Chapter 1
Challenges in the Realm of Embedded Real-Time Image Processing

Philippe Millet, Michael Grinberg and Magnus Jahre

Abstract The development of power-efficient solutions gives new embedded products the ability to analyse images and thereby brings more intelligence to embedded systems – providing more and better services of higher quality as well as advanced capabilities such as self-adaptation and autonomy. This will allow cars to drive safer, medical devices to assist surgeons, and autonomous drones to find people that have gotten lost. For small-series products, one needs to find an embedded platform that provides enough performance, does not exceed the target price, and has sufficiently low power consumption. As these requirements are typically conflicting, image processing engineers spend considerable time identifying the best possible trade-off for their algorithm implementation on the chosen platform. Providing a common platform that allows the efficient implementation of image processing systems across diverse application domains – a key objective of our TULIPP project – requires a solid understanding of the constraints and challenges of each domain. In this paper, we report the key challenges we identified within the medical, Unmanned Aerial Vehicle (UAV), and automotive domains to aid the community in developing the next generation of embedded image processing systems.

1.1 Introduction

Embedded computing refers to a computing solution that performs a dedicated function within a larger mechanical or electrical system. The larger system is often

Philippe Millet
Thales, Palaiseau, France, e-mail: philippe.millet@thalesgroup.com

Michael Grinberg
Fraunhofer IOSB, Karlsruhe, Germany, e-mail: michael.grinberg@iosb.fraunhofer.de

Magnus Jahre
Norwegian University of Science and Technology, Trondheim, Norway,
e-mail: magnus.jahre@ntnu.no

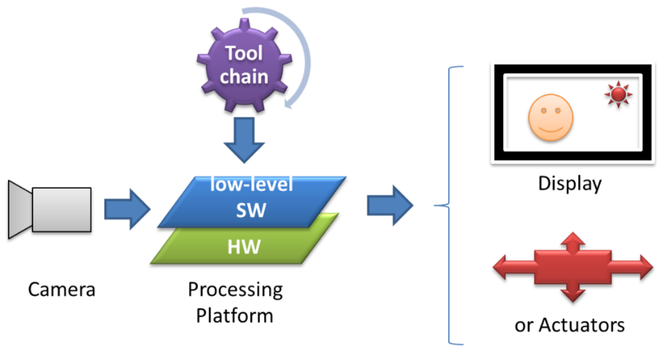

Fig. 1.1: A typical embedded image processing system. The system (1) captures input with one or more sensors, (2) uses a computing platform to analyse the captured image, and (3) either outputs an enhanced image or takes action.

mobile, commonly process sensor data, and may need to perform computation within given deadlines (i.e., real-time constraints). Embedded systems bring functions such as control and automation to devices in common use today – e.g., mobile phones, washing machines, cameras, pacemakers, TVs, and alarm clocks. A recent study found that 98% of all manufactured microprocessors are components of embedded systems [1].

Even though the spectrum of embedded computing solutions is very wide, from a bird's-eye view, common characteristics and concerns can still be found when comparing typical embedded computers to general-purpose ones. We can notice that embedded solutions are always faced with small or highly constrained volume, weight constraints, and limits on power consumption or heat dissipation. This is referred as SWaP (Size, Weight, and Power) constraints.

These constraints have a drastic impact on the embedded computing platform. For instance, the platform is constrained with respect to the processor's capabilities and the size of the memories (RAM as well as Flash memory). The storage capabilities are also limited; there is often no hard drive. Further, there is almost always links to sensors and actuators. When the device has access to a network, the system can use cloud resources to provide more advanced features. For instance, the Google GPS application on Android mobile phones uses the network servers to compute the best way to reach a given destination.

Figure 1.1 illustrates an embedded image processing system in more detail. Generally speaking, an image processing platform is composed of a hardware platform on top of which low-level software is implemented to operate the hardware (e.g., an operating system, libraries, and the application itself). Further, a set of tools, generally called a tool-chain, must also be available to develop applications. Typically, the image processing platform uses one or more sensors (e.g., a camera or camera-like device) to capture images. Through a dedicated algorithm, higher-level information will be extracted from the image. This result is then used to take action or output enhanced information with the image.

Since the sensor outputs frames at a given rate, the platform must be able to process the images at the same rate to not lose any information. In some systems, the loss of frames may have dire consequences such as an aircraft crashing (i.e., hard real-time systems). In soft real-time systems, loosing frames means that the system is failing to perform its desired function. Most embedded image processing systems have (soft or hard) real-time constraints which means that the system must be dimensioned to process data at the same rate as it is produced. To achieve this, a holistic view of the system is required.

Increasing single-thread performance typically requires increasing clock frequency which in turn increases power consumption [4]. Alternatively, the system can be partitioned into different components that work in parallel which enables improving performance while keeping clock frequency constant. Power-efficiency can be improved even further by mapping each system component to a computing device that is specialised to the core computation performed by that particular component (e.g., [12, 13, 7]). This makes the hardware platform heterogeneous, and heterogeneous platforms are typically more efficient than homogeneous platforms [2].

Unfortunately, heterogeneity comes at a cost. More specifically, the programmer is forced to specialise each system component to its target computing device which reduces programmability and versatility. This often means using dedicated programming languages or restricted Application Programming Interfaces (APIs). In addition, the programmer is exposed to the conventional challenges of parallel programming which for instance include taking into account task scheduling and orchestrating data transfers between tasks. This complexity means that advanced image processing systems benefit significantly from using an operating system as it provides convenient services such as transparently mapping and scheduling tasks onto hardware compute devices. Another way of making development for heterogeneous platforms more manageable is to provide extensive tool support (e.g., STHEM [10]). This enables the developer to quickly assess how well the application performs in relation to key requirements such as frame rate and power consumption.

A key challenge of applying specialisation to embedded image processing systems is that the appropriate trade-off point among the conflicting requirements differ between application domains. Thus, a good understanding of the each domain's key challenges is necessary. In the Tulipp project [5], we addressed this issue by studying key applications within the medical, Unmanned Aerial Vehicle (UAV), and automotive domains. We now outline the key challenges of these domains in Section 1.2, 1.3, and 1.4, respectively. Finally, we discuss the efficient implementation of Convolutional Neural Networks (CNNs) in Section 1.5 as CNNs are likely to become a widely used component of future embedded image processing systems.

1.2 Image Processing Challenges in the Medical Domain

The medical domain. As defined by the physicians, Medicine is an art based on science. Doctors have to diagnose, to make prognosis, and to make decisions based partly on protocols and scientific examination of the patient. The difficulties they face are mostly to be able to understand what is going wrong with only partial information of a human being. The human body is such a complex system that it requires a lot of practice and experience for doctors to deal with it.

Even if medicine is an art, it is a highly technical domain. Technological improvements enable medical staff to benefit from more accurate measurements and imagery. Medical imaging is the visualisation of body parts, organs, tissues, or cells for clinical diagnosis and preoperative imaging. The global medical image processing market is about $15 billion a year. The imaging techniques used in medical devices include a variety of modern equipment in the field of optical imaging, nuclear imaging, radiology, and other image-guided intervention. The radiological method, or X-ray imaging, renders anatomical and physiological images of the human body at a very high spatial and temporal resolution.

Imagery is one of the key mechanisms to improve diagnostic accuracy, reduce the time spent to cure patients, or to increase the level of control while administering the cure. It also allows for faster surgery, smaller cuts in the body, and faster patient recovery. All these improvements allow reducing the costs to cure, which is a priority for insurance companies and governments.

The TULIPP medical use case. The TULIPP medical use case focuses on X-ray instruments and thereby addresses a significant part of the market share. More specifically, we focused on the mobile C-arm which is a perfect example of a medical system that improves surgeon efficiency. This device shows the doctor a real-time view from inside the body of the patient during the operation, allowing for small incisions instead of wide-cuts and more accurately targeting the desired region. As a result, the patient recovers much faster and the risk of nosocomial diseases is reduced. The drawback of this technique is the radiation dose which is 30× higher than what we receive from our natural surroundings each day. This is a significant problem for the medical staff that performs such interventions all day long, several days a week.

While the X-ray sensor is very sensitive, lowering the emission dose increases the level of noise on the pictures, making them unreadable. This can be corrected with proper image processing. More specifically, it is possible to divide the radiation by a factor of 4 and restore the original quality of the picture by applying specific noise reduction algorithms running on high-end PCs. Unfortunately, in such a confined environment as an operating room, crowded with staff and equipment, where size and mobility matters, this is not convenient.

Another issue is that regulations require that all radiation that the patient is exposed to must have a specific purpose. Thus, each photon that passes through the patient and is received by the sensor must be delivered to the practitioner; no frame should ever be lost. This creates the need to manage side-by-side strong real-time constraints and high-performance computing.

In the Tulipp project, our heterogeneous hardware platform provides computing power comparable to a standard desktop computer while being comparable to the size of a smartphone. Thus, Tulipp makes it possible to lower the radiation dose while maintaining image quality. We achieved this goal by taking a holistic approach to image processing system development (see Chapter **??** for more details).

1.3 Image Processing Challenges in the UAV Domain

The UAV domain. The term Unmanned Aerial Vehicle (UAV) refers to any flying aircraft without humans on board. In recent years, the usage of UAVs in different application fields has increased significantly. Currently, most important markets for UAVs are aerial photogrammetry, panoramic photography, precision farming, surveillance, and reconnaissance. Further application fields are for instance rescue, law enforcement, logistics, and research. The usage of such systems in the entertainment domain is growing especially fast. This development is boosted by the constantly rising commercial market for small UAVs providing broad accessibility, diversity, and low costs. Essential enhancements to UAV usage are expected from improvement of their capabilities; perception and intelligent evaluation of the environment make many new applications possible. Many of those applications can greatly benefit from intelligent on-board image processing. However, most of the image processing algorithms are developed in high-level programming languages such as C/C++ and are quite complex. Optimising them for an embedded system is, hence, a quite challenging task.

The Tulipp UAV use case. In most cases, UAVs are carrying a sensor payload that allows them to accomplish their respective tasks. Even though we are used to hear about autonomous drones, most of the current systems are still remotely piloted by humans. A human on the ground has to permanently monitor both the drone flight in order to avoid collisions with obstacles and the payload of the drone in order to successfully accomplish the desired mission (e.g., to capture the desired data).

The simultaneous operation of the UAV and of the sensor payload is a challenging task. Mistakes may be fatal either with regard to the mission success or – much worse – with regard to mission safety. Besides, there might be a limitation of the UAV operation area due to the need for a constantly available communication link between the UAV and the remote control station.

A major improvement could be made if UAVs were capable of autonomous navigation or at least had an obstacle detection and avoidance capability. Such a capability can be achieved by means of additional sensors, such as ultrasonic sensors, radars, laser scanners, or video cameras, that monitor the UAV's surroundings as shown in Figure 1.2.

Fig. 1.2: Obstacle detection and avoidance for UAV.

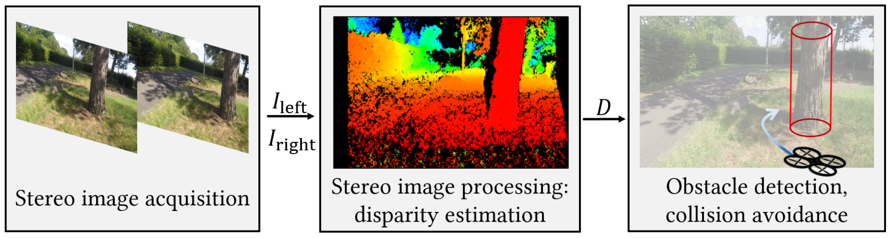
Stereo image acquisition
I_{left}
I_{right}
Stereo image processing: disparity estimation
D
Obstacle detection, collision avoidance

Fig. 1.3: Obstacle avoidance using a stereo camera setup. Obstacle detection is done based on disparity images D that are computed from the stereo camera images I_{left} and I_{right}. A disparity is the displacement of a pixel in one stereo camera image with respect to the corresponding object pixel in the image of the second camera. The smaller the distance to an object, the larger is the disparity of its pixels. In this figure, disparities are visualised by means of a false-color image. Small disparities are shown in blue, large in red.

However, the ultrasonic sensors have a very limited range, the radar sensors might "overlook" non-metal objects, and the laser scanners are heavy and energy-intensive and thus not well suited for with UAVs that already have tight weight and power constraints. A pragmatic solution can be achieved using a lightweight stereo video camera setup with cameras orientated in the direction of flight. Hence, the UAV use case in TULIPP deals with high-performance stereo vision algorithms for UAV obstacle avoidance.

Stereo vision based obstacle detection is usually done by analysing the so called depth or disparity images. They are computed from the stereo camera images and encode distances to objects in the captured scene as shown in Figure 1.3. The most challenging part in this case is the disparity estimation algorithm. This is particularly true for "good" dense disparity estimation algorithms which are based on pixel matching with global optimisation approaches.

Within TULIPP we implemented a collision avoidance system that is based on disparity image analysis and performs the following functions:

1. Synchronous image acquisition from two video cameras.
2. Stereo image processing to compute disparity images.

3. Obstacle detection and collision avoidance.

We describe in more detail how we implemented these functions in Chapter **??** of this book. This required utilising the heterogeneous Tulipp compute platform to enable real-time operation while appropriately balancing weight, performance, and power consumption [9]. While implemented for UAVs, the technology is easily portable to other vehicles and particularly cars.

1.4 Image Processing Challenges in the Automotive Domain

The automotive domain. Advanced Driver-Assistance Systems (ADAS) – which help the driver to focus on what is important on the road – are developing at a very fast pace. Sustained by available and sufficiently performant technology, devices are embedding more and more intelligence to analyse the driver and the car's environment and might even act and control the car in case of danger. Since this technology saves lives, it is strongly supported by governments and insurance companies.

Having more electronic devices in a car comes with its own set of significant challenges. The first challenge is the power consumption. When the number of compute platforms increases, power consumption goes up and the trend towards requiring more computation and more powerful processors exacerbates this issue. A second challenge is that the number of sensors is also increasing at a fast pace. More cameras are added to better understand the whole environment of the car, but also to interpret the behaviour of the passengers and to supervise the driver's actions. Images will also be linked with other sensors in the car and sensor fusion algorithms will be required for the car to fully understand the current situation and make the right decision.

Ideally, the car should be able to foresee situations before they are encountered. To reach this goal, the car must be able to predict the behaviours of the other cars. Since legacy cars will still be operational for many years, this problem cannot be solved completely by the cars communicating with each other. Thus, the car must rely on advanced techniques to analyse the behaviour based on what it sees, just like humans do. Human drivers continuously learn how to interpret traffic, and it is not unlikely that cars will have to develop learning capabilities of their own.

ADAS systems are costly. The technology is typically first developed for high-end cars, but is commonly introduced into the consumer market after only a few years. While the target price of the high-end version may not be an issue, the implementation for consumer cars must be as cheap as possible. The consumer market is the main objective since the larger volume results in higher return on investment.

The Tulipp automotive use case. The strict requirements of ADAS systems pushes technology development with respect to hardware, software, and algorithms. In the Tulipp project, we focused on implementing pedestrian detection using Viola-Jones classifiers on the heterogeneous Tulipp hardware platform using a High-Level

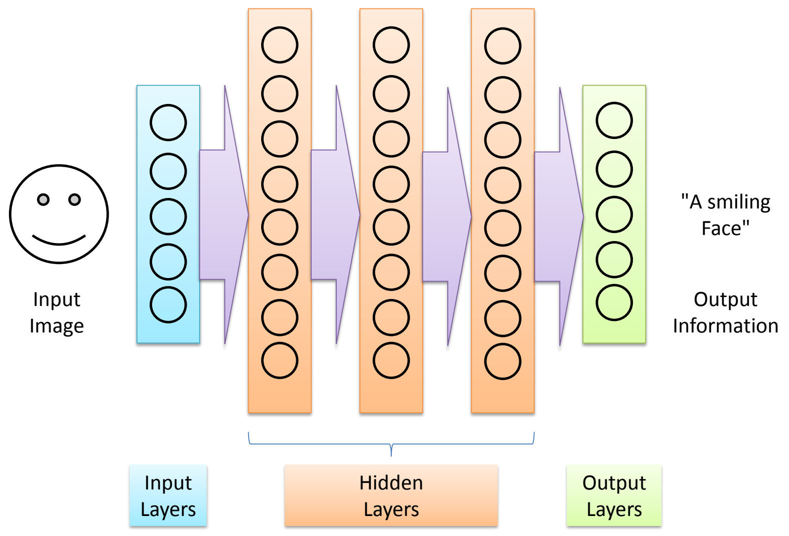

Fig. 1.4: The basic structure of a Deep Neural Network (DNN).

Synthesis (HLS) [6, 11]. The key result was that we were able to implement a near-real-time pedestrian detection system with a fraction of the manpower required to implement similar systems using a traditional Register Transfer Level (RTL) approach.

1.5 Looking into the Future: Neural Networks

Over the last 5 years, Deep Neural Networks (DNNs) have emerged as a promising approach for analysing images with high quality of results and low error rates. This family of algorithms follows a massively parallel and distributed computation paradigm, as it emulates a very large number of neurons operating in parallel. As shown in Figure 1.4, the neurons are organised into layers, each of which operates successive and complementary processing. The first layer, the input layer, is in charge of extracting the pixels from the image and may apply a first convolution or a filter to the image. The last layer is the output layer. It is in charge of collecting the information from the neural network. The internal layers between the input and output layers are in charge of splitting and partly characterising the image through a list of features and then combining the features to identify and classify the information extracted

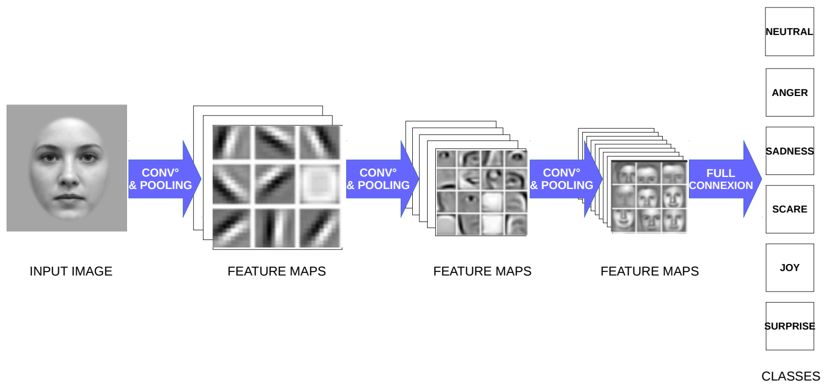

Fig. 1.5: Basic structure of a Convolutional Neural Network (CNN)

from the image. Each neuron within one layer provides a set of outputs that are sent to other neurons on the next layer through connections called synapses. Each synapse is weighted, and the set of all the network's synaptic weights form the set of so-called network parameters.

Various types of DNNs exist, and different DNNs mostly differ by the connection policy between hidden layers. Providing all-to-all connection between layers (dense layers) allows to build Multi-Layer Perceptrons (MLP) which are used for simple data classification. For image processing, Convolutional Neural Networks (CNNs), in which the connection policy emulates convolutional filters, are commonly used. As shown in Figure 1.5, the convolutional layers are used to hierarchically extract visual features. Pooling layers are used to down-sample visual features, thus reducing the compute requirements. At the end of the so-called feature-extraction stage, dense layers are used to perform classification based on extracted features.

The vast majority of computation is performed by convolutional layers. The reason is that the convolutional stage is much deeper than the classification stage [8, 3] and that convolutional layers are more complex than pooling layers. Thus, accelerating CNN algorithms mainly rely on accelerating convolution which is not an easy matter when targeting embedded applications [14]. The two main constraints are: (1) the amount of computation to process the convolutions is too high to be performed in real-time, and (2) insufficient memory and bandwidth are available to store and use the parameters of the network. For example, the network used by Burkert et al. [3] requires performing more than 800k convolutions in the first layer.

Fortunately, neural networks are very static. Once a network has been trained, the data path between the layers is known and predictable. This characteristic makes it possible for hardware designers to fine-tune architectures to efficiently execute CNNs (e.g., [12]). Dedicated processors with parallel accelerators for convolutions can be combined with highly efficient memory access and large memory capacity. Due to the predictability of the CNNs, the memory system might be highly hierarchical and

Table 1.1: Summary of embedded image processing challenges.

Type	Challenge
Sensors	More sensors to capture the whole environment. More cameras with better quality and larger image sizes.
Algorithms	More complex, requiring more processing from the hardware. More information will be extracted from the images. More intelligence that emerges from the images and from other sources (other kinds of sensors, communication between drones or cars, etc.).
Energy and Power	The energy consumption should ideally remain constant. While this might not be possible, it must be managed as more energy means bigger batteries with higher costs and weight. Meeting the power-budget typically means that higher processing-efficiency is required.
Development Costs	Development costs should be as low as possible and time-to-market as short as possible. To achieve this, the development must rely on advanced development and analysis tools, operating systems, standard libraries, and reusable APIs.
Customer Price	The markets addressed by TULIPP are highly competitive. Therefore, the final cost of the system must be controlled to be able to offer it at a price customers can afford.

automatic code generation tools could then be helpful to schedule the execution of the network and the flow of data between the memories and the processing units of the architecture.

1.6 Conclusion

We have now outlined the key challenges of the application domains we focused on in the TULIPP project. Overall, we find that more and more processing is required to process data from an ever-increasing number of sensors that provide data with increasing resolution (see Table 1.1). To respond to these trends, hardware platforms must evolve to meet the requirements of current and future image processing algorithms by providing more specialised hardware with larger yet low-power memory systems. As the applications continue to evolve, a key challenge is to provide versatile computing platforms capable of delivering high performance for the most important computational patterns within each domain.

Acknowledgements This work is a synthesis of experiences from the work in the TULIPP project. Thus, there are many people that have contributed through discussions during the project and contributions to different project-internal documents. TULIPP was funded by the European Horizon 2020 programme (grant agreement #688403).

References

1. Barr, M.: Real men program in C. `https://webpages.uncc.edu/~jmconrad/ECGR4101Common/Articles/Real%20men%20program%20in%20C.pdf` (2009)
2. Borkar, S., Chien, A.A.: The future of microprocessors. Communications of the ACM **54**(5), 67 (2011)
3. Burkert, P., Trier, F., Afzal, M.Z., Dengel, A., Liwicki, M.: Dexpression: Deep convolutional neural network for expression recognition. arXiv preprint arXiv:1509.05371 (2015)
4. Horowitz, M., Indermaur, T., Gonzalez, R.: Low-power digital design. In: Symposium on Low Power Electronics, pp. 8–11 (1994)
5. Kalb, T., Kalms, L., Göhringer, D., Pons, C., Marty, F., Muddukrishna, A., Jahre, M., Kjeldsberg, P.G., Ruf, B., Schuchert, T., Tchouchenkov, I., Ehrenstrahle, C., Christensen, F., Paolillo, A., Lemer, C., Bernard, G., Duhem, F., Millet, P.: TULIPP: Towards ubiquitous low-power image processing platforms. In: Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), pp. 306–311 (2016)
6. Kalb, T., Kalms, L., Göhringer, D., Pons, C., Muddukrishna, A., Jahre, M., Ruf, B., Schuchert, T., Tchouchenkov, I., Ehrenstråhle, C., Peterson, M., Christensen, F., Paolillo, A., Rodriguez, B., Millet, P.: Developing Low-Power Image Processing Applications with the TULIPP Reference Platform Instance. In: C. Kachris, B. Falsafi, D. Soudris (eds.) Hardware Accelerators in Data Centers, pp. 181–197. Springer International Publishing (2019)
7. Koraei, M., Fatemi, O., Jahre, M.: DCMI: A scalable strategy for accelerating iterative stencil loops on FPGAs. ACM Transactions on Architecture and Code Optimization **16**(4), 36:1–36:24 (2019)
8. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks **3361**(10), 1995 (1995)
9. Ruf, B., Monka, S., Kollmann, M., Grinberg, M.: Real-time on-board obstacle avoidance for UAVs based on embedded stereo vision. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XLII-1**, 363–370 (2018)
10. Sadek, A., Muddukrishna, A., Kalms, L., Djupdal, A., Podlubne, A., Paolillo, A., Goehringer, D., Jahre, M.: Supporting utilities for heterogeneous embedded image processing platforms (STHEM): An overview. In: Applied Reconfigurable Computing (ARC) (2018)
11. Synective Labs: The automotive use-case. `http://tulipp.eu/wp-content/uploads/2019/03/2018_VISION18_SYN.pdf` (2018)
12. Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P., Jahre, M., Vissers, K.: FINN: A framework for fast, scalable binarized neural network inference. In: Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 65–74 (2017)
13. Umuroglu, Y., Jahre, M.: An energy efficient column-major backend for FPGA SpMV accelerators. In: Proceedings of the International Conference on Computer Design (ICCD), pp. 432–439 (2014)
14. Verhelst, M., Moons, B.: Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to IoT and edge devices. IEEE Solid-State Circuits Magazine **9**(4), 55–65 (2017)