aiWare™
Hardware innovations
for automotive AI

Exploiting parallelism in NN workloads to realize scalable, high performance NN acceleration hardware

written by: Tony King-Smith

Abstract: Many automotive system designers, when considering suitable hardware platforms for executing high performance NNs (Neural Networks) frequently determine the total compute power by simply adding up each NN's requirements – the total defines the capabilities of the NN accelerator needed. Or does it?

The reality is almost all automotive NN applications comprise a series of smaller NN workloads. By considering the many forms of parallelism inherent in automotive NN inference, a far more flexible approach, using multiple NN acceleration engines, can deliver superior results with far greater scalability, cost effectiveness and power efficiency.

Looking “under the hood” of AI workloads

When considering the design of a hardware platform capable of executing the AI workloads needed for automated driving, many factors need to be considered. However, the biggest one is uncertainty: what capabilities does the hardware actually need to execute the worst case NN workload, and how much performance do I need to safely and reliably execute that? And how much of the time do I need to run that worst case workload?

The challenge becomes even harder when designing an SoC (System on Chip) for automotive. Semiconductor vendors must sell their products to multiple customers to get sufficient return on the massive investment required to bring a new SoC to market. But how do they decide what workloads will best represent what customers will want to execute, when automotive AI is such a fast-moving and constantly evolving field? And since it takes at least 3-4 years from design start to the first chips being fully qualified for production use, how do we know that the assumptions made at the start of the design process for indicative workloads are in any way correct, or even appropriate?

Since 2015 we have been developing our in-house hardware, software and systems skills to understand what it really means to deploy scalable AI in high volume production vehicles.

Also, as the complexity and sophistication of AI workloads continues to rise, so does power consumption to execute them. Given the enormous amount of compute power we can now integrate on one chip, one way SoC designers tackle this problem is to make use of what is known as “dark silicon”: switching off all power to areas of the chip not being used.

This approach has been extensively used in mobile phones for the past few years, and can make a substantial difference to power consumption – but only if the chip design is conceived in alignment with the software from the start to utilize it. This relies on the fact that in reality the AI subsystems will use multiple different workloads, that are switched in and out from time to time according to the current operating conditions and environment (eg highway, urban or parking modes, low vs high volumes of traffic, different weather conditions).

Without a deep understanding of the range of workloads and how they each operate, SoC designers are forced to target the worst case, which often means large parts of the chip’s capabilities are rarely used. For an automated vehicle, that means unnecessary cost, and much higher power consumption. That is why AImotive develops both software and hardware technologies, so we can take a holistic approach to system design. And nowhere does that have greater impact than when designing high performance NN inference systems for automated driving.

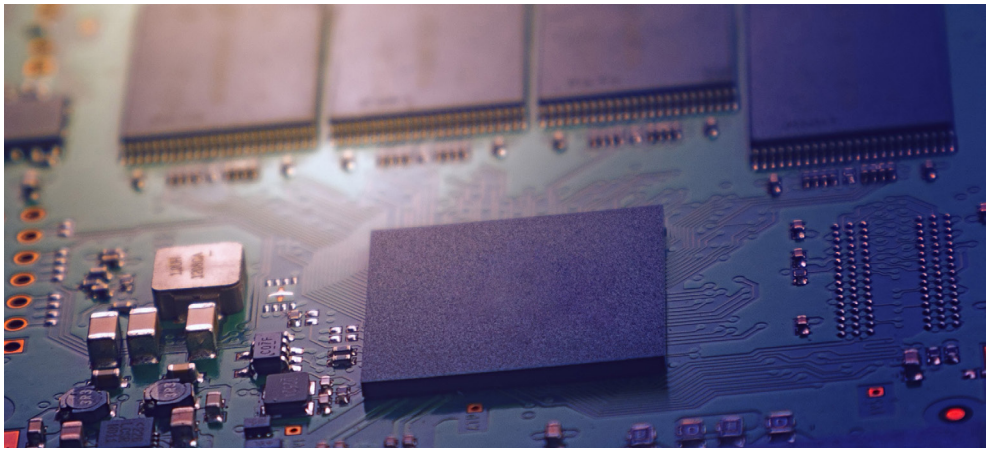
The total is the sum of many parts

When designing automated driving solutions based on CNNs, designers are faced with many challenges, such as:

- What are the worst case workloads I need to execute, and do I know exactly what those workloads will be?
- How do I enable continuous improvement over the life of the system, as NN research continues to advance rapidly after I have frozen the design of the hardware and/or processing chips?
- How can I make my hardware platform scalable, so I can allow for performance upgrades, and also enable smaller or larger configurations depending on the vehicle model or sensors used?

What is the workload?

Once challenge of designing hardware systems is that almost always the final software is far from ready when the hardware design is being finalized. Indeed, if a new chip is part of the solution, that needs to be finalized 3-4 years ahead of SOP (Start of Production), when often even the higher level architecture is not yet finalized, and implementation has barely begun.



The design of chips for an AI-enabled vehicle are often finalized at least 3-4 years before the car enters volume production, well before the AI algorithms are finalized, or even the final sensor configurations used

Traditionally, engineers solve these problems by over-specifying the hardware platform. By building in a level of contingency for speed, memory etc, the hardware designers can be confident that the hardware will be able to support the final solution. And we all know that the performance requirements never shrink during implementation!

Executing NNs requires extremely high performance engines, measured in many tens or hundreds of TOPS (Trillions of Operations Per Second). If you try to build in contingency, these engines get even bigger and more power-hungry. When considering tough cost and power constraints, the pressure is inevitably to reduce the size of these NN accelerators. That doesn't fit well with a conservative design methodology, where so much about the final AI software or NNs is not known.

THIS IS WHY WE DESIGNED AIWARE HARDWARE IP FIRST AND FOREMOST AS A HIGHLY SCALABLE ARCHITECTURE.

Our aiWare hardware IP complement our modular aiDrive software technology portfolio, by offering our customers and partners alternatives to mainstream solutions that are often over-specified. By bringing thought leadership to hardware-software codesign at both the board and chip level AImotive is therefore best positioned to help our partners create the most optimized product that can best adapt to future market trends.

Do we need one big NN accelerator engine?

The simple answer: no! When we were designing the architecture for aiWare, we looked at all the different NN workloads within an AI-based system, which together consumed more than 100 TOPS of processing horsepower. Based on experience with our own aiDrive software, as well as extensive discussions with many OEMs and Tier1s about how they are building their own solutions, we could see that in practice it is never one single large "monolithic" NN workload: it is a series of NN workloads, some executing the same task in parallel, some pipelined

AI systems use multiple NNs in various ways, breaking down the task into a series of modules. Furthermore, as raw data (eg the pixels received by a camera sensor, or the points generated by a 3D radar) is processed through the AI system, it is often the case that some work is done for each sensor before being combined with data from other sensors. This initial workload, known sometimes as pre-processing, often dominates the total TOPS budget. It's just one of the ways that parallelism inherent in a NN-based system can be leveraged to design more flexible hardware platforms.

Do you need to scale all performance equally?

Another challenging trend of NN research is that as knowledge grows, and demands increase for greater safety and robustness, the performance demands scale upwards, sometimes demand multiple times more NN performance. Often these advances in AI technology can result in significant improvements in safety and quality of sensing and decision making. So how do we take advantage of these advances in a vehicle designed for a 10-20 year operating life?

Having a single NN accelerator core is, by definition, limited in capacity, however powerful. What happens when you exceed the capabilities of that engine? And if that NN accelerator is integrated into an SoC, will you be forced to move to a new SoC? That will require re-validation of all the software of every function executing on that SoC – not just the NN parts. That is extremely costly and time-consuming, if the only increase you really want is for the NN accelerator to have greater capacity?

Should hardware scale over lifetime?

As experience grows rapidly for integrating AI into automated vehicles, so too does the need for modularity and scalability. These days, cars often use common platforms for the underlying chassis, which can then be adapted to the various different models. OEMs and Tier1s are now starting to apply similar concepts to the vehicle software, by bringing together all the different software components – sometimes distributed over 50 or 100 ECUs (Electronics Control Units) – into a common software platform.

As knowhow continues to evolve in NN research for improved safety and robustness, and software in vehicles becomes more frequently updated during the car's operational life, methods for enabling scalability and upgradeability of AI is becoming ever more important

As cars become increasingly upgradeable during their lifetime, the electronics needs to also move towards a standard, modular, scalable and upgradeable hardware platform. But the different types of processors need to be upgradable separately: in particular the NN acceleration hardware, where algorithms and workloads are likely to change dramatically every year for the foreseeable future as this exciting field continues to evolve and mature, and we all learn how AI can best be deployed for automated driving.

The evolution of optimized embedded computing from generalized computing

Embedded computing is a highly sophisticated area. Over the past 30 years, as chip integration surged forward delivering ever more processing power at lower power consumption, new computing architectures have evolved to fully exploit the enormous potential of large scale chip integration.

As performance scales, increasing use is made of more specialized processors – sometimes known as accelerators – that do a very specific range of tasks much better than a general purpose processor (such as an Intel x86, Arm or RISC-V CPU). Some well-known examples include:

- The GPU – Graphics Processing Unit – that revolutionized user interfaces and applications forever on anything from PCs to smart phones
- The Crypto accelerator – that enabled real-time high speed encryption on enormous volumes of data, in anything from a central banking system to a chip and pin credit card
- The video accelerator – that enabled us to watch hour after hour of HD quality films on a small handheld smartphone, and capture incredibly high quality video from the smallest cameras and smartphones, and even drones

Each of these specialized processors delivers orders of magnitude greater performance – but only for the tasks they are designed to accelerate. For everything else, they are pretty useless! Since they are complex, they therefore need to be used sufficiently to justify their cost in any system.

NN accelerators – the latest disruptor

Engineers have been designing NN accelerators for decades. It is a perfect textbook challenge: a simple, highly parallel task that needs to be executed billions of times per second. However, with the almost unimaginable integration now possible with the latest chip technologies, NN accelerators can now be created delivering up to 100s of TOPs, by some of the biggest names such as Google, Facebook, Nvidia and Intel.

These have been initially very advanced, high power designs targeting data centers, where power and cost constraints are minimal: achieving the highest performance is everything.

However, when designers try to use those same chips in something as highly constrained as an automated vehicle, things start to go wrong:

- The chips are far too expensive to be viable for volume production vehicles
- They consume far too much power
- They are not designed for real-time operation
- They are not sufficiently reliable to be used in an application involving human life

The automotive market needs highly optimized solutions

When you break down what goes into today's car, it is amazing what is achieved for the price. The sheer breadth of different technologies, functions and capabilities is breathtaking. Furthermore, these days each car manufacturer seems capable of producing more different and unique models every year: every possible shape, size and configuration.

But in order to do that, car manufacturers have been refining for decades the art of cost optimization. Every component is cost-engineered to only do exactly what is needed, nothing more; every function has to be extensively checked and certified as capable of doing its job for 10-20 years. And it has to always work: you don't want to be rebooting your car at 130 kph on a major highway!

We have established that AI is by far the best way to implement automated driving. But unlike many other parts of a car, AI is evolving so rapidly that any design started today will almost certainly be obsolete by the time it reaches production, let alone after 5-10 years in the market. So how do we deploy NNs in cars in a way that enables us to keep up?

Parallelism is the key to scalability

While the total NN processing power needed for a car might be 100 TOPS or more, that can be broken down into a number of smaller workloads. Each of these is unlikely to exceed 1-10 TOPS on its own. This can be referred to as “workload parallelism”.

NN processing itself is actually relatively simple: we just need to execute a lot of it! Whereas a general-purpose CPU needs to execute hundreds of different instructions of many different types, a NN processor need only execute a very few highly specialized instructions. However, whereas a CPU executes instructions sequentially (read one piece of data, then do something to it, then write a result out, etc) – a NN accelerator needs to execute one command on sometimes millions of pieces of data. This is known as “data parallelism”.

LET’S NOW SEE HOW SOME OF THESE CONCEPTS CAN BE APPLIED TO IMPLEMENTING SCALABLE HIGH PERFORMANCE NN PROCESSING.

Workload parallelism

In an automated vehicle, we have multiple sensors each gathering data at high speed. That data needs to be processed and combined to deliver one set of controls, based on inputs from all the different sensors, plus other information such as desired route, map information, planned trajectory etc.

AImotive has particularly strong expertise in systems where cameras are the primary sensor, so the example here will focus mainly on cameras. A simple case of workload parallelism is doing pre-processing of data from each camera separately. If each camera requires x TOPS to achieve the required IPS (Inferences Per Second), we can execute these NNs in several different ways, as shown in Figure 1a and 1b:

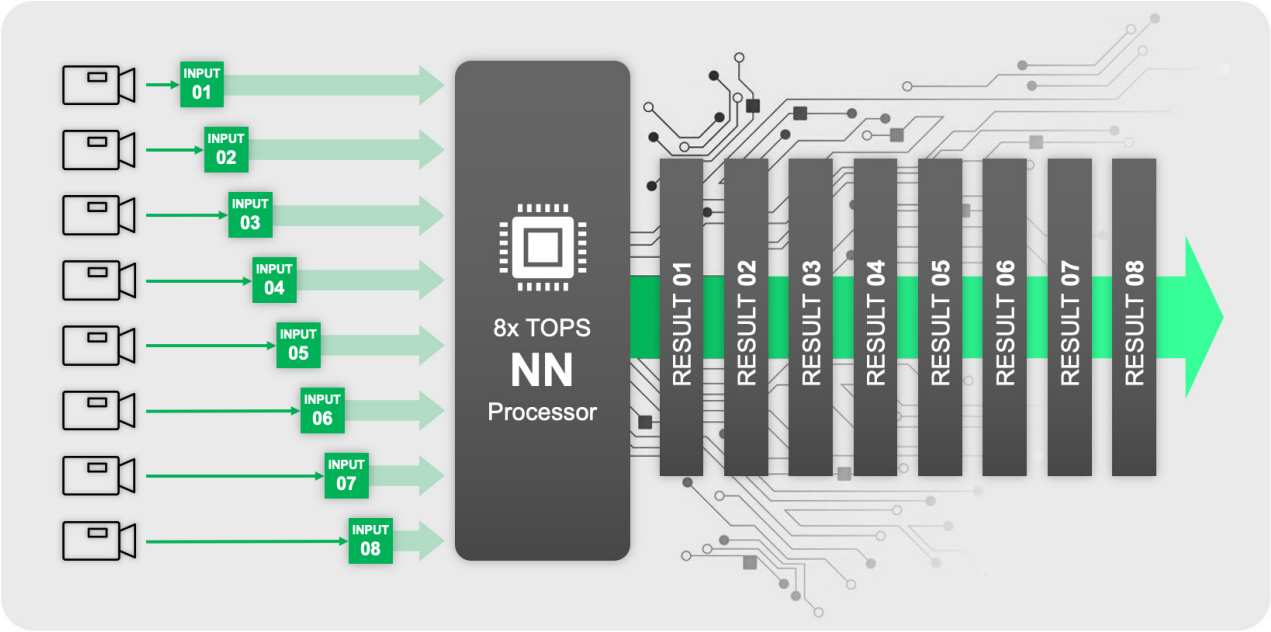


Figure 1a: sequential input workloads through one large NN accelerator

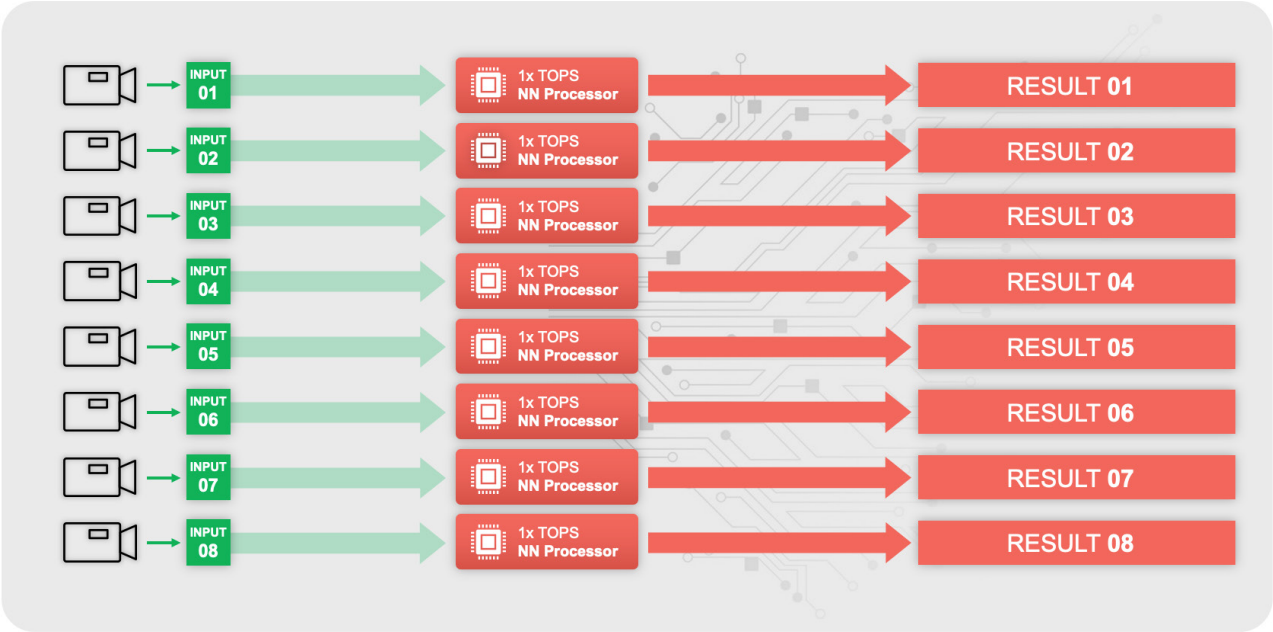
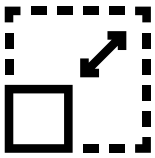
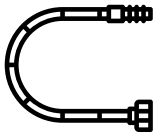


Figure 1b: parallel input workloads through multiple smaller NN accelerators

In principle, both solutions will produce the same results at similar speeds. However from a modularity, flexibility and scalability perspective there are several differences:



1. Scalability: if the sensors are upgraded, only the processors related to the upgraded sensors need to be upgraded. Also, only the sensor subsystem itself needs to be replaced, which is often far easier than replacing a central processor



2. Flexibility: A chip containing an accelerator capable of achieving 1x TOPS consumes 8x less power than a chip with a processor capable of 8x TOPS and is much cheaper. Therefore, it is more flexible, and can be manufactured in significantly higher volumes making it cheaper per TOPS. Also, can be installed in more places around the vehicle, such as next to the camera sensor itself, within an e-mirror enclosure etc

3. Upgradeability: if only upgrading the front end perception algorithms, then only the NN processor executing that needs to be upgraded; the rest of the AI system can remain unchanged

This approach is what is known as distributed computing: a series of smaller, cheaper processors distributed physically around the system. The debate as to which is better – centralized or distributed – has raged ever since the first microprocessors were invented.

For a car manufacturer, one key attraction of this approach is that not all the power consumption is focused in one small area, but instead distributed physically around the vehicle making it easier to dissipate heat. Another advantage is a distributed processor can be made more robust – if one fails, the rest are unaffected. However this approach is also a great way to implement NN acceleration in a centralized processor. The scalability of using

multiple smaller NN accelerators means that the central processor can be more finely configured for the end application, by only adding sufficient NN acceleration for the sensors used in that model, rather than the largest possible configuration. Cost engineering central processors will be key, and leveraging the flexibility of multiple smaller NN accelerators will deliver significant benefits for cost optimization.

But perhaps the most attractive advantage for Tier1s or OEMs is that if the NN accelerator is itself built from multiple smaller NN accelerators, it is far easier to implement a wide range of mid-life hardware upgrades without having to replace the central processor itself. This subsystem-level upgrade approach is particularly attractive for Tier1s seeking to maximize the value of their subsystem (ECU) products.

Workload parallelism: sensor fusion

It is well established that each type of sensor has strengths and weaknesses: no one sensor is the ideal solution. Therefore combining data from different types of sensors, such as cameras, radars, lidars and ultrasonics, will result in a higher quality result. This is known as "sensor fusion", and takes a variety of different forms.

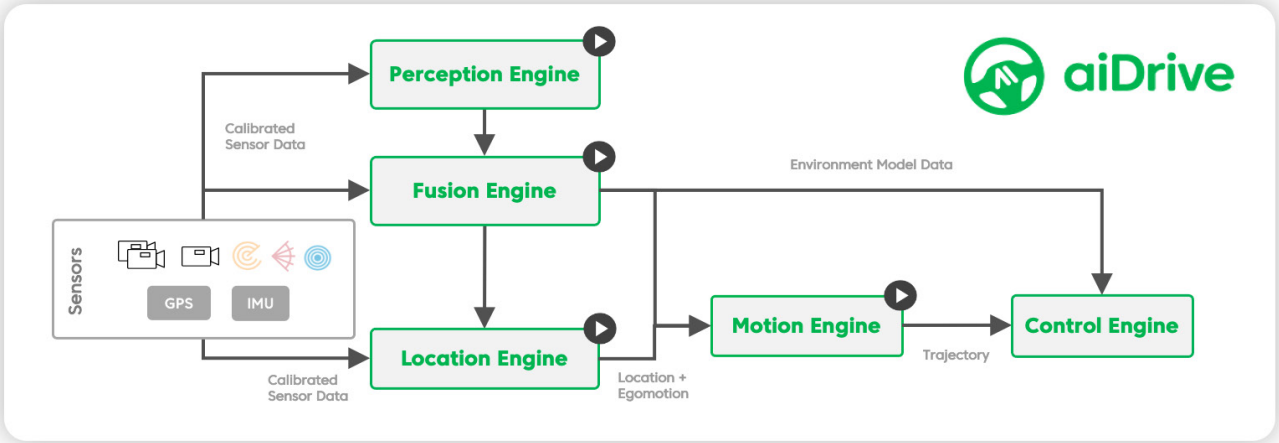


Figure 2: Sensor fusion is a key function of most automated driving solutions

Also multiple sensors may be combined to create a more powerful part of the solution, for example 4 cameras for a 360 degree surround view system. Since the data all comes from the same type of sensor, it is easier to perform fusion on this data. This can take two forms:

1 / **Data level fusion** (also known as “early” or “forward” fusion) (figure 3): the raw data from all the sensors are merged together, and one large NN is executed on it to perform object detection, before being passed on to a separate perception engine.

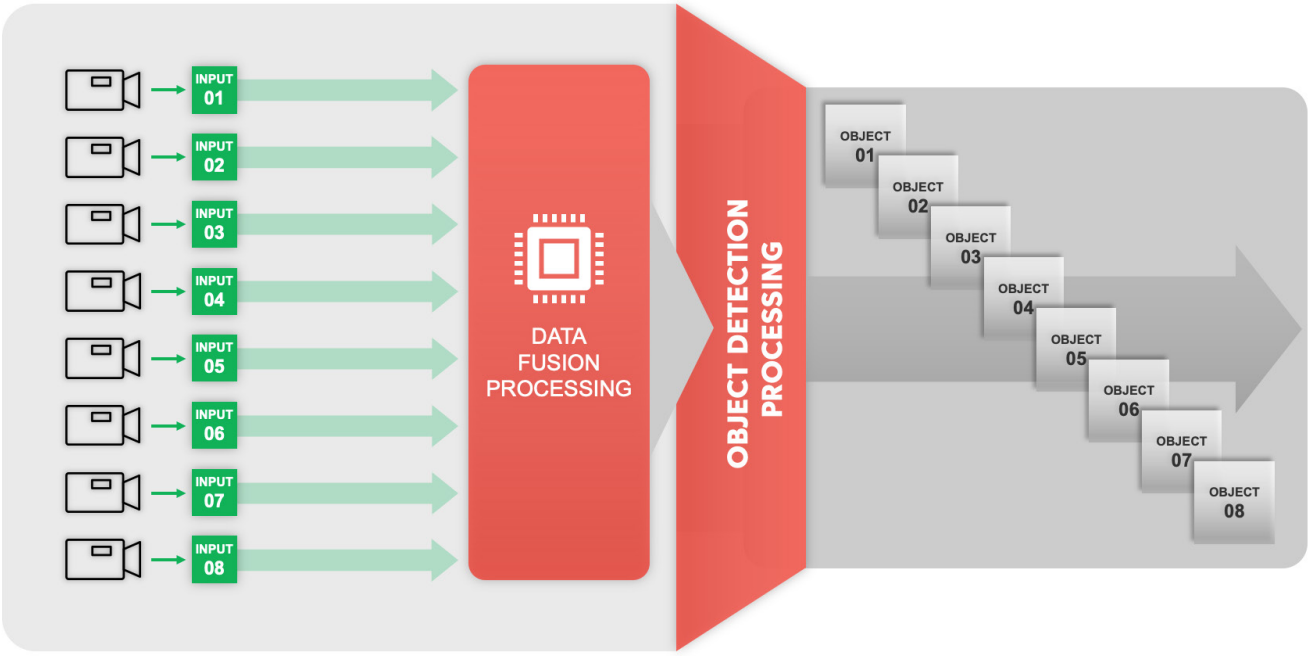


Figure 3: data (early) fusion takes data from a group of sensors, and creates a large, unified data map on which object detection is performed

2 / Object fusion (figure 4): (also known as “late” fusion) (figure 4): each sensor performs object detection locally, then only the objects detected from all sensors are brought into an object fusion engine, which then performs perception. One key advantage of object fusion is that the data sent to the central perception engine is orders of magnitude smaller than sending all the raw sensor data; indeed it is also significantly smaller than the fused data. Since the transmission of large amounts of data in a vehicle is a major challenge – for reliability as well as cost and power reasons – this is an area of considerable interest in the emerging area of sensor fusion.

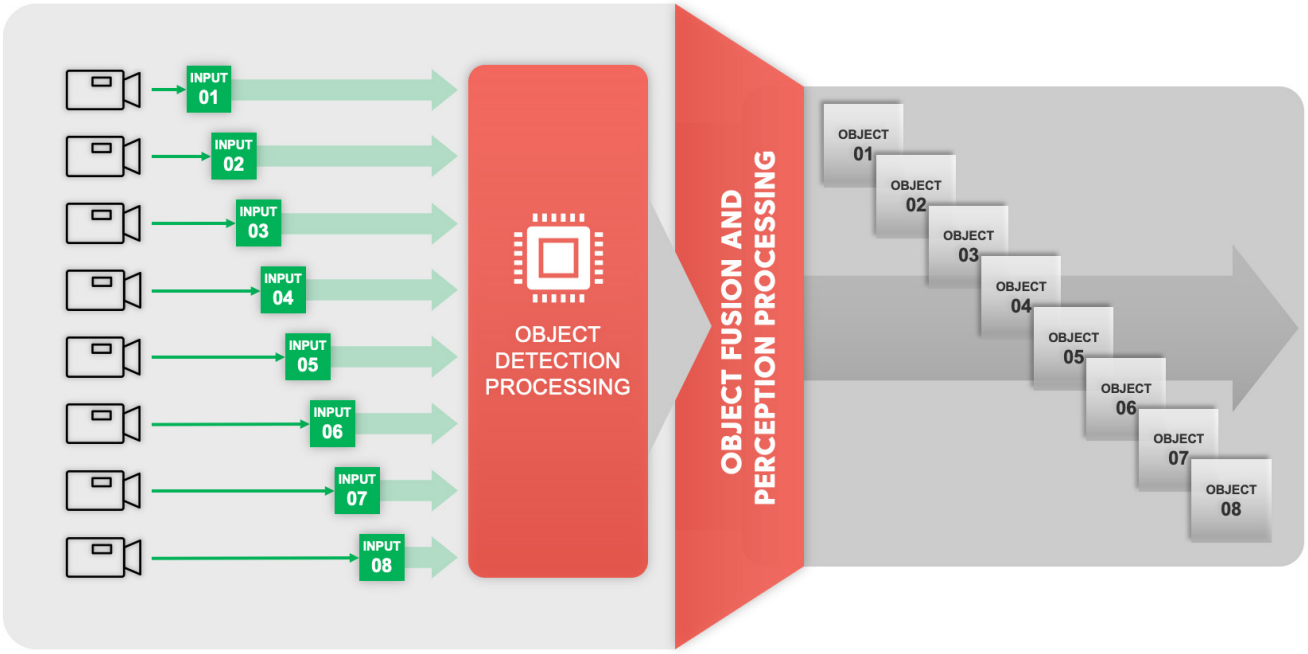


Figure 4: object (late) fusion takes objects from multiple object detection engines, and creates a smaller size unified object map on which final perception is performed

However another advantage of object (late) fusion is that each object detector can be executed separately, leading to opportunities to parallelise the execution:

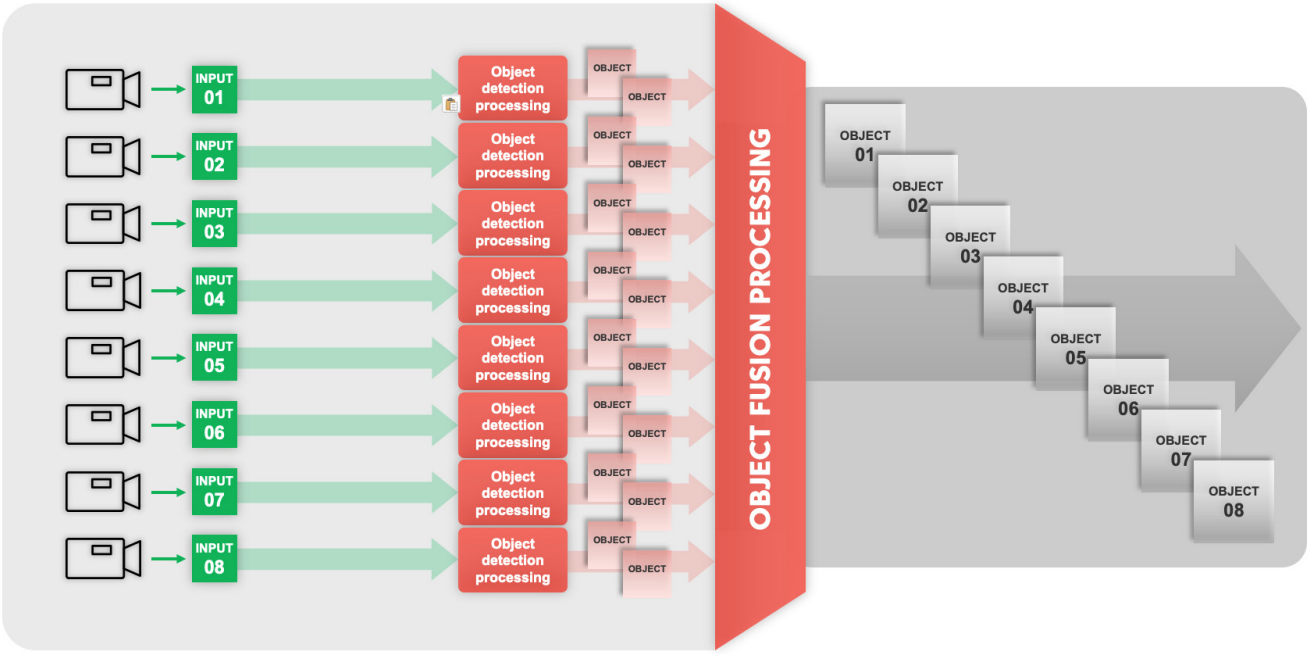


Figure 5a: Object detection can be performed on each image separately, with all fusion taking place in the Object Fusion engine

In Figure 5a, one separate NN engine can be used for each image. However, since each object detection workload is not so large as the total workload, these could be shared between a smaller number of parallel engines – four (Figure 5b) or even two (Figure 5c) depending on the object detection workload being performed. In this way, a variety of multi-core schemes can be used – either within one chip or using multiple chips – to achieve a highly scalable yet high performance Object detection subsystem.

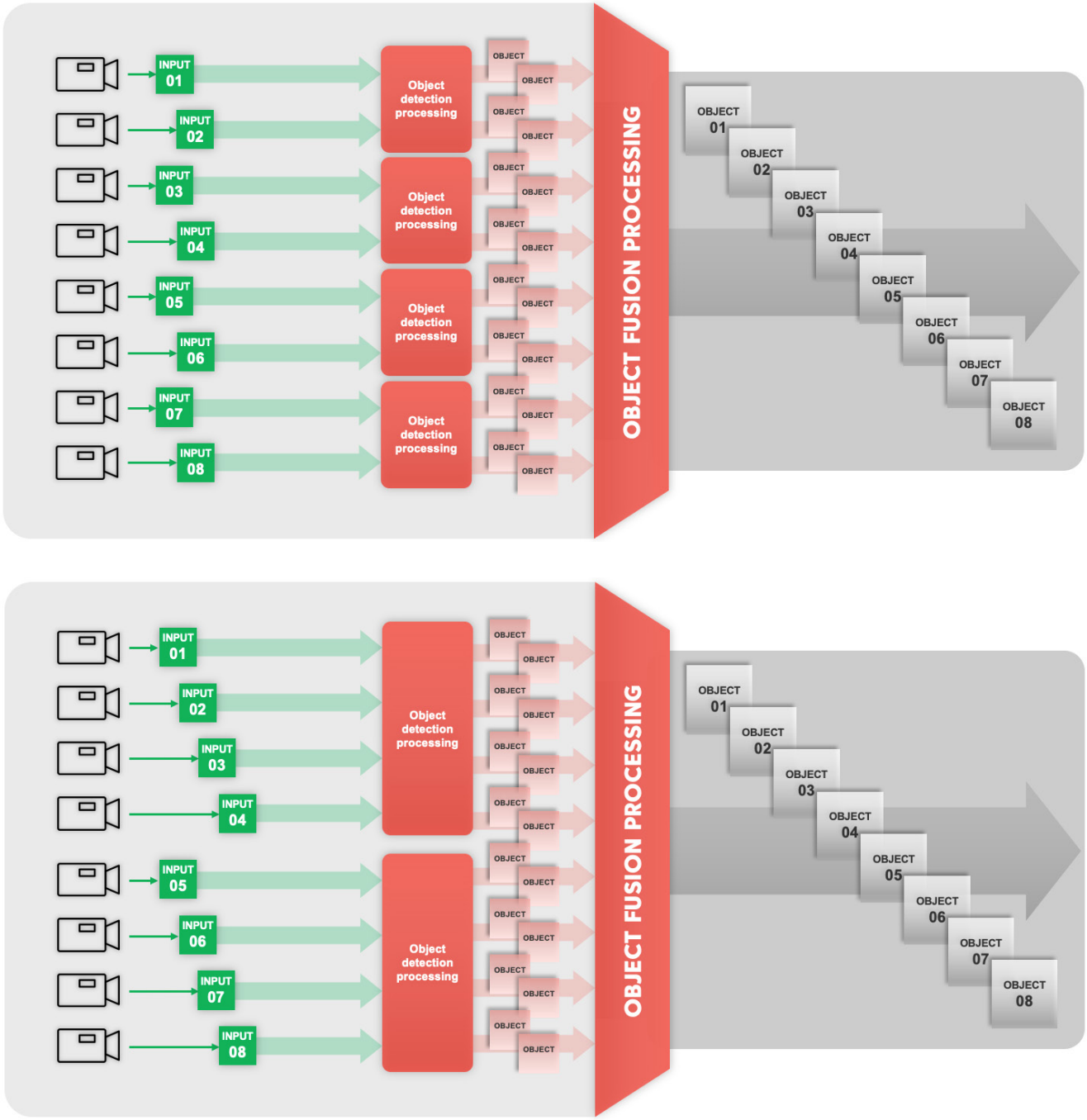
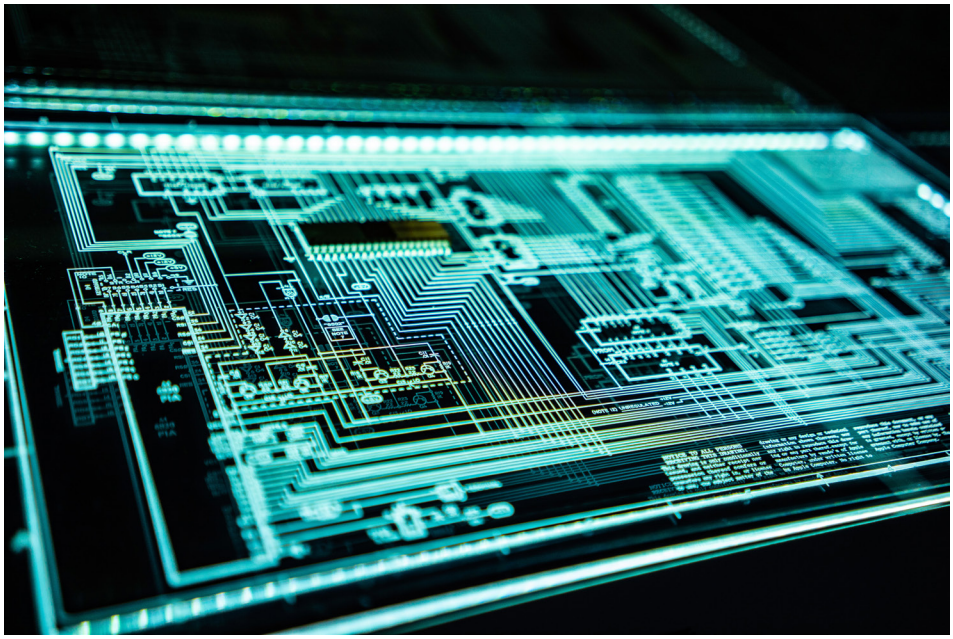


Figure 5b & 5c: Depending on the processing power needed for object detection, multiple inputs can be allocated to each engine

As explained earlier, having a more modular distributed architecture means the system can be more easily upgraded over time. Indeed, AI sensor configurations for low, mid-range and high end versions of vehicle models can be much more easily offered to customers using this approach.



Data and network parallelism: the aiWare approach

When we designed the aiWare architecture, we studied in great depth exactly what happens within every NN we used in our own aiDrive software solutions. We also exhaustively surveyed – and continue to do so – state of the art best practice and latest research on NN architectures. What were the best topologies? What layer types were most used? What sorts of optimizations are most effective when moving a NN trained in the lab to an embedded real-time inference environment?

> **THE AIWARE ARCHITECTURE MAKES EXTENSIVE USE OF MANY FORMS OF DATA , DERIVED FROM A DEEP UNDERSTANDING AND ANALYSIS OF A WIDE RANGE OF CNN TOPOLOGIES ALONGSIDE MULTI-LEVEL HARDWARE PARALLELISM TO MAXIMIZE CONCURRENT ALU OPERATIONS ACROSS EVERY PART OF AIWARE EVERY CLOCK CYCLE.**

The result is a highly scalable, power-efficient NN inference acceleration engine capable of sustaining >95% MAC utilization efficiency while delivering extremely low latency for some of the most demanding automated driving applications.

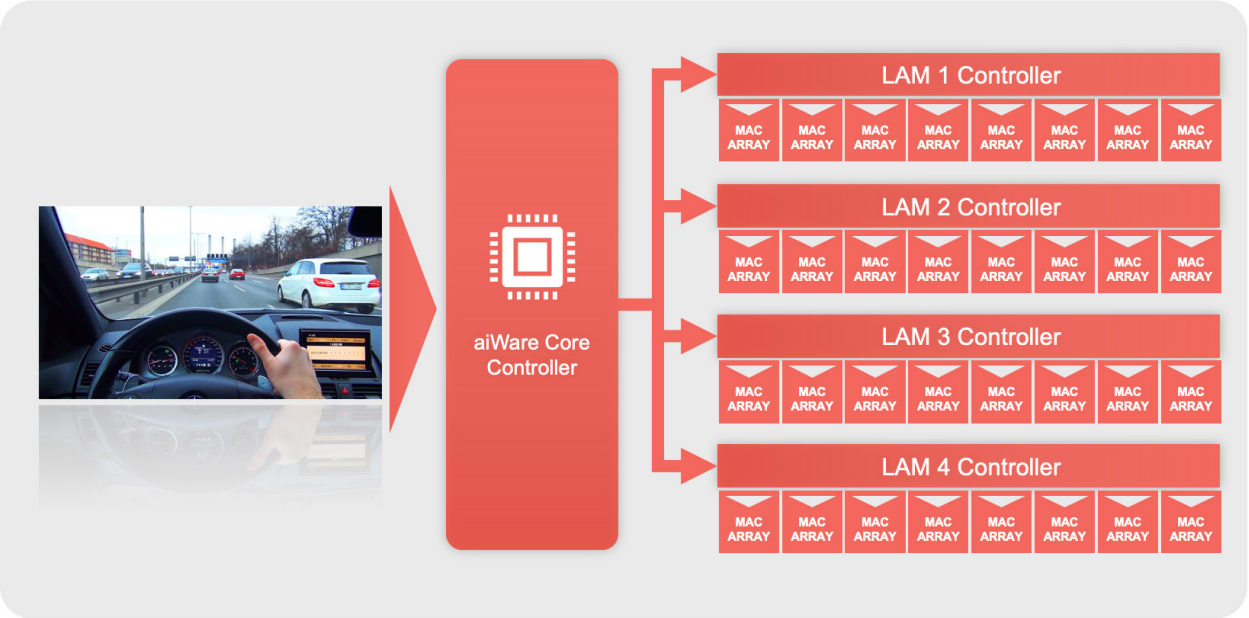


Figure 6: aiWare breaks down the input image into regions, which are then allocated to multiple parallel LAM engines. Each LAM engine is further broken down into multiple MAC Array execution units, each of which has multiple MACs executing in parallel.

The aiWare hardware IP core can be scaled from 1 to 32 TOPS per core, while maintaining extremely high efficiency. However, by adopting some of the concepts in this white paper, a wide range of multi-core aiWare configurations can be created delivering >100 TOPS without needing the significant added complexity of multi-processor technologies such as NOCs, coherent caches or demanding high speed wide buses. For example, a simple multi-core accelerator containing four aiWare cores can deliver >100 TOPS over full automotive temperature range in 16nm process technologies – ideal for a central processing cluster, or for zone controllers. Alternatively, using a chiplet approach with each chiplet containing one or two aiWare cores, a wide range of SIPs can be implemented.

A similar approach can be taken for multi-core implementations within a large SoC. With the potential for 7nm or 5nm processes to integrate astonishing amounts of logic and memory, multiple aiWare cores can be implemented within one SOC using these more straightforward approaches to parallelism.

By focusing on workload parallelism, these cores can be connected to the I/O, NOC and/or memory subsystems directly, rather than having to use complex multi-processor technologies such as coherent caches, bus snooping or complicated synchronization schemes. Also, the multi-core approach can be used to implement an elegant power island based power management system, switching on and off those cores needed as required to enable dynamic optimization of system power.

Because of the extensive use of so many forms of parallelism, aiWare can form the heart of of a highly scalable, modular, upgradeable NN acceleration solution supporting all the most intense NN computation needed in tomorrow's L2-L4 vehicles from the most luxurious to the most basic.

Other forms of parallelism

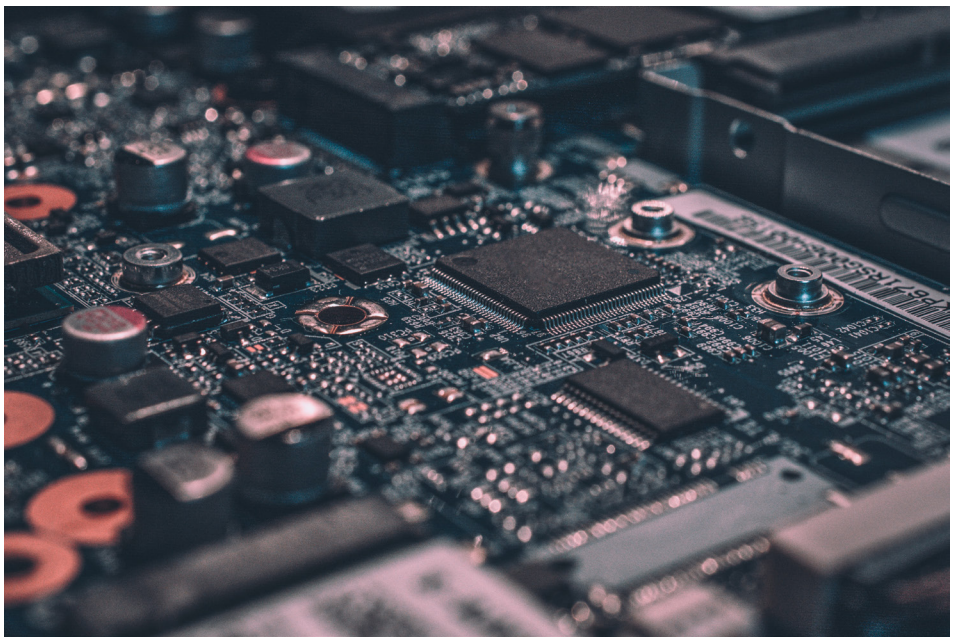
Beyond workload and data parallelism, other forms of parallelism exist. For example, by analysing closely the dataflow between one layer and the next, two or more layers can be executed at the same time by aiWare hardware. For example, as the convolution kernel is computed for each data point in a feature map in one layer, the result from that convolution could be then used to calculate the following activation or pooling function while the convolution continues to be calculated on later data points. In aiWare, this means using a C-LAM (computing convolutions) to share data with an F-LAM (computing an activation or pooling function).

> **THIS IS BUT ONE EXAMPLE OF WAYS AIWARE IS HIGHLY OPTIMIZED TO MINIMIZE LATENCY, RESULTING IN THE HIGHEST POSSIBLE THROUGHPUT IN THE SHORTEST TIME.**

Note this is quite different to "batching", a technique used extensively in ML (Machine Learning). Batch processing in principle is executing multiple tasks by grouping them together to improve efficiency on any computer.

For ML, it is used to optimize memory usage during the Training phase of ML, by keeping all data for a set of images (or other inputs) in memory while performing multiple calculations on that data. That's great for optimizing computation of non-real time ML training, but no good for real-time inference in a vehicle. That's why NN accelerators designed for ML are usually almost always inappropriate for embedded in-vehicle applications, despite their claims for extremely high TOPS.

Since aiWare was conceived for the sole purpose of accelerating AI Inference in a vehicle in real time, it was designed only to operate using "batch size=1", which means "process every input from start to finish the instant you receive it". This ensures that aiWare delivers the Inference result with the shortest possible latency.



Conclusions

The debate of how best to implement high performance - single large extreme performance computation engines, or arrays of much smaller processors - will continue to rage for as long as computer system engineering exists. However it is important to understand that the goals are more than just having sufficient computing power: it is also how to deliver that computation performance in a way that meets the needs of the end application.

We believe that aiWare is one of the most flexible yet highly optimized NN acceleration solution available today for automotive AI. Thanks to its very high MAC utilization over a wide range of imaging or sensor fusion workloads, aiWare solutions can achieve superior performance to many other well-known chip solutions claiming 2x-3x greater TOPS capacity. And since aiWare is in itself extremely power-efficient delivering as much as 4 TOPS/W under worst-case process corners and temperature and supply voltage extremes, high performance AI can be deployed in the widest possible range of physical locations and system architectures.

Automotive applications are unique in their demands for a combination of ultra-low latency and power consumption over an extremely wide temperature range. It is s not easy to achieve the most flexible, cost-effective solution that will be easy to deploy in volume production yet comply with the most stringent safety demands while operating for hours every day over a long lifetime.

It is that background complemented by an in-depth understanding of parallelism in its many forms at both hardware and application level, that make AImotive's aiWare NN acceleration hardware IP stand out from the crowd.

Tony King-Smith
Executive Advisor at AImotive

Tony is a respected electronics technology executive, with more than 40 years' experience advising and managing R&D and marketing teams. He is best known as the former Chief Marketing Officer of Imagination Technologies. Tony is an internationally recognized expert in SoCs, semiconductors, embedded software and intellectual property, and has held senior management positions with British Aerospace, Inmos, LSI Logic, Hitachi (Renesas) and Panasonic.