CONVOLVE

Seamless design of smart edge processors

GRANT AGREEMENT NUMBER: 101070374

Deliverable D1.1

Initial requirements and use-cases

European Commission

Title of the deliverable	Initial requirements and use-cases
WP contributing to the deliverable	WP 1
Task contributing to the deliverable	Task 1.1, 1.2
Dissemination level	PU – Public
Due submission date	30/04/2023
Actual submission date	29/04/2023
Author(s)	Tobias Piechowiak, Clément Laroche, Riccardo Miccini, Benjamin Cramer, Andre Guntoro, Lara Arche Andradas, Egbert Jaspers
Internal reviewers	Mounir Ghogho (UIR) Tobias Grosser (UED) Manil Gomony

Document Version	Date	Change
V0.1	17/03/2023	Initial ToC
V0.2	25/04/2023	Revision
V0.3	29/04/2023	Final version

Contents

1 Introduction 4
1.1 Intro to Chipset metrics and benchmarking 4
1.2 The Convolve project within EU Horizon's innovation program 5
2 Overview on Use-cases 6
2.1 Deep Noise Suppression & Speech Quality Prediction (GN Audio) 6
2.2 Acoustic Scene Analysis (Bosch) 16
2.3 On-board Computer Vision (TASE) 21
2.4 Video-based Traffic Analysis (Vinotion) 26
3 Individual Requirements 28
3.1 Vinotion 28
3.2 GN Audio 29
3.3 Bosch 30
3.4 TASE 30
4 Conclusion 31
5 Description of requirement categories 32

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 imminent advantages of edge processing[1]. Factory robots, cars, as well as audio processes, for example, need high-speed processing with minimal latency or power requirements.

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 Intro to Chipset metrics and benchmarking

As stated above, although every use-case is different, we will try to define consolidated metrics that cover all use-cases and are accepted by all parties involved – although each entity might define their own cut-off thresholds for success for each metric and may prioritize each metric differently.

A survey of the relevant literature reveals four main dimensions, or "pillars" that are often used to characterize and benchmark a given solution and are suitable for our purposes. These are:

- Performance (P)
    - Inference speed on test datasets
    - I/O latency for real-time applications
    - MACs / cycle

[1] https://www.datamation.com/edge-computing/pros-cons-edge-computing/

- Operations per second (OPS)
- Real-time factor (RTF)

- Efficiency (E)
  - Average power for inference on test dataset (W)
  - Peak power (W)
  - TOPS/Watts
  - TOPS / mm^2
- Quality (Q)
  - Model accuracy
  - Use-case dependent metric
    - Speech Quality
    - Sound classification accuracy
    - Feature accuracy
- Size (S)
  - Model size & memory requirements for parameters
    - Quantized vs. non-quantized.
  - SW memory

For given (required) application accuracy, often used metrics are (from the computing area), energy-efficiency (using either TOPS/Watt, or Joule/Op) and area efficiency (TOPS/mm^2). These 2 can also be combined into energy-area efficiency, taking their product.

Based on the above, it would also be possible to create a single-value metric V as a linear combination of those individual pillars:

V = function (P, E, Q, S).

where each user's priorities on where to put the focus of development can be considered.
Note, that while linear combinations are a first order approximation of the weighted performance, they usually provide a simple and easy-to-understand way for aggregating multiple performance dimensions. Furthermore, non-linear combinations in most cases, might be too complex to interpret and understand for obtaining the factors' individual contribution and might provide no extra benefit for an overall perception of factor contribution.

Some of the above performance metrics are naturally use-case specific – for example, the *real-time factor* is used in audio processing. It is computed as the ratio between processing time of a time frame and the hop-size. Thus, if processing on one time frame takes longer than the delay between advancing to the next frame, no real-time processing is possible.

1.2 The Convolve project within EU Horizon's innovation program

The primary aim of this document is to identify and define practical application use-cases from various domains that would prove beneficial to be executed on a secure edge processor with ultra-low power consumption. These use-cases come from three different companies, covering:

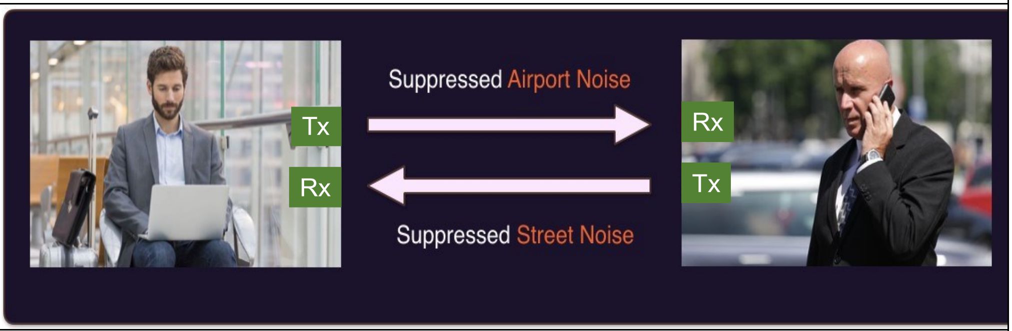
- Audio processing (GN Audio, Bosch)
- Satellite image processing (TASE)
- Video processing (Vinotion)

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 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 is discussed. Moreover, the authors of each use-case have provided software code that will be stored in a Git repository hosted by TU Eindhoven. This repository can be used as a benchmarking tool and a basis for future development.[2]

2 Overview on Use-cases

2.1 Deep Noise Suppression & Speech Quality Prediction (GN Audio)

Use case title	Deep Noise Suppression / Speech Enhancement
Owner	**GN Audio**
Other partners involved	Bosch
Visualization of the use case	
Use case description	*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 or intelligibility. 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 denoising tasks. It is considered a "hot topic" in the wider communication and computer industry, with large companies and academy dedicating massive resources to it[3] although not necessarily focusing on edge processing.

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

[3] https://arxiv.org/pdf/2202.13288.pdf

Use case neural network models	Here, we will focus on several models for speech enhancement that have received ample citations in recent publications and have been shown to be effective for the task. They use a combination of different layers and activation operations i.e., convolutional (1D Convolutional, 2D Convolutional), recurrent (Long Short-term Memory (LSTM), 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: UNet a. Inputs: Short-term Fourier Spectrogram (STFT) b. Layers: 2D Convolutions, 2D Transposed Convolutions, 2D Pooling c. Activations: Leaky ReLU d. Skip-connections: Yes 2. Name: Demucs Denoiser a. Inputs: Raw audio waveform b. Layers: 1D Convolutions, 1D Transposed Convolutions, Long Short-Term Memory (LSTM) c. Activations: ReLU d. Skip-connections: Yes 3. Name: NsNet2 a. Inputs: Short-term Mel Spectrogram b. Layers: Fully connected, Gated Recurrent Units (GRU) c. Activations: ReLU, Sigmoidal d. Skip-connections: No Note, that these models and their structures could be combined to even form different architectures. For example, *NsNet2* consist of mainly recurrent layers and could very well benefit from a convolutional layer as input for additional effective feature extraction. Furthermore, several relevant research questions arise from the presented architectures. For example, skip connections are memory heavy from a HW perspective - could those architectures that use them be modified to perform as well without them? Note also, that not all of them must be considered in the project.
Statements of Needs	Much better (high-quality speech) and power efficient noise suppression running on the edge as part of a *Tx* processing line before transmission of signal.
CONVOLVE objectives addressed	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 behavior of the models, both on the model level as well as

<table>
<tr><td></td><td>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).</td></tr>
<tr><td>CONVOLVE WPs involved</td><td>WP4, WP5, (WP6)</td></tr>
<tr><td>Quantified baseline at the start of the project</td><td>As mentioned above we want to assess the proposed models by the following performance metrics. Here, we describe the baseline model (Unet) which can run on actual hardware and follow-up on the larger models that should run on any future chipset.

Here, we will give concrete numbers for one model that has been discussed in literature and comes very close to the some of the architectures discussed for Convolve purposes.

The numbers are either based on computations from torchinfo or from information in literature (see footnote). We will give information on

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

<table>
<tr><td rowspan="2">Model</td><td rowspan="2"># Parameters</td><td>MACs / s float16 x float16</td><td>Memory (Parameters) float16</td><td>Memory BW float16</td><td rowspan="2">Power @ 0.8 V</td><td rowspan="2">Power / Mmacs</td><td rowspan="2">Latency</td></tr>
<tr><td>MACs / cycle float16 x float16</td><td>Memory (Parameters) Int8</td><td>Memory BW Int8</td></tr>
<tr><td rowspan="2">NsNet[4] type on described target</td><td rowspan="2">0.99 M</td><td>43 M</td><td>3.95 MB</td><td>0.55 MB / sec</td><td rowspan="2">66 mW</td><td rowspan="2">~ 1.5 mW</td><td rowspan="2">1.7 msec</td></tr>
<tr><td>2 (from paper[4])</td><td>0.98 MB</td><td>0.14 MB / sec</td></tr>
</table>
This table should be considered as the baseline on which to measure any new model deployment on. 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 Digital Signal Processing (DSP) parameters:

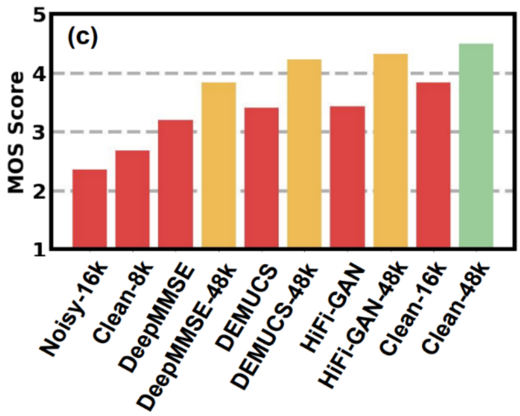
• Block frame-size: 400 samples (@257 one-sided FFT-bins)
• Hop-size: 100 samples</td></tr>
</table>

[4] https://arxiv.org/abs/2210.07692

- *Sampling rate*: 16 kHz

In DSP, hop-size refers to the number of samples between consecutive analysis frames in a signal processing algorithm. It is typically used in time-frequency analysis, such as in spectrogram or Mel-spectrogram calculations. Note, that numbers will change when other DSP settings are used. Generally, one trades latency for compute requirements, smaller *Hop-sizes* decreasing latency but increasing compute requirements.

Regarding use-case specific metrics, with current denoising frameworks available relatively high speech quality is achieved, typically Mean-opinion scores (MOS) between 3 and 4 for a bandwidth of 8 kHz. As a reference for our claims, we can regard the following figure 1[5]. Here, *red* gives the baseline for 8 kHz bandwidth, *green* gives the full band reference and *yellow* the respective denoiser with a bandwidth extension as a post-processing.



TYPICAL MOS SCORES FOR DENOISING ALGORITHMS WITH AND WITHOUT SUCCESSIVE BANDWIDTH EXTENSION. NUMBERS BEHIND THE NAME INDICATE BANDWIDTH. FROM: BANDWIDTH EXTENSION IS ALL YOU NEED

Goals at the end of the project in defined metrics

The following shows similar numbers as for the baseline model but for larger and more effective models that could ideally be deployed to a Neural Processing Unit (NPU), including operations needed per cycle like the previous table but without exact power figures since these models have not been deployed.

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
Demucs	18.8 M	4350 M	75.5 MB	72 MB/s

[5] https://pixl.cs.princeton.edu/pubs/Su_2021_BEI/ICASSP2021_Su_Wang_BWE.pdf

		> 44 @ *100MHz*	18.9 MB	18 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

Metric	Unit	Value
Performance	I/O latency RTF MMACs	< 2 msec < 0.8 > 4350
Power Consumption	mW / MMACs Full-cycle load	< 0.1 < 0.5 W
Memory	MB	>
Quality	VISQOL[6] Mean-opinion score (MOS)	> 4

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

Besides this very concrete model-related metrics, we want any future SoC also to fulfil the following requirements: latency and real-time factor. 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 HW elements involved in the use case	• Sufficient memory close to processor(s) as indicated by metrics requirements. • High memory bandwidth to and from processor(s) • Parallel processing pipeline that consisting of multiple processors. • Variable clock rate depending on NN load
Key SW elements involved in the use case	• 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 function needs to be supported. • Profiler for memory management • Profiler for processing management • Emulator for simulating SoC “off-line” • CUDA support

[6] https://arxiv.org/pdf/2004.09584

	• cuDNN support
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 *(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 requirement(s)	In general, GNA is not overly concerned about overall lack of privacy or safety 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). However, there are two main lines of security aspects GNA is interested in: 1. **Protection of intellectual property** in form of neural network models 2. **Secure update of firmware and neural network models** to the edge device Here, 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 and train. This can and has been threatened by e.g., using reverse engineering techniques. In general, **reverse engineering** refers to the process of analysing a product or system to understand its design, function, or components. In the context of artificial intelligence, reverse engineering[7] can be used to extract information about the inner workings and design of a neural network model. One common method of reverse engineering neural networks can be the **teacher-student method**[8]. In this approach, a large, complex neural network (the teacher) is trained on a dataset, and then a smaller, simpler neural network (the student) is trained to mimic the behaviour of the teacher network. By examining the output of the student network, an attacker can gain insights into the inner workings of the teacher network, potentially revealing proprietary information about the model's architecture or training data. To prevent reverse engineering through the teacher-student method, one approach is to introduce noise or other markers to the output of the student network, making it more difficult for an attacker to discern useful information. Another approach is to use adversarial training, in which the student network is trained to resist attempts at reverse engineering by introducing deliberate misdirection or obfuscation.

[7] https://arxiv.org/abs/1711.01768

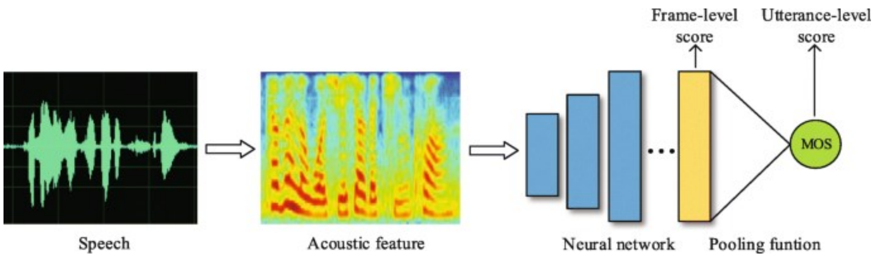
[8] https://arxiv.org/abs/1503.02531

	An additional method of reverse engineering neural networks can be through the analysis of their output alone, without knowledge of the underlying architecture or training data. For example, an attacker might input a series of carefully constructed test cases to the network and analyze its responses to infer information about its internal structure or decision-making process. To mitigate this special type of reverse engineering, one approach could be to **introduce watermarking**[9] into the output of the neural network. Watermarking involves adding a small, non-noticeable signal to the output of the network that can be used to identify the source of the output. This can deter attackers from attempting to reverse engineer the network, as the presence of the watermark can reveal the actions of the attacker. **Secure firmware updates** are referring 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. One approach to secure firmware updates is to use encryption to protect the update process. This involves encrypting the firmware update using a cryptographic key, which is then securely transmitted to the device much the same as done for any symmetric and asymmetric encryption schemes. The device uses the key to decrypt and verify the firmware update, ensuring that it is authentic and has not been modified during transmission. This would help to prevent attackers from intercepting and modifying the firmware update and ensures that only authorized updates are installed on the device. In a similar fashion, **neural model updates** through encryption involves the use of encryption to protect the transmission and storage of updates to neural networks, which are commonly used in machine learning applications. As before, this is important to prevent attackers from intercepting or tampering with the updates, which could lead to degraded performance, security vulnerabilities or IP infringements. One final contemplation to secure firmware or model updates is the potential use of h*omomorphic encryption*[10]. The main advantage of homomorphic encryption is that it allows computations to be performed on encrypted data without first decrypting it. This means that sensitive data, e.g., the new firmware, can remain private and secure, even while it is being processed and used in computations. Potentially, it could even be used for secure AI inference on audio data, allowing the model to perform computations on the encrypted data, storing it in

[9] A survey of deep neural network watermarking techniques (arxiv.org)

[10] https://homomorphicencryption.org

	batches/buffers and only perform decryption process on a longer timescale than the actual audio processing which could reduce resources for the actual decryption process. However, there might be challenges associated with using homomorphic encryption for AI model inference, such as the computational overhead involved in performing computations on encrypted data. But advances in the field of homomorphic encryption have made the technology increasingly practical, and it might become feasible for use in the audio-data domain.

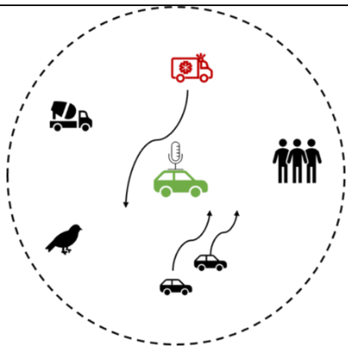
Use case title	Speech Quality Prediction
Owner	GN Audio
Other partners involved	N/A
Visualization of the use case	
Use case description	The field of speech quality prediction can be divided into full-reference (also known as intrusive), which requires a clean reference signal to compare against, and 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, POLQA, VISQOL 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. However, due to the lack of reference signals in real-world scenarios, most of the focus will be on reference-less methods. Some of the main unintrusive speech quality estimator models in the literature as DNSMOS, NISQA, and QualityNet. DNSMOS is a convolutional model featuring four blocks consisting of 2D convolution, ReLU activation, max pooling, and dropout, followed by two dense layers. It is trained on a set of 600 noisy speech clips that have been processed through a variety of noise-suppression algorithms, whereas the target MOS were gathered through a human subjective listening test based on the ITU-T P.808 standard. Similarly, NISQA comprises a convolutional frame-wise feature extractor, followed by a self-attention block to model temporal dependencies, and finally an attentive pooling mechanism. This model can be trained to predict both MOS and four additional quality dimensions.

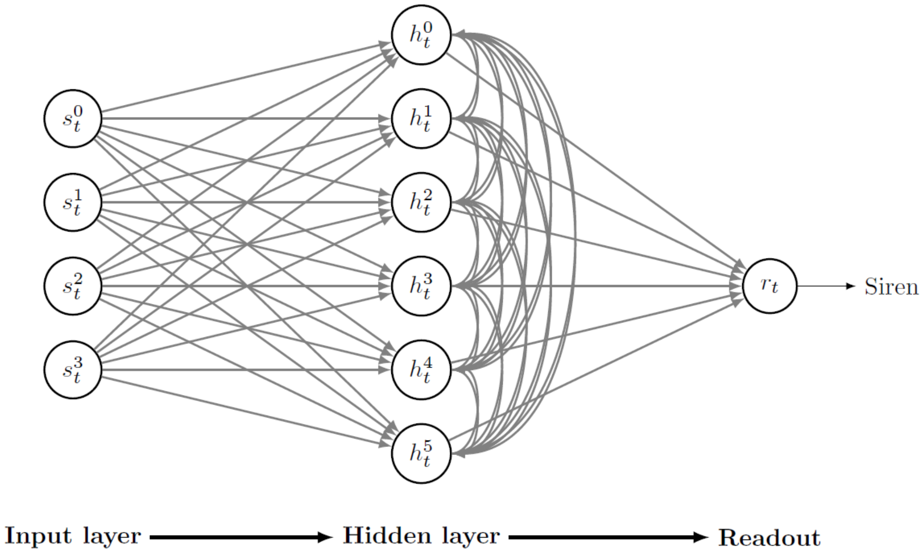
	Finally, QualityNet uses a bidirectional-LSTM block followed by fully connected layers predicting frame-wise MOS predictions which are then averaged together to provide a global score. To make the model causal, the bidirectional-LSTM blocks can be substituted with LSTM layers. Although new prediction models are constantly being developed, these models exemplify the use of three deep learning primitives, namely convolution, attention mechanisms, and recurrent units, and thus provide a strong foundation for further experiments aimed at reducing their computational footprints in terms of size, number of operations, and inference latency. This will be achieved through several optimization techniques, including quantization/binarization.
Statements 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 addressed	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).
CONVOLVE WPs involved	WP4, WP5
Quantified baseline at the start of the project	As for noise-suppression we mainly focus on following metrics for characterizing network performance: • Multiply-accumulate per seconds • Memory footprint • Memory bandwidth • Power • Power / Mmacs (Million MACs) • Latency Here, the info comes again mainly from torchinfo. 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			
QualityNet	120 K	5.3 M	0.45 MB	5 MB	7.6 mW	1.5 mW	< 3ms
			0.11 MB	1.4 MB			

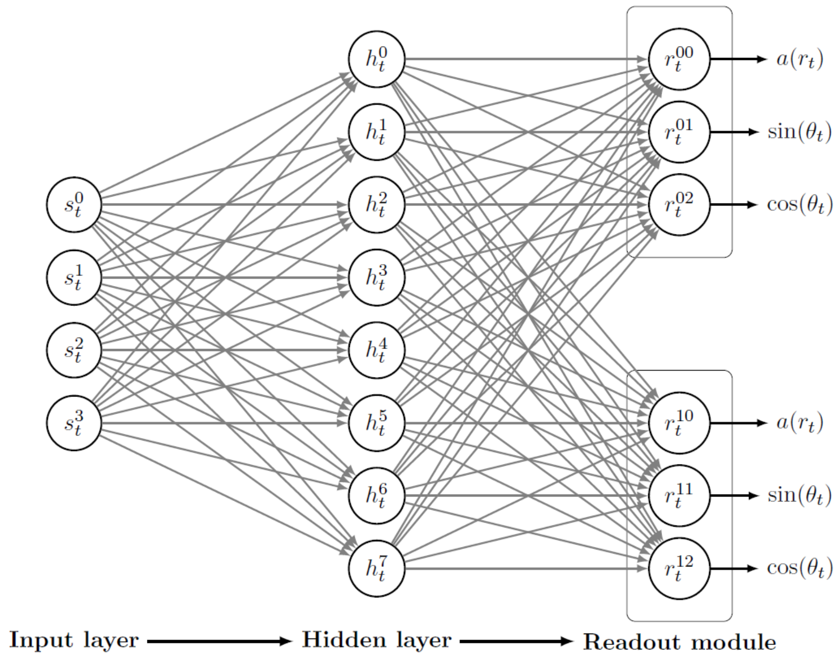
Goals at the end of the project	Effective non-intrusive speech quality prediction running seamlessly and very power efficiently on the edge for constant monitoring. Power requirements should have been decreased by at least a factor of 10 relative to the numbers in the above table.
Key HW elements involved in the use case	• Sufficient memory close to processor(s) as indicated by metrics requirements. • High memory bandwidth to and from processor(s) • Parallel processing pipeline that consisting of multiple processors. • Variable clock rate depending on NN load.
Key SW elements involved in the use case	• 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 function needs 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.

2.2 Acoustic Scene Analysis (Bosch)

Use case title	Acoustic Scene Analysis
Owner	Robert Bosch GmbH
Other partners involved	N/A
Visualization of the use case	
Use case description	Acoustic scene analysis aims to extract information about the environment from the sound signal(s) recorded by a receiver. In the scope of the proposed use-case, we focus on typical traffic scenes as recorded by a car equipped with microphones. Among others, these scenes include the sounds of moving emitters like cars and emergency vehicles, but also signals of static (non-moving) sources. Based on this superposition, we aim to extract information about the identity and position of different emitters of interest. These environments pose challenging conditions since the underlying signals potentially exhibit a high degree of temporal as well as spatial overlap and show diverse and complex structures. In addition, different signal-to-noise ratios (SNRs), source object as well as signal amplitudes need to be tackled which all require robust edge processing approaches to the topic of acoustic scene analysis.
Statements of Needs	The information contained in the recorded acoustic signals has the potential to not only augment existing sensory data, but, moreover, provides additional and safety critical features in situations where movie or comparable data may not be sufficient or even available. Hence, the analysis of the acoustic environment in an edge computing scenario is of general interest to further enhance autonomous driving.
Use case neural network models	Within the scope of CONVOLVE, we focus on recurrent neural architectures to approach the analysis of acoustic scenes. In that process, we aim to investigate the following tasks and associated neural architectures: 1. **Siren detection** Within this first task, audio recordings of traffic scenes are used to detect siren sounds. For that purpose, we consider recurrent architectures to predict the presence of siren sounds within the input signal.

	 Here, we exemplarily show a detection network with a single recurrently connected hidden layer. The key components of this architecture are: • an input layer implemented by a Mel-Spectrogram of the audio signal, • a hidden layer (set of hidden layers) composed of GRUs or LSTMs, • and a readout comprising of a single linear unit with sigmoidal activation. The network's response is evaluated on every time frame of the Mel-Spectrogram by thresholding the output of the readout unit: In case its activation exceeds a threshold of 0.5, the frame is assumed to contain a siren signal during inference. For the training of the detection NN, publicly available dataset[11] can be used. **2. Sound source tracking** Here, a recurrent architecture is tasked to predict the position of emergency vehicles in simulated acoustic scenes. For that purpose, a set of readout modules is connected in series to the actual recurrent architecture. Each readout module comprises of three linear units to output distance and angular information of a single source.

[11] Asif, M., Usaid, M., Rashid, M., Rajab, T., Hussain, S., & Wasi, S. (2022). Large-scale audio dataset for emergency vehicle sirens and road noises. Scientific data, 9(1), 599.

<table>
<tr>
<td></td>
<td>


In the attached visualization of the network architecture, we omitted recurrent connections within the hidden layer for clarity. Note that the number of readout modules imposes an upper limit on the number of sources that can be tracked simultaneously. The key components of these tracking networks are:

an input layer implemented by a Mel-Spectrogram of multi-channel audio signals,
a hidden layer composed of either GRUs or LSTMs,
a set of readout modules each comprising of three linear units with sigmoidal and tanh activation functions to constrain the range of the prediction to their physical plausible range.

Due to the lack of applicable tracking data, we draw on data augmentation strategies to emulate the sounds of moving sources along pre-defined trajectories based on audio recordings and/or simulated signals of stationary sources[12].
</td>
</tr>
<tr>
<td>CONVOLVE objectives addressed</td>
<td>Energy efficiency is of central interest for this use case since the total power budget available within a car is limited (objective 1). Moreover, a rapid adaptation to changing requirements is desired to stay flexible (objective 2). Having detailed information about emergency vehicles is safety critical and hence demands for reliable hardware and security against attacks during inference as well as system updates (objective 3). Last, the use case of acoustic scene analysis constitutes an ideal smart edge application. Here, the events of interest (siren signals) rarely occur which is why smart adaptation mechanisms are a promising method to further enhance efficiency.</td>
</tr>
</table>

[12] Damiano, S., & van Waterschoot, T. (2022). Pyroadacoustics: a Road Acoustics Simulator Based On Variable Length Delay Lines. In Proceedings of the 25th International Conference on Digital Audio Effects.

CONVOLVE WPs involved	The proposed neural architectures are refined within the scope of CONVOLVE. Hence, WP4 is tightly involved in this use-case. Further, a close interaction with WP2, WP3, WP4, WP5 and WP6 is targeted.
Quantified baseline at the start of the project	For both tasks, PyTorch models of the proposed neural architectures are available that facilitate a first set of benchmarks and an estimate of the computational footprint. Currently, all networks are trained on Mel-Spectrograms extracted from the raw audio data. In the following we evaluate one specific network topology for each of the two presented tasks. All reported metrics refer to the processing of a single input (time) frame of the Mel-Spectrograms. **1. Siren detection** For the siren detection task, we consider the following parametrization and recurrent architecture: • The input contains a single channel audio signal sampled at 8 kHz. • Based on that, a Mel-Spectrogram with 64 channels is calculated with a hop length of 100 ms and a window size of 50 ms. While the choice of the hop length ensures to meet the throughput constraints, the lower window size reduces the computations associated with the feature extraction and leaves room for neural network calculations, thereby facilitating real-time processing of the input data. • The actual neural network comprises of a single hidden layer composed of 100 GRUs. • The activations of the hidden units are forwarded to a single linear readout unit with sigmoidal activation function. It is noteworthy, that there might be further space for optimization by reducing the window size to reduce the computational footprint of the feature extraction and to allow for more time-consuming neural network calculations (e.g., deeper neural networks with larger hidden layers) by guaranteeing real-time processing at the same time. However, for the current GPU implementation the window size used for the spectrogram calculation dominates the total duration spend on the processing of a single input frame. **2. Siren Tracking** For the tracking of siren sounds in 2D space, we stick to the following parametrization and architecture: • The input contains four channel audio signals sampled at 16 kHz. • Based on that, Mel-Spectrograms with 64 channels for each of the raw audio channels are calculated with a window and hop length of 64 ms.

- These features serve as input for a single recurrently connected hidden layer composed of 1000 GRUs.
- Their activation is forwarded to two readout modules, each with 3 linear units (1 sigmoidal, 2 tanh activations).

Again, the throughput is dominated by the signal processing in the current implementation and could potentially be improved by smaller window sizes.

In the table below, we summarize important model details as well as performance metrices associated with the NN processing.

Model	#Parameter	Mult-Adds (M)	Parameter memory (MB)	Metrics	
				Performance	Throughput (frames/s)
Detector	49 901	0.05	0.2	Acc > 0.95	> 10
Tracker	3 780 006	3.78	15.13	wMSE	> 10

Currently, all parameters and observables are represented by 32-bit floating-point values. Since the models are currently available as GPU implementation, their power budget is assumed to reside in the range 5-100 W. The reported throughput values correspond to the duration required to evaluate the network for a single time step and the period imposed by the window size of the Mel-Spectrogram.

It is noteworthy that the actual value of the performance potentially depends on the parametrization of the feature extraction and, i.e., trade-offs between the spectral resolution and the computational footprint are possible for a final hardware implementation. The current values are chosen to capture a broad frequency range. At low SNRs, high frequency components could be less impacted by road noises and could hence boost performance.

Goals at the end of the project

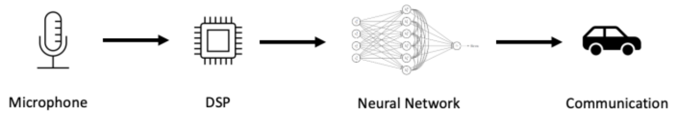
At the end of the project, we would like to have efficient hardware implementations of the proposed models. Further, we aim for a reduction of the model sizes by

1. compression techniques like quantization (8-bit int) as well as pruning,
2. binarization and
3. the implementation of dynNNs.

With these methods, the computational footprint of the presented models could potentially be reduced to facilitate an efficient implementation. As outlined above, the evaluation of the proposed models is limited by the window size used for the Mel-Spectrogram calculation. Hence, our main goals concern smart feature extractions as well as the power budget of the final implementation. The following table summarizes the goals.

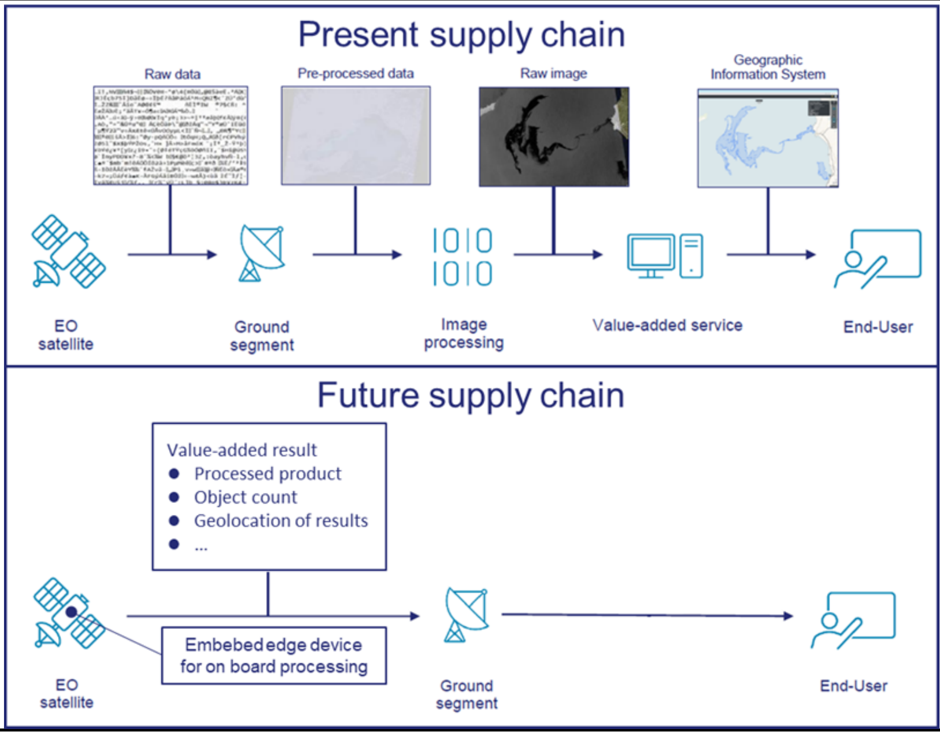
Model	Parameter memory (MB)	Metrics

<table>
<tr><td></td><td>
<table>
<tr><th></th><th>Int8</th><th>Performance</th><th>Throughput (frames/s)</th><th>Power (mW)</th></tr>
<tr><td>Detector</td><td>0.05</td><td>Acc > 0.95</td><td>> 10</td><td>< 100</td></tr>
<tr><td>Tracker</td><td>3.8</td><td>wMSE</td><td>> 100</td><td>< 100</td></tr>
</table>

It is noteworthy that the above stated parameter memory represents the target for conventional compression techniques, assuming an 8-bit integer precision as a target. The application of dynNNs has the potential to further improve upon these metrics.</td></tr>
<tr><td>Key HW elements involved in the use case</td><td>• Feature extraction (DSP blocks)
• Mixed precision
• Tolerance regarding harsh environments (radiation, vibration, temperature, ...)
• Time multiplexing (parallel processing, smart mapping)
• Adjustable clock frequency depending on input (for dynNN applications)
• Efficient always-on detection</td></tr>
<tr><td>Key SW elements involved in the use case</td><td>• Signal processing (MelSpectrogram, AmplitudeToDB)
• Recurrent layers (GRU/LSTM)
• Linear layers
• Activation functions (Sigmoidal, Tanh)
• Model updates</td></tr>
<tr><td>Interactions</td><td>
</td></tr>
</table>

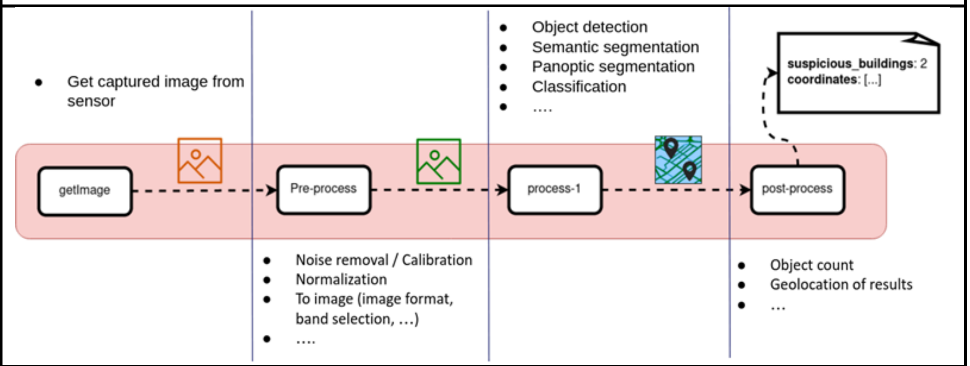
2.3 On-board Computer Vision (TASE)

Use case title	On-board Computer Vision
Owner	Thales Alenia Space España
Other partners involved	TUE, Vinotion

Visualization of the use case	
Use case description	The images generated by Earth Observation (EO) satellites are traditionally downlinked to ground for processing and analysis. 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. EO images present a series of peculiarities related to the sensors that generate them and the conditions under which they are captured, which must be considered during the process of processing the image and which are detailed below as they are considered highly relevant for the rest of the WPs: • They have a large variability, among others, in the number of channels, value ranges, size, resolution and intensity, depending on the type of sensor generating the data. • They have a spatial reference of the captured area called georeferentiation and a temporal reference of when the capture was made. This information is usually sent within the image as attached metadata. • They are large, so it is common to use tile processing. It is important not to lose the location of the tiles in the full image to be able to recover this location information once the processing is finished. • They can contain from one to thousands of bands. To monitor the processing and make them visible to the human eye, the images must be converted at some point to monochrome or RGB

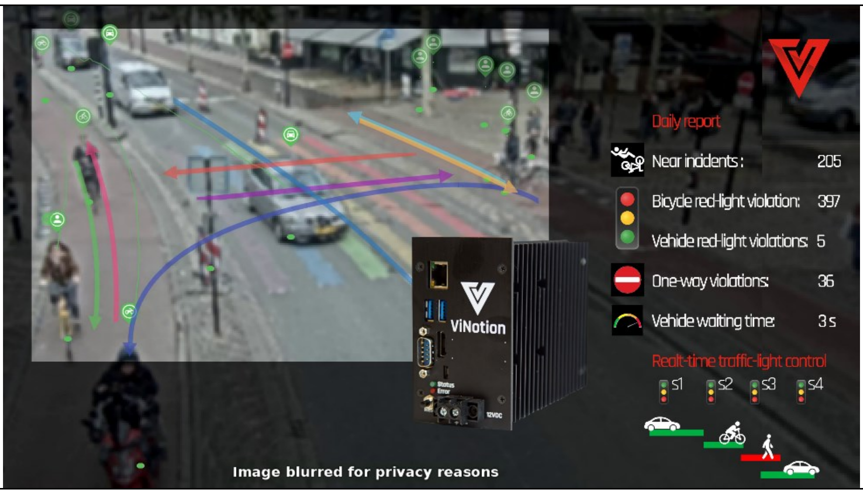
	representations by assigning individual bands or combinations of bands to the destination channels of the image. This visualization has its own problems associated with it (dark images, contrast stretching, radiometric and geometric corrections, speckle noise,…).
Use case neural network models	Convolutional Neural Networks (CNN) has been used in Thales Alenia Space for performing computer vision tasks applied to satellite imagery both in ground software products and embedded experimentation. Very deep backbones, usually pre-trained on large datasets like the general ImageNet or the space-oriented dataset called SpaceNet, are typically specialized on specific tasks using the available training data. Due to the reduced datasets in real problems, it is usually hard to train models from scratch thus transfer learning is a commonly used to bypass this limitation. From the main computer vision tasks (image classification, object detection semantic segmentation and instance segmentation) TAS activity has been focused on object detection and semantic segmentation as main tasks but in some use cases, multiple tasks are combined to filter the scenes and simplify the achievement of the main task. Object detection consists of detecting objects in an image and their spatial location within the image. Bounding boxes (rectangles) are used to delimit the object shape. In the object detection task, the algorithms are usually classified in two-step and one-step methods. The two-step algorithms use two models, one for extracting regions of interest and a second model for classifying and refining the localization of the objects. On the other hand, one-step algorithms use only one model for localizing and classifying the objects in an image in just one pass. Both two-step methods like Faster R-CNN[1] and Mask R-CNN[2] and one-step methods like YoloV2[3] have been used to achieve the task in 3-band raster images generated from multispectral and synthetic aperture radar (SAR) imagery and in single band images generated from SAR sensors. Semantic segmentation task consists of classifying each pixel in an image from a predefined set of classes. Typically, each class is assigned a color, and therefore replacing the pixel value with the color of the class produces a new image that represents the results. In the semantic segmentation task, DeepLabV3[4] and BiSeNet V2[5] models have been used in 3-band images (RGB combination among others), generated from high resolution multispectral imagery.
Statements 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. Cadastres, asset tracking,). To build the future supply chain, it is necessary: • 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 addressed	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 prevented, 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 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).
CONVOLVE WPs involved	WP2, WP3, WP4 & WP5
Quantified baseline at the start of the project	Current processing is done sequentially in a pipeline fashion with containerized steps. Images are processed using EO libraries which uses exclusively CPU and RAM memory requiring at least 8GB of dedicated RAM to transform the images. Computer vision stages that use deep learning techniques mainly use GPU resources. Due to the depth of the CNN used and context required, 4GB to 8GB of the GPU memory are used currently by the applications (depending on the computer vision task). The following GPUs have been used so far: • Nvidia Tesla M60 (Azure NVv3-series VMs): o Memory: 8GB

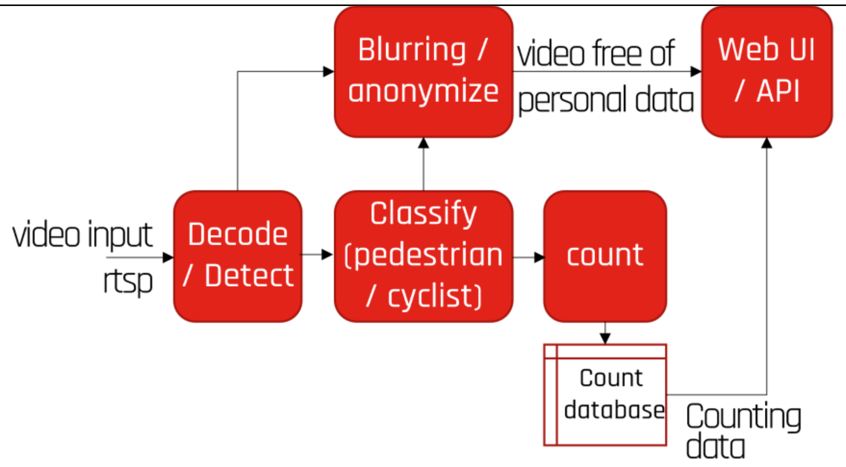
	o Max power Consumption: 300W • Nvidia Quadro RTX 4000 o Memory: 8GB o Max power consumption: 160W
Goals at the end of the project in defined metrics	• Design capable of supporting different neural networks more than 50 layers deep. • Power consumption objective: o < 0,1 mW/MMACs o Maximum dissipation compatible with a flight design < 20 W (per device, SoC). o At board level, maximum power dissipation < 50 W (with as many SoCs working in parallel as possible).
Key HW elements involved in the use case	• Enough memory close to processor(s). • High memory bandwidth to and from processor(s) • Parallel processing pipeline that consisting 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 has 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
Key SW elements involved in the use case	• All convolutional layers need to be supported. • All recurrent layers need to be supported. • Dense layers need to be supported. • All current activation function needs 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	

[1] [1506.01497] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (arxiv.org)
[2] [1703.06870] Mask R-CNN (arxiv.org)
[3] [1612.08242v1] YOLO9000: Better, Faster, Stronger (arxiv.org)
[4] [1706.05587v3] Rethinking Atrous Convolution for Semantic Image Segmentation (arxiv.org)
[5] [2004.02147v1] BiSeNet V2: Bilateral Network with Guided Aggregation for Real-time Semantic Segmentation (arxiv.org)

2.4 Video-based Traffic Analysis (Vinotion)

Use case title	Video-based traffic analysis
Owner	ViNotion
Other partners involved	TU/e, Thales
Visualization of the use case	
Use case description	ViSense is 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 m^2 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.
Statements 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 provided 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 addressed	Obj1: Energy efficient mechanisms also support Obj4 on Smart edge processing enabling mechanisms. Compact, cost-efficient and advanced visual interpretation requires a CONVOLVE approach. For privacy protection and secure traffic control, also Obj3 (security and reliability mechanisms) is important.
CONVOLVE WPs involved	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 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 TX2 platform. 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, colour conversion, scaling, image stabilization, object blurring, a webserver for webservices and dashboarding, etc. The 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. Most of the resources of the TX2 system are currently reserved for a single ANN performing object detection and classification (mainly GPU resources) and for a tracking algorithm (combination of GPU and CPU resources). Video decoding is currently offloaded to a dedicated on-board hardware.
Goals at the end of the project	The future of ULP AI processing and deep learning is inevitable. We want significant power-efficiency improvement to reduce 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 PoE cameras is limited.
Key HW elements involved in the use case	Current HW elements are: • 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. • ~ 20 Watts

Key SW elements involved in the use case	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	

3 Individual Requirements (See here)

3.1 Vinotion

ID	Requirement (short and as specific as possible)	Dependency (ID)	Complexity (1 High, 3 Low)	Priority (1 High, 3 Low)
Architectural requirements				
A1	Allow a diverse set of processing blocks in a pipeline: e.g., video coding, color conversion, detection, tracking, projective transformations, etc. to facilitate heterogenous processing with highly efficient inter-communication		2	1
A2	Low power per operation: For small product design without dissipation concerns and allowing complex applications		2	2
A3	The most important hardware interfaces are 1Gbit Ethernet, UART, OTG, USB, HDMI, M2 slot, Mini PCIe, reset	A1	3	1
A4	Accelerators for video encoding / decoding	A1	3	1
A5	GPU for rendering	A1	3	1
A6	Deep learning accelerator	A1	2	1

A7	Compatible with open DNN frameworks (Pytorch desired)	A1, A6	2	2
A8	Compliant with Yocto		2	2
A9	At least 16 GB RAM and 32 GB eMMC	A1, A4, A5, A6	3	2
A10	Ambient temperature –30 to +60 deg			3
Behavioural requirements				
B1	The system should host a webserver to give full freedom of implementing interactions	A1		1
B2	Boot on Power,	A1		1
B3	Scalability in Power usage if not all compute resources are used or needed		1	2
Functional requirements				
F1	Real-time video processing	A4		
F2	Low latency to enable traffic control			
F3	Run several NN models simultaneously		2	1
F4	Data security using Trusted Platform Module or alternative			
Non-Functional requirements				
NF1	Flexibility for large variety of video sensors types: Frame-rates, resolutions, 8 – 16 bits, number of colour channels, etc		3	2
NF2	Future proof: new algorithms including AI are advancing in a rapid pace		1	2
NF3	Tooling for rapid application development in software	A1	1	2

3.2 GN Audio

ID	Requirement (short and as specific as possible)	Dependency (ID)	Complexity (1 High, 3 Low)	Priority (1 High, 3 Low)
Architectural requirements				
A1	Support a flexible DSP pipeline – STFT and Mel transformations could even be accelerated. Raw signal inputs should be supported as well optionally.		2	1
A2	Low power operations in both DSP and accelerator pipeline for complex NN model deployment	A1	2	1
A3	A “lambda” layer – wraps arbitrary processing into NN layer	A1, A2	1	3
Behavioural requirements				
B1	Feedback of speech quality assessment to user so the user can intervene manually		3	3

B2	Knob for changing the amount of (denoising) processing taking place to empower individual preferences		3	3
Functional requirements				
F1	Low latency for always retaining an RTF < 1	A1		
F2	Seamless dynamism – switching between models must be imperceptible		2	1
F3	Run several NN models simultaneously		2	1
Non-Functional requirements				
NF1	Multi-channel inputs	NF2	2	3
NF2	Multi-modal inputs – i.e., sound, vibration sensors, PPG)		1	3
NF3	Rapid mapping of applications and quick development iterations	A1	2	1

3.3 Bosch

ID	Requirement (short and as specific as possible)	Dependency (ID)	Complexity (1 High, 3 Low)	Priority (1 High, 3 Low)
Architectural requirements				
A1	Support for streaming multi-channel raw audio signals		2	1
A2	DSP	A1	2	1
A3	Composable structure	A1, A2	1	1
A4	Low power operation	A1, A2, A3	1	1
Behavioural requirements				
B1	Channelling the prediction to other sub-systems	A3, A4	3	1
Functional requirements				
F1	Model switching from detector to tracker network if siren is detected and communication of predictions	B1, NF2	1	1
Non-Functional requirements				
NF1	Always-on	A4	2	1
NF2	Fast and dynamic reconfiguration (model switching, parametrization, …)	A1, A2, A2, F1	1	1

3.4 TASE

ID	Requirement (short and as specific as possible)	Dependency (ID)	Complexity (1 High, 3 Low)	Priority (1 High, 3 Low)
Architectural requirements				

A1	Due to the difficulty of accessing hardware, reconfigurable systems are always desired.	F1	2	2
A2	Fully scalable architecture (depending on the type of mission, memory and processing requirements vary greatly)		2	1
A3	Support of a wide range of layers (convolutional, dense, sequential,…), activation functions.	A4, A7, A8	2	2
A4	Compatible with open DNN frameworks (PyTorch desired)	A3, A7, A8	2	2
A5	Design compatible with space-grade components		3	1
A6	Self-healing design for maximum life extension (desirable up to 15 years)		1	3
A7	Compatible with standard light Operating Systems	A3, A4	2	2
A8	Storage is a main driver in the design due to the need of processing big volumes of data (images in the order of GB)	A3, A4	3	2
Behavioural requirements				
B1	A confident sovereign solution (without cyber risks, embedded application integrity)		3	1
Functional requirements				
F1	Possible continuous integration of HW/SW improvements and new features.	A1	2	1
F2	Model agnostic solution		3	1
F3	Possibility to orchestrate processing steps (containerized if possible)		3	1
Non-Functional requirements				
NF1	Allow maintaining the highest level of technicality during the satellite lifetime		2	3
NF2	Highly flexible (compatible with a large type of sensors)		3	2
NF3	Processing capability shall allow to execute the computer visions tasks described above (table 3.3)		2	1
NF4	The integrity of the SW being deployed on-board shall be prevented at all times.	B1	1	1

4 Conclusion

In this document, we define consolidated metrics that cover all use-cases and are used by all parties involved. The four main dimensions or "pillars" that are often used to characterize and benchmark a given solution include *Performance, Efficiency, Power Consumption, Quality, and Size.*

These metrics can be combined into a single-value metric, V, which reflects each user's priorities on where to put the focus of development.

The use-cases presented in this document highlight the specific requirements for *audio processing, satellite image processing, and video processing*, and how they relate to the overall objectives of Convolve which are:

1 Power efficiency
2 Dynamism of models processing pipeline
3 Security & Privacy

Furthermore, based on the use-cases presented and the consolidated metrics defined, we formulated concrete requirements that will serve as a working baseline for U/C optimization and for the remaining work-packages.

These requirements will consider the specific needs of each use-case and the priorities of the involved parties. By doing so, we aim to ensure that the developed solution meets the necessary performance, efficiency, power consumption, quality, and size requirements for each use-case.

5 Description of requirement categories

- **Architectural requirements:** Are those requirements which have a measurable effect on the system's architecture (they can be SW and HW). Some examples could be: an specific database (i.e. my system needs an oracle data base), there cannot be connections outside my local system, my system need Ethernet connection, it should run on a FPGA, etc.

- **Behavioural requirements:** these requirements define how the system's users (can be human beings or other systems) interact with the current system. Some examples could be: A knob is needed to controlled the different modes of my system, a touchscreen that allows user's to re arrange elements, the system should generate X output if receives an Y input, the motor must stop if receives an-input from Z.

- **Functional requirements**: define in a top-level way what you want your system to do. Example: if Y then Z, my system will show an image on a screen, etc

- **Non-Functional requirements:** are those requirements which defines how the system should work, it is related to the quality of the system. Examples: My system should not be on hold for more than 0.1 second, my system should be able to stop if, etc.