# RECONFIGURABLE PROCESSOR FOR DEEP LEARNING IN AUTONOMOUS VEHICLES

*Yu Wang[1*2*], Shuang Liang[3*], Song Yao[2*], Yi Shan[2*], Song Han[2*4*], Jinzhang Peng[2*] and Hong Luo[2*]*

[1*]Department of Electronic Engineering, Tsinghua University, Beijing, China
[2*]Deephi Tech, Beijing, China
[3*]Institute of Microelectronics, Tsinghua University, Beijing, China
[4*]Department of Electrical Engineering, Stanford University, Stanford CA, USA

*Abstract - The rapid growth of civilian vehicles has stimulated the development of advanced driver assistance systems (ADASs) to be equipped in-car. Real-time autonomous vision (RTAV) is an essential part of the overall system, and the emergence of deep learning methods has greatly improved the system quality, which also requires the processor to offer a computing speed of tera operations per second (TOPS) and a power consumption of no more than 30 W with programmability. This article gives an overview of the trends of RTAV algorithms and different hardware solutions, and proposes a development route for the reconfigurable RTAV accelerator. We propose our field programmable gate array (FPGA) based system Aristotle, together with an all-stack software-hardware co design workflow including compression, compilation, and customized hardware architecture. Evaluation shows that our FPGA system can realize real-time processing on modern RTAV algorithms with a higher efficiency than peer CPU and GPU platforms. Our outlook based on the ASIC-based system design and the ongoing implementation of next generation memory would target a 100 TOPS performance with around 20 W power.*

Keywords - Advanced driver assistance system (ADAS), autonomous vehicles, computer vision, deep learning, reconfigurable processor

## 1. INTRODUCTION

If you have seen the cartoon movie WALL-E, you will remember when WALL-E enters the starliner Axiom following Eve, he sees a completely automated world with obese and feeble human passengers laying in their auto driven chairs, drinking beverages and watching TV. The movie describes a pathetic future of human beings in the year of 2805 and warns people to get up from their chairs and take some exercise. However, the inside laziness has always been motivating geniuses to build auto driven cars or chairs, whatever it takes to get rid of being a bored driver stuck in traffic jams.

At least for now, people find machines genuinely helpful for our driving experience and sometimes they can even save peoples lives. It has been nearly 30 years since the first successful demonstrations of ADAS [1][2][3], and the rapid development of sensors, computing hardware and related algorithms has brought the conceptual system into reality. Modern cars are being equipped with ADAS and the numbers are increasing. According to McKinseys estimation [4], auto-driven cars will form a 1.9 trillion dollars market in 2025. Many governments like those in the USA [5], Japan [6] and Europe [7][8][9] have proposed their intelligent transportation system (ITS) strategic plans, which have drawn up timetables for the commercialization of related technologies.

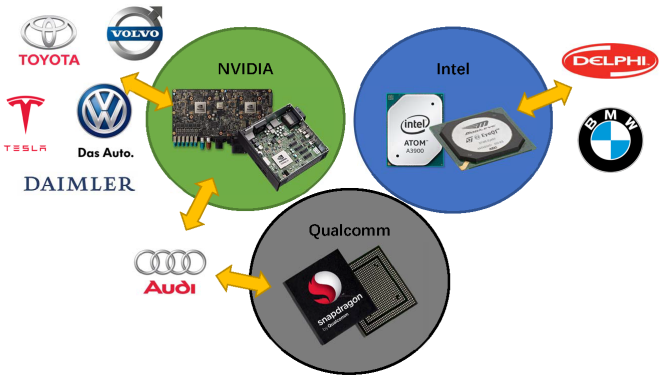


**Figure 1**. The market pattern of automotive cars.

In current ADASs, machine vision is an essential part; it is also called autonomous vision [10]. Since the conditions of weather, roads and the shapes of captured objects are complex and variable with little concern for safety, the anticipation for high recognition accuracy and rapid system reaction to these is urgent. For state-of-the-art algorithms, the number of operations has already increased to tens and hundreds of giga-operations (GOPs). This has set a great challenge for real time processing, and correspondingly we need to find a powerful processing platform to deal with it.

Fig. 1 shows a pattern of the current market of automotive cars. NVIDIA is leading the market with its Drive series GPU platforms, and has already built cooperation with car manufacturers like Audi, Tesla, Daimler, etc. Intel is also focusing on this area. It has acquired many relevant companies such as Mobileye, Nervana, Movidius and Altera, and has collaborated with BMW and Delphi to build its ecosystem circle. It has also released products such as Atom A3900 for the automotive scene[11]. Another chip giant Qualcomm is also trying to make inroads in this market. It has release dedicated processors like Snapdragon 602A and 820A chips [12], and it has bought NXP to strengthen its impact in the ADAS market.

Many ADAS solutions have chosen graphic processing unit (GPU)-based systems to carry their autonomous vision algorithms, not only because of their powerful computational ability since GPU-based systems can offer massive parallelisms in datapaths and the latest GPU processors can offer a throughput of several TOPS such as the NVIDIA Drive PX2 system [13] with Xavier chips, but also because of the robust and efficient developing environment support such as CUDA [14] and cuDNN [15].

While GPU can offer a computing speed of TOPS, the power consumption can often be the bottleneck for in-car system implementation as some modern GPUs can cost 200-300 W. One solution is to improve the power efficiency, and this can be achieved through the dedicate logic customization, and reconfigurable processors can be a suitable choice. One representative reconfigurable processor is FPGA. FPGA suppliers Xilinx and Altera have already introduced their FPGA products into ADAS scenarios such as Zynq-7000 [16] and Cyclone-V [17] series SoC. While the power is around 10 W, FPGA can also get a peak performance of around 100 GOPS. Together with the features of multi-threading, parallel processing and low latency, FPGA could be expected to be a favorable choice for autonomous vision systems.

Naturally, we can convert an FPGA design into an application-specific integrated circuit (ASIC), and the circuit system can further improve its efficiency by at least one order of magnitude with its reconfigurability maintained, which makes ASIC another mainstream ADAS solution. Suppliers including Qualcomm, Intel, Infineon, and Texas Instruments have released their ASIC-based SoC products for ADAS. One representative product is Intel Mobileyes EyeQ4 chip [18], which will be released in 2018 and can get a 2.5 TOPS performance drawing only 3-5 W. The low power feature makes it quite suitable for in-car supercomputing.

Both the chances and challenges for reconfigurable in-car systems lie ahead. This article will firstly analyze the development of modern RTAV algorithms, then evaluate the performance of each hardware platform, and finally discuss how we can build a more efficient reconfigurable system for RTAV.

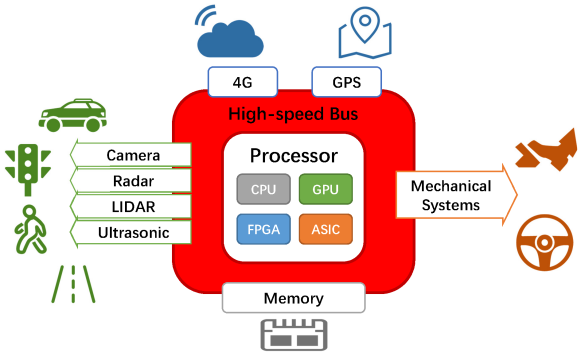


**Figure 2**. A block diagram for ADAS system description.

## 2. TRENDS IN AUTONOMOUS VISION

### 2.1. An overview of an ADAS system

An ADAS system collects data of the surrounding environment from sensors and remote terminals such as cloud servers and satellites, and makes real-time recognition of surrounding objects to assist drivers or automatically make judgements for a better driving experience and avoid potential accidents. A typical system is depicted in Fig. 2. As we can see, there could be a series of sensors on vehicles such as cameras, radars, LIDAR and ultrasonics to get input for a real-time surrounding condition description, and processors will react to give driver warnings or control the mechanical system of the vehicle in some certain circumstances with the trained algorithm models stored in the memory. Communication interfaces can help to locate cars with map data, and can obtain traffic information from datacenters and even offload some compute-intensive tasks to cloud servers, and this can be even more powerful in the future as much faster communication protocols like 5G is already on the way.

Various functions can be achieved with an equipped ADAS system, and autonomous vision has taken up a great portion of this. As we can see from Fig. 3, functions such as vehicle detection (VD), lane departure warning (LDW), forward collision warning (FCW), pedestrian detection (PED), traffic sign recognition (TSR), etc. are achieved by the autonomous vision system itself or together with audio and radar systems. Hence, it is important to find an efficient solution for autonomous vision processing. Next, we will take an overview of the vision algorithms, and present an analysis of potential hardware carriers.

### 2.2. Traditional algorithms of autonomous vision

For most autonomous vision functions such as PED, VD, LDW, TSR, etc., the kernel algorithm can be generalized into a 2D object detection question. As shown in Fig. 4, a traditional detection process consists of the following stages: image preprocessing, region of interest (ROI) selection, feature extraction and classification.

For traditional algorithms, usually steps like gain and expo-

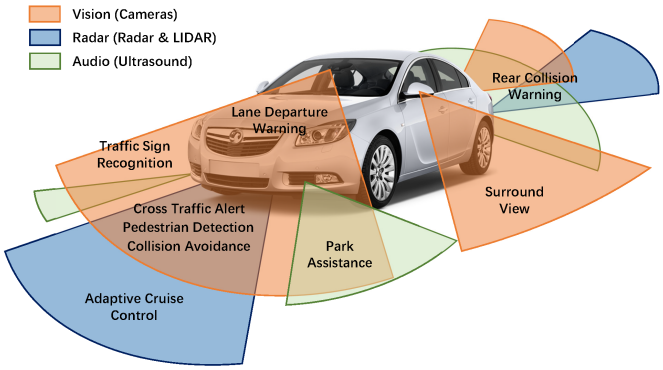


**Figure 3**. Common functions in ADAS system.

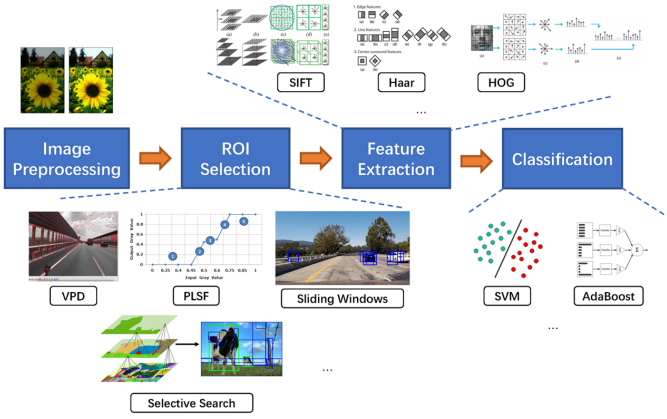


**Figure 4**. Workflow of traditional detection algorithms.

sure adjustment and image rectification would be performed to preprocess the collected images. ROI selection methods depend on the type of task, such as vanishing point detection (VPD) [19] and piecewise linear stretching function (PLSF) [20] are used in LDW, and sliding window methods are taken in PED, VD and TSR. It would be time consuming to execute an exhaustive ROI search, so various optimizations are also taken for ROI selection. Broggi et al. [21] use morphological characteristics of objects and distance information. Uijlings et al. [22] propose a selective search approach to efficiently generate ROIs. For feature extraction, various manually designed features such as Scale-Invariant-Feature-Transform (SIFT) [23], Histogram-of-Oriented-Gradients (HOG) [24], Haar [25], etc. have been widely used in detection tasks. For classification, combined simple classifiers like AdaBoost [26] and support vector machines (SVMs) [27] are popular to work with traditional features. Some part based methodologies also appear to reduce the complexity of the overall task, such as Felzenszwalb et al. [28] proposes a deformable part model (DPM) to break down the objects into simple parts.

### 2.3. The rise of convolutional neural network (CNN)

In recent years, the rise of CNN has set off a revolution in the area of object detection. A typical CNN consists of a

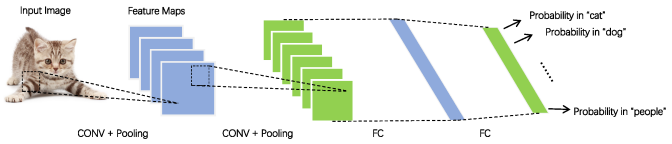


**Figure 5**. A typical CNN architecture.

number of layers that run in sequence as shown in Figure 5. Convolutional layer (CONV layer) and fully-connected layer (FC layer) are two essential types of layer in CNN, followed by optional layers such as pooling layers for down-sampling and normalization layers. The first CONV layer takes an input image and outputs a series of feature maps, and the following CONV layers will extract features to higher levels layer by layer through convolving the input feature maps with filters. After CONV layers, FC layers will classify the extracted features and output the probability of each category that the input image might belong to.

State-of-the-art CNN models have achieved outstanding performance in computer vision areas. Take image classification as example, in 2012 Krizhevsky et al. announced an 8-layer CNN model AlexNet [29] which achieved 84.7% top-5 accuracy on ImageNet [30], which was far beyond the performance of conventional algorithms. Five years have passed, many organizations such as Google [31][32][33][34], Oxford [35], Microsoft [36] have been focusing on novel CNN model designs with more complex computing patterns, and the accuracies of the top models have already surpassed the human vision level [37].

The excellent performance of CNN is because the generic descriptor extracted from CNN that trained on large scale datasets is much richer than the traditional manually designed features, and can be used for various tasks with some fine tuning [38]. Hence for object detection problems, CNN-based algorithms can get a much better performance than the traditional ones.

The workflows of different detection algorithms are shown in Fig. 6. R-CNN was first proposed [39]. It generates a set of region proposals with selective search, warp/crop each region into a static size, then extracts the feature maps with CONV layers, and finally completes the classification with FC and SVM layers. Since R-CNN needs to run CONV layers for every region proposal which is very expensive in computations, SPP-net has appeared [40]. It merely needs to compute CONV layers only once with spatial pyramid pooling to transfer feature maps into fixed length vectors for FC layers. Based on SPP-net, Fast R-CNN was designed by Girshick et al. [41] which used multi-task loss to train the classifier and bounding-box (BB) localizers jointly, with single-sized ROI pooling to the feature maps of the last CONV layer which are projected with region proposals. Then Ren et al. [42] proposed Faster R-CNN, using the region proposal network (RPN), which was actually a Fast R-CNN network, to generate region proposals and to get rid of the large computations of traditional region proposal methods, and reused the Fast

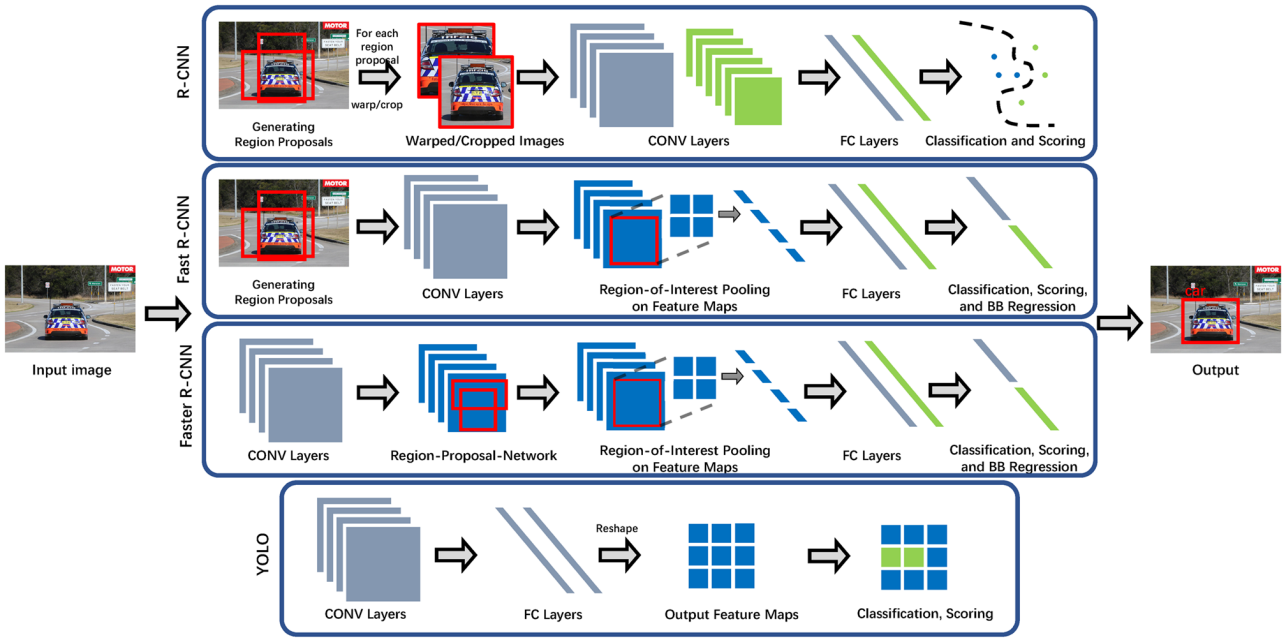


**Figure 6**. The processing flow of typical CNN-based detection methods.

R-CNN model to train the classifier and BB localizers. Unlike the former algorithms which could only get satisfying mean Average Precision (mAP) performance with the weakness of slow speed, Faster R-CNN can achieve real-time processing since it benefits from RPN and can get a 5fps speed with one NVIDIA K40 GPU. Redmon et al. designed YOLO [43] which directly took the whole input images to train the model, and classifies each pixel in the output feature maps. This equals to dividing the input image into several cells and doing the classification inside each cell, which avoids the expensive process for proposals and can be around seven times faster than Faster R-CNN to realize a more feasible real-time detection with acceptable accuracy drop.

These detection algorithms have shown outstanding performance on a PASCAL VOC dataset [44]. However, for the autonomous vision scene, the detection mission would be much tougher since the objects will be presented in much worse quality for the big variance of object scale and the incomplete captured object shape. Therefore, we need to optimize the way we obtain proposals during our detection algorithms. The corresponding representative benchmark for autonomous vision is KITTI [45], and various algorithms have been proposed for the dataset. We have selected some top ranked detection algorithms and have listed them in Table. 1. Actually, most of these algorithms have taken CONV layers to extract the features based on the classic CNN models with small revisions followed by application dependent FC layers. We compare the CONV layers of classic CNN models in Table. 2. As we can see, giga MACs need to be solved for each input frame. Together with FC layers and considering the number of region proposals, in order to realize real-time processing, the hardware needs to provide a throughput speed of over 100-1000 GOPS. With the growing number of image data collected from cameras, future requirement of

**Table 1**. Top-ranked detection algorithms on KITTI.

| Algorithm | Target object (Moderate level) | | |
|---|---|---|---|
| | Car | Pedestrian | Cyclist |
| MS-CNN [46] | 89.02% | 73.70% | 75.46% |
| SubCNN [47] | 89.04% | 71.33% | 71.06% |
| SDP+RPN [48] | 88.85% | 70.16% | 73.74% |
| 3DOP [49] | 88.64% | 67.47% | 68.94% |
| Mono3D [50] | 88.66% | 66.68% | 66.36% |
| SDP+CRC [48] | 83.53% | 64.19% | 61.31% |
| Faster R-CNN [42] | 81.84% | 65.90% | 63.35% |

**Table 2**. Comparison of CONV layers in classic CNN models.

| Model | AlexNet [29] | VGG-16 [35] | Inception v1 [31] | ResNet-50 [36] |
|---|---|---|---|---|
| Top-5 Error | 19.8% | 8.8% | 10.7% | 7.0% |
| # of Weights | 2.3M | 14.7M | 6.0M | 23.5M |
| # of MACs | 666M | 15.3G | 1.43G | 3.86G |

computing speed could reach 10-100 TOPS. To build such a powerful processor with programmability and a power consumption of less than 30 W is a challenging task, and we will discuss the contenders in the next section.

## 3. PROCESSORS FOR REAL-TIME AUTONOMOUS VISION

### 3.1. Heterogeneous platforms for CNN acceleration

As the CNN algorithm rapidly develops, so have the related hardware accelerator designs, in recent years. The work of

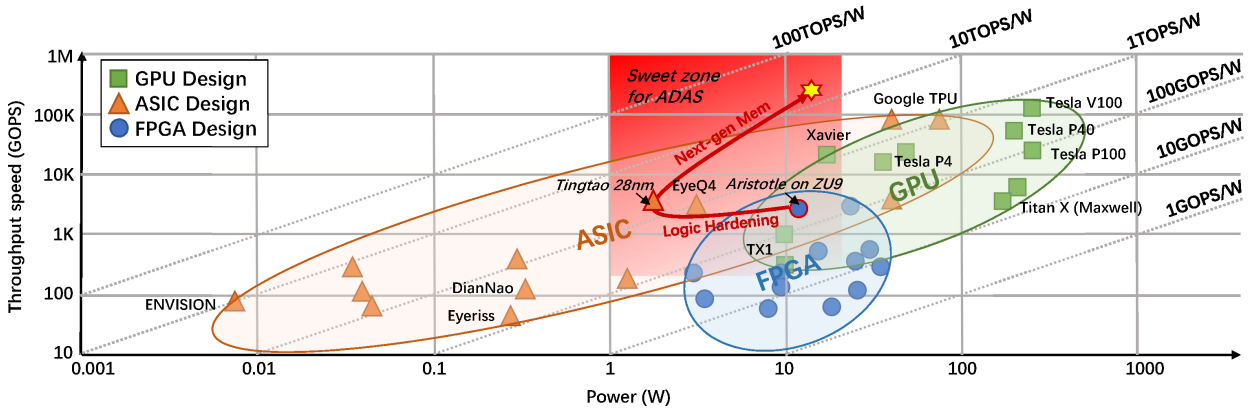


**Figure 7**. Hardware designs of CNN accelerators on different platforms and development route for RTAV accelerator in ADAS.

(Source by:https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/)

[51] shows the comparison between neural network accelerators, as depicted in Fig. 7. We can see from the image that GPUs are among the top tier of computing speeds, but the power consumption is also very high. The freshly released NVIDIA Tesla V100 can get an astounding computing speed of 120 TOPS [52], with a power consumption of 300 W. This can be useful in datacenter scenarios for cases like model training where power is not the main concern. There are also some GPUs designed for low-power embedded environments, like NVIDIA Jetson TX1 mobile GPU, which brings a 300 GOPS speed on VGG-16 and a peak performance of 1 TOPS with only a 10 W cost [53]. The large general purpose stream processors on chip might bring a considerable parallelism, but the efficiency remains a question. With the technology of 28nm, the NVIDIA TITAN-X and TX1 GPU can only get an efficiency of 20-100 GOPS/W.

To improve the efficiency, we need to customize the inside logic of processing elements (PEs) to enhance processing parallelism and optimize memory access patterns. FPGA could be a suitable initial selection, since it can provide a large amount of computing and memory resources and enough reconfigurability with programmable interconnection to map common algorithms on. In Fig. 7 we can see that there have been many FPGA designs. The top designs, including our Aristotle system on the Xilinx ZU9 FPGA platform, can get a throughput speed at around 2 TOPS, which is quite close to the same technology generation NVIDIA TITAN-X GPUs, but of almost 10 times better efficiency. This proves the capability of FPGA of being a strong competitor.

As we can see, most CNN layers consist of MAC operations and have similar computing patterns which could be possibly generalized and parameterized. Therefore, with mature hardware architecture and processing flow, it is feasible to harden the original FPGA accelerator design into an ASIC chip with a programmable interface for reconfigurability, which can further improve performance. Kuon et al. [54] have measured the performance gap between FPGA and ASIC. It is said that the critical-path delay of ASIC is three to four times less than FPGAs, and the dynamic power consumption ratio is approximately 14 for FPGA to ASIC, while the average chip area of ASIC is also 18 times smaller than FPGA. This means we can realize a much better performance with ASIC within a given hardware area. ASIC designs have the relatively better energy efficiency, mostly between 100 GOPS/W to 10 TOPS/W. They have shown excellent performance in low-power area, and as we can see from Fig. 7 some representative designs such as DianNao [55], Eyeriss [56] and Envision [57] are showing a performance of around 100 GOPS with only milli-watt level power consumption. The efficiency can even reach 10 TOPS/W at extreme low voltage status. To the other side, those ASICs with larger chip sizes are capable of offering more abundant PEs and memory bandwidth, which can lead to a faster throughput speed, such as Googles TPU [58] which can get a peak performance of 86 TOPS. From the business aspect, a large quantity production of ASIC could also reduce the overall cost. However, note that the deep-learning algorithms for RTAV have a quite short evolving cycle, usually within six to nine months. Moreover, the benchmarks for RTAV are also far from perfect and new tasks appear nearly every year. While ASICs time to market is no less than one year, there is a potential risk of incompatibility between hardware processors and fresh algorithms and application scenes. Solution providers need to make a risk-return analysis.

Recently, some breakthroughs have taken place in the area of near-memory and in-memory computing. The 3-D memory can offer an order of magnitude higher bandwidth and several times power consumption than 2-D memory, such as Hyper Memory Cube (HMC) proposed by Micron [59], which uses through silicon vias (TSV) to stack the dynamic random-access memory (DRAM) on top of the logic circuit. Through this method, the memory bandwidth can be increased by an order of magnitude from 2-D memory, and the power consumption can be five times less. There have already been some designs combining the CNN accelerator architecture with HMC [60][61]. Another technology is to embed the computation inside memory, such as memristor

[62]. It can realize a MAC operation through the summation of currents from different memristor branches. This avoids the data movement and can save energy. Recent simulation works such as ISAAC [63] and PRIME [64] have evaluated the efficiency of memristors in CNN acceleration.

An ideal ADAS system should be able to offer a computing speed of over 200 GOPS with no more than 40 W, and hence we can mark the sweet zone for ADAS systems as the red painted area in Fig. 7. Inside this sweet zone, we can sketch a development route for the reconfigurable processors for RTAV acceleration, shown as the dark red curve. Starting from the FPGA design, we can climb up through logic hardening for an efficiency of above 1 TOPS/W, and with the help of the implementation of next generation memory technology, the bandwidth can be broaden and the memory access cost could be reduced, which can lead to an even higher efficiency, to more than 10 TOPS/W. We use the yellow star to indicate our target in Fig. 7. With a larger die size, a throughput speed of over 100 TOPS could be expected, which can be a suitable choice for an ideal RTAV system.

### 3.2. Chances and challenges for reconfigurable processors

In the area of RTAV, chances and challenges coexist for a wide application of reconfigurable processors. The following features of reconfigurable processors will bring them opportunities:

1) **Programmability**. Reconfigurable processors can offer a pool of logic and memory resources on-chip. Considering the fast evolving RTAV algorithms, it is not hard for users to update the on-chip functions after they bought it from the supplier.

2) **Reliability**. For example, the industrial grade FPGAs can stably work in a temperature range between $-40°C \sim 100°C$. This makes them able to satisfy the requirement of standards AEC-Q100 and ISO 26262.

3) **Low-power**. The power consumption for reconfigurable processors is no more than 30 W. Low-power consumption is suitable for the in-car environment.

4) **Low-latency**. Since algorithms mapped onto reconfigurable processors provide deterministic timing, they can offer a latency of several nanoseconds, which is one order of magnitude faster than GPUs. A quick reaction of ADAS systems is essential to dealing with sudden changes on the road.

5) **Interfaces**. Unlike GPU which can only make communication through the PCI Express protocol, both ASIC and FPGA designs can provide huge interface flexibility, which can be very helpful for ADAS system integration.

6) **Customizable logic**. Recently there has been great progress in the area of model compression, including data quantization and sparsity exploration. For general purpose processors like CPU and GPU, only fixed data types could be supported and the memory access pattern would be regular. Reconfigurable processors can offer fine-grained customizability which can support data type as low as to 1 bit, and specialized controllers could be introduced to deal with irregular sparsity inside the models.

7) **Multi-thread processing**. For ADAS systems, it would be best for different algorithms to be processed simultaneously, such as LDW would work on grayscale images while PD would process RGB images. Reconfigurable processors can provide vast spatial parallelism for algorithms to work in individual channels.

However, challenges remain for the wide use of reconfigurable processors such as:

1) **Programming language gap**: Most developers use high-level programming languages to build their project, while for reconfigurable processors they need to start from the bottom-level hardware and describe the logic with register-transfer level (RTL) hardware description language (HDL) such as Verilog and VHDL.

2) **Limited on-chip resource**: There is limited area for on-chip arithmetic and memory resource to map the tiled algorithm on spatially. This might form a bottleneck for some large-scale algorithms.

3) **Limited off-chip bandwidth**: To communicate reconfigurable processors with off-chip memories like DDR, the bandwidth is often limited by the clock frequency of the controller and the width of data wires.

### 3.3. Related reconfigurable processors

There have been many excellent reconfigurable processor designs for deep learning models. Initial designs are mostly based on FPGAs. Chakaradhar et al. [65] proposed a runtime reconfigurable architecture for CNN on FPGA with dedicated switches to deal with different CNN layers. Zhang et al. [66] used a nested loop model to describe CNN and designed the on-chip architecture based on high-level synthesis optimizations. Suda et al. [67] presented an OpenCL-based FPGA accelerator with fully-connected layers also implemented on-chip.

ASIC-based reconfigurable processors have appeared in recent years. The representative work is Diannao [55] and its subsequent series [68][69][70], which focused great efforts on memory system optimization. Eyeriss [56] focused on the dataflow optimization and used smaller PEs to form a coarse-grained computing array. ENVISION [57] utilized a dynamic-voltage-accuracy-frequency-scaling (DVAFS) method to enhance its efficiency and reached 10 TOPS/W with low voltage supply. Googles TPU [58] has been the recent star with large on-chip memories and has reached a similar throughput speed to peer GPUs withdrawing much less energy.

Most of these precedent reconfigurable processors have their own features with partial optimization of the entire flow, but few consider the entire flow of the neural network accelerator system. Therefore, the on-chip utilization rate of different CNN layers will eventually fluctuate [58] which may drag down the overall efficiency of the system, and there has been a large space left for improvement from the aspect of

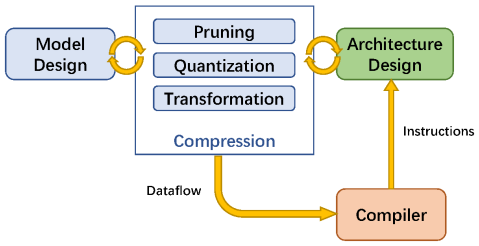


**Figure 8**. The software-hardware co-design workflow of our system.

software. With this motivation, we will introduce our system design in the following section.

## 4. SOFTWARE-HARDWARE CO-DESIGN FOR A RECONFIGURABLE AUTONOMOUS VISION SYSTEM

### 4.1. The overall system workflow

What we have already achieved is an FPGA-based system called *Aristotle* to target CNN acceleration, which can deal with various CNN-based applications and can be conveniently mapped onto different FPGA platforms. For a better processing performance, we should reduce the software workload and improve the hardware utilization rate. Accordingly, we design the software hardware co-design workflow of our Aristotle system depicted in Fig. 8. To reduce the workload, we compress the models using software methods like quantization, pruning and matrix transformation. To improve the utilization rate, the compiler will take the compressed model and hardware parameters of different FPGA platforms as inputs, and execute a task tiling with dataflow optimizations to generate instructions for the hardware. The hardware architecture will exploit the parallelism on-chip for higher throughput with proper granularity choice and datapath reuse. The details will be introduced as follows.

### 4.2. Compression methods

Usually, an algorithm model is trained in floating-point form, but there exists redundancy. Previous work has shown that it is not necessary to represent every datum with 32-bit, and an appropriate data quantization would not hurt the overall accuracy of the model. In Fig. 9 we have made an experiment of quantization on state-of-the-art CNN models, and as we can see from an 8-bit quantization brings little loss to the accuracy. A lower bit-width can directly compress the size of memory footprint, and can bring chance to share datapath consists of integrated DSP blocks. We can implement two multipliers for 8-bit inputs with one $25 \times 18$ DSP block on Xilinx platform.

Another method is to implement a pruning process to the pre-trained model, in order to decrease the number of connections inside a model [71]. It has been proved that some of the connections that have weights close to zero will make a small impact on the output pixel, and can be pruned without much

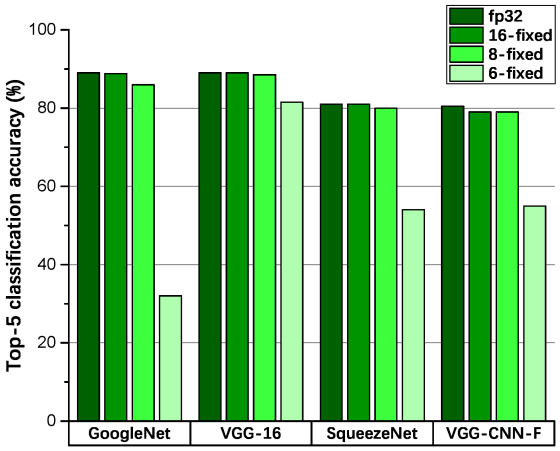


**Figure 9**. Quantization results for different CNN models.

**Table 3**. Comparison of compression ratio between quantization, pruning and matrix transformation methods at different accuracy loss levels (baseline 32-bit floating-point).

| Accuracy Loss | SVD | Quantization Only | Pruning Only | Quantization and Pruning |
|---|---|---|---|---|
| 0% | - | 5.8x | 10.3x | 27.0x |
| 1% | 5.4x | 14.1x | 15.6x | 35.7x |
| 2% | 6.5x | 14.9x | 19.1x | 37.0x |
| 4% | 6.9x | 15.4x | 22.9x | 37.7x |

loss and the loss can be further healed by retraining. Table 3 has shown that if we combine pruning and quantization together, the compressed model size would be the smallest with negligible accuracy loss. Together with Huffman coding, the model size of AlexNet can be reduced by 35 times, and that of VGG-16 can be reduced by 49 times. We should notice the randomness of sparsity from pruning, which is tough to be efficiently used for hardware execution. To deal with this case, we add some constraints to limit the pruned connections in regular patterns, and this can increase the number of all zero channels for more skips during the acceleration process.

Moreover, we can see that inside the basic MAC operations of CNN, multiplication is always the most resource consuming operation, so reducing the number of multiplications can also enhance the hardware performance. Matrix transformation like Winograd [72] and FFT [73] can achieve this goal by targeting different sizes of filters. Take Winograd transformation as example, if we tile the input feature maps into $6 \times 6$ blocks and convolve it with $3 \times 3$ filters, through transformation we can reduce the number of multiplications by 2.25 times and replace them with cheap add and shifting operations.

With all these compression methods above, we can reduce the workload of the original model, which will benefit the on-chip memory and arithmetic resources and system throughput speed.

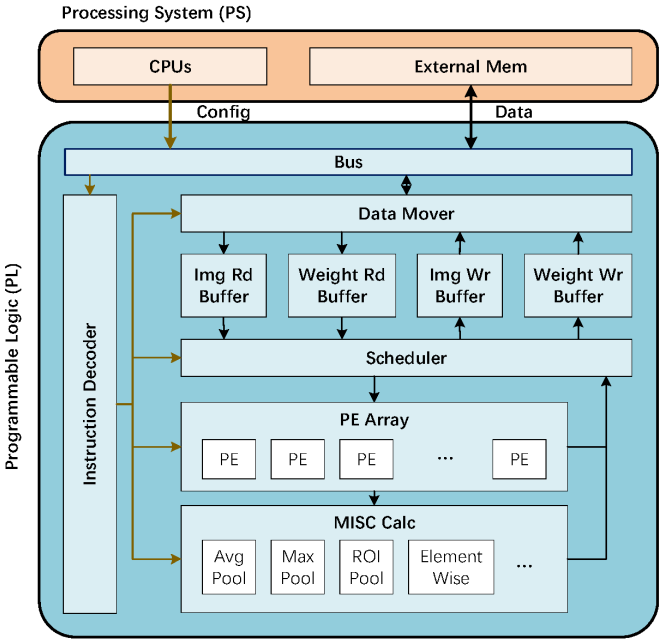


**Figure 10**. Our CPU+FPGA system architecture.

### 4.3. Hardware architecture design

Our Aristotle hardware architecture design [74] is given in Fig. 10. A CPU+FPGA accelerator design is adopted, which consists of two parts: the processing system (PS) and the programmable logic (PL). PS contains the low-power CPU processors and the external memory, which offers programmability and data capacity. Instructions will be transferred into PL and decoded to implement the control of PL. PL is the on-chip design where the majority of the CNN accelerator logic is located, and can be scalable due to the chosen FPGA platform. PEs are placed inside PL for parallel MAC operations, which can complete the convolving process through multiple iterations. Some functions that cannot be efficiently accelerated with PE, such as several kinds of pooling and an element-wise dot product, will be contained inside a MISC calculation pool for optional use. On-chip buffers will be provided to offer data reuse opportunities controlled by a scheduler, and communicate with external memories using a data mover such as a direct memory access controller (DMAC). Such hardware architecture design can be easily shared between layers which are friendly to instruction generation and high-level programming.

Instead of combining every multiplication of one filter window together, we split the computing kernel into smaller granularity, which can avoid the waste of arithmetic resource while dealing with a large filter size or window stride, and can ensure a regular data access pattern for easier control. Furthermore, a smaller granularity of PE can increase the chance of skipping for sparsity, which can save the overall execution time of the system.

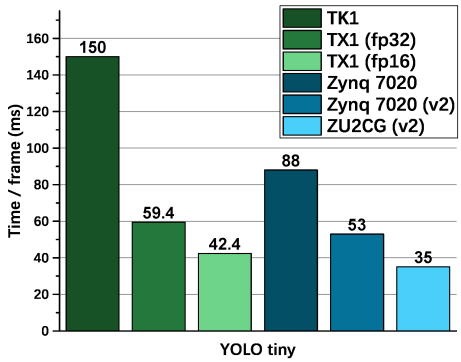


**Figure 11**. Evaluation results of YOLO-tiny on mobile GPUs and different FPGA platforms.

**Table 4**. Evaluation results of SSD on CPU, GPU and FPGA platforms.

| Platform | Intel Xeon E5-2640 v4 | NVIDIA GTX 1080TI GPU | Xilinx ZU9 FPGA | |
|---|---|---|---|---|
| Task | SSD (YOLO) | | | SSD (YOLO) Pruned |
| Operations (GOPs) | 16.6 | | | 7.4 |
| fps | 4.88 | 183.48 | 9.09 | 20.00 |
| Power (W) | 90 | 250 | 14 | |
| Efficiency (fps/W) | 0.054 | 0.734 | 0.649 | 1.429 |

### 4.4. Performance evaluation

We use the YOLO algorithm to evaluate our Aristotle system, which is the most popular real-time detection algorithm in the RTAV area. Fig. 11 shows the comparison of performance on different platforms. We can see that compared with the same level mobile GPU platforms our system can reach a similar performance. However, the power consumption of our Zynq-7020 and ZU2 based systems are around 3 W, while the power of GPU is 15 W. Moreover, the peak performance of TK1 is 326 GOPS and that of TX1 is 1 TOPS, while the peak performance of our FPGA platforms is only around 100 GOPS. These can prove a much better efficiency of our system design.

We also use the YOLO version SSD [75] algorithm to compare our larger FPGA systems with CPUs and GPUs. SSD is an optimized algorithm based on YOLO with multi-scale feature extractions which can improve the ability to capture small objects. Table. 4 lists the results on different platforms. We can see that both GPU and FPGA solutions can reach a faster performance than the Intel Xeon CPU. The power consumption of the NVIDIA GTX 1080TI GPU can get up to 250 W, while the value of FPGA is only 14 W. From the perspective of efficiency, with the pruning method implemented, our design can get an efficiency almost twice that of 1080TI GPU.

Furthermore, we have tested a Densebox [76] model on our

**Table 5**. Evaluation results of Densebox on GPU and FPGA platforms.

| Platform | NVIDIA GTX 1080TI GPU | | Xilinx ZU9 FPGA |
|---|---|---|---|
| Input Size | 640x360 | | |
| Task | Densebox | Densebox Pruned | |
| Operations (GOPs) | 28 | 1.2 | |
| fps | 150 | 330 | 300 |
| Power (W) | 250 | | 14 |
| Efficiency (fps/W) | 0.60 | 1.32 | 21.43 |
| Recall | 0.875 | | |

platform and a peer GPU. Densebox is an end-to-end fully convolutional network (FCN) which has been widely used in face detection applications, and face detection is an essential part of the in-vehicle driver status recognition, such as drowsiness detection. We have pruned the model with the method mentioned in clause 4.2 from 28 GOPs to 1.2 GOPs, with the recall rate staying the same. Table. 5 shows that with the help of pruning, our ZU9-based platform can reach twice the speed of the 1080TI GPU. The GPU can also get a 330 fps with the pruned model, but the utilization rate of model sparsity is quite low considering the peak performance of 1080TI is almost 10.6 TOPS, which results in an efficiency which is 16 times worse than our ZU9 FPGA, reflecting the fit between our compression methods and our hardware system.

### 4.5. Tingtao: an ASIC-based reconfigurable accelerator

Our ASIC-based reconfigurable accelerator Tingtao is already on schedule. The PS of Tingtao is an ARM Cortex-A5 processor, and the PL includes two deep-learning processing units (DPUs), each containing 2048 MAC PEs and works at 500MHz. Some necessary interfaces for RTAV application are also integrated. Tingtao has taken a 28nm CMOS technology and is projected to provide a peak performance of 4 TOPS at a power of 3 W, which is slightly better than the EyeQ4 product. With the compression method and compiling optimization introduced, the performance could get even better. As shown in Fig. 7, Tingtao has filled the sparse area of 1 to 10 W of power and TOPS level throughput. We are also planning to try a larger design for our next version, and we will pay efforts in the ongoing research of the implementation of emerging memory technology based on our precedent work [64] for the target of our development route.

## 5. CONCLUSION

This article has reviewed the algorithms for RTAV applications of ADAS, a comparative analysis has been done over different types of platforms, and an enumeration of chances and challenges for reconfigurable RTAV platforms. We have introduced the software-hardware co-design workflow for our reconfigurable RTAV system, with detailed hardware architecture design and implemented compression methods, which ensure an efficient execution with programmability. Evaluation shows that our system can get the best efficiency among peer processors with a satisfying real-time processing performance. An ASIC-based solution can further exploit the efficiency, which means a similar throughput speed with the FPGA-based Aristotle system and an energy cost of one order of magnitude less.

There are some other deep learning models utilized in RTAV applications. Recurrent neural network (RNN) is one of them, and the long-short term memory (LSTM) model [77] shows excellent performance in classifying, processing and predicting time series. This feature can be helpful for object tracking and action predicting functions in ADAS systems. We have not expanded on this topic in this article, but we have already released a similar design based on our Aristotle system framework [78], which has proved the capability of processing various deep learning models.

Future RTAV processors need to offer a 10-100 TOPS throughput speed with less than 30 W power, and to realize this we could count on the rapid development of workload compression such as extreme low-bitwidth CNNs [79][80][81][82] and novel pruning ideas [83][84], hardware design such as dataflow optimization [85][86] and sparsity supported architecture [87][88], and emerging memory technology implementation [60][89]. We are confident that with all those mentioned above, the reconfigurable products will thrive in the ADAS market.

## REFERENCES

[1] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine vision and applications*, vol. 1, no. 4, pp. 223–240, 1988.

[2] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, "Vision and navigation for the carnegie-mellon navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.

[3] E. D. Dickmanns and B. D. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 199–213, 1992.

[4] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs, *Disruptive technologies: Advances that will transform life, business, and the global economy*. McKinsey Global Institute San Francisco, CA, 2013, vol. 180.

[5] J. Barbaresso, G. Cordahi, D. Garcia, C. Hill, A. Jendzejec, and K. Wright, "USDOT's intelligent transportation systems (ITS) ITS strategic plan 2015-2019," Tech. Rep., 2014.

[6] Cabinet of Japan, "Statement on "forging the world-leading it nation"," Tech. Rep., 2013.

[7] Directive 2010/40/EU of the European Parliament and of the council, "Directives on the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport," Tech. Rep., 2010.

[8] European Comission. Directorate-General for Mobility and Transport, *White Paper on Transport: Roadmap to a Single European Transport Area: Towards a Competitive and Resource-efficient Transport System*. Publications Office of the European Union, 2011.

[9] European Comission, "Preliminary descriptions of research and innovation areas and fields, research and innovation for europe's future mobility," Tech. Rep., 2012.

[10] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *arXiv preprint arXiv:1704.05519*, 2017.

[11] Intel, "Intel atom processor e3900 series." [Online]. Available: https://www.qualcomm.com/solutions/automotive/drive-data-platform

[12] Qualcomm, "Drive data platform." [Online]. Available: https://www.qualcomm.com/solutions/automotive/drive-data-platform

[13] N. Corp., "NVIDIA drive PX - the AI car computer for autonomous driving," 2017. [Online]. Available: http://www.nvidia.com/object/drive-px.html

[14] NVIDIA CUDA, "NVIDIA CUDA C programming guide," *Nvidia Corporation*, vol. 120, no. 18, p. 8, 2011.

[15] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cuDNN: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[16] Xilinx, "Xilinx automotive Zynq-7000." [Online]. Available: https://www.xilinx.com/publications/prod_mktg/ZynqAuto_ProdBrf.pdf

[17] Altera, "A safety methodology for ADAS designs in FPGAs." [Online]. Available: https://www.altera.com/en_US/pdfs/literature/wp/wp-01204-automotive-functional-safety.pdf

[18] Mobileye, "The evolution of EyeQ," 2017. [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/

[19] J. Son, H. Yoo, S. Kim, and K. Sohn, "Real-time illumination invariant lane detection for lane departure warning system," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1816–1824, 2015.

[20] V. Gaikwad and S. Lokhande, "Lane departure identification for advanced driver assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 910–918, 2015.

[21] A. Broggi, M. Bertozzi, A. Fascioli, and M. Sechi, "Shape-based pedestrian detection," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 215–220.

[22] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[23] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[24] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[25] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[26] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*. Springer, 1995, pp. 23–37.

[27] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[34] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, 2017, pp. 4278–4284.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[38] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[39] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision*. Springer, 2014, pp. 346–361.

[41] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[42] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[44] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[45] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[46] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision*. Springer, 2016, pp. 354–370.

[47] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 924–933.

[48] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2129–2137.

[49] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, 2015, pp. 424–432.

[50] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156.

[51] NICS Lab of Tsinghua University, "Neural network accelerator inference," 2017. [Online]. Available: https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/

[52] NVIDIA, "NVIDIA Tesla V100," 2017. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-v100/

[53] ——, "NVIDIA Jetson - the embedded platform for autonomous everything," 2017. [Online]. Available: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

[54] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 26, no. 2, pp. 203–215, 2007.

[55] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *ACM Sigplan Notices*, vol. 49, no. 4. ACM, 2014, pp. 269–284.

[56] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[57] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10 tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 246–257.

[58] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," *arXiv preprint arXiv:1704.04760*, 2017.

[59] J. Jeddeloh and B. Keeth, "Hybrid memory cube new dram architecture increases density and performance," in *VLSI Technology (VLSIT), 2012 Symposium on*. IEEE, 2012, pp. 87–88.

[60] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and efficient neural network acceleration with 3d memory," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2017, pp. 751–764.

[61] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*. IEEE, 2016, pp. 380–392.

[62] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[63] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 14–26.

[64] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 27–39.

[65] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 247–257.

[66] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170.

[67] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-s. Seo, and Y. Cao, "Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 16–25.

[68] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 609–622.

[69] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3. ACM, 2015, pp. 92–104.

[70] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "Pudiannao: A polyvalent machine learning accelerator," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1. ACM, 2015, pp. 369–381.

[71] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[72] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4013–4021.

[73] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *arXiv preprint arXiv:1312.5851*, 2013.

[74] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

[75] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[76] L. Huang, Y. Yang, Y. Deng, and Y. Yu, "Densebox: Unifying landmark localization with end to end object detection," *arXiv preprint arXiv:1509.04874*, 2015.

[77] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[78] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang *et al.*, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA." in *FPGA*, 2017, pp. 75–84.

[79] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[80] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[81] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[82] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[83] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[84] V. Sze, T.-J. Yang, and Y.-H. Chen, "Designing energy-efficient convolutional neural networks using energy-aware pruning," 2017.

[85] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer cnn accelerators," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 2016, pp. 1–12.

[86] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[87] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-x: An accelerator for sparse neural networks," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 2016, pp. 1–12.

[88] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017, pp. 27–40.

[89] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 541–552.