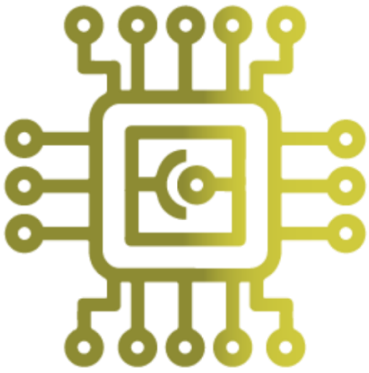

CONVOLVE

Seamless design of smart edge processors

GRANT AGREEMENT NUMBER: 101070374

Deliverable D1.3

Update of requirements and use cases

Title of the deliverable	Update of requirements and use cases
WP contributing to the deliverable	WP 1
Task contributing to the deliverable	Task 1.2 – 1.4
Dissemination level	PU – Public
Due submission date	30/06/2024
Actual submission date	29/08/2024
Author(s)	VIN, GNA, BOS, TASE
Internal reviewers	Egbert Jaspers (VIN) Marc Geilen (TUE)

Document Version	Date	Change
V0.1	04/07/2024	Initial table of contents
V0.2	09/08/2024	First complete draft
V1.0	14/08/2024	Final version for review
V1.1	29/08/2024	Final version

Deliverable Summary

The use cases in the CONVOLVE project consider a wide range of applications such as audio processing, video processing and satellite imagery. Each of these application use-cases have requirements that are very different from each other, and the consequences of not adhering to these requirements might have severe impact on usefulness and usability. During the initial phase of the project, the requirements of the use-cases were determined. This document outlines the updated requirements for the use cases based on the research results and progress made in the CONVOLVE project.

1. Introduction

Edge artificial intelligence (AI) starts with edge computing. Edge AI refers to AI algorithms that process locally on hardware devices and can process data without any type of connection. This means operations such as data acquisition can occur without streaming or storing data in the cloud because of the need for immediate response, besides preventing the cloud from being overloaded by data traffic that can easily be avoided by following the edge processing approach. This is important, because there are an increasing number of cases where device data cannot be handled via the cloud. Furthermore, there are some important advantages of edge processing[1]. Factory robots, cars, as well as audio processes, for example, need high-speed processing with minimal latency even though power availability is limited.

For example, imagine a self-driving car suffering from cloud latency while detecting objects on the road, or operating brakes or steering wheels. Any delay in data processing will result in a slower response from the vehicle. If the slowdown is such that the vehicle does not respond in time, this could result in an accident.

In audio processing, sound needs to be processed with minimal possible latency and low power to be able to run on headsets and in-ear devices.

In the specific case of satellite imagery, it is increasingly being used to predict natural disasters and to assist immediately after they occur and in dangerous situations such as strategic conflicts or illegal activities. Having the right response in the right place at the right time is key to the success of these activities. This is achieved by processing the imagery on board the same satellites that generate it, thus downloading only useful information or insights to the actor that requires it.

However, for each application use-case, the requirements might be very different as well as the consequences of not adhering to those. Therefore, any metric or benchmark must take different factors into account.

1.1 Definition of use-cases

The primary aim of this document is to identify and define practical application use-cases from

[1] The Pros and Cons of Edge Computing: https://www.datamation.com/edge-computing/pros-cons-edge-computing/

various domains that benefit from being executed on a secure edge processor with ultra-low power consumption and determine relevant requirements for the design of the edge processor. These use-cases come from four different companies, covering:

- Audio processing for in-ear hearing devices (GN Audio, Bosch)
- Audio processing for surround sensing of cars (Bosch)
- Satellite image processing (Thales Alenia Space España)
- Video processing for road-side traffic monitoring (ViNotion)

The audio processing use-case focuses on enhancing speech quality, suppressing noise and audio source tracking in vehicles, while the satellite image processing use-case aims to detect and track changes in recorded images in near real-time. Lastly, the video processing use-case involves video traffic analysis for tracking persons and vehicles in real-time.

For each use-case, specific requirements are outlined, and their relevance to CONVOLVE's broader goals are discussed. Moreover, the authors of each use-case have provided software code that is stored in a Git repository hosted by TUE. This repository can be used as a benchmarking tool and a basis for future development[2].

1.2 Update on requirements

This document presents the original use-cases with updated requirements. The updates compared to the initial requirements are briefly summarized for each use-case, so that it is clear which parts of the use-case have been changed and are updated.

[2] CONVOLVE Git repository: https://gitlab.tue.nl/es/convolve/

2. Overview on use-cases

2.1 Deep Noise Suppression / Speech Enhancement (GN Audio)

Owner	Other partners involved
GN Audio	N/A

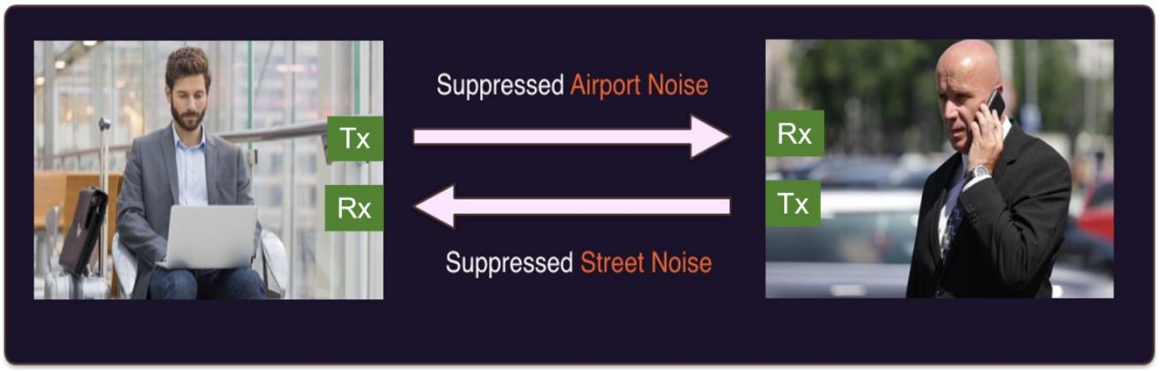

Figure 1: Visualization of the use case.

Deep Noise Suppression (DNS) or Speech Enhancement aims to improve the quality of both Tx (uplink) and Rx (downlink) speech signals by reducing background noise, thereby improving their quality and intelligibility (see Figure 1). This is a very challenging task, especially due to the vast amount of complex acoustic situations that may arise in the real world, such as the presence of an undesired speaker (referred to as "jammer") close to the main user's microphone. Thus, Speech Enhancement can be seen as an umbrella term for more specific (edge-) denoising tasks.

Utilization of neural network models

In D1.1, three models were originally taken into consideration for use as example neural networks in the Convolve program.
During the project, one single model called the NsNet-type model evolved as the focus.
It uses a combination of different types of layers and activation operations i.e., Gated Recurrent Unit (GRU) and pooling operations as well as ReLU, Leaky ReLU and Sigmoidal activations. In the following we will provide a short overview of the main characteristics of those models.

1. Name: **NsNet2[3] (Main Focus)**
    1.1. Inputs: Short-term Mel Spectrogram, Short-term Fourier Transform
    1.2. Layers: Fully connected, Gated Recurrent Units (GRU)
    1.3. Activations: ReLU, Sigmoidal
    1.4. Skip-connections: No

Statement of needs

Much better (high-quality speech) and power efficient noise suppression running on the edge as part of a Tx processing pipeline before transmission of signal.

[3] Data augmentation and loss normalization for deep noise suppression (NsNet2): http://arxiv.org/abs/2008.06412

CONVOLVE objectives & involvement

Objectives

State-of-the-art high-fidelity audio use case running on the edge devices with unprecedented power efficiency (Objective 1). Efficiency can and should be achieved by dynamic behaviour of the models, both on the model level as well as on compiler level (Objective 2). Any updates to model structure, weights or other firmware should be performed in a secure and encrypted way without the risk of side-channel attacks (Objective 3).

WPs involvement

WP4, WP5, (WP2, WP6)

Quantified baseline at the start of the project

Here, we will state concrete numbers for the NsNet2 model that have been discussed in literature and come very close to the some of the architectures discussed for Convolve purposes as a baseline.

The KPI's used here are listed below. They are either based on computations from torchinfo or taken from literature[4]. We will give requirements on

- Multiply-accumulate operations per seconds
- Memory
- Memory bandwidth
- Power
- Power / Mmacs (Million MACs)
- Latency

Model	# Parameters	MACs / s float16 x float16 MACs / cycle float16 x float16	Memory (Parameters) float16 Memory (Parameters) Int8	Memory BW float16 Memory BW Int8	Power @ 0.8 V	Power / MMacs	Latency
NsNet2 type on described target	0.99 M	43 M 2 (from paper)	3.95 MB 0.98 MB	0.55 MB / sec 0.14 MB / sec	66 mW	~ 1.5 mW	1.7 msec

This table should be considered as the baseline on which to measure any new model deployment. Note, float16 numbers are taken from the referenced paper while int8 numbers are estimated based on basic arithmetic calculations.

The audio examples used for this assessment use the following DSP parameters:

4 Accelerating RNN-based Speech Enhancement on a Multi-Core MCU with Mixed FP16-INT8 Post-Training Quantization: http://arxiv.org/abs/2210.07692

- Block frame-size: 400 samples (@257 one-sided FFT-bins)
- Hop-size: 100 samples
- Sampling rate: 16 kHz

Updated KPI's for the end of the project

The following shows similar numbers as for the baseline model, but for larger and more effective models that could ideally be deployed to a NPU, including operations needed per cycle like the previous table but without exact power figures, since these models have not been deployed. However, the numbers are chosen so, as to constitute a real improvement.

Model	# Parameters	MACs / s	Memory (Parameters) Float32	Memory BW Float32
		MACs / cycle (min. required)	Memory (Parameters) Int8	Memory BW Int8
NsNet2	3.6 M	693 M	14.3 MB	5.5 MB/s
		> 7 @ 100MHz	3.5 MB	1.4 MB/s

The audio examples used for this assessment use the following DSP parameters:

- FFT-frame size: 128 samples
- Frame-hop size: 64 samples
- Sampling rate: 16 kHz

Note that these numbers do change when other DSP settings are used. Generally, one trades latency for computational requirements, smaller FFT-sizes decrease latency, but increase compute requirements.

Metric	Unit	Value
Performance	I/O latency RTF	< 1 msec < 0.8
Power Consumption	mW / MMACs Full-cycle load	< 0.1 < 50 mW
Quality	VISQOL5 Mean-opinion score (MOS)	> 4

Besides these very concrete model-related metrics we want any future SoC to also fulfill the following requirements. Latency is defined as the end-to-end latency – from sound input to sound output on any defined SoC or SoM. It is important since larger audio latencies yield undesired echo and sound coloration effects.

Real-time Factor (RTF) is defined as the ratio between the time for a model to process a frame of audio data compared to the iteration of frames through the signal – the so-called frame-hop size.

Key elements involved

Hardware elements

- Sufficient memory close to the processor(s) as indicated by metrics requirements.
- High memory bandwidth to and from processor(s)
- Parallel processing pipeline that consists of multiple processors or accelerators.
- Variable clock rate depending on NN load

Software elements

- All PyTorch convolutional layers (nice-to-have)
- All PyTorch recurrent layers (nice-to-have)
- Most spread activation functions (need-to-have)
- Profiler for memory management (nice-to-have)
- Profiler for processing management (nice-to-have)
- CUDA support (nice-to-have)
- cuDNN support (nice-to-have)

Low-level algorithmic elements

As mentioned in parantheses, the described software elements can be regarded as a rough outline to support the current audio-related use-cases.
However, most of the points constitute nice-to-haves' and not a need-to-haves' for the following argument - the operations necessary for e.g. Recurrent Layers can be even split up further to more low-level operations[5]. In this sense, what is pivotal to compute outputs of Recurrent as well as Dense Layers are mainly these components:

- Hadamard Matrix Product (Element-wise matrix multiplication)[6]
- Normal Matrix Product, General Matrix-Multiplication (GEMM)[7]

Furthermore, the Hadamard product can even be represented as Normal Matrix Multiplication. This means that in the end the *Normal Matrix Product* is the only minimal-viable operation that needs to be supported for the audio use-cases – in case of resource or time constraints, it thus would constitute the most relevant focus of any effort for optimization towards a neural-network accelerator.

Interactions

The HW elements should be optimized to maximally exploit AI SW elements and be optimized for a) high-speed data processing bandwidth, b) efficient memory access (maybe compute in memory), and c) parallel computing.
Additionally, HW elements should also be optimized for power efficiency to minimize energy consumption and reduce costs, especially for larger-scale AI applications. Finally, HW elements should be designed to support the specific requirements of the AI applications, such as image recognition, speech enhancement, or autonomous driving, to provide the necessary performance and accuracy for the tasks at hand.

[5] GRU layer PyTorch: https://pytorch.org/docs/stable/generated/torch.nn.GRU.html
[6] Hadamard matrix product: https://en.wikipedia.org/wiki/Hadamard_product_(matrices)
[7] Matrix multiplication (GEMM): https://en.wikipedia.org/wiki/Matrix_multiplication

Security requirements

In general, GNA is not overly concerned about overall privacy or safety issues from the user's perspective, since no data is stored on the device, and data transmission commonly occurs within inherently insecure channels (air medium) or channels where security is ensured by the underlying transmission protocol (i.e., Bluetooth or other RF protocols). However, there are two main lines of security aspects that GNA is interested in:

1. Protection of intellectual property in the form of neural network models
2. Secure update of firmware and neural network models to the edge device

The protection of intellectual property refers to the protection of the specific neural network architecture and specialized data from copying or inferring, which can be expensive to acquire or develop. This can and has been threatened by e.g., using reverse engineering techniques.

Secure firmware updates refer to the process of updating the software that controls the hardware components of an (edge) device, such as a smartphone, IoT device or headset in a secure and trusted manner. This is important to ensure that the device remains secure and up to date, as vulnerabilities or bugs in the firmware could potentially be exploited by attackers to gain unauthorized access or cause damage to the (edge) device.

More details on the approach towards those two goals can be found in D1.1.

Updates with respect to D1.1

From a GNA perspective, we

- updated the KPIs (*power requirement, required MMACs, latency, …)* to reflect a true but realistic improvement of those
- focus workload and resource on one single use-case model to make workflow more effective across any work package
- refine the type of operation that should be the focus for any acceleration (BEMM)

To confirm or reject the defined KPIs for this concrete use-case the following tools will be applied:

- ZigZag
- Stream
- GVSOC

and other neural accelerator emulator tools. Details on these emulator tools can be found in D6.1[8]

[8] CONVOLVE_D6.1_Modular_Architecture_Template_Definition_v1.1.pdf

2.2 Speech Quality Prediction (GN Audio)

Owner	Other partners involved
GN Audio	N/A

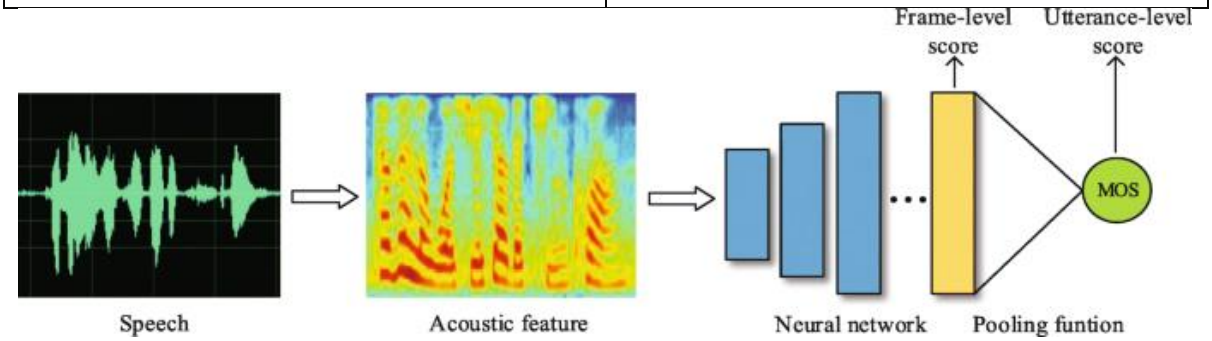

Figure 2: Visualization of the use case.

The field of speech quality prediction can be divided into full-reference (also known as intrusive), which requires a clean reference signal to compare against, and reference-less (also termed non-intrusive), which operates on the given signal only.

While there exist several full-reference metrics based on DSP or perceptual models (PESQ[9], POLQA[10], ViSQOL[11], etc) that correlate nicely with a human-attributed MOS, these are often computationally expensive, and it might be beneficial to "approximate" them using an optimized ANN-based implementation that can run on an accelerator (an example ANN for speech quality prediction is depicted in Figure 2). However, due to the lack of reference signals in real-world scenarios, most of the focus will be on reference-less methods.

Utilization of neural network models

Some of the main unintrusive speech quality estimator models in the literature as DNSMOS[12], NISQA[13], and QualityNet[14].

During the project, the focus mainly turned to DNSMOS to make efficient use of resources. The following operations are present in the DNSMOS network architecture:

- 2D Convolutions (kernel size 3x3)
- MaxPool layers
- Fully connected layers
- ReLu activations

[9] Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs: https://www.itu.int/rec/T-REC-P.862

[10] Perceptual objective listening quality prediction: https://www.itu.int/rec/T-REC-P.863

[11] ViSQOL v3: An Open Source Production Ready Objective Speech and Audio Metric: https://arxiv.org/abs/2004.09584

[12] DNSMOS: A Non-Intrusive Perceptual Objective Speech Quality metric to evaluate Noise Suppressors: https://arxiv.org/abs/2010.15258

[13] NISQA: A Deep CNN-Self-Attention Model for Multidimensional Speech Quality Prediction with Crowdsourced Datasets: https://arxiv.org/abs/2104.09494

[14] Quality-Net: An End-to-End Non-intrusive Speech Quality Assessment Model based on BLSTM: https://arxiv.org/abs/1808.05344

Statement of needs

Constant and seamless monitoring of the speech quality serves as the feedback metric for any speech enhancement system to react to sudden changes in enhancement processing.

CONVOLVE objectives & involvement

Objectives

Running a speech quality predictor constantly in the background requires the model to be highly power efficient (Objective 1 and optimally dynamic Objective 2), changing processing dependent on the environment. Any updates to model structure or weights should be performed in a secure and encrypted way without the risk of side-channel attacks (Objective 3).

WPs involvement

WP4, WP5, WP6

Quantified baseline at the start of the project

As for noise-suppression we mainly focus on following metrics for characterizing network performance:

- Multiply-accumulates per second
- Memory
- Memory bandwidth
- Power
- Power / MMacs (Million MACs)
- Latency

Here, this info comes again mainly from torchinfo[15]. It was assumed that model inference was performed on the same chipset as for the noise-suppression examples before. This yielded a 1.5 mW consumption per million macs.

Note, for DNSMOS, the memory bandwidth is significantly higher than for the other models, mainly due to the large bandwidth requirements for CNNs.

Model	# Parameters	MACs / s float16 x float16	Memory (Parameters) float16	Memory BW float16	Power @ 0.8 V	Power / MMacs	Latency
		MACs / cycle float16 x float16	Memory (Parameters) Int8	Memory BW Int8			
DNSMOS	33 K	2.47 M	0.13 MB	17 MB	3.7 mW	1.5 mW	< 3ms
		0.04 @ 100 MHz	0.04 MB	4.3 MB			
Quality Net	120 K	5.3 M	0.45 MB	5 MB	7.6 mW	1.5 mW	< 3ms
			0.11 MB	1.4 MB			

[15] TorchInfo: https://pypi.org/project/torchinfo/

Updated KPI's for the end of the project

Effective non-intrusive speech quality prediction running seamlessly and very power efficiently on the edge for constant monitoring. Power requirements should be decreased by at least a factor of 2 relative to the numbers in the baseline table above.

Key elements involved

Hardware elements

- Sufficient memory close to the processor(s) as indicated by metrics requirements.
- High memory bandwidth to and from processor(s)
- Parallel processing pipeline that consists of multiple processors.
- Variable clock rate depending on NN load.

Software elements

- All PyTorch convolutional layers need to be supported.
- All PyTorch recurrent layers need to be supported.
- PyTorch Dense layers need to be supported.
- All current PyTorch activation functions need to be supported.
- Profiler for memory management
- Profiler for processing management
- Emulator for simulating SoC "off-line"
- CUDA support
- cuDNN support

Interactions

The HW elements should be optimized to maximally exploit AI SW elements and be optimized for a) high-speed data processing, b) efficient memory access (compute in memory), and c) parallel computing.

Additionally, HW elements should also be optimized for power efficiency to minimize energy consumption and reduce costs, especially for larger-scale AI applications. Finally, HW elements should be designed to support the specific requirements of the AI application, such as image recognition, speech enhancement, or autonomous driving, to provide the necessary performance and accuracy for the task at hand.

Security requirements

Here, the requirements are identical to those that were formulated for the *Speech Enhancement* use-case (section 2.1).

Again, for this use-case, there are two main lines of security aspects GNA is interested in:

1. **Protection of intellectual property** in the form of neural network models
2. **Secure update of firmware and neural network models** to the edge device

For more details, please refer to Page 12 in this document or D1.1.

Updates with respect to D1.1

From a GNA perspective, we

- updated the KPIs (mainly *power requirements) to* still reflect a true but realistic improvement of those defined in the baseline
- focus workload and resource on one single use-case model (DNSMOS)

Again, the tools that should be applied to confirm the realism of the defined KPIs are:

- ZigZag
- Stream
- GVSOC

Details on the named tools can be found in D6.1[16]

[16] CONVOLVE_D6.1_Modular_Architecture_Template_Definition_v1.1.pdf

2.3 Acoustic Scene Analysis (Robert Bosch GmbH)

Owner	Other partners involved
Robert Bosch GmbH	N/A

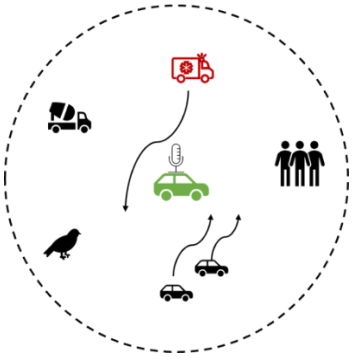

Figure 3: Illustration of a typical acoustic traffic scene. Here, information about emergency vehicles based on siren sounds should be extracted.

Bosch focuses on the detection and tracking of siren sounds in typical traffic scenarios (see Figure 3). For that purpose, two distinct neural network architectures are proposed to solve both tasks independently. Both are currently implemented on a Raspberry Pi that serves as a baseline for further benchmarking and guides the development for implementations on more special-purpose hardware considered in the Convolve project.

Utilization of neural network models

In the Convolve project, we aim to utilize recurrent and convolutional neural architectures to perform the detection of siren sounds and the tracking of the underlying sources. In the following, the models for each of the two tasks are highlighted. In summary, the following neural network components are involved:

1. Conv2d layers
2. MaxPool2d layers
3. GRU layers
4. Linear layers
5. Tanh, Sigmoid and ReLU activations

1. Siren detection

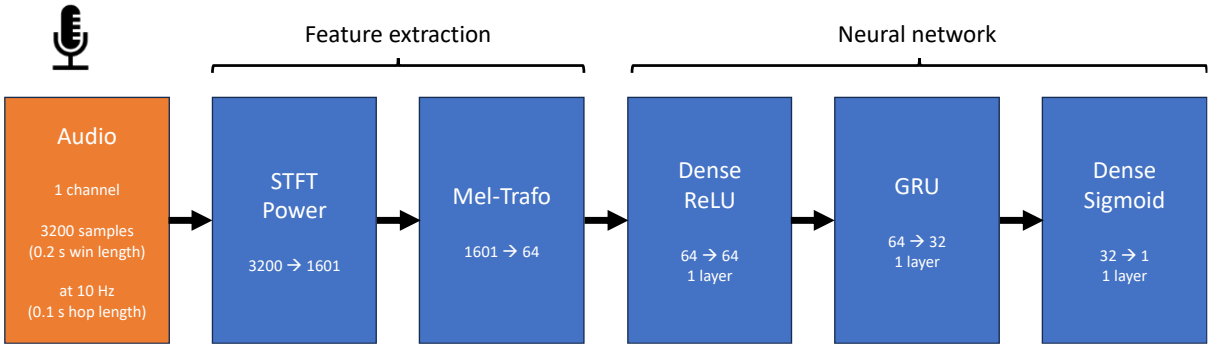

Figure 4: Processing chain and neural network architecture for the siren sound detection use case. Mel-spectrograms features are calculated based on a single channel audio recording. These features are then processed by a dense layer and the output thereof is sequentially processed in time by a GRU-layer. The detection result is indicated by dense layer with sigmoid activation function.

For the **detection** network, Mel-spectrogram features - extracted from a single-channel audio stream - are used as network input. The network comprises of a linear layer with ReLU activation function, followed by a GRU layer that processes the features for each window sequentially. A linear readout with sigmoid activation function is then used to output the prediction (Figure 4). To implement a modified spiking version of this network, a leaky integrate-and-fire layer is used instead of the GRU layer, and the readout is replaced by a leaky integrator layer.

2. Sound source tracking

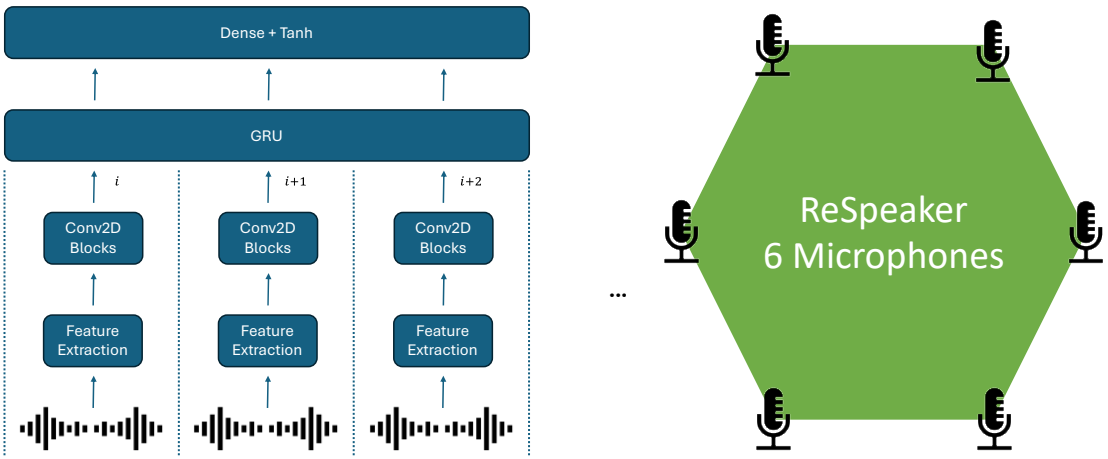

Figure 5: Data handling and neural network architecture for the siren sound tracking use case (left). A sequence of 2D features is extracted from the multi-channel input audio stream. These features are processed by a sequence of Conv2D blocks. For each window, the resulting feature maps of the last block are flattened and used as input to a GRU layer. On top of the GRU activations, a linear readout with Tanh activation function is trained to output the tracking information. The microphone geometry considered is chosen to coincide with the ReSpeaker 6-Mic setup (right).

The architecture of the **tracking model** has been slightly changed to enhance robustness and to foster more efficient implementations on the available accelerators (Figure 5). First, the

microphone geometry has been fixed based on the point-demo implementation on the RaspberryPi. Hence, the ReSpeaker 6-mic array geometry is used for benchmarking. Based on the recorded multi-channel audio signals, two-dimensional feature maps are extracted by employing a sliding window approach. Next, these features are processed by a sequence of Conv2d-blocks. The feature maps of the last block serve as input to a GRU-layer that sequentially processes the generated feature maps in time, i.e. across windows. A linear readout with Tanh activation function is trained to output the predictions.

Statement of needs

As outlined in deliverable D1.1, the consideration of acoustic features has the potential to augment and enrich the data of other sensor modalities, especially for autonomous driving.

CONVOLVE objectives & involvement

Objectives

The objectives remain the same. In the following, we provide a summary of D1.1:

1. Energy efficient solutions are required since the total power budget available within a car is limited.
2. A rapid adaptation to changing requirements is desired.
3. The information obtained is safety critical and demands security against attacks and reliable hardware.
4. Usually, siren sound events occur sparsely in time, which is why smart adaptation mechanisms are relevant to enhance efficiency.

WPs involvement

WP2, WP4 and WP5 (WP3, WP6).

Quantified baseline at the start of the project

For the point demos, both proposed neural architectures are implemented on the Raspberry Pi hardware. As outlined above, the ReSpeaker 6-Mic array has been considered, due to its availability, interfacing as well as visualization options. In more detail, a single microphone has been utilized for the detection and all six microphones for the tracking use case. The network's prediction is visualized with the 12 LEDs on the microphone array.

This demo in combination with software simulations on conventional hardware allowed us to identify bottlenecks of the initial implementation and in turn facilitated the development of the two neural architectures as described above. The main bottlenecks were the following:

1. Large recurrent hidden layers: For both sub-parts of the use case the hidden layer sizes have been reduced by inserting suitable encoder blocks that in turn facilitated a reduction of the hidden layer sizes.
2. Feature extraction methods: The feature dimensionality directly impacts the size of the input layer of each network. Especially for the tracking use case, the multi-microphone setup could translate to a high dimensionality of the input feature space. In turn, efficient extraction methods have been used to reduce the overall size of the early layers of each network.

The table below summarizes key model characteristics as well as the first benchmark results obtained on a Raspberry Pi 4B with 8GB RAM.

Model	#MACs (M)[17]	#Params (k)	Model Size (MB)[18]	Metrics			
				Performance	Throughput (Frames/s)	Average Latency (ms)[19]	System Power (W)[20]
Detector	0,01	13,6	0,05	Acc > 0,95	~10	~ 6	10
Tracker	62,21	279,9	1,12	Avg err < 30°	~10	~ 41	

The values presented in the table correspond to a model with FP32 precision. The number of MACs and parameters as well as the corresponding model sizes have been obtained from torchinfo[21]. The update interval of 100 ms is based on system requirements and is determined by the algorithm. The throughput and average latency have been measured on a Raspberry Pi 4B.

Compression techniques have been applied in software simulations that have the potential to further reduce the computational overhead of any hardware implementation. In addition, sparsity has been considered by implementing equivalent spiking neural networks and the binarization of activation functions. While the 8-bit integer quantization is assumed to be relevant for most of the implementations, sparsity could further boost the performance of our networks on more specialized hardware.

Goals at the end of the project

The main goal at the end of the project would be to reduce power consumption and to demonstrate the detection and tracking of siren sounds with the resulting solution in real-time. Here, the above-mentioned algorithmic changes could contribute to an efficient solution.

Key elements involved

The key elements involved basically remain the same. Due to the focus on a specific microphone array, the maximum number of audio channels is now determined by the six available microphones. Apart from that, the processing chain stays the same (see Figure 5).

The following list of elements as well as the interactions are mainly taken from D1.1 and augmented with the additional layer types required for the siren sound tracking network.

Hardware elements

- Parallel processing
- Mixed precision
- Adjustable clock frequency
- Efficient always-on detection

[17] Number of MACs of neural network for a single prediction.
[18] For parameters in FP32 precision.
[19] Average latency values include the feature extraction and neural network calculations for a single prediction.
[20] No detailed measurement available for the power consumption. The provided value constitutes an estimate for the full system.
[21] Yep, T. (2020). torchinfo [Computer software]. https://github.com/TylerYep/torchinfo

- Tolerance regarding harsh environments (radiation, vibration, temperature, ...)

Software elements

- Signal processing (FFT, iFFT, MelSpectrogram, AmplitudeToDB)
- Recurrent layers (GRU/LSTM)
- Linear layers
- Activation functions (Sigmoid, Tanh, ReLU)
- Conv2D layers
- MaxPool2d layers

Interactions

Figure 6 depicts the processing chain presented in D1.1. The audio signals are recorded with a microphone array and features are extracted by drawing on signal processing techniques. These features are in turn processed by the neural networks and the prediction thereof is communicated.

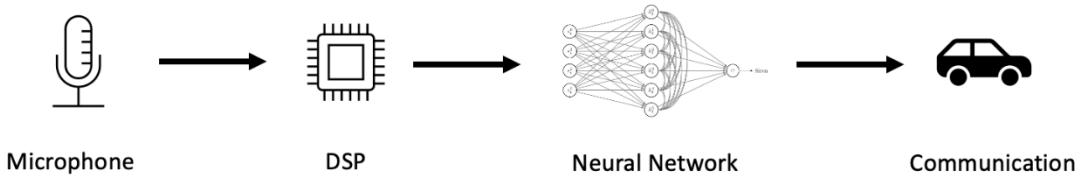

Figure 6: Processing chain from recorded auto to communication of the predictions.

Security requirements

The security requirements have been described in deliverable D3.1. In the following, we provide a short summary. The availability of detailed knowledge about emergency vehicles in the vicinity around a car constitutes safety-critical information. Because of this, not only the model and its parameters have to be secured, but also the communication between all components must be considered in terms of privacy and attacks (see Figure 6). The latter could involve physical access to the system like the blocking of microphones, the removal of hardware components or even a manipulation of the communication which could in turn lead to wrong actions with detrimental outcome. Lastly, potential model updates after the initial deployment must be considered.

Updates with respect to D1.1

In summary,

1. the network architectures have been refined to alleviate bottlenecks of the initially suggested neural architectures and the requirements have been updated accordingly.
2. The compression targets and sparsity constraints have been adjusted to draw on the benefits of the newly proposed networks.

2.4 On-board Computer Vision (Thales Alenia Space España)

Owner	Other partners involved
Thales Alenia Space España	TUE, UIR

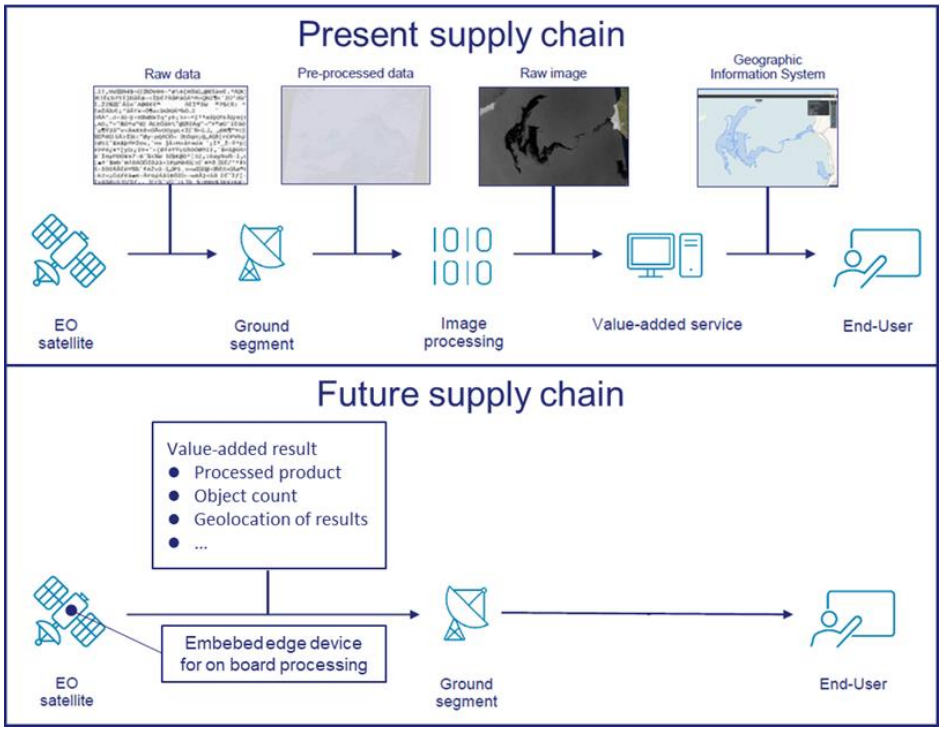

Figure 7: Visualization of the use case.

The images generated by Earth Observation (EO) satellites are traditionally downlinked to ground for processing and analysis (see top part of Figure 7). This causes congestion of the communications channel and represents a security breach for certain confidential images. The downloading of the images can only take place at specific moments of the orbit, then slowing down decision making, this delay is even increased sometimes since the analyses are performed only by highly experienced experts who are not the decision makers.

Embedded systems have strong restrictions on the availability of resources and, therefore, condition the depth of the analysis that can be carried out in-place of the captured scenes.

Utilization of neural network models

In D1.1, two computer vision tasks were taken into consideration for the definition of the use case to be addressed in the CONVOLVE project, object detection and semantic segmentation of satellite images. After an initial assessment, the problem of performing a semantic segmentation on specific hyperspectral imagery was selected as the base use case as it present multiple steps that can be modified to improve the performance and reduce the energy consumption and represents an unsolved problem for onboard processing (see bottom part of Figure 7).
Additionally, TAS has ongoing mission called IMAGIN-e that can potentially hosts an experimental model for onboard processing of hyperspectral captures of the Earth surface during the period 2024/2025 and thus, could bring the opportunity to perform a live validation of the improvements on real flight hardware and on a real space mission.

In this base use case, after exploring different architectures, the model DeeplabV3+ with the following particularities was selected as the baseline model:

- The input layer processes hyperspectral imagery of 202 bands (channels), obtained from the EnMap mission optic instrument.
- Three datasets for lands usage and land cover are provided with the same optical imagery but different segmentation masks obtained from different missions (already co-registered).
- A ResNext50 is used as backbone for the feature extraction process.
- A segmentation map with one channel per segmentation class is generated as the output of the model

Statement of needs

The present supply chain:

- Downloads all imagery (raw data) taken without discriminating whether it is potentially useful.
- Involves latencies of hours from the time the raw data is obtained at the satellite sensor until the end user is aware of the valuable data.
- Requires specific functional blocks designed specifically for each mission.
- Potential security vulnerabilities when downloading raw data from imagery that could be compromised or modified (defense. cadasters, asset tracking,).

To build the future supply chain, it is necessary to:

- Download only the information useful to the end user.
- Make valuable information available to the end user in a few minutes.
- Have a generic architecture independent of the mission.
- If only the result (the inference) is downloaded, it is more difficult to detect its usefulness. If an enhanced image (value-added product) is downloaded, the security gap remains the same and would need to be improved.
- Added to this is the need to ensure the integrity of the processing SW installed on board and to prevent potential malware from being uploaded from the ground.

CONVOLVE objectives & involvement

Objectives

In an environment as hostile as space and where information must inevitably travel through an air channel to ground, the security and reliability of the on-board systems is crucially important. The integrity of the SW being deployed on-board shall be always preserved, if needed including dedicated mechanisms to cover this (Objective 3).

In addition, the power available on a satellite is limited and managed to keep all systems alive throughout the life of the mission as they age. In addition, the size of the images to be processed will make it inevitable to have several devices working in parallel, so it is essential that their power consumption be kept to a minimum (Objective 1).

Finally, new trends and players in the space market are setting much shorter design and development times, which must be matched or improved to stay in the competition (Objective 2).

WPs involvement

WP2, WP3, WP4 & WP5

Quantified baseline at the start of the project

As mentioned before, the model DeepLabV3+ with a ResNext50 encoder was selected as the baseline model for processing images of 128x128px with 202 bands/channels.

The table below shows the initial KPIs for the baseline model:

Model	Model Size (MB)	# Trainable Params	# FLOPS	Inference Time (s)	Mean Power (W)
DeepLabV3+ (Resnext50 encoder)	218.33MB	26.7M	4.8B	1.49	22.44

These KPIs were measured using a trained model that is running in a docker container on a flight computer with the following characteristics:

- ARM Cortex-A72 CPU with 16 cores running at 2GHz
- 32GB of DDR4 SDRAM
- No graphics hardware acceleration

Goals at the end of the project

- Design capable of supporting different neural networks more than 50 layers deep.
- Power consumption objective:
  - < 0,1 mW/MMACs
  - Maximum dissipation compatible with a flight design < 20 W (per device, SoC).
  - At board level, maximum power dissipation < 50 W (with as many SoCs working in parallel as possible).

Key elements involved

Hardware elements

- Enough memory close to processor(s).
- High memory bandwidth to and from processor(s)
- Parallel processing pipeline that consists of multiple processors.
- Variable clock rate depending on NN load
- Radiation-tolerant components
- High junction temperature range (-50ºC, 150ºC)

Some specific algorithms of the use case have been tested on the following HW platforms:

- Satellite space grade Xilinx Kintex Ultrascale XCKU115 2:
  - Up to 5,520 DSPs and 75.9 Mb of embedded RAM
  - Alpha Data ADM PCIE 8K5 (2 DDR4 2400MT/s)
- Versal AI Core Series VCK190

Software elements

- All convolutional layers need to be supported.

- All recurrent layers need to be supported.
- Dense layers need to be supported.
- All current activation functions need to be supported.
- No restriction on ML library but PyTorch is preferred
- Full tool ecosystem is provided for easy deployment on SoC.
- Emulator for simulating SoC "off-line"
- CUDA support
- cuDNN support

Interactions

The interactions between the different stages for analysis of a satellite image are depicted in Figure 8.

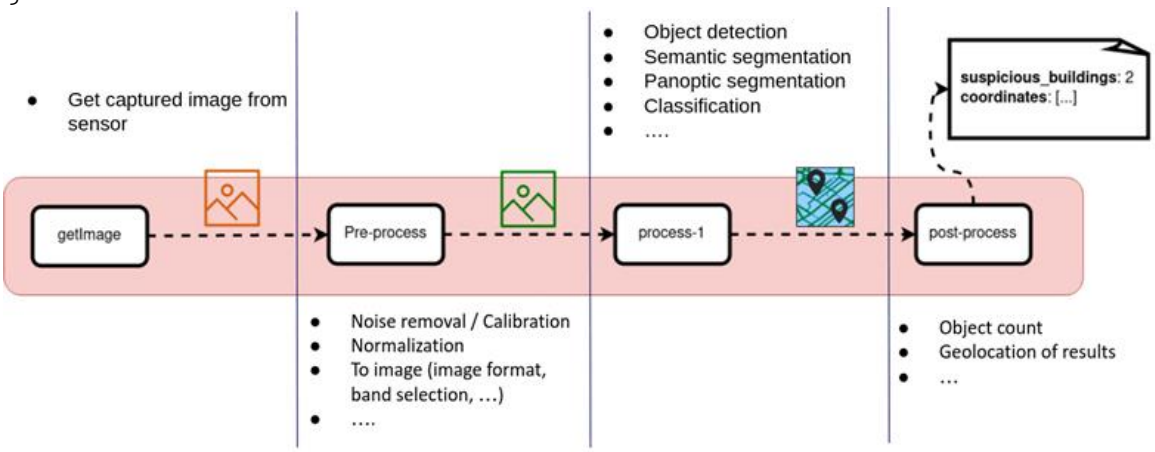

Figure 8: Visualization of the interactions between different processes

Security requirements

The usage of robust encryption protocols is paramount because satellite systems are susceptible to various cyber threats, including signal jamming, spoofing, and data interception which can lead to the unauthorized disclosure of sensitive information. Ensuring the integrity and authenticity of the data transmitted from satellites is crucial to prevent malicious actors from injecting false information or disrupting communication channels.

The prevention against model reverse-engineering is crucial to avoid the extraction of sensitive information. Weight extraction attacks can reveal the internal parameters of a model, which can then be used to replicate the model, steal intellectual property, or uncover training data that might include private or proprietary information.

Updates with respect to D1.1

The specific computer vision task, the type of sensor used, and the baseline model with its KPIs, and the security requirements have been updated to reflect the current research direction defined throughout the project from its inception to the present.

2.5 Video-based traffic analysis (ViNotion)

Owner	Other partners involved
ViNotion	TUE

Figure 9: Visualization of the use case.

ViSense is an edge-based video analysis system that reads, analyzes and processes real-time video from a standard CCTV/RGB camera (24/7 measurements) for surveillance, traffic management, incident detection, crowd management and various other traffic cases. By utilizing artificial intelligence (AI) software, ViSense makes it possible to register movements of all objects including pedestrians, bicycles and vehicles. This technology provides a large amount of information about the objects with high accuracy up to 98% in streets of 20 m wide and squares up to 500 m2 with a single camera sensor. For example, ViSense can provide insight into densities, paths travelled from objects (trajectories), heatmaps, row length, statistics of near incidents, etc. Figure 9 depicts an example image of a scene containing a busy crossing where all traffic participants are being detected and followed. The arrows indicate the counting directions that are of interest and on the right-hand side various high-level metrics about red light, speeding and one-way violations are displayed.

Utilization of neural network models

Convolutional Neural Networks (CNN) are used for performing visual object detection and classification. Objects are classified into one or multiple categories and their locations in the image are estimated by 2D bounding boxes.

The neural network model utilized in the ViSense application is one of the key components in the system because the performance of the analysis system depends on reliable object detection and classification. Furthermore, the computational complexity of the neural network is significant with respect to other parts of the analysis software. Therefore, the focus is on acceleration and optimization of the object detection and classification network.

Typically, single stage object detection models are computationally more efficient and better suited for embedded hardware. These models mostly use operations that are widely available in

all frameworks. Well-known object detection models are the SSD[22] and YOLO architectures[23]. In this project, the focus is on optimization and porting of the YOLOv6 model[24].

More specifically, the YOLOv6s6 configuration is utilized for object detection.
The YOLOv6s6 network takes an image as input and passes it through a series of convolutional layers in the backbone. The resulting features are then further refined in a neck and the final 2D bounding-box detections and classifications are generated by three detection heads. The operations used in the YOLOv6s6 network are as follows:

- 2D convolutions (3x3 and 1x1 kernels)
- Transpose, concatenate and reshape operations
- Max pooling
- Up sampling convolutions
- ReLu and Sigmoid activations
- Non-maxima suppression for 2D bounding boxes

Statement of needs

A ViSense system can be deployed for large scale systems like highways and train stations, comprising more than 1000 cameras. Using AI for automatic interpretation, requires significant computational power and should be performed on the edge for several reasons: 1) it provides anonymization near the sensor and protects privacy; 2) it does not form a computational bottleneck in the cloud where the sensor data is used for traffic control or crowd control; 3) it preserves communication bandwidth; 4) It reduces a single-point-for-failure due to the distributed nature of the processing.

CONVOLVE objectives & involvement

Objectives

Energy efficiency (Objective 1) is an important aspect for enabling smart edge processing applications (Objective 4). Compact, cost-efficient and advanced visual interpretation requires a CONVOLVE approach. Efficient and effective development is important for fast integration in custom applications (Objective 2). For privacy protection and secure traffic control, also Objective 3 (security and reliability mechanisms) is important.

WPs involvement

WP3 and WP4 are important to translate the user requirements to system requirements. As the development of AI technologies and their exploitation are advancing rapidly, it is important to create systems with a high amount of flexibility for the programmer and easy to use tools

[22] SSD: Single Shot MultiBox Detector: https://arxiv.org/abs/1512.02325
[23] You Only Look Once: Unified, Real-Time Object Detection: https://arxiv.org/abs/1506.02640;
YOLO9000: Better, Faster, Stronger: https://arxiv.org/abs/1612.08242;
YOLOv3: An Incremental Improvement: https://arxiv.org/abs/1804.02767;
YOLOv4: Optimal Speed and Accuracy of Object Detection: https://arxiv.org/abs/2004.10934;
YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications: https://arxiv.org/abs/2209.02976;
YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors: https://arxiv.org/abs/2207.02696
[24] YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications: https://arxiv.org/abs/2209.02976 ;
YOLOv6 v3.0: A Full-Scale Reloading: https://arxiv.org/abs/2301.05586

allowing high-level programming languages to be mapped to the underlying hardware while abstracting from the complexity.

Quantified baseline at the start of the project

We have a product exploiting ANN on a Nvidia Jetson family of platforms. The firmware contains a pipeline of processing functions and needs an integral approach for efficiency improvements. The building blocks consist of video decoding, ANN for object detection and classification, tracking, color conversion, scaling, image stabilization, object blurring, a webserver for webservices and dashboarding, etc.

The initial product runs 1 full HD video stream with 2 x 512x512 ANN template at 4 fps including tracking at 30 fps with more than 100 objects on a TX2 at typical 15 W power on a TX2 system. The product is ported to a Xavier NX system and capable of performing video analysis on 6 video channels (full HD) in parallel. The ANN for object detection and classification is performed for all channels simultaneously on the GPU and is computationally most expensive. The detection model used is the YOLOv6s6 architecture with an input resolution of 1280x768 pixels. Video decoding is currently offloaded to dedicated on-board hardware.

For the object detection model, the following benchmarks have been made as a baseline. The inference time of the YOLOv6s6 model has been measured with full precision (fp32) and when quantized to half precision (fp16) and integer precision (int8). The accuracy numbers are not reported, since we found that the quantization has negligible impact on detection and classification performance. Inference time denotes the average inference time over multiple passes for a batch size of 1.

System	Precision	Inference time / latency (ms)
Xavier NX (Jetpack 5.0)	fp32	102
	fp16	29
	int8	15

The following table provides a more detailed overview of the inference performance of the YOLOv6s6 model (FP32) on the Xavier NX hardware platform measured with TorchInfo.

Model	# Parameters	MACs	Memory (Parameters) Float32	Mean Power
YOLOv6s6	94.6 M	121 G	296 MB	15 Watts

No detailed profiling is available for power consumption for the baseline model. However, global system measurements on the NX hardware indicate an average power consumption of 15 Watts during processing a video stream of 25 frames per second and performing detection (inference of YOLOv6s6) at 5 frames per second. During inference of the model, peeks in power consumption of up till 45 Watts have been measured.

Goals at the end of the project

Considering trends of AI development and the continues increase of computational power per Watt, conform Moore's law, the future of Ultra-Low Power (ULP) AI processing and deep learning is inevitable. We want significant power-efficiency improvement to reduce the costs of the power supply and a passive thermal design and allow a week of operation on a 0,5 kg battery

operated system. Such a low-power design would also enable in-camera AI processing, since the power budget of Power-over-Ethernet (PoE) cameras is limited.

Key elements involved

Hardware elements

HW elements for the TX2 system:

- 256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
- Dual-Core NVIDIA Denver 2 64-Bit CPU
- Quad-Core ARM® Cortex®-A57 MPCore
- 8GB 128-bit LPDDR4 Memory 1866 MHx - 59.7 GB/s
- 32GB eMMC 5.1
- Accelerators for H264 encoding and decoding
- Total power consumption of ~20 Watts

HW elements for the Xavier NX system:

- 384-core NVIDIA Volta™ architecture GPU with 48 Tensor Cores
- 6-core NVIDIA Carmel Arm®v8.2 64-bit CPU
- 8GB 128-bit LPDDR4x 59.7GB/s
- 16GB eMMC 5.1
- Accelerators for H264 encoding and decoding
- Total power consumption of ~20 Watts

Software elements

- Gstreamer for video input reading
- Proprietary component-based framework (C++)
- Jetpack 4.6, CUDA/cuDNN, TensorRT for ANN deployment, OpenCv, etc.
- Typical ANN architectures that are used are: SSD and YOLO (v4/v6/v7)

Interactions

All interactions between the different software components are performed on a single hardware platform. Figure 10 depicts the main software components that are present in the ViSense application, the arrows illustrate the interactions with each other.

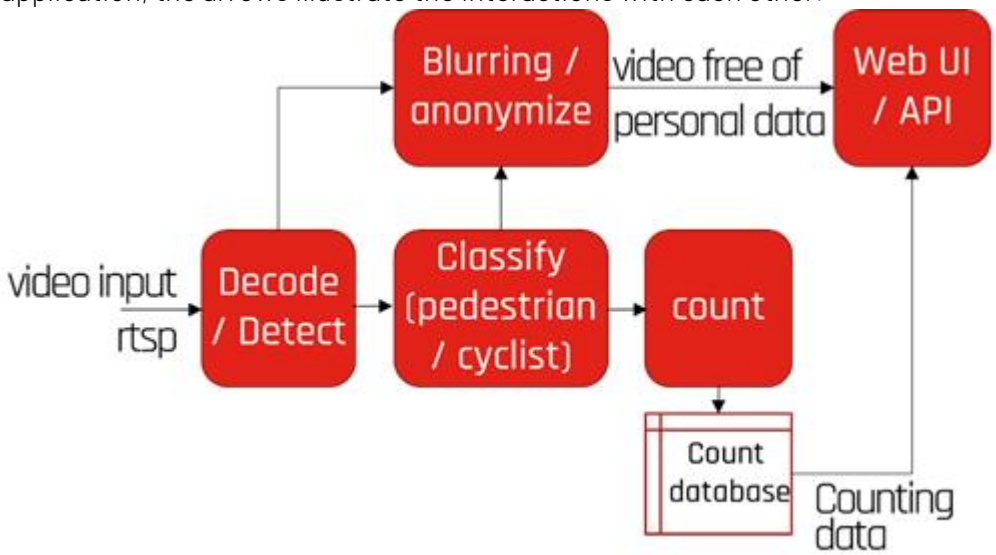

Figure 10: Visualization of the interactions between different software components on the hardware.

Security requirements

The use of camera-based traffic sensors imposes a privacy risk. Following the GDRP regulations, video from a public space might contain the appearance of people and license plates of vehicles and is defined as personal data. To prevent this personal data from being leaked from the system, the privacy-by-design paradigm of prof. Jaap-Henk Hoepman[25] should be considered. Risks should be mitigated by applying the following design strategies: minimize information; hide information; separate data structures; aggregate personal information; inform data subjects; control the access and maintenance of data; and enforce regulations. These strategies are addressed as follows:

Minimize: Only metadata is stored such as counting results, object class, object speed and object trajectories. Video data that is received from the camera (tier) by the edge device is analysed and not stored.

Hide: Analysis data is stored on a local storage. When accessing the edge tier, the user must provide authentication (username, password) to obtain rights to retrieve the stored data or to access the web UI. Data without personal information is sent to clients using HTTPS or web socket connections over VPN.

Separate: The data mentioned in point 1 is stored in separate storages on the same device. For example, object trajectories are stored separately from their classifications, counting results are stored separately from the camera configuration. Databases containing object information are only linked together using unique identifiers (UUIDs) but require access to the different containers. These UUIDs are not traceable to individual persons.

Aggregate: Personal information in the form of video images containing natural persons is discarded directly after usage and any visual data about detected objects is discarded after analysis.

Inform: Since the video feed of the camera is not stored or used for surveillance, and no personal information is detected or stored, informing the public is not strictly required but with a sign near the camera system, the people are notified and ensured that their privacy is respected.

Control: Since no personal data is stored nor exposed outside the edge device, the right of access, rectification, erasure, and restriction does not apply.

Enforce: A privacy document describes the technical safeguards that the product uses to protect the privacy of the data subjects. Customers can use this information for a Data Protection Impact Assessment (DPIA) and should be legally enforced by a Data Protection Officer.

[25] Privacy Design Strategies (The Little Blue Book): https://www.cs.ru.nl/~jhh/publications/pds-booklet.pdf

Updates with respect to D1.1

The requirements have been updated and extended in the following ways.

- Utilization of neural network models: A brief discussion on ANN models for object detection has been added, including a motivation to focus on the object detection and classification model for optimization because of the significant computational load with respect to the other components in a video analysis system for traffic surveillance.
- Quantified baseline at the start of the project: Besides the TX2-based system for analysis of a single video channel, an additional system with Xavier NX hardware is described for multi-channel video processing (6 channels). This updated system is more recent and has more computational power while having comparable power consumption as the TX2-based system. Additionally, a benchmark of the object detection model on the NX hardware has been added as a baseline.
- Security requirements: An additional note with respect to the security requirements is made with respect to privacy risks. Security is an important aspect for designing privacy related systems.

3. Summary & Conclusion

This document described the updated requirements of the use cases of the applications that are considered in the CONVOLVE project. In total there are five application use-cases. Three applications aim at audio processing for localizing, filtering and enhancing sound signals, one targets satellite imaging, and the last use-case focusses on video processing. Each of the use cases is discussed individually and the updated requirements are highlighted with respect to the initial requirements in an individual subsection.

Although there are no major updates with respect to the initial requirement specifications, there have been made several refinements on the KPI's, baseline benchmarks and the utilized techniques to reflect the current research direction in the CONVOLVE project.