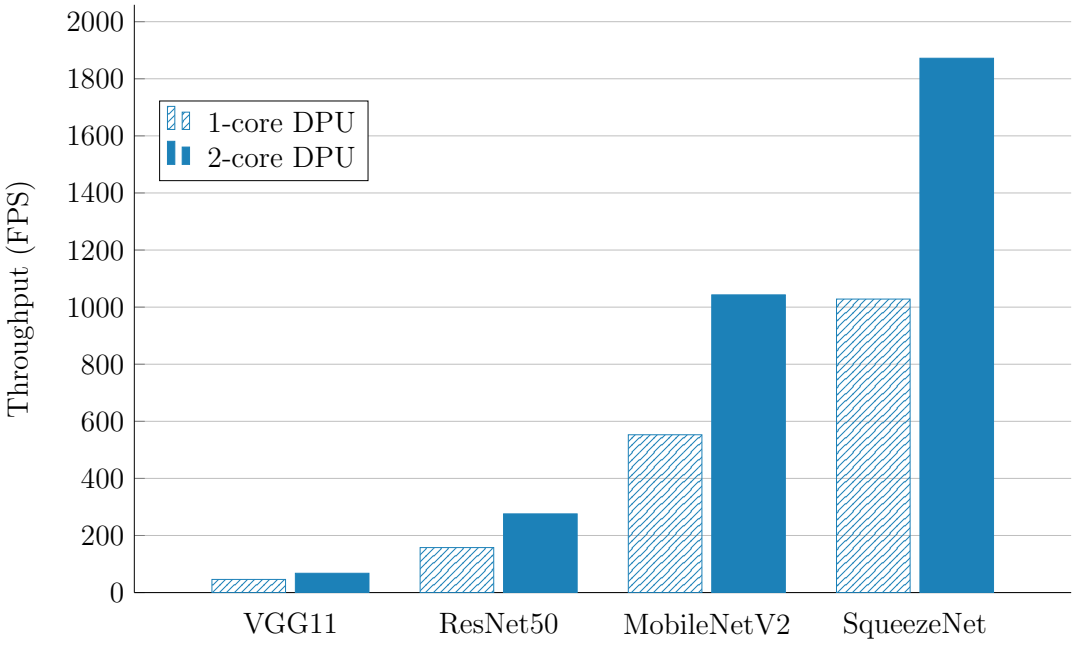

Figure 5.3: Throughput results for the four studied models, on 1 and 2 processing cores of the DPU.

5.3.2 Binary Semantic Segmentation CNN Model Porting Process

In order for the binary semantic segmentation LD-UNet model to be implemented on the FPGA it is quantized from 32-bits to 8-bits with fixed point arithmetic. The quantization is performed with the Vitis AI quantizer using, first, the PTQ solution with a calibration batch of the training dataset. The performance results of the quantized LD-UNet, with PTQ, are shown in Table 5.3. We observe that the quantization is being detrimental to the performance of the model. For this, we utilize the Vitis AI quantizer to perform an additional post-training Fast Finetuning step where the parameters of the quantized model are being adjusted to further reduce the loss function value. For this Fast Finetuning step, we provide 2000 patches that correspond to ~9% of the training dataset. From Table 5.3 we observe that after Fast Finetuning, the quantized model can once again achieve satisfactory performance results with little degradation compared to the corresponding floating point one.

Table 5.3: Performance of Quantized LD-UNet

Model	Acc.	IoU	Rec.	Prec.
LD-UNet Floating-Point	94.57	84.47	92.42	90.75
LD-UNet PTQ	74.60	21.41	21.67	94.75
LD-UNet PTQ & FF	91.92	78.34	91.54	84.46

The processing architecture for the application of semantic segmentation is implemented on the Xilinx ZCU104 FPGA board using the Xilinx Vitis and Vivado 2021.1 and Vitis AI 2.0. The DPU IP with 2 processing cores and the pre-processing accelerator operate at 300MHz and the Vivado power analysis reports a total on-chip power of 14.111 W for the entire design. The resources utilization of the FPGA is shown in Table 5.4. We observe that the DPU IP is a very resource-hungry design whereas the pre-processing accelerator adds only little overhead to the resources utilization. The total utilization also includes auxiliary components such as interconnections, clock managers, etc.

Table 5.4: Resources Utilization for Implemented Design on ZCU104

Component	LUTs	FFs	DSPs	RAMBs
2-Core DPU IP	97K (42%)	195K (42%)	1380 (80%)	169 (54%)
Pre-Proc. Acc.	8.7K (4%)	11K (2%)	26 (1%)	9.5 (3%)
Total	114K (49%)	220K (48%)	1406 (81%)	212 (68%)

The execution time results for the acceleration of the pre-processing, the post-processing and the entire application, including the LD-UNet execution on the DPU, for a single input image patch, is shown in Fig. 5.4. We observe that the baseline pre-processing that includes resizing and scaling the input data in software on the ARM PS of the FPGA achieves a speedup of 1.94 when executed on a dedicated hardware HLS kernel in the PL. Furthermore, the baseline post-processing implementation of calculating the sigmoid activation function and performing thresholding on the output can be sped-up by a factor of 14.52 with our optimized LUT-based implementation. With the above optimizations the total speedup for the entire application is 1.64. Finally, when benchmarking the CNN inference on the DPU, standalone, utilizing multithreading, a throughput of 632 FPS can be achieved with 2 threads, an improvement of 85% when compared to the 342 FPS of a single thread.

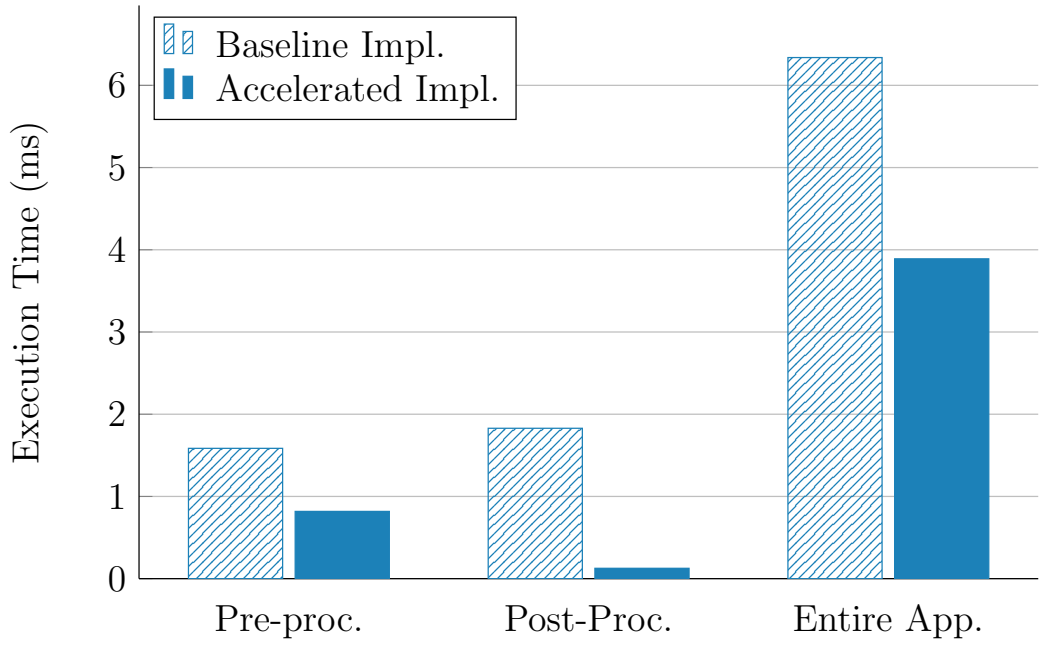

Figure 5.4: Execution time for the baseline and accelerated implementations of pre-processing, post-processing and the entire application on the FPGA.

Chapter 6

High Throughput & Fault-Tolerant FPGA–VPU Interfacing

The current chapter focuses on design and development of the interfacing system between an FPGA and the Intel Movidius Myriad 2 VPU. The interfacing system targets data frame transfers between the two devices in computing architectures which are utilized in on-board payload data processing applications and particularly in the context of the European Space Agency (ESA) activity ”FPGA Accelerated DSP Payload Data Processor Board”. Consequently, it needs to be designed with high-throughput capabilities in order to support high bit-rate sensors and with a fault-tolerance mechanism in order to support the entire computing system mitigate the effects of space radiation on its components. The remainder of this chapter is structured as follows. First, in Section 6.1 the background regarding heterogeneous computing architectures in space is discussed. Then, in Section 6.2 the design of the Myriad 2 Interface (M2 IF) module, designed and developed on the FPGA side, is presented in detail, focusing on its high-throughput and fault-tolerance characteristics. Finally, Section 6.3 presents the results from the extensive testing campaign of the interfacing system in several experimental hardware setups.

6.1 Background

The NewSpace era relies on novel technological approaches in space applications. The advances in small form factor satellites, such as SmallSats and CubeSats, broaden the scope of the EO missions and attract new areas of research for payload processing in space. Furthermore, there is a rapidly growing interest in the use of artificial intelligence in space. In particular, deep learning and CNNs are already being deployed as a solution for many on-board computer vision tasks including data reduction, remote sensing data processing, vision-based navigation and more [72].

With these technological advances come corresponding challenges such as the increased remote sensing data generated by satellite instruments and the computational demands of modern image processing and artificial intelligence algorithms. As a result, the space-qualified general-purpose processors, such as the radiation-hardened LEON CPUs, fail to keep up with the processing requirements of future missions. Moreover, constraints in the power budget and the dependability requirements of each space mission, drive the research community and the industry to revisit the computing architectures for space avionics. In order to address the above challenges, the interest is focused on heterogeneous and mixed-criticality computing architectures. In these architectures, rad-hard components are complemented by Commercial Off-The-Shelf (COTS) accelerators for payload data processing and especially image processing and deep learning tasks which are not mission-critical [62]. Specialized COTS SoC accelerators, such as VPUs and SoC FPGAs, offer attractive trade-offs between SWaP-C, processing performance and development flexibility [73, 74].

In the ESA activity "FPGA Accelerated DSP Payload Data Processor Board" (ESTEC contract 4000126129/18/NL/AF) such a heterogeneous computing system is designed and a prototype high-performance platform is developed, the High-Performance Compute Board (HPCB). The HPCB platform aims to support the next generation of deep learning-based image processing on-board by handling payload data from multiple high bit-rate instruments simultaneously and at the same time apply system-level fault mitigation techniques when they are required. In order to be modular, the HPCB platform consists of a carrier board and three mezzanine extension cards for parallel processing and Triple Modular Redundancy (TMR) configurations. The prototype GR-VPX-XCKU060 carrier board and GR-HPCB-FMC-M2 mezzanine card are shown in Fig. 6.1 and Fig. 6.2 respectively. The carrier board, includes the GR716 [75] radiation-tolerant microcontroller with the role to supervise the operation of the entire platform. Furthermore, it includes the Xilinx Kintex Ultrascale XCKU060 [76] which serves as the flow manager of the system by receiving data from multiple on-board sensors, perform any required transcoding and forward data to the specialized COTS accelerators, the working memory and the mass-memory of the system. Finally, the carrier board can facilitate three mezzanine cards with COTS accelerators which can operate in redundant modes. Each card, includes the Myriad 2 VPU for accelerating image processing and deep learning tasks on-board. The entire computing architecture is illustrated in Fig. 6.3.

6.2 Design

In the current section, the design of the interfacing system between the FPGA and the Myriad 2 VPU is introduced. First, 6.2.1 describes the details of the digital design of the M2 IF module on the FPGA. Then, 6.2.2 elaborates on the design of the fault-tolerance mechanism of the interfacing system.

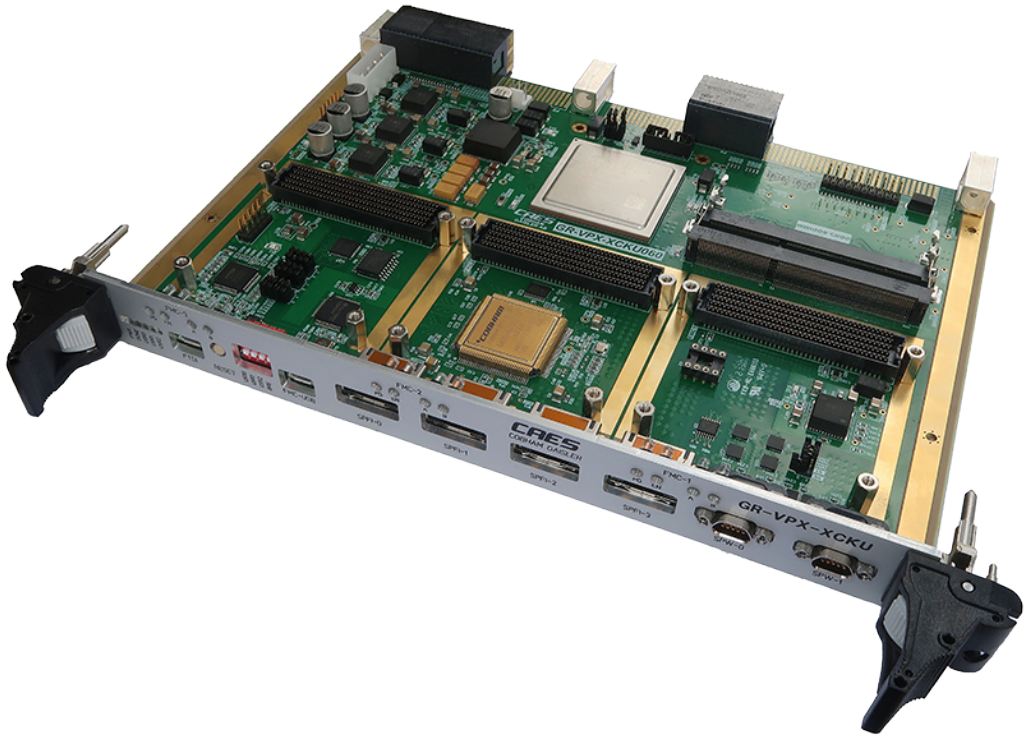

Figure 6.1: The GR-VPX-XCKU060 carrier board of the HPCB prototype system.

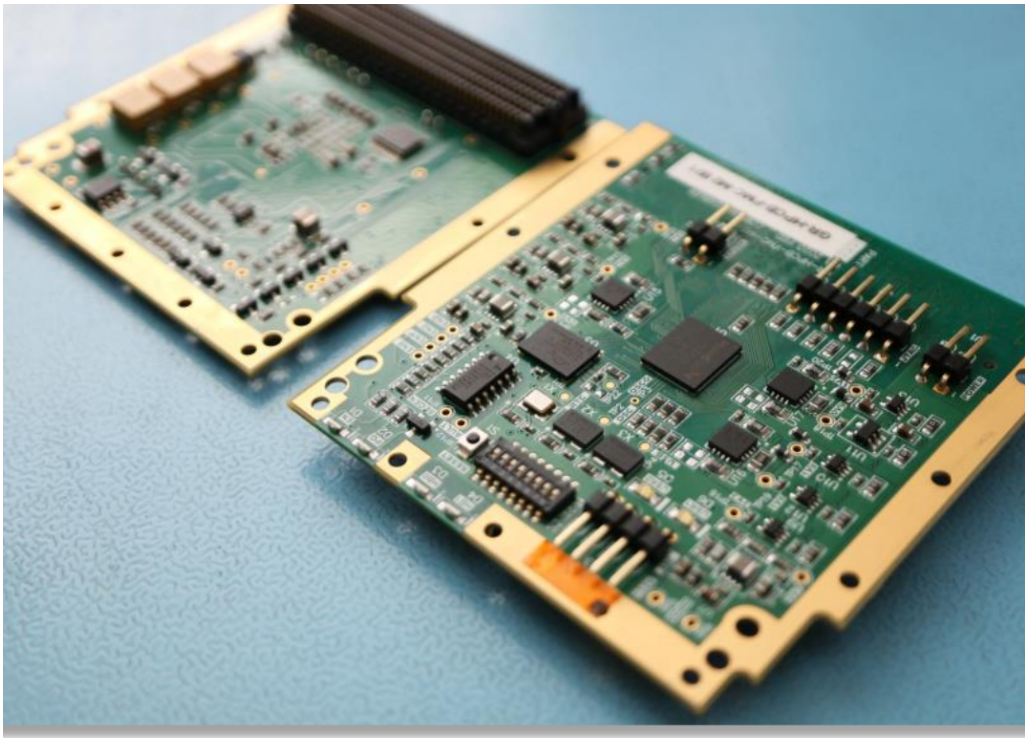

Figure 6.2: The GR-HPCB-FMC-M2 mezzanine card of the HPCB prototype system.

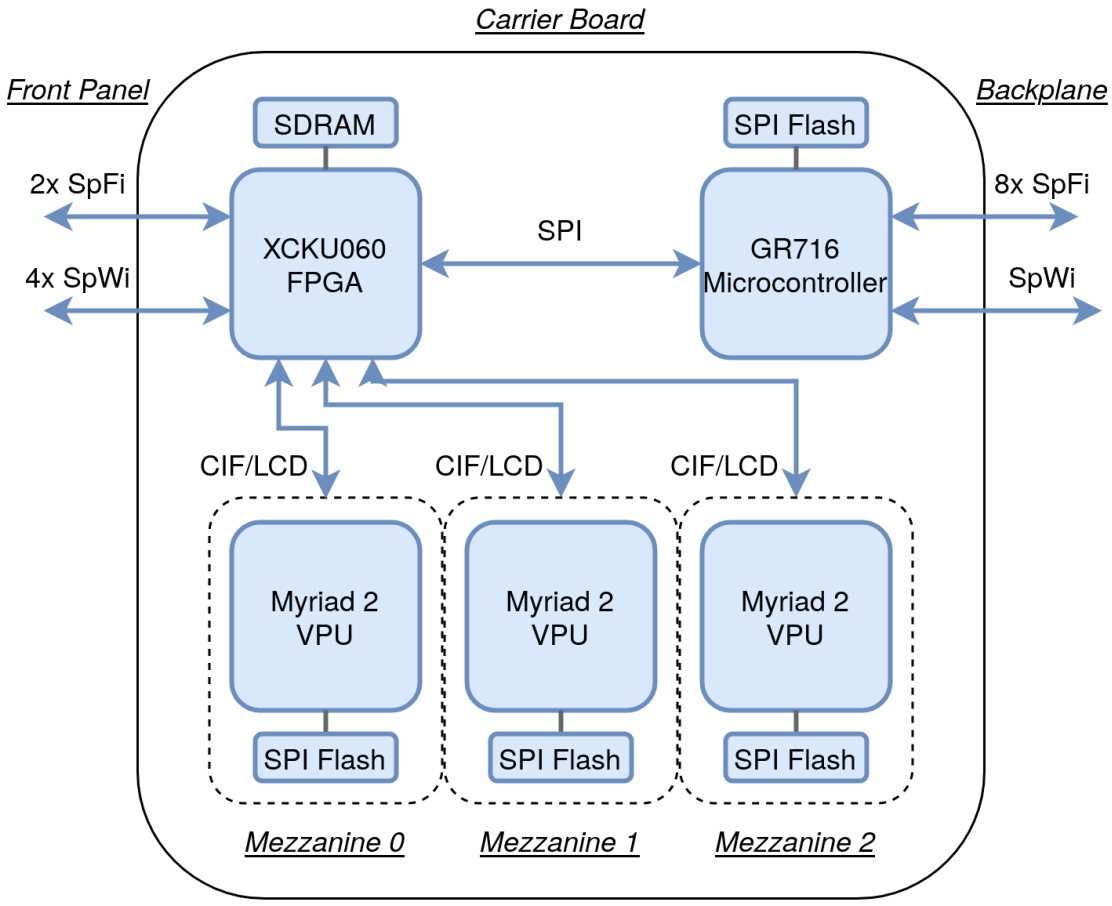

Figure 6.3: The block diagram of the HPCB platform computing architecture.

6.2.1 Design of the Interfacing System

The M2 IF module on the FPGA side, handles the communication with Myriad 2 over the CIF and LCD interfaces. The CIF interface is utilized for transmitting images, or even generic data in the form of frames, from the FPGA to the Myriad 2. Similarly, the FPGA receives frames from the Myriad 2 via the LCD interface. The block diagram of the internal architecture of the M2 IF module is shown in Fig. 6.4.

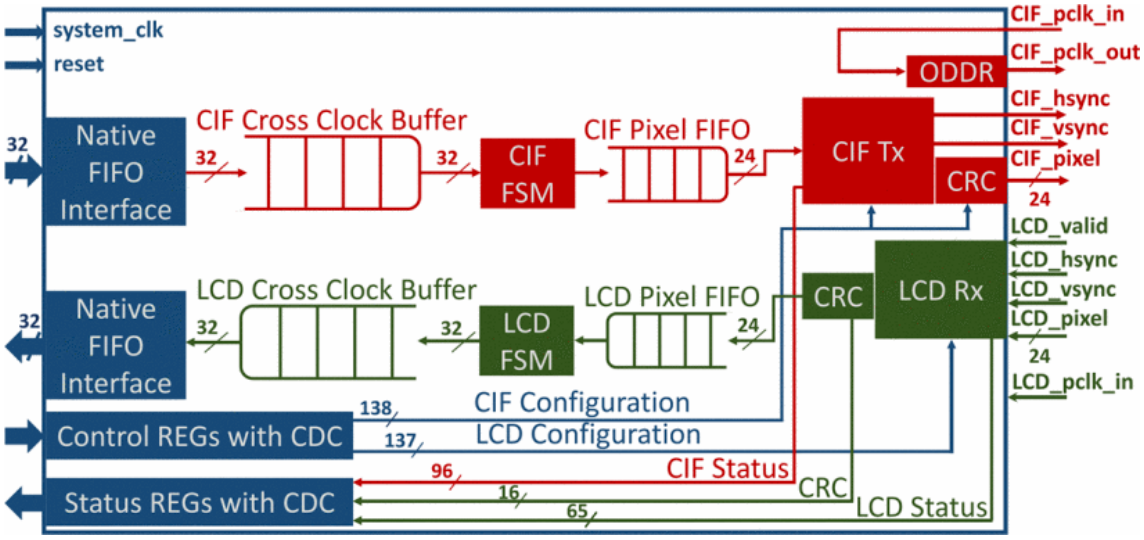

Figure 6.4: Block diagram of M2 IF module in the FPGA.

Regarding clocking and resets, the M2 IF operates on a system clock which is used for the Control and Status registers, as well as the native FIFO interfaces of the internal buffers. An asynchronous reset is forwarded from the top-level entity to all the internal components of the M2 IF module. Regarding the CIF interface clocking, all components on the CIF clock domain operate on the CIF_pclk_in clock signal. For source-synchronous operation of the CIF interface, which is the default mode of operation, the CIF_pclk_in should be provided by a Mixed-Mode Clock Manager (MMCM), external to the M2 IF, configured for the desired operating frequency. Then the CIF_pclk_in clock is forwarded to Myriad 2, by means of an Output Double Data Rate (ODDR) register, at the CIF_pclk_out signal. Alternatively, if Myriad 2 provides the clock of the CIF interface, the external MMCM handling it can feed it to the M2 IF again via the same CIF_pclk_in signal. The LCD interface clocking is based on the LCD_pclk_in signal which is used by the all components on the LCD clock domain. For source synchronous operation of the LCD interface, which is the default mode of operation, the LCD_pclk_in should be provided by an MMCM, external to the M2 IF, which performs clock recovery from the LCD clock provided by the Myriad 2. If source synchronous operation is not opted, the LCD_pclk_in should be provided by an MMCM that generates a clock of the desired operating frequency. All other CIF and LCD signals, both for synchronization and data, are appropriately assigned to/from registers inside the VHDL design in order to be able to

pack these registers into I/O Blocks (IOBs) of the FPGA. The IOB packing is performed by a simple constraint for each of these signals. Any Clock Domain Crossing (CDC) logic is implemented by independent clock FIFOs.

Regarding the buffers of the M2 IF, the CIF and LCD Cross Clock Buffers and the CIF and LCD Pixel FIFOs are VHDL implementations for generic RAM blocks and FIFOs. The CIF and LCD Cross Clock Buffers have a word width of 32 bits which can contain several pixels, while their depth is configurable and is indicated at compile time by the corresponding VHDL constant. Along with their native FIFO interface, they also expose programmable full and empty signals with corresponding thresholds configured by VHDL constants. They include logic for crossing from the FPGA engine clock domain to/from the CIF and LCD clock domains respectively. For injection of a frame into the M2 IF module, write operations can be performed to the CIF Image Buffer through its native interface with flow control based on the full and empty signals. In the case of data reception from the M2 IF, read operations can be performed to the LCD Cross Clock Buffer via its native interface with the availability of the result data inside the LCD Cross Clock Buffer indicated by the empty signal. The CIF and LCD Pixel FIFOs have a word width of 24 bits, equal to the CIF and LCD parallel interfaces width, and a configurable, at compile time, depth by means of a corresponding VHDL constant. The CIF Pixel FIFO, operates on the CIF clock domain and holds one CIF pixel per 24-bit word while the LCD Pixel FIFO operates on the LCD clock domain and holds one LCD pixel per 24-bit word.

When it comes to runtime programmability and status reporting, the M2 IF module includes two sets of registers for interfacing with the rest of the FPGA engine. The Control Registers are written from control modules external to the M2 IF while the Status Registers are written from the M2 IF side. The Control Registers, contain information regarding the CIF and LCD interfaces timing parameters and the CIF and LCD pixels bit-depth. The Status Registers, contain information regarding the number of frame transmissions and receptions over the CIF and LCD interfaces and also report any errors such as overflows, underruns and CRC mismatches.

The M2 IF also includes the CIF and LCD FSMs which operate on the CIF and LCD clock domain and control the read and write operations between the corresponding Cross Clock Image Buffers and Pixel FIFOs. The CIF FSM is configured during runtime with regards to the width, height and pixel bit-depth of the CIF frame to be transmitted based on the values of the corresponding Control Registers. When the CIF FSM begins operation, it reads one 32-bit word from the CIF Image Buffer per clock cycle which can contain several distinct pixels as indicated by the corresponding Control Register. These distinct pixels from each word are then written, one by one, to the CIF Pixel FIFO. The above read and write operations are fully pipelined in order to sustain the maximum bit-rate supported by the width and operating frequencies of the interfaces. The LCD FSM is configured during runtime with regards to the specifications of the LCD frame to be received, in an identical way to the CIF FSM. During consecutive clock cycles, the LCD FSM reads 24-bit words where each contains a pixel of the indicated bit-depth. These pixels are concatenated into

one 32-bit word which is forwarded to the LCD Image Buffer in a pipelined manner as with the CIF interface.

Finally, M2 IF includes the CIF Tx and LCD Rx components. The CIF Tx component performs transmission of images to the Myriad 2 over the CIF interface. The CIF Tx implements the CIF interface synchronization protocol by controlling both the timing of the HSync and VSync synchronization pulses as well as the read operations from the CIF FIFO. The number of clock periods for which the synchronization pulses are active is indicated by the CIF timing parameters provided from the Control Registers of M2 IF. Similarly to above, the LCD Rx component handles the reception of frames from Myriad 2 over LCD. For this, it implements the LCD interface synchronization protocol with the HSync and VSync synchronization pulses timed based on the LCD timing parameters provided from the Control Registers of M2 IF.

6.2.2 Design of the Fault-Tolerance Mechanism

The interfacing system between the FPGA and the Myriad 2 VPU, based on the CIF and LCD interfaces in the form of image frame transfers, is designed to be fault-tolerant. In particular, payload data frames exchanged between both sides are protected by a 16-bit CRC field. The CRC values are appended to the end of each frame by means of a frame footer. The footer has the form of an entire CIF or LCD frame row with the first pixels containing the 16-bit CRC, while the rest of the pixels are zero-padded. The CRC algorithm used for both CIF and LCD is CRC-16-CCITT, where the polynomial is 0x1021 $(x^{16}+x^{12}+x^5+1)$ with an initial value of 0x0. On the FPGA side, the modules which are responsible for handling the CRC values of the CIF and LCD frames are depicted in Fig. 6.5 and Fig. 6.6 correspondingly and are explained in detail in the following paragraphs.

The module that is responsible for formulating and appending the CRC on CIF frames transmitted to the VPU is shown in Fig. 6.5. While a CIF frame transmission is active, the CIF CRC module reads one pixel value at each clock cycle from the CIF Pixel FIFO and forwards it to the CIF transmitter. The CIF transmitter supports frames with bit-depth of 8, 16 and 24 bits. For this, the CIF CRC contains three instances of a CRC16 calculator, which operate in parallel and calculate on-the-fly the CRC values of frames with 8, 16 and 24-bit depth. The CRC16 calculators are controlled by the counter-based Frame Footer FSM. When the active portion of the frame has been forwarded, the CRC16 calculation is complete. Then, the Frame Footer FSM selects the output of the CRC16 calculators based on the corresponding bit-depth and forwards the additional footer row that contains the CRC16 value in the first pixels. When the entire CIF frame is received on the VPU side, the VPU calculates the corresponding CRC16 value and compares it with the received one.

The transmission of the LCD frames from the VPU to the FPGA follows the same principles as that of the CIF frames. When an LCD frame of payload data has been for-

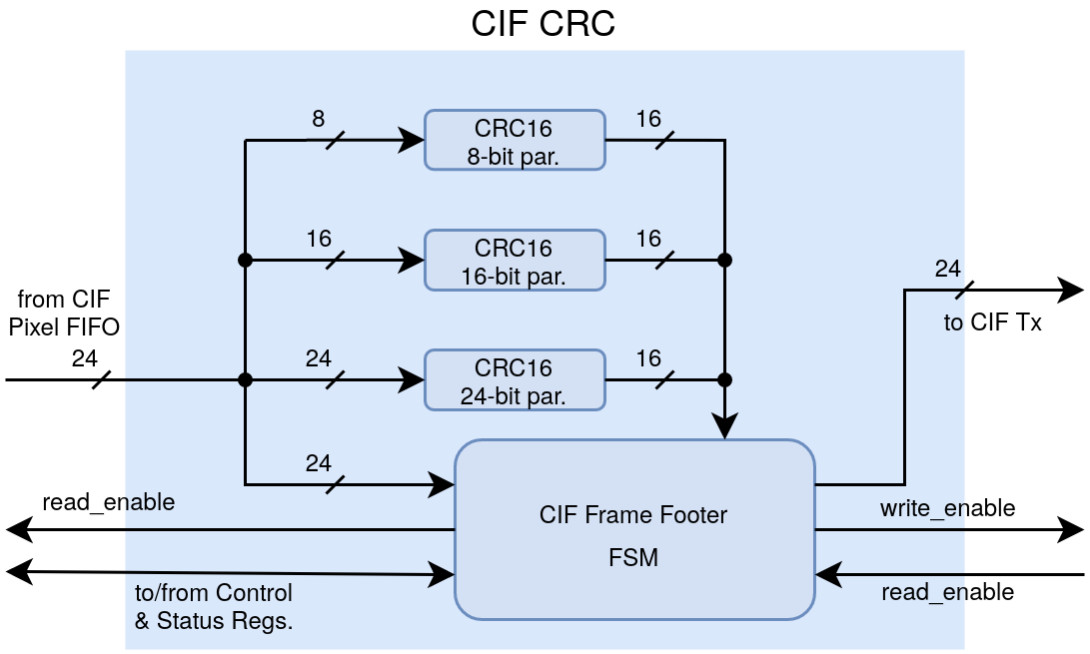

Figure 6.5: The CIF CRC FPGA module.

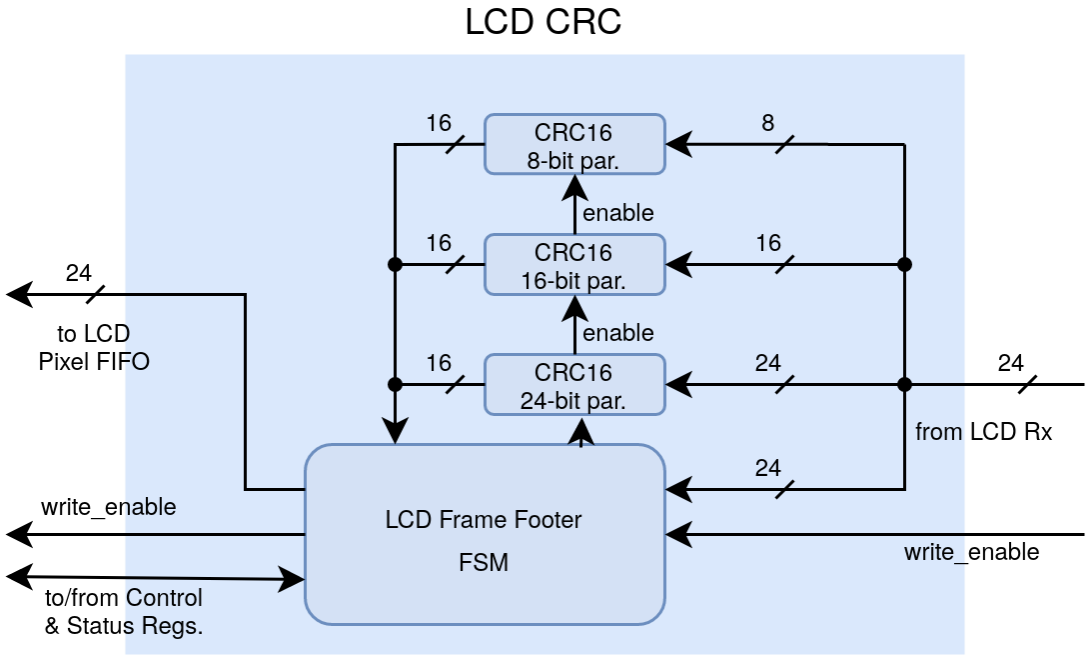

Figure 6.6: The LCD CRC FPGA module.

mulated in the VPU, the VPU calculates the CRC16 value, includes it to the footer of the frame and starts the frame transmission over LCD. The module that handles the CRC calculation and comparison on the FPGA side is shown in Fig. 6.6. The LCD receiver feeds the LCD CRC module with one pixel value at each clock cycle. Three distinct CRC16 modules operate in parallel on different bit-widths of the incoming pixel in order to support all the three different pixel bit-depths. When the active portion of the LCD frame has been received, the CRC16 value is calculated and the Frame Footer FSM handles the reception of the additional footer row. It extracts the received CRC value from the correct position of the footer and performs the comparison with the calculated one. The comparison result is reported to the corresponding status registers of the FPGA.

The above fault-tolerance mechanism of the interfacing system can contribute to the fault mitigation strategies of the entire HPCB computing system in several ways. First, when performing critical data transfers, such as boot images and other configuration data, from the FPGA to the Myriad 2 VPU over CIF, these CIF data frames are protected by the CRC footer. If such a critical CIF frame results to a CRC mismatch when received by the Myriad 2, it can be dropped in order to avoid to misconfiguration of the VPU. Second, when the three Myriad 2 VPU accelerators of the HPCB architecture operate in a triple modular redundancy mode, then the processed frames received from the Myriad 2 over LCD are protected by the CRC footer. If a CRC mismatch occurs this is reported to the voter of the entire FPGA system. In this way, the HPCB system can perform triple voting to identify the correct results and moreover, identify VPUs which consistently malfunction in order to take additional measures such as rebooting them.

6.3 Evaluation and Results

The current section presents the details of the testing campaign and corresponding results of the high-throughput and fault-tolerant interfacing system presented in the previous section.

First, we evaluate the resource utilization of the M2 IF module in the FPGA. For this the M2 IF module, is implemented on the commercially available Xilinx Zynq-7010 FPGA and the resource utilization results are reported in Table 6.1. In the table, the resource utilization for the entire M2 IF module without the CIF and LCD CRC components which are responsible for the fault-tolerance mechanism of the interfacing system are presented, along with the resource utilization for the CIF and LCD CRC components only. The results show, that the FPGA modules which are responsible for the fault-tolerance mechanism are very efficient in terms of resources and introduce only a minimal resource overhead compared to the entire M2 IF module.

Following the results on the resource utilization of the M2 IF module, the testing campaign and corresponding results are explained in detail. During the testing campaign, the

Table 6.1: Resources utilization for the M2 IF module on Zynq-7010

Component	LUTs	FFs	DSPs	RAMBs
M2 IF w/o CRC	2977 (17%)	1232 (4%)	2 (3%)	10 (17%)
CIF CRC + LCD CRC	347 (2%)	305 (1%)	2 (3%)	0 (0%)

system is evaluated on a number of different experimental hardware setups, starting from simple setups with commercially available components to increasingly complex setups which also include prototype hardware. The motivation behind this campaign is, first, to enable testing of the M2 IF VHDL module on commercial hardware before the manufacturing of prototype hardware is complete. This allows for standalone and modular testing before the integration of the M2 IF module and the interfacing system to the entire prototype HPCB computing system. Second, this evaluation campaign enables early testing of the prototype hardware as it becomes available in order to identify potential issues and drive revisions of PCB designs early in the production phase. Based on the above, the experimental hardware setups used in the current evaluation campaign are the following:

- Setup A: The Myriad 2 Eyes-of-Things (EoT) evaluation board combined with a commercial FPGA board, both of which were operated locally.
- Setup B: The prototype GR-HPCB-FMC-M2 mezzanine card combined with a commercial FPGA board, both of which were operated locally.
- Setup C: The prototype GR-HPCB-FMC-M2 mezzanine card combined with the prototype GR-VPX-XCKU060 carrier board, accessed and operated remotely.

Regarding the experimental hardware setup A, it includes the commercial Xilinx VC707 FPGA development board. In this setup, the Myriad 2 EoT board is connected to the VC707, via the XM105 FMC debug card, with jumper wires. The setup is depicted in Fig. 6.7. A Linux workstation is connected to and controls both the VC707 and the EoT boards. On the FPGA side, the entire M2 IF module is implemented along with a testbed VHDL design which includes auxiliary components such as clock generators, user LEDS, etc. On the Myriad 2 side, an application based on the CIF and LCD controllers is developed which supports simple generation, transmission, reception and checking of frames. With this setup, first, simple standalone CIF frame transmissions and LCD frame receptions are tested. Then, in order to create a complete testbed for simultaneous testing of both interfaces, a loopback application is enabled in the Myriad 2 and a corresponding loopback design is introduced in the FPGA. In this loopback scenario, test frames of various patterns are generated in Myriad 2 and are transmitted to the FPGA over LCD, one-by-one. After LCD frame reception on the FPGA side, the received frame is read from the M2 IF module and is re-written to it for transmission over CIF to the Myriad 2. After the CIF transmission, Myriad 2 performs pixel-by-pixel checking of the frame

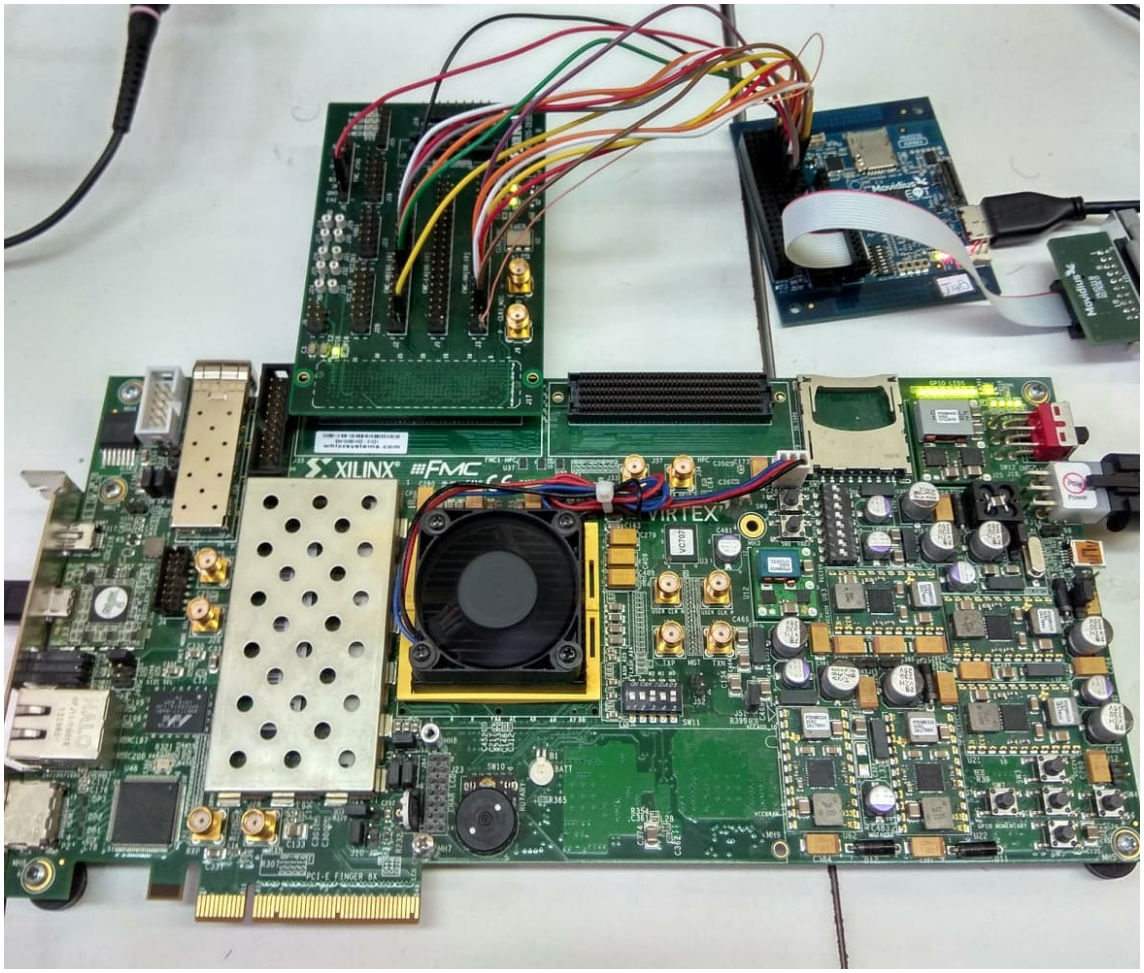

Figure 6.7: The experimental hardware setup A with the EoT and VC707 boards.

for errors. This setup, validates the correct functionality of the M2 IF VHDL module for both the CIF and LCD interfaces without any of the target prototype being available at the time. The maximum achieved operating frequency is 40 MHz for the transmission of 1024×1024 frames with 4-bit pixels without any errors. The above results are constrained, first, by the availability of only a limited number of pins for the CIF and LCD interfaces on the EoT board. Second, a number of electrical issues are identified in this setup. In particular, all of them have to do with cross-talk effects between the signals of the same interface or even between the two different interfaces.

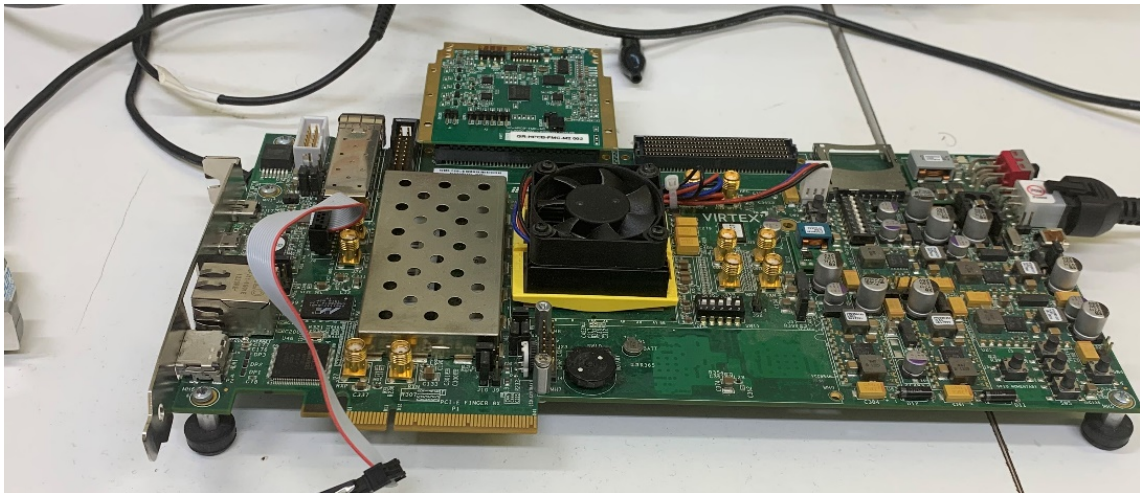

Figure 6.8: The experimental hardware setup B with the GR-HPCB-FMC-M2 mezzanine card and the VC707 board.

Following the evaluation of the interfacing system on the experimental hardware setup A, the evaluation on setup B is performed. In setup B, the prototype GR-HPCB-FMC-M2 mezzanine card is connected to the FMC interface of the commercial VC707 development board as shown in Fig. 6.8. Similarly to setup A, the loopback application and loopback design is implemented on the Myriad 2 and FPGA side respectively. With the experimental hardware setup B, the maximum operating frequency that is achieved is 50 MHz for transmission and reception of frames up to 2048×2048 pixels with a bit-depth of 16 pixels. The cross-talk effects which were identified in setup A, were reduced with the use of the FMC interface instead of the jumper wires. However, an issue regarding the presence of several devices (FPGA and Myriad 2) in the same JTAG chain was identified. Moreover, another issue regarding a clock signal from the Myriad 2 being forwarded to a non-clock capable pin of the FPGA resulted in limitations in the maximum operating frequency on this setup. The identification of these issues very early in the testing phase contributed to the following prototype revisions of the GR-HPCB-FMC-M2 mezzanine card to include corrections to address them.

The final experimental hardware setup consists of the entire HPCB prototype platform and is shown in Fig. 6.9. In this setup, the loopback application is once again used

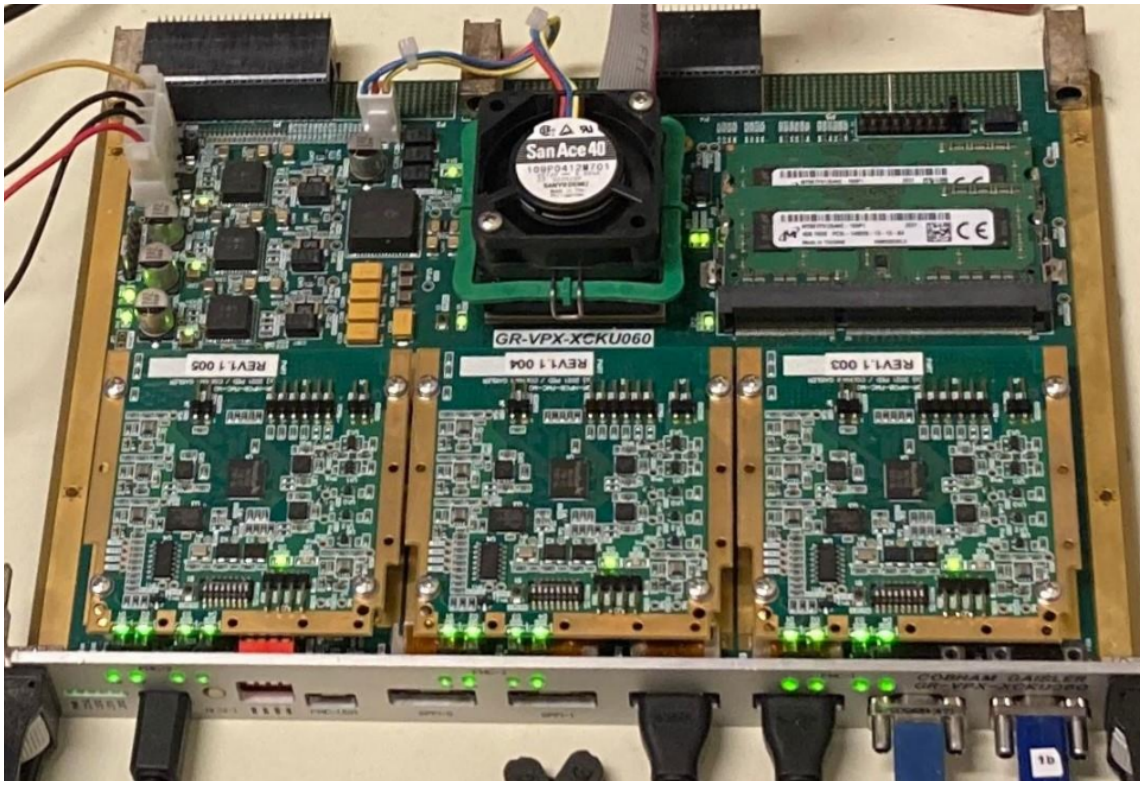

Figure 6.9: The evaluation setup with the GR-HPCB-FMC-M2 mezzanine card combined with the GR-VPX-XCKU060 carrier board.

to evaluate the maximum performance of the interfacing system standalone on the final prototype hardware and to validate the correct functionality of the fault-tolerance mechanism. For the CIF interface, error-free transmission of frames of 2048x2048 pixels of 16-bits each at 150 MHz is achieved which is the maximum frequency of the Myriad 2 CIF interface according to the specification. For the LCD interface, error-free transmission of frames with an ascending scale data pattern of 2048x2048 pixels of 16-bits each, at 150 MHz is achieved which is the maximum frequency of the Myriad 2 LCD interface according to the specification. For the same LCD frames with a checkerboard pattern error-free LCD transmission at 100 MHz is achieved, showing that the frame pattern can impact the maximum achieved frequency of the interface due to cross-talk issues even in the most advanced hardware setup of the campaign.

Chapter 7

Conclusions & Future Work

The current doctoral thesis focused on CNNs for computer vision applications in the context of their acceleration and deployment on edge devices. The thesis made distinct contributions in four different challenges regarding techniques for the efficient optimization of CNN models performance as well as techniques for CNN hardware accelerator architectures design.

First, the thesis proposed the SunMask generation image processing method, in order to improve the accuracy of image regression CNNs on the tasks of irradiance estimation and forecasting. The proposed image processing method utilized the solar coordinates in the sky as well as the mapping function of the lens of the fisheye camera and resulted in accurate identification of the position of the sun in the image in all cases. When the proposed image processing method was applied to the sky images before these were processed by the image regression CNNs, it was shown to improve the accuracy of the irradiance results produced by the CNNs. An extensive study on the irradiance estimation task, on four popular CNN models with various model sizes, as well as on the forecasting task, for three different forecast horizons, showed that the proposed SunMask generation method can consistently improve the accuracy of the irradiance results in all cases.

Furthermore, for the task of binary semantic segmentation for on-board cloud detection using satellite images, the thesis proposed a novel CNN model. The proposed model, called LD-UNet, was a lightweight variant of the, well-known for semantic segmentation, U-Net model which combined several CNN techniques such as depthwise separable convolutions, dilated convolutions and residual blocks in order to result in limited computational cost and memory footprint without significantly sacrificing semantic segmentation performance. The results of LD-UNet on a well-defined test dataset showcased that it can provide a competitive trade-off between CNN model size and semantic segmentation performance when compared to lightweight state-of-the-art models in the literature evaluated on the same test dataset.

Additionally, the thesis focused on porting the CNN models of the aforementioned

irradiance estimation and cloud detection applications, to the programmable engine of the Xilinx MPSoC FPGA, towards enabling their deployment in edge computing applications. The porting process and development flow was based on the Vitis AI framework and explored the capabilities of several quantization solutions in order to minimize the effects of quantization to the models as much as possible. Moreover, an acceleration approach was showcased for accelerating different processes of a single computer vision task by utilizing heterogeneous resources of the SoC FPGA. The acceleration of the CNN models on the edge-oriented FPGA resulted in real-time processing rates for the applications of CNN-based irradiance estimation from sky images as well as cloud detection from satellite imagery.

Finally, an interfacing system between an FPGA and a deep learning and image processing accelerator, the Myriad 2 VPU, was designed and developed. The interfacing system was developed in the context of a prototype on-board payload data processing architecture and platform, and took into account the high-throughput requirements of the system for the bi-directional data transfers between the FPGA and the Myriad 2. The interfacing system also implemented a fault-tolerance mechanism in order to support the system-level fault mitigation strategy of the entire HPCB platform. The extensive testing campaign performed during the thesis verified the interfacing system in commercial as well as prototype hardware platforms and helped identify issues in the prototype hardware early in the development and production phase.

The research conducted in the context of the thesis highlights the following directions for future work:

1. Regarding image processing and deep learning-based irradiance forecasting, more advanced deep learning models such as ConvLSTMs, which can process temporal sequences of images, could be studied and employed for improved irradiance forecasting accuracy.

2. Regarding CNNs for semantic segmentation of satellite imagery, the resources requirements of more complex CNNs specialized for hyperspectral images and for multiclass semantic segmentation should be studied and optimized towards their effcient deployment on on-board accelerators.

3. Regarding the acceleration approach of accelerating different processes on the heterogeneous resources of the SoC FPGA, even more techniques could be employed for accelerating additional processes of a computer vision task such as employing additional hardware kernels on the PL of the FPGA or using SIMD optimizations for processes executed on the ARM-based PS of the FPGA.

4. Regarding the high-throughput and fault-tolerant interface for the CNN accelerator, the data transfer times should be further evaluated in combination with extensive image processing benchmarks on the Myriad 2, in order to minimize the data transfers overhead.

List of Abbreviations

AI Artificial Intelligence.

ANN Artificial Neural Network.

API Application Programming Interface.

ASIC Application-Specific Integrated Circuit.

CDC Clock Domain Crossing.

CIF Camera Interface.

CNN Convolutional Neural Network.

COTS Commercial Off-The-Shelf.

CPU Central Processing Unit.

CRC Cyclic Redundancy Check.

CUDA Compute Unified Device Architecture.

DCNN Deep Convolutional Neural Network.

DHI Diffuse Horizontal Irradiance.

DNI Direct Normal Irradiance.

DPU Deep Learning Processor Unit.

DSP Digital Signal Processing.

EO Earth Observation.

EoT Eyes-of-Things.

ESA European Space Agency.

FF Fast Finetuning.

FN False Negative.

FP False Positive.

FPGA Field-Programmable Gate Array.

FPS Frames per Second.

FS Forecast Skill.

FSM Finite State Machine.

GHI Global Horizontal Irradiance.

GPU Graphics Processing Unit.

GT Ground Truth.

HDL Hardware Description Language.

HLS High-Level Synthesis.

HPCB High-Performance Compute Board.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

IoT Internet of Things.

IoU Intersection over Union.

LCD Liquid Crystal Display.

LD-UNet Lightweight Dilation UNet.

LSTM Long Short-Term Memory.

LUT Lookup Table.

M2 IF Myriad 2 Interface.

MAE Mean Absolute Error.

MLP Multilayer Perceptron.

MMCM Mixed-Mode Clock Manager.

MSE Mean Square Error.

nRMSE Normalized Root Mean Square Error.

ODDR Output Double Data Rate.

OLI Operational Land Imager.

PL Programmable Logic.

PM Persistence Model.

PS Processing Subsystem.

PTQ Post Training Quantization.

PV Photovoltaic.

QAT Quantization Aware Training.

ReLU Rectified Linear Unit.

RISC Reduced Instruction Set Computer.

RMSE Root Mean Square Error.

RTL Register-Transfer Level.

SI Sky Imager.

SIMD Single Instruction, Multiple Data.

SoC System-on-Chip.

SoM System-on-Module.

SVM Support Vector Machine.

SWaP-C Size, Weight, Power and Cost.

TIRS Thermal Infrared Sensor.

TMR Triple Modular Redundancy.

TN True Negative.

TP True Positive.

TPU Tensor Processing Unit.

VHDL VHSIC Hardware Description Language.

VLIW Very Long Instruction Word.

VPU Vision Processing Unit.

Bibliography

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Online]. Available: https://doi.org/10.1109/5.726791

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017. [Online]. Available: https://doi.org/10.1145/3065386

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015. [Online]. Available: https://arxiv.org/abs/1512.00567

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9. [Online]. Available: https://www.doi.org/10.1109/CVPR.2015.7298594

[7] Intel, "Intel Movidius Vision Processing Units (VPUs)," date accessed 01-03-2023. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html

[8] NVIDIA, "Jetson Nano Developer Kit," date accessed 01-03-2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[9] Xilinx, "Zynq UltraScale+ MPSoC," date accessed 01-03-2023. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

[10] 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. [Online]. Available: https://arxiv.org/abs/1704.04861

[11] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[12] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," 2017. [Online]. Available: https://arxiv.org/abs/1707.01083

[14] S. D. Miller, M. A. Rogers, J. M. Haynes, M. Sengupta, and A. K. Heidinger, "Short-term solar irradiance forecasting via satellite/model coupling," *Solar Energy*, vol. 168, pp. 102–117, 2018, advances in Solar Resource Assessment and Forecasting. [Online]. Available: https://doi.org/10.1016/j.solener.2017.11.049

[15] A. Ayet and P. Tandeo, "Nowcasting solar irradiance using an analog method and geostationary satellite images," *Solar Energy*, vol. 164, pp. 301–315, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X18301993

[16] L. E. Ordoñez Palacios, V. Bucheli Guerrero, and H. Ordoñez, "Machine Learning for Solar Resource Assessment Using Satellite Images," *Energies*, vol. 15, no. 11, p. 3985, May 2022. [Online]. Available: http://dx.doi.org/10.3390/en15113985

[17] B.-Y. Kim, J. W. Cha, and K.-H. Chang, "Twenty-four-hour cloud cover calculation using a ground-based imager with machine learning," *Atmospheric Measurement Techniques*, vol. 14, no. 10, pp. 6695–6710, 2021. [Online]. Available: https://amt.copernicus.org/articles/14/6695/2021/

[18] Q. Li, W. Lu, and J. Yang, "A Hybrid Thresholding Algorithm for Cloud Detection on Ground-Based Color Images," *Journal of Atmospheric and Oceanic Technology*, vol. 28, no. 10, pp. 1286 – 1296, 2011. [Online]. Available: https://journals.ametsoc.org/view/journals/atot/28/10/jtech-d-11-00009_1.xml

[19] R. A. Rajagukguk, R. Kamil, and H.-J. Lee, "A Deep Learning Model to Forecast Solar Irradiance Using a Sky Camera," *Applied Sciences*, vol. 11, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/5049

[20] H.-M. Zuo, J. Qiu, Y.-H. Jia, Q. Wang, and F.-F. Li, "Ten-minute prediction of solar irradiance based on cloud detection and a long short-term memory

(LSTM) model," *Energy Reports*, vol. 8, pp. 5146–5157, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484722007375

[21] Y. Chu, M. Li, and C. F. Coimbra, "Sun-tracking imaging system for intra-hour DNI forecasts," *Renewable Energy*, vol. 96, pp. 792–799, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148116304529

[22] A. Niccolai and A. Nespoli, "Sun Position Identification in Sky Images for Nowcasting Application," *Forecasting*, vol. 2, no. 4, p. 488–504, Nov 2020. [Online]. Available: http://dx.doi.org/10.3390/forecast2040026

[23] Q. Paletta and J. Lasenby, "A Temporally Consistent Image-based Sun Tracking Algorithm for Solar Energy Forecasting Applications," in *NeurIPS 2020 Workshop on Tackling Climate Change with Machine Learning*, 2020. [Online]. Available: https://www.climatechange.ai/papers/neurips2020/8

[24] Q. Paletta, G. Arbod, and J. Lasenby, "Benchmarking of deep learning irradiance forecasting models from sky images – An in-depth analysis," *Solar Energy*, vol. 224, pp. 855–867, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X21004266

[25] H. Wen, Y. Du, X. Chen, E. Lim, H. Wen, L. Jiang, and W. Xiang, "Deep Learning Based Multistep Solar Forecasting for PV Ramp-Rate Control Using Sky Images," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1397–1406, 2021.

[26] H. Jiang, Y. Gu, Y. Xie, R. Yang, and Y. Zhang, "Solar Irradiance Capturing in Cloudy Sky Days–A Convolutional Neural Network Based Image Regression Approach," *IEEE Access*, vol. 8, pp. 22 235–22 248, 2020.

[27] F. Wang, Z. Zhang, H. Chai, Y. Yu, X. Lu, T. Wang, and Y. Lin, "Deep Learning Based Irradiance Mapping Model for Solar PV Power Forecasting Using Sky Image," in *2019 IEEE Industry Applications Society Annual Meeting*, Sep. 2019, pp. 1–9.

[28] S. Song, Z. Yang, H. Goh, Q. Huang, and G. Li, "A novel sky image-based solar irradiance nowcasting model with convolutional block attention mechanism," *Energy Reports*, vol. 8, pp. 125–132, 2022, iCPE 2021 - The 2nd International Conference on Power Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484722004127

[29] G. Bahl, L. Daniel, M. Moretti, and F. Lafarge, "Low-Power Neural Networks for Semantic Segmentation of Satellite Images," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 2469–2476.

[30] L. Jiao and W. Gao, "Refined UNet Lite: End-to-End Lightweight Network for Edge-precise Cloud Detection," *Procedia Computer Science*, vol. 202, pp. 9–14, 2022, international Conference on Identification, Information and Knowledge in

the internet of Things, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922005361

[31] J. Zhang, X. Li, L. Li, P. Sun, X. Su, T. Hu, and F. Chen, "Lightweight U-Net for Cloud Detection of Visible and Thermal Infrared Remote Sensing Images," *Optical and Quantum Electronics*, vol. 52, no. 9, p. 397, Sep 2020. [Online]. Available: https://doi.org/10.1007/s11082-020-02500-8

[32] S. I. Venieris and C.-S. Bouganis, "fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs," in *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. Institute of Electrical and Electronics Engineers (IEEE), May 2016, pp. 40–47. [Online]. Available: http://dx.doi.org/10.1109/FCCM.2016.22

[33] S. I. Venieris and C. S. Bouganis, "fpgaConvNet: Mapping Regular and Irregular Convolutional Neural Networks on FPGAs," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–17, 2018.

[34] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–74. [Online]. Available: https://doi.org/10.1145/3020078.3021744

[35] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, and K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, dec 2018. [Online]. Available: https://doi.org/10.1145/3242897

[36] V. Rybalkin, A. Pappalardo, M. Ghaffar, G. Gambardella, N. Wehn, and M. Blott, "Finn-l: Library extensions and design trade-off analysis for variable precision lstm networks on fpgas," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. Los Alamitos, CA, USA: IEEE Computer Society, aug 2018, pp. 89–897. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/FPL.2018.00024

[37] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 35–47, 2018.

[38] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to fpgas," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 1–12.

[39] V. Leon, G. Lentaris, E. Petrongonas, D. Soudris, G. Furano, A. Tavoularis, and D. Moloney, "Improving Performance-Power-Programmability in Space Avionics with Edge Devices: VBN on Myriad2 SoC," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 3, mar 2021. [Online]. Available: https://doi.org/10.1145/3440885

[40] C. Adams, A. Spain, J. Parker, M. Hevert, J. Roach, and D. Cotten, "Towards an Integrated GPU Accelerated SoC as a Flight Computer for Small Satellites," in *2019 IEEE Aerospace Conference*, 2019, pp. 1–7.

[41] F. C. Bruhn, N. Tsog, F. Kunkel, O. Flordal, and I. Troxel, "Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space," *CEAS Space Journal*, vol. 12, no. 4, pp. 551–564, Dec 2020. [Online]. Available: https://doi.org/10.1007/s12567-020-00321-9

[42] J. España Navarro, A. Samuelsson, H. Gingsjö, J. Barendt, A. Dunne, L. Buckley, D. Reisis, A. Kyriakos, E. A. Papatheofanous, C. Bezaitis, P. Matthijs, J. P. Ramos, and D. Steenari, "High-Performance Compute Board - A Fault-Tolerant Module for On-Boards Vision Processing," in *OBDP2021 - 2nd European Workshop on On-Board Data Processing (OBDP2021)*, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5521624

[43] R. C. Amorim, R. Martins, P. Harikrishnan, M. Ghiglione, and T. Helfers, "Dependable MPSoC framework for mixed criticality applications," in *OBDP2021 - 2nd European Workshop on On-Board Data Processing (OBDP2021)*, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5521521

[44] P. Kuligowski, G. Gajoch, M. Nowak, and W. Sładek, "System-level hardening techniques used in the COTS-based data processing unit," in *OBDP2021 - 2nd European Workshop on On-Board Data Processing (OBDP2021)*, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5521521

[45] M. Y. Mehmood, A. Oad, M. Abrar, H. M. Munir, S. F. Hasan, H. A. u. Muqeet, and N. A. Golilarz, "Edge Computing for IoT-Enabled Smart Grid," *Security and Communication Networks*, vol. 2021, p. 5524025, Jul 2021. [Online]. Available: https://doi.org/10.1155/2021/5524025

[46] C. Feng, Y. Liu, and J. Zhang, "A taxonomical review on recent artificial intelligence applications to PV integration into power grids," *International Journal of Electrical Power & Energy Systems*, vol. 132, p. 107176, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142061521004154

[47] H. Zsiborács, N. H. Baranyai, A. Vincze, L. Zentkó, Z. Birkner, K. Máté, and G. Pintér, "Intermittent Renewable Energy Sources: The Role of Energy Storage in the European Power System of 2040," *Electronics*, vol. 8, no. 7, p. 729, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/7/729

[48] X. Chen, Y. Du, E. Lim, L. Fang, and K. Yan, "Towards the applicability of solar nowcasting: A practice on predictive PV power ramp-rate control," *Renewable Energy*, vol. 195, pp. 147–166, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148122008266

[49] F. Lin, Y. Zhang, and J. Wang, "Recent advances in intra-hour solar forecasting: A review of ground-based sky image methods," *International Journal of Forecasting*, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016920702100176X

[50] B. Juncklaus Martins, A. Cerentini, S. L. Mantelli, T. Z. Loureiro Chaves, N. Moreira Branco, A. von Wangenheim, R. Rüther, and J. Marian Arrais, "Systematic review of nowcasting approaches for solar energy production based upon ground-based cloud imaging," *Solar Energy Advances*, vol. 2, p. 100019, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667113122000079

[51] T. E. Pickering, "The MMT all-sky camera," in *Ground-based and Airborne Telescopes*, L. M. Stepp, Ed., vol. 6267, International Society for Optics and Photonics. SPIE, 2006, pp. 448 – 454. [Online]. Available: https://doi.org/10.1117/12.672508

[52] H. Carreira Pedro, D. Larson, and C. Coimbra, "A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2826939

[53] A. Andreas and T. Stoffel, "REL Solar Radiation Research Laboratory (SRRL): Baseline Measurement System (BMS); Golden, Colorado (Data); NREL Report No. DA-5500-56488," 2019. [Online]. Available: http://dx.doi.org/10.5439/1052221

[54] M. P. Véstias, R. P. Duarte, J. T. de Sousa, and H. C. Neto, "Moving Deep Learning to the Edge," *Algorithms*, vol. 13, no. 5, p. 125, 2020. [Online]. Available: https://www.mdpi.com/1999-4893/13/5/125

[55] Q. Paletta, A. Hu, G. Arbod, and J. Lasenby, "ECLIPSE: Envisioning CLoud Induced Perturbations in Solar Energy," *Applied Energy*, vol. 326, p. 119924, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261922011813

[56] Q.-K. Tran and S.-k. Song, "Computer Vision in Precipitation Nowcasting: Applying Image Quality Assessment Metrics for Training Deep Neural Networks," *Atmosphere*, vol. 10, no. 5, p. 244, 2019. [Online]. Available: https://www.mdpi.com/2073-4433/10/5/244

[57] V. Le Guen and N. Thome, "A Deep Physical Model for Solar Irradiance Forecasting with Fisheye Images," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 2685–2688.

[58] H. T. C. Pedro, D. P. Larson, and C. F. M. Coimbra, "A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods," *Journal of Renewable and Sustainable Energy*, vol. 11, no. 3, p. 036102, 2019.

[59] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, "pvlib python: a python package for modeling solar energy systems," *Journal of Open Source Software*, vol. 3, no. 29, p. 884, 2018. [Online]. Available: https://doi.org/10.21105/joss.00884

[60] R. Naushad, T. Kaur, and E. Ghaderpour, "Deep Transfer Learning for Land Use and Land Cover Classification: A Comparative Study," *Sensors*, vol. 21, no. 23, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/8083

[61] R. S. Reddy, D. Lu, F. Tuluri, and M. Fadavi, "Simulation and prediction of hurricane Lili during landfall over the central gulf states using MM5 modeling system and satellite data," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 36–39.

[62] G. Furano, G. Meoni, A. Dunne, D. Moloney, V. Ferlet-Cavrois, A. Tavoularis, J. Byrne, L. Buckley, M. Psarakis, K.-O. Voss, and L. Fanucci, "Towards the Use of Artificial Intelligence on the Edge in Space Systems: Challenges and Opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, pp. 44–56, 2020.

[63] Mohajerani, Sorour, "95-Cloud: An Extension to 38-Cloud Dataset." [Online]. Available: https://github.com/SorourMo/95-Cloud-An-Extension-to-38-Cloud-Dataset

[64] Sorour Mohajerani. 38-Cloud: A Cloud Segmentation Dataset. [Online]. Available: https://github.com/SorourMo/38-Cloud-A-Cloud-Segmentation-Dataset

[65] S. Mohajerani and P. Saeedi, "Cloud and Cloud Shadow Segmentation for Remote Sensing Imagery Via Filtered Jaccard Loss Function and Parametric Augmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 4254–4266, 2021. [Online]. Available: https://doi.org/10.1109%2Fjstars.2021.3070786

[66] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241.

[67] S. Mohajerani and P. Saeedi, "Cloud-Net: An end-to-end Cloud Detection Algorithm for Landsat 8 Imagery," 2019. [Online]. Available: https://arxiv.org/abs/1901.10077

[68] Xilinx, "Vitis AI User Guide UG1414 (v2.5)," https://docs.xilinx.com/r/en-US/ug1414-vitis-ai, June 2022, date accessed 01-03-2023.

[69] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1325–1334.

[70] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving Post Training Neural Quantization: Layer-wise Calibration and Integer Programming," 2020. [Online]. Available: https://arxiv.org/abs/2006.10518

[71] Xilinx, "Vitis Vision Library," https://www.xilinx.com/products/design-tools/vitis/vitis-libraries/vitis-vision.html, date accessed 01-03-2023.

[72] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[73] G. Lentaris, I. Stratakos, I. Stamoulias, D. Soudris, M. Lourakis, and X. Zabulis, "High-performance vision-based navigation on soc fpga for spacecraft proximity operations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 1188–1202, 2020.

[74] V. Leon, G. Lentaris, E. Petrongonas, D. Soudris, G. Furano, A. Tavoularis, and D. Moloney, "Improving Performance-Power-Programmability in Space Avionics with Edge Devices: VBN on Myriad2 SoC," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 3, mar 2021. [Online]. Available: https://doi.org/10.1145/3440885

[75] Cobham Gaisler, "GR716 - LEON3FT Microcontroller," https://www.gaisler.com/index.php/products/components/gr716, date accessed 01-03-2023.

[76] Xilinx, "Kintex Ultrascale FPGAs," https://www.xilinx.com/products/silicon-devices/fpga/kintex-ultrascale.html, date accessed 01-03-2023.

List of Publications

[1] C. Vagionas, R. Maximidis, I. Stratakos, A. Margaris, A. Mesodiakaki, M. Gatzianas, K. Kanta, P. Toumasis, G. Giannoulis, D. Apostolopoulos, E. A. Papatheofanous, G. Lentaris, D. Reisis, D. Soudris, K. Tsagkaris, N. Argyris, D. Syrivelis, P. Bakopoulos, R. M. Oldenbeuving, C. G. H. Roeloffzen, P. W. L. van Dijk, I. Dimogiannis, A. Kontogiannis, H. Avramopoulos, A. Miliou, N. Pleros, and G. Kalfas, "End-to-End Real-Time Service Provisioning over a SDN-controllable analog mmWave Fiber-Wireless 5G X-haul Network," *Journal of Lightwave Technology*, pp. 1–10, 2023. [Online]. Available: https://doi.org/10.1109/JLT.2023.3234365

[2] E. A. Papatheofanous, V. Kalekis, G. Venitourakis, F. Tziolos, and D. Reisis, "Deep Learning-Based Image Regression for Short-Term Solar Irradiance Forecasting on the Edge," *Electronics*, vol. 11, no. 22, 2022. [Online]. Available: https://doi.org/10.3390/electronics11223794

[3] E. A. Papatheofanous, P. Tziolos, V. Kalekis, T. Amrou, G. Konstantoulakis, G. Venitourakis, and D. Reisis, "SoC FPGA Acceleration for Semantic Segmentation of Clouds in Satellite Images," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2022, pp. 1–4. [Online]. Available: https://doi.org/10.1109/VLSI-SoC54400.2022.9939585

[4] V. Leon, E. A. Papatheofanous, G. Lentaris, C. Bezaitis, N. Mastorakis, G. Bampilis, D. Reisis, and D. Soudris, "Combining Fault Tolerance Techniques and COTS SoC Accelerators for Payload Processing in Space," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/VLSI-SoC54400.2022.9939621

[5] A. Kyriakos, E. A. Papatheofanous, C. Bezaitis, and D. Reisis, "Resources and Power Efficient FPGA Accelerators for Real-Time Image Classification," *Journal of Imaging*, vol. 8, no. 4, 2022. [Online]. Available: https://doi.org/10.3390/jimaging8040114

[6] K. Kanta, P. Toumasis, K. Tokas, I. Stratakos, E. A. Papatheofanous, G. Giannoulis, I. Mesogiti, E. Theodoropoulou, G. Lyberopoulos, G. Lentaris, D. Apostolopoulos,

D. Reisis, D. Soudris, and H. Avramopoulos, "Demonstration of a Hybrid Analog-Digital Transport System Architecture for 5G and Beyond Networks," *Applied Sciences*, vol. 12, no. 4, 2022. [Online]. Available: https://doi.org/10.3390/app12042122

[7] C. Vagionas, R. Maximidis, I. Stratakos, A. Margaris, A. Mesodiakaki, M. Gatzianas, K. Kanta, P. Toumasis, G. Giannoulis, D. Apostolopoulos, E. A. Papatheofanous, G. Lentaris, D. Reisis, D. Soudris, K. Tsagkaris, N. Argyris, D. Syrivelis, P. Bakopoulos, R. M. Oldenbeuving, C. G. H. Roeloffzen, P. W. L. van Dijk, I. Dimogiannis, A. Kontogiannis, H. Avramopoulos, A. Miliou, N. Pleros, and G. Kalfas, "End-to-End Real-Time Service Provisioning over a SDN-controllable 60 GHz analog FiWi X-haul for 5G Hot-Spot Networks," in *Optical Fiber Communication Conference (OFC) 2022*, 2022, p. Th4A.7. [Online]. Available: https://doi.org/10.1364/OFC.2022.Th4A.7

[8] V. Leon, C. Bezaitis, G. Lentaris, D. Soudris, D. Reisis, E. A. Papatheofanous, A. Kyriakos, A. Dunne, A. Samuelsson, and D. Steenari, "FPGA & VPU Co-Processing in Space Applications: Development and Testing with DSP/AI Benchmarks," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2021, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ICECS53924.2021.9665462

[9] J. España Navarro, A. Samuelsson, H. Gingsjö, J. Barendt, A. Dunne, L. Buckley, D. Reisis, A. Kyriakos, E. A. Papatheofanous, C. Bezaitis, P. Matthijs, J. P. Ramos, and D. Steenari, "High-Performance Compute Board - A Fault-Tolerant Module for On-Boards Vision Processing," in *OBDP2021 - 2nd European Workshop on On-Board Data Processing (OBDP2021)*, 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5521624

[10] E. A. Papatheofanous, D. Reisis, and K. Nikitopoulos, "LDPC Hardware Acceleration in 5G Open Radio Access Network Platforms," *IEEE Access*, vol. 9, pp. 152 960–152 971, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3127039

[11] P. Toumasis, K. Kanta, K. Tokas, I. Stratakos, E. A. Papatheofanous, G. Giannoulis, I. Mesogiti, E. Theodoropoulou, G. Lyberopoulos, G. Lentaris, D. Apostolopoulos, D. Reisis, D. Soudris, and H. Avramopoulos, "Demonstration of FPGA-based A-IFoF/mmWave transceiver integration in mobile infrastructure for beyond 5G transport," in *2021 European Conference on Optical Communication (ECOC)*, 2021, pp. 1–4. [Online]. Available: https://doi.org/10.1109/ECOC52684.2021.9606061

[12] E. A. Papatheofanous, D. Reisis, and K. Nikitopoulos, "The LDPC Challenge in Software-Based 5G New Radio Physical Layer Processing," in *2021 IEEE International Mediterranean Conference on Communications and Networking*

(MeditCom), 2021, pp. 312–317. [Online]. Available: https://doi.org/10.1109/MeditCom49071.2021.9647697

[13] I. Stratakos, E. A. Papatheofanous, D. Danopoulos, G. Lentaris, D. Reisis, and D. Soudris, "Towards sharing one FPGA SoC for both low-level PHY and high-level AI/ML computing at the edge," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021, pp. 76–81. [Online]. Available: https://doi.org/10.1109/MeditCom49071.2021.9647576

[14] A. Kyriakos, V. Kitsakis, A. Louropoulos, E. A. Papatheofanous, I. Patronas, and D. Reisis, "High Performance Accelerator for CNN Applications," in *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2019, pp. 135–140. [Online]. Available: https://doi.org/10.1109/PATMOS.2019.8862166

[15] A. Kyriakos, E. A. Papatheofanous, B. Charalampos, E. Petrongonas, D. Soudris, and D. Reisis, "Design and Performance Comparison of CNN Accelerators Based on the Intel Movidius Myriad2 SoC and FPGA Embedded Prototype," in *2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, 2019, pp. 142–147. [Online]. Available: https://doi.org/10.1109/ICCAIRO47923.2019.00030

[16] V. Kitsakis, E. A. Papatheofanous, D. Reisis, G. Lentaris, and D. Soudris, "Parity Based In-Place FFT Architecture for Continuous Flow Applications," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 97–100. [Online]. Available: https://doi.org/10.1109/ICECS.2018.8617990