cREAtIve: Reconfigurable Embedded Artificial Intelligence

Invited Paper

Poona Bahrebar
Ghent University
Ghent, Belgium
poona.bahrebar@ugent.be

Leon Denis
ETRO—Vrije Universiteit Brussel
Brussels, Belgium
ldenis@etrovub.be

Maxim Bonnaerens
Ghent University—imec—IDLab
Ghent, Belgium
maxim.bonnaerens@ugent.be

Kristof Coddens
Melexis
Ieper, Belgium
kco@melexis.com

Joni Dambre
Ghent University—imec—IDLab
Ghent, Belgium
joni.dambre@ugent.be

Wouter Favoreel
Flir Systems Trading
Meer, Belgium
wouter.favoreel@flir.com

Illia Khvastunov
Melexis
Ieper, Belgium
ikh@melexis.com

Adrian Munteanu
ETRO—Vrije Universiteit Brussel
Brussels, Belgium
acmuntea@etrovub.be

Hung Nguyen-Duc
XenomatiX
Leuven, Belgium
hung.nguyenduc@xenomatix.com

Stefan Schulte
Flir Systems Trading
Meer, Belgium
stefan.schulte@flir.com

Dirk Stroobandt
Ghent University
Ghent, Belgium
dirk.stroobandt@ugent.be

Ramses Valvekens
easics
Leuven, Belgium
ramses@easics.be

Nick Van den Broeck
XenomatiX
Leuven, Belgium
nick.vandenbroeck@xenomatix.com

Geert Verbruggen
easics
Leuven, Belgium
geert@easics.be

Abstract

cREAtIve targets the development of novel highly-adaptable embedded deep learning solutions for automotive and traffic monitoring applications, including position sensor processing, scene interpretation based on LiDAR, and object detection and classification in thermal images for traffic camera systems. These applications share the need for deep learning solutions tailored for deployment on embedded devices with limited resources and featuring high adaptability and robustness to changing environmental conditions. cREAtIve develops knowledge, tools and methods that enable hardware-efficient, adaptable, and robust deep learning.

CCS Concepts

• **Computing methodologies** → **Neural networks**; **Machine learning**; **Artificial intelligence**; • **Computer systems organization** → **Sensors and actuators**; **Embedded systems**; • **Hardware** → **Reconfigurable logic applications**; **Robustness**.

CF '21, May 11–13, 2021, Virtual Conference, Italy

ACM ISBN 978-1-4503-8404-9/21/05…$15.00
https://doi.org/10.1145/3457388.3458857

Keywords

Deep learning, Automotive and traffic monitoring, Object detection and classification, Embedded devices, Position sensors, Point cloud processing, Reconfiguration, Robustness

ACM Reference Format:
Poona Bahrebar, Leon Denis, Maxim Bonnaerens, Kristof Coddens, Joni Dambre, Wouter Favoreel, Illia Khvastunov, Adrian Munteanu, Hung Nguyen-Duc, Stefan Schulte, Dirk Stroobandt, Ramses Valvekens, Nick Van den Broeck, and Geert Verbruggen. 2021. cREAtIve: Reconfigurable Embedded Artificial Intelligence: Invited Paper. In *Computing Frontiers Conference (CF '21), May 11–13, 2021, Virtual Conference, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3457388.3458857

1 Introduction

Artificial Intelligence (AI) and in particular Deep Learning (DL) solutions have been successfully introduced in many domains for their efficiency, especially in classification tasks. Systems that involve embedded sensors, such as security cameras and self-driving cars, require state-of-the-art DL algorithms to perform robust detection, tracking, recognition, or navigation. However, such systems possess limited bandwidth, have zero latency tolerance, and are constrained by privacy issues. They therefore require the use of *specific embedded hardware* and cannot rely on cloud-based solutions. Consequently, tailored DL solutions must be developed to meet the constraints on accuracy, data storage, power consumption, and latency to cope with *limited resources*.

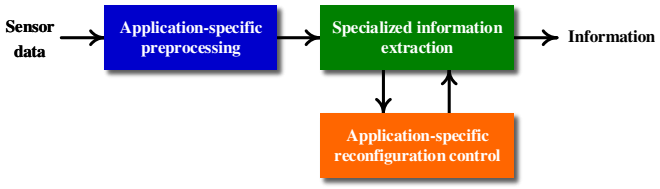


Figure 1: Splitting up the highly complex problems into simpler neural network problems.

The project cREAtIve [1] is built around the development of novel **highly-adaptable embedded DL solutions for automotive and traffic monitoring applications**. In particular, cREAtIve will improve the state-of-the-art in the following three applications:

- Stable sensor measurements for position sensors,
- Scene interpretation based on LiDAR point clouds,
- Object detection and classification in thermal images for traffic camera systems.

These applications share the need for a DL implementation tailored to deployment on *embedded devices with limited resources* while still being *robust* to changing environmental conditions.[1]

The core of our project vision assumes that most available DL networks are over-dimensioned. Since they are designed to be generally applicable, they are much more powerful than what is needed for most specific applications. As shown in Fig. 1, we aim to achieve robustness by exploiting application-specific information. We use Neural Networks (NNs) to automatically remove as much variability at the input side as possible. In addition, we allow the remaining information extraction system to specialize to the stable aspects of its operating conditions and to automatically reconfigure when these conditions change. In this way, our methodology effectively decomposes a highly complex problem into a combination of simpler, and therefore, less computation-intensive NNs.

2 Consortium as a Whole

The cREAtIve project centers around the common research needs of the three applications (i.e. sensors, point cloud processing, and object detection in thermal images), which will be optimized by the respective application partners (i.e. Melexis [2], XenomatiX [3], and FLIR [4]), with the help of the research partners.

The three research lines at the core of cREAtIve are (i) DL solutions that have to be implemented on (ii) embedded systems with limited resources, and with a specific focus on (iii) adapting the systems to changing environmental conditions. These three research lines are fully covered by the research partners (i.e. ETRO-VUB [5] for specific DL algorithms, DSLab-UGent [6] for DL implementations, and HES-UGent [7] for adaptability). easics [8] also sits at the core of the project with its expertise in resource-limited implementations.

[1] An example of where *specialized learning* can be particularly beneficial is traffic monitoring, which has to be robust to highly variable weather conditions and various environments where the cameras are placed.

3 Innovation Goals and Recent Achievements

3.1 Robust AI-empowered Sensors

Traditional Hall effect and magnetoresistive sensors are widely used in the automotive industry. Typical use cases include determining vehicle velocity, combustion engine ignition timing, anti-lock braking systems, etc. The simplicity and accuracy of hall sensors made them very successful, but they tend to be more and more affected by stray magnetic fields present in vehicles, resulting in perturbations of the sensor readouts. These fields are produced by electric currents that can be particularly large (e.g. currents of hundreds of amperes to power the traction motors).

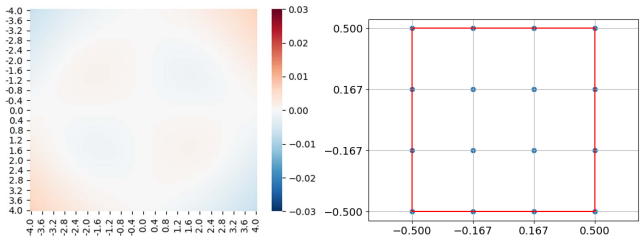
In cREAtIve, we have investigated AI-based solutions to filter out perturbations in the hall effect sensor of Melexis. Concretely, a NN is designed to predict one-dimensional (1D) movements using noisy hall sensor readouts.

3.1.1 Data Generation — In order to train a NN, synthetic data was generated through physically-based simulations. A visualization of the simulated magnetic field for a 2-pole magnet is provided in Fig. 2a. The x and y axes represent the spatial position of the field. Its strength along the x-axis, denoted as B_x, is color coded. Similar data are computed for B_y and B_z. Final sensor readouts are obtained by sampling, transforming and interpolating those fields based on the magnet positions, orientations, and sensor configurations.

An example of a sensor consisting of a rectangular 1 mm^2 die with the hall plates uniformly distributed is also shown in Fig. 2b. As a proof of concept, 1D magnet movements are predicted by the NN. In our experiments, the x-axis is taken as the to-measure axis. Gaussian noise is added along the y-axis, rotational orientation of the magnet, and sensor readouts.

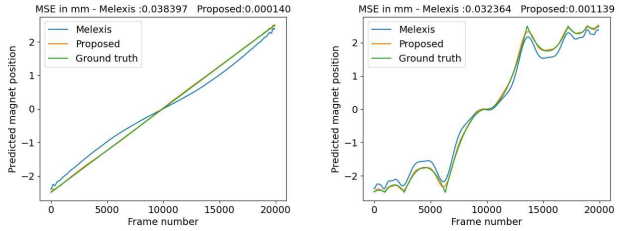
3.1.2 Experimental Assessment — The designed NN is compared with one of the state-of-the-art Hall effect sensors of Melexis. To preserve potential proprietary technology, no details regarding the network architecture are disclosed.

For our experiments, we have employed a uniform 1 mm^2 2×2 sensor with the Hall plates being sensitive along the z-axis. We note that this is merely a test case, and that other configurations might produce superior results. We also remark that the flexibility in sensor layout is enabled by the ability of the NN to produce accurate magnet predictions regardless of Hall plate placement, unlike traditional solutions which enforce specific hall plate positioning, such as the arc-tangent method. The results when predicting the



(a) (b)

Figure 2: (a) The simulated magnetic field and (b) an example sensor of 1 mm^2 with 16 hall plates used for data generation.



(a) (b)

Figure 3: Comparison of the prediction accuracy of the proposed NN and Melexis' current solution for a 2×2 hall sensor with 1 mm^2 die.

magnet trajectories are depicted in Fig. 3. The same network was used for both test cases. From the figure, it is clear that the proposed AI-based solution vastly outperforms the state-of-the-art. Additional experiments also revealed that results were consistent irrespective of the magnet trajectory.

3.2 Deep Learning for Point Cloud Processing

The advent of geometric DL has led to the development of novel methods that generalize structured deep neural models to non-Euclidean domains (e.g. point clouds [15], graphs [23], and manifolds [19]). The ability to apply Deep NNs (DNNs) on point clouds is particularly important for the automotive industry due to the rising popularity of LiDAR for navigation purposes. Due to their underlying data capturing methods, raw point clouds captured by LiDAR (or any Time-of-Flight camera) inherently exhibit noise. Those perturbed points make it more difficult to perform operations such as object detection and point cloud registration, especially for objects at larger distances. In this context, the main contribution of cREAtIve is designing a digital twin of XenomatiX's XenoLidar [3], allowing the generation of sufficient and diverse training data for DL-based noise removal. Unlike previous LiDAR simulations [16, 22], the proposed method exploits GPU rendering capabilities to significantly reduce the time necessary for acquiring synthetic data.

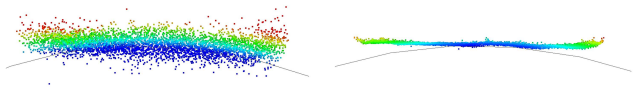
3.2.1 GPU-Accelerated Data Acquisition — The principle idea behind the LiDAR simulation is reversing the 3D rendering pipeline employing data stored in the framebuffer. The LiDAR is simulated by sampling the framebuffer and computing its corresponding 3D location in camera space using the z-buffer. Each such sample corresponds to a ray being cast in the scene with the position of the LiDAR coinciding with the camera. Camera parameters should be chosen to mimic the LiDAR's field of view.

Generating datasets using the previously described method produces clean point clouds, which are regarded as the ground truth by the NN. Training data is generated by applying Gaussian noise to each point with respect to the ray's directions. Similar to real captured data, the amplitude of the noise is a function of the distance with respect to the emitter. Compared to the full simulation of XenomatiX that relies on raycasting, the aforementioned simulation speeds up data acquisition thousandfold. We note though that deviations from the true LiDAR exist. The digital twin uniformly casts rays, which does not reflect the XenoLidar's sampling. Furthermore, the small baseline between receiver and emitter is not considered. Nevertheless, our experiments show that the proposed method proves sufficiently accurate for creating robust models for denoising purposes of real scenes captured by the XenoLidar.

3.2.2 Experimental Results — The architecture of the denoising NN was inspired by [13]. Its design, however, lies outside the scope of cREAtIve and is therefore not detailed. Rather, we will evaluate the synthetic data and its capability of training NNs operating on real data. To do so, two different scenarios are investigated, the first one being a controlled environment used to calibrate the XenoLidar. Specifically, this setup consists of the LiDAR being placed 20 meters in front of a calibration wall.

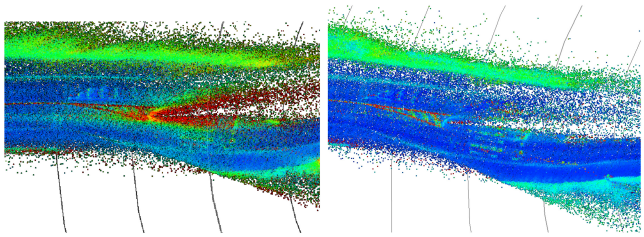
A top-down view of the noisy and denoised point clouds for one specific frame captured by the XenoLidar is provided in Fig. 4. The colors indicate the measured distance with blue starting at 20m and red being 21m and beyond. The grey line indicates a distance of 20m. The figure clearly demonstrates the denoising capability of the NN, hence, significantly improving the measurements of the calibration wall.

The second dataset comprises a real driving scenario. For this scene, no ground truth is known. Figure 5 shows a top-down view of the captured scene. Unlike the previous experiment, multiple frames are stitched, producing a denser point cloud. The colors indicate the amount of reflected signal, with blue and red indicating low and high reflectivity, respectively. When closely examining the figure, it is clear that more structure is obtained in the point clouds after denoising. Features such as road markings and pedestrian crossings are much more refined in Fig. 5b compared to Fig. 5a.



(a) (b)

Figure 4: The (a) input and (b) denoised point cloud results obtained for the wall calibration test scene.



(a) (b)

Figure 5: The results obtained for stitching real world LiDAR data (a) prior and (b) after denoising.

3.3 Design of novel Deep Learning Object Detection and Classification Algorithms for Thermal Imaging

Training DNNs for specific hardware is particularly difficult due to the lack of sufficient training data. On the other hand, manually annotating the data is very tedious and time consuming, and in many cases barely feasible due to the vast amount of data required. As shown in Section 3.2, a possible solution is to generate synthetic data through the use of a digital twin. Another solution is to automate the labeling process.

For this use case, cREAtIve has investigated the automated labeling of thermal images using NNs operating on the RGB spectrum. Real-life thermal and RGB video sequences are simultaneously captured by a single FLIR camera. Labels are extracted from the RGB data and projected to the thermal domain. The main difficulty here lies with the thermal and conventional RGB camera having different camera specifications (e.g. focal length, field-of-view, etc). Using the acquired labels, a NN can be trained specifically for thermal images.

3.3.1 Data Projection — Since the intrinsics and extrinsics are different for both cameras, a simple affine transformation does not suffice for mapping the labels from the RGB spectrum to the thermal domain. Rather, a projection must be applied.

We compute the transformation matrix **T** that minimizes the distances of points in the RGB domain and their projections in the thermal image by solving an optimization problem. To facilitate this process, visually distinctive points (such as the corner of a car or top front mirror) are chosen. We empirically determined that computing a reliable transformation matrix requires roughly 100 points, ideally covering a large portion of the image planes. It is important to note that a different camera configuration would require a different **T**. For this work, each reference point was carefully selected manually. This process, however, can be automated using standard computer vision techniques.

3.3.2 Experimental Results — For evaluating the proposed method, labels were extracted from the RGB video sequence using the model from [24] trained on the COCO dataset [18]. Next, the acquired labels are projected to the thermal channel using the techniques described previously and a NN operating solely on the thermal spectrum was trained. We note that for the employed video sequences, RGB and thermal streams are perfectly synchronized in time. No time synchronization is therefore necessary.

Figure 6 provides results when using the RGB and thermal networks. The labels extracted from the RGB domain are shown in Fig. 6a, whereas their projection to the thermal spectrum is given in Fig. 6b. Results obtained when applying the NN designed specifically for thermal data is shown in Fig. 6c. Finally, the results obtained from inference on the thermal channel using the RGB network is shown in Fig. 6d. Note that the projected labels are indeed of high quality, but not always capture the full object in the thermal domain due to visibility differences of both cameras. Yet, the NN trained specifically for thermal data using those labels does detect the complete car.

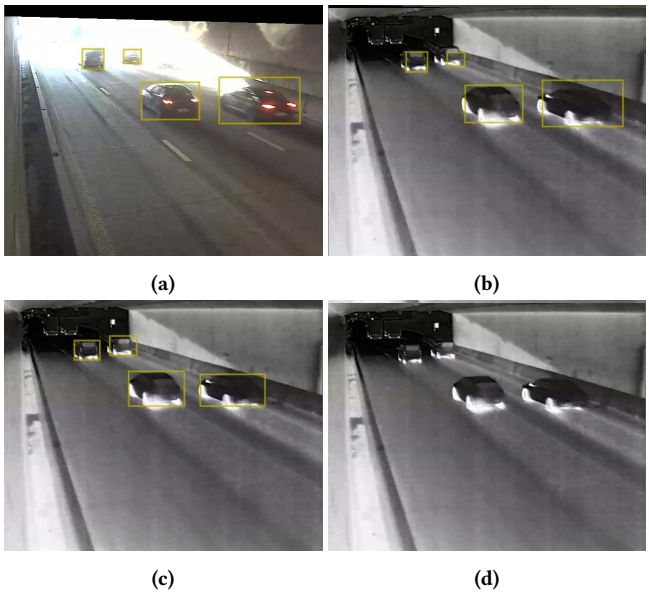


Figure 6: The results obtained by the NNs. The acquired labels for RGB (a) are projected on the thermal image (b). Inference for an infrared image using NNs trained for thermal and RGB data are shown in (c) and (d), respectively. Note that the RGB network fails to detect any cars when applied on thermal data.

3.4 Hardware Trade-offs for Deep Networks

In embedded systems, resources and power consumption are constrained. At the same time, we need low latency implementations. This drives embedded systems for vision applications towards hardware solutions specifically tailored to the problem at hand.

cREAtIve investigates the benefits of FPGA implementations against more mainstream architectures such as GPUs, DSPs, or processors. DL networks implemented on an embedded system require many trade-offs to be made to ensure the network adheres to the physical constraints. These trade-offs in the context of FPGA implementations are quantified to be used for the network training algorithm. We also ensure that the chosen network topology can be easily scaled and ported to other FPGA instances with different requirements. Furthermore, we have investigated scaling and associated resource scheduling methods.

3.4.1 Optimizing Object Detection Models for Embedded Systems — In computer vision, it is common to downscale models by changing the backbone network (e.g. from VGG to MobileNet), reducing the input resolution (e.g. from 512×512 to 300×300 in SSD), reducing the number of layers (e.g. from 50 to 18 in ResNet) or a combination of these. To further reduce the computational cost, techniques such as pruning and quantization are often utilized.

In cREAtIve, two additional model compression techniques were developed specifically for object detection, an important vision task in embedded systems which is deployed on our FLIR cameras. Our first technique focuses on the *backbone* or feature extractor network of the object detector. These networks are usually pretrained

on a very general dataset such that they can be used in a wide range of scenarios. When using such a backbone network in an embedded object detector for a specific domain (e.g. traffic monitoring), multiple parts of this network can be pruned as they are redundant or have no influence in the specific task. In this project, we developed a pruning method that uses dropout-based sensitivity analysis to determine the redundant convolutional layer in the feature extractor.

Our second technique compresses the *head* of the object detector, which predicts classes and exact locations of the objects in an image. The raw output of the NN is a list or predictions for a predefined grid of possible object locations. This large network output then goes through a post-processing step to produce the final detection predictions. In this project, we reduce the predictions made by the network by pruning those that can be covered by neighboring predictions. This technique is able to save both computations that happen during inference of the NN as well as the often costly computations that happen in the post-processing step.

3.4.2 Hardware Design Space Exploration — On the hardware side of the design space exploration, we built a tool that lets users configure easics AI core to be deployed on FPGAs. This tool takes two sets of inputs:

(1) Hardware configuration input such as the Multiply-And-Accumulate (MAC) units of the convolution engine and size of the different data buffers.
(2) The network specifications of the DL model, in ONNX format [12].

Based on these inputs, the tool estimates the required FPGA resources as well as the inference time of the DL model. The former reports a list of commercially available FPGAs that satisfy the required resources together with the resource utilization for each listed FPGA. The latter estimates the running time of the DL model when it would run on the actual FPGA. Given a fixed DL model, one can change the hardware configuration input to evaluate the performance of different hardware setups. Similarly, given a fixed hardware configuration, one can change the DL model structure to see the effect on the hardware utilization and latency.

3.5 Adaptable Embedded Deep Networks

One of the objectives of cREAtIve is to enable the adaptation of the resource-efficient implementations for the three targeted applications to specific environmental conditions, without giving up on the resource constraints. It is impossible to provide separate hardware solutions for each different situation and change between them as we do not have enough resources to store several full FPGA configurations. The challenge hence is to investigate how adaptation can be best implemented in hardware, given the resource constraints.

Most Convolutional NN (CNN) implementations on FPGAs typically rely on DSP blocks to implement MAC operations for efficiency reasons [20, 25]. Therefore, using cost-optimized FPGA boards (which contain a limited number of DSP blocks) for the CNN implementations is a challenging task.

Weight pruning has been shown to be an efficient method to reduce the computational complexity of a CNN while maintaining accuracy [21]. Magnitude-based weight pruning [17] removes weights below some threshold value. The threshold value itself may vary depending on the application, specific environmental conditions, network layer, etc., and thus, may have to be adapted.

To exploit the substantial opportunity offered by weight pruning, a hardware controller is designed to automatically identify and perform the sparse matrix operations using Lookup Tables (LUTs) instead of the DSP blocks. As a result, by allocating the DSP blocks to the non-sparse matrix operations, this approach enables a better management of the DSP block resources, allows smaller FPGA boards to perform more work, and hence lowers the overall system cost.

Furthermore, in the scope of the development of nearbAI [9], the easics AI core product family, some techniques have been investigated to address run-time NN changes (e.g. changes in weight, weight accuracy, conditional execution, etc.) with minimal or no delay.

3.5.1 Hardware Controller — The overall system to implement the controller consists of three main modules which are connected using the AXI bus interface:

(1) **CPU Controller** for scheduling the operations and inputs,
(2) **FPGA Controller** for weight pruning and performing the matrix multiplications, and,
(3) **Main Controller** to interface between CPU and FPGA Controllers. The Main Controller monitors the input stream and triggers the FPGA Controller to select the appropriate parameters for weight pruning.

Due to the lack of space, we will focus on the MUX-based FPGA Controller which is shown in Fig. 7. As can be seen, a number of multiplexers are used to introduce the weight pruning capability and adjust the threshold parameter values. The controller consists of similar modules for each layer type of the CNN (i.e. Convolutional, Fully-connected, Pooling). When the threshold value needs to be modified, the Main Controller triggers the first MUX which controls the MUXs in each layer type. Furthermore, if the Output Evaluator detects sparse matrix operations, LUTs are used to implement the computations. Otherwise, the input tensor matrices are multiplied by the weights using the available DSP blocks.

Based on our evaluations on the Xilinx SoC board, 1349 LUTs are required to implement the FPGA Controller with weight pruning capability. This overhead is acceptable if enough DSP computations can be saved by weight pruning and it can be further reduced by 8% using the Parameterized Reconfiguration (PR) technique [14].

3.5.2 Reconfigurable Hardware Implementation — We have investigated run-time modifications on a CNN model to optimize the trade-off between speed and accuracy depending on conditions only known at run-time. A number of techniques are developed to allow dynamic switching between weights and weight resolutions, and to support these types of modifications on the easics AI core. The implementation follows the same scheme as described in Section 3.5.1 where a Main Controller is responsible for taking global decisions (e.g. which set of parameters is needed), and then configures the FPGA Controllers. Applications include coping with

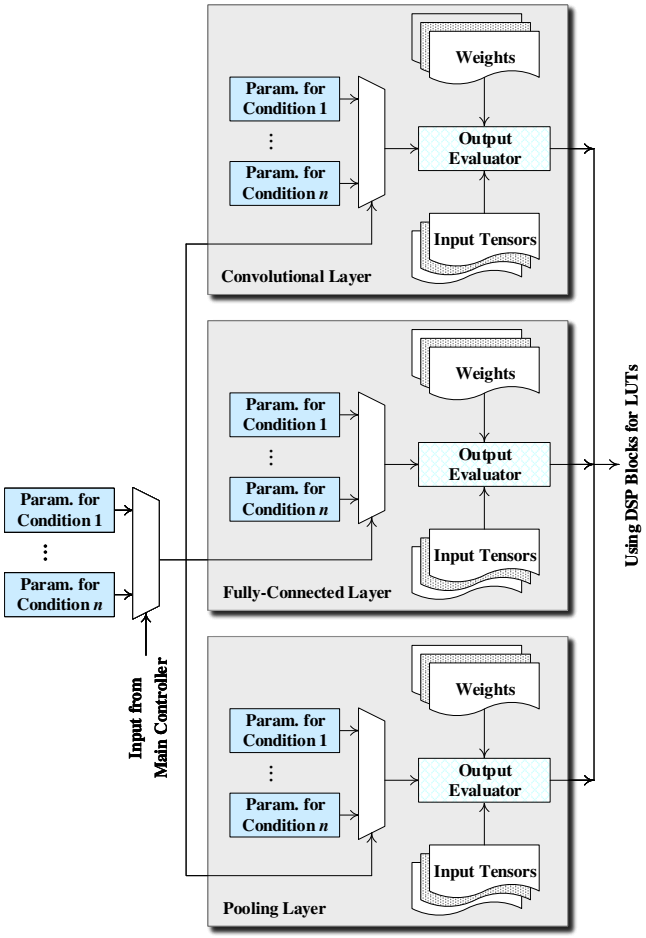


Figure 7: MUX-based FPGA Controller with weight pruning capability.

the varying signal-to-noise ratio of sensor readings (e.g. due to varying illumination of an image sensor).

4 Project Outcome

The consortium will showcase two physical demonstrators to validate the innovation goals of the capabilities of cREAtIve:

(1) **Traffic Monitoring Demonstrator** — This demonstrator will illustrate the DL-based solutions for object detection and classification in thermal images acquired with FLIR cameras (Section 3.3), assess the hardware-performance trade-offs for deep networks (Section 3.4), and highlight the adaptation capabilities of the devised deep networks (Section 3.5).

(2) **Automotive Demonstrator** — This demonstrator will illustrate the sensor denoising methodology (Section 3.1) as well as show the capabilities offered by DL-based processing of point clouds generated by the LiDAR systems of XenomatiX (Section 3.2), enabling an improved object detection, tracking and classification compared to existing conventional solutions. The hardware-performance trade-offs as well as the adaptation capabilities of the devised networks (Sections 3.4 and 3.5) will also be demonstrated.

Furthermore, the project provides a proof of concept of adaptive CNN implementations on an automatically reconfigurable FPGA and show the benefits of this approach to at least one of the applications.

Acknowledgments

This work was executed within the imec.icon project cREAtIve, a research project bringing together academic researchers and industry partners. The cREAtIve project was co-financed by imec [10] and received project support from Flanders Innovation & Entrepreneurship (VLAIO) [11] (project nr. HBC.2018.0532).

References

[1] https://www.imec-int.com/en/what-we-offer/research-portfolio/creative/.
[2] https://www.melexis.com/en.
[3] https://xenomatix.com/.
[4] https://www.flir.eu/.
[5] http://www.etrovub.be/.
[6] https://www.ugent.be/ea/idlab/en.
[7] https://www.ugent.be/ea/elis/en/csl/hardware-and-embedded-systems.
[8] https://www.easics.com/.
[9] https://www.easics.com/deep-learning-fpga/.
[10] https://www.imec-int.com/en.
[11] https://www.vlaio.be/nl/andere-doelgroepen/flanders-innovation-entrepreneurship.
[12] Open Neural Network Exchange (ONNX). https://onnx.ai/, 2019.
[13] Bolsee, Q., and Munteanu, A. CNN-based denoising of Time-of-Flight depth images. In *Proc. IEEE Int. Conf. on Image Processing (ICIP)* (2018), pp. 510–514.
[14] Bruneel, K., Abouelella, F., and Stroobandt, D. Automatically mapping applications to a self-reconfigurable platform. In *Proc. ACM/IEEE Design Automat. Test in Europe Conf. (DATE)* (2009), pp. 964–969.
[15] Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR)* (2017), pp. 77–85.
[16] Gusmão, G., Barbosa, C., and Raposo, A. Development and validation of LiDAR sensor simulators based on parallel raycasting. *Sensors 20*, 24 (2020), 71–86.
[17] Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Proc. Int. Conf. on Neural Inf. Process. Syst. (NIPS)* (2015), p. 1135–1143.
[18] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common objects in context. In *Proc. European Conf. on Computer Vision (ECCV)* (2014), pp. 740–755.
[19] Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recog. (CVPR)* (2017), pp. 5115–5124.
[20] Nguyen, D., Kim, D., and Lee, J. Double MAC: Doubling the performance of convolutional neural networks on modern FPGAs. In *Proc. ACM/IEEE Design Automat. Test in Europe Conf. (DATE)* (2017), pp. 890–893.
[21] O'Keeffe, S., and Villing, R. Evaluating extended pruning on object detection neural networks. In *Proc. Irish Signals and Systems Conf. (ISSC)* (2018), pp. 1–6.
[22] Wang, F., Zhuang, Y., Gu, H., and Hu, H. Automatic generation of synthetic LiDAR point clouds for 3-D data analysis. *IEEE Trans. on Instrumentation and Measurement 68*, 7 (2019), 2671–2673.
[23] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph. 38*, 5 (2019), 1–12.
[24] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. Detectron2. https://github.com/facebookresearch/detectron2, 2019.
[25] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., and Cong, J. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proc. ACM/SIGDA Int. Symp. on FPGAs* (2015), p. 161–170.