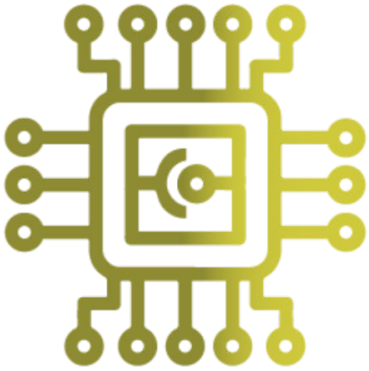

CONVOLVE

Seamless design of smart edge processors

GRANT AGREEMENT NUMBER: 101070374

Deliverable D4.1

Roadmap document for neural networks

Title of the deliverable	Roadmap document for neural networks
WP contributing to the deliverable	WP 4
Task contributing to the deliverable	Task 4.1
Dissemination level	RE - Restricted to a group specified by the consortium
Due submission date	30/04/2023
Actual submission date	29/04/2023
Author(s)	Benjamin Cramer (BOS), Simon Davidson (MAN), James Garside (MAN), Friedemann Zenke (FMI), Alaa Zniber (RAB), Tobias Piechowiak (MAN), Edward Jones (MAN), Quassim Karrakchou (UIR), Lara Arche Andradas (TASE), Alejandro Mousist (TASE), Thomas Verlest (AXE), Mounir Ghogho (RAB), Manil Dey Gomony (TUE), Egbert Jaspers (VIN), Andre Guntoro (BOS), Riccardo Miccini (GAN), Shreya Kshirasagar (BOS)
Internal reviewers	Alexa Kodde (CLAI)

Document Version	Date	Change
V0.0	30/03/2023	Table of content and main document structure
V0.2	22/04/2023	First integrated version
V1.0	29/04/2023	Final document

Deliverable Summary

The CONVOLVE project employs a two-step iterative approach to ultimately develop smart edge processors by addressing the whole design stack. The end of each phase is characterized by demos, guiding the next iteration. Algorithmic developments have to propagate through the full design stack to culminate in one of these demos. Hence, a roadmap of algorithmic development in the context of the planned demos is of great importance. Within this document, the key algorithmic principles required to show the full potential for each use-case in the demos are highlighted.

This document "D4.1 Roadmap document for neural networks" is a deliverable under the deliverable lead of BOS of the Work package No. 4 "Algorithmic principles for ultra-low power neural network (NN) processing", task T4.1 "Roadmap for energy efficient high performance neural networks" under the task lead of FMI, and sets out the "Roadmap for energy efficient high performance neural networks" including objectives in Chapter 1, state of the art in Chapter 2, roadblocks in Chapter 3, research plans for each use-case, data representations for spiking neural networks and learning strategies in Chapter 4, as well as a summary of the subprojects and plans in Chapter 5.

1. Objectives

Recently, methods from the field of Deep Learning (DL) achieved tremendous successes. The underlying models reach high accuracy in a vast range of industry-relevant applications. To be applicable in real-world scenarios, these increasingly complex models demand for efficient implementations on resource-constrained edge devices.

Apart from these models that are built to boost performance on specific applications, efforts have been made to develop methods that allow the reduction of the computational footprint of the proposed solutions on an algorithmic level. These strategies encompass model compression methods like pruning and quantization. Moreover, the utilization of spiking- as well as the instantiation of dynamic neural networks (DynNNs) promises to further reduce computational overhead.

All these approaches promise more efficient neural networks (NNs) and are therefore highly relevant for the development of the CONVOLVE project. Moreover, competitive solutions are fostered by a hand-in-hand development of soft- and hardware spanning all levels of the design stack and a constant re-evaluation, keeping awareness on the roadmap and requirements of the use-cases.

1.1. WP objectives

WP4 focuses on the development of algorithms and models for ultra-low-power NNs. In that scope, strategies like model compression, sparsity, DynNNs and online learning are considered as well as their application for the edge. Being generic, these methods are applied to both,

artificial neural networks (ANNs) and spiking neural networks (SNNs) for a set of industry-relevant use-cases.

1.2. WP contribution to CONVOLVE's objective

Anchoring algorithmic strategies to reduce the computational footprint of the deployed NN models deeply into the full design stacks fosters more efficient and flexible solutions. Further, the joint development with all other levels of the design stack will target all CONVOLVE objectives:

1. 100x improvement in energy efficiency,
2. 10x reduced design time,
3. Provide hardware security and real-time guarantees,
4. And smart edge applications.

The interconnections of algorithms and models with the rest of the design stack helps to achieve these four objectives being closely guided by the use-cases and the demos. The iterations help to refine the requirements and, hence, the demonstration of novel algorithmic strategies serves an inevitable role to achieve the CONVOLVE objectives.

1.3. WP contribution to other WPs

In general, the development of novel solutions within each WP (work package) is a collaborative effort that is driven by knowledge inherited from prior work of each partner. Ideally, this knowledge can serve as a basis for further research within the WP itself. Within WP4, specialists within the fields of model compression, DynNNs, SNNs as well as learning strategies jointly work on solutions facilitating the feasibility of the CONVOLVE objectives described in Section 1.2.

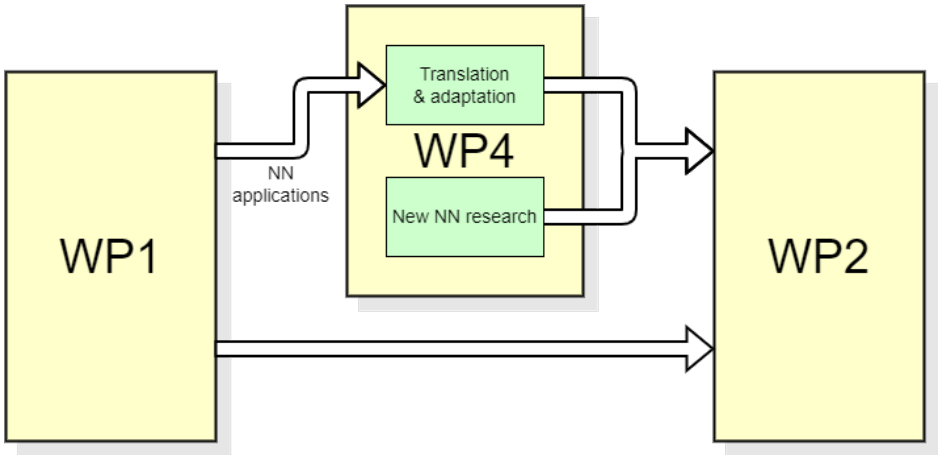

FIGURE 1: OVERVIEW OF WP INTERACTIONS. THE ALGORITHMIC DEVELOPMENT WITHIN WP4 IS MAINLY DRIVEN BY THE USE-CASES SPECIFIED IN WP1. BOTH, WP1 AS WELL AS WP4 SERVE AS STRONG DRIVERS FOR THE DEVELOPMENT OF HARDWARE ACCELERATORS WITHIN WP2.

Reaching the CONVOLVE objectives is of course a collaborative effort that also involves other WPs (Figure 1). More specifically, the strongest input dependency comes from WP1, where

existing and new applications relevant to the industrial partners are collected, analysed and identified for common factors and adaptation to (for example) new learning strategies are made. This information is then used to inspire the design development of other WPs in general and of WP4 in particular. The second strong interaction involves WP2, i.e. the accelerator design: Any newly proposed algorithm by WP4 needs to be efficiently represented by hardware components. Hence, WP4 serves as a driver for WP2. As an example, the work on SNNs needs to identify specific acceleration functions, especially those associated with timed operations, prior to the actual hardware design. In contrast, other operations targeting the support of ANNs already known to developers will be dealt with first. However, it is expected that there will be significant overlap between many functions.

It is noteworthy that some early development within WP4 is assumed to be self-contained, i.e. without specific input from the industrial partners and the use-cases. However, the most important objective of any algorithmic development should be the application to one of the industry relevant use-cases. By collaborating closely with these partners, WP4 can ensure that their needs are met and that the project as a whole is successful.

2. State of the art

In the following section, an overview of existing algorithmic approaches to the topic of ultra-low-power NN processing is provided. Here, we focus on methods that are highly relevant in the context of the CONVOLVE project and the respective objectives.

2.1. Model compression

Model compression techniques aim to reduce computational complexity as well as the size of NN models to promote implementations in resource constrained scenarios (Neill, 2020). In the following, we shortly review quantization (Section 2.1.1) and pruning methods (Section 2.1.2).

2.1.1. Quantization

Quantization aims to change the representation of numerical values on digital devices by using a mapping function. Ideally, the number of bits used to represent parameters and observables should be kept at a minimum while preserving the original performance of the NN model on a given task. As a result, the reduced computational footprint fosters efficient implementations of NN models on resource-constrained devices with limited memory bandwidth (Gholami, et al., 2021). Often NN models are available as GPU implementations with a 32-bit floating-point precision leaving room for improvements by drawing on quantization strategies. The advantages of compression techniques can play a crucial role in reducing carbon footprint and overall, enhancing energy efficiency.

Different approaches to quantization have been considered in the past (for a complete review see (Gholami, et al., 2021)). In a so-called **post-training quantization**, calibration data is used to estimate the clipping ranges and scaling factors of the mapping function for a pre-trained NN model (Gholami, et al., 2021). Based on this knowledge, the model is quantized afterwards. For some applications, **quantization-aware training** is required to recover the original performance (Nagel, et al., 2021; Gholami, et al., 2021). Here, a trained model is quantized by drawing on the mapping function and subsequently re-trained based on a small set of data samples.

Different levels of granularity for the quantization have been used, since a shared quantization scheme for the whole NN model might lead to a severe reduction in accuracy. According to (Gholami, et al., 2021), the clipping range for the weights has been determined layer-wise (Krishnamoorthi, 2018), group-wise (Shen, et al., 2020), and channel-wise (Zhang, Yang, Ye, & Hua, 2018). Moreover, some layers might be more sensitive to the quantization and therefore demand a higher precision in a so-called **mixed-precision** quantization. Despite the associated benefits, finding suitable representations is a challenging task (Gholami, et al., 2021) that has been addressed by methods like reinforcement learning (Wang, Liu, Lin, Lin, & Han, 2019) and neural architecture search (Wu, et al., 2019).

The optimization target for any quantization approach on the overall NN performance already highlights the dependency on the actual model as well as data used for training and inference. We will discuss the implications thereof in the roadblock section.

2.1.2. Pruning

Parameter pruning is another strategy to reduce the computational footprint of deep learning systems. Instead of reducing the numerical precision of individual parameters, pruning purports to remove specific parameters that are not needed for performing a given task, thereby sparsifying the network architecture. Combined with suitable hardware that can exploit such parameter sparsity, pruning presents a promising way of further reducing neural networks' memory and computational footprint for edge applications.

We distinguish between post-hoc pruning after training and pruning at initialization. One of the most common post-hoc approaches is magnitude-based pruning, whereby small weights are removed from the network following training. Magnitude-based pruning is often accompanied by weight regularization during training and is usually followed by additional rounds of fine-tuning in which the pruned weights are now masked out of the computational graph.

While post-hoc pruning is an easily implementable form that commonly reduces the parameter count by 95% or more with little impact on task performance, it still requires training a dense network from scratch. The method, therefore, does not benefit from the computational savings during network training, which can be a limiting factor when training on neuromorphic hardware substrates on which parameter memory is limited.

Pruning at initialization has moved increasingly into the focus (Bellec, Kappel, Maass, & Legenstein, 2017) motivated by the recently formulated Lottery Ticket Hypothesis (Frankle & Carbin, The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, 2018), which posits that most dense networks contain a sparse subnetwork that can be trained from scratch

to high task performance. However, despite tremendous efforts (Tanaka, Kunin, Yamins, & Ganguli, 2020; Lee, Ajanthan, & Torr, 2019; Liu & Zenke, 2020), most pruning approaches fail to reach the full potential of lottery tickets acquired by training densely connected networks (Frankle, Dziugaite, Roy, & Carbin, 2021). Further, most pruning approaches result in massive parameter savings in the dense layers of convolutional neural networks. In contrast, most of the computational cost resides in the convolutional layers in such architectures. These limitations thus warrant a closer look into how to improve pruning methods intelligently and specifically reduce the computational cost in convolutional layers, e.g., by pruning filters rather than individual weights, which is one of the goals of the project.

2.2. Dynamic neural networks

Typical approaches to reducing the processing complexity of deep learning models include model compression and response approximation. They aim at reducing the model size by using sparsity, adding collaborative layers (Lee J. L.-H., 2021) or designing tiny architectures from scratch (Liberis, Dudziak, & Lane, 2021). Even though they are memory-efficient, they still trade off accuracy to reduce the processing latency since they rely on a "one-size-fits-all" approach that processes all inputs identically.

Recently, dynamic neural networks (DynNNs) were introduced to make the processing complexity at the inference stage input-dependent. The idea behind DynNNs is borrowed from biological NNs which adapt the neural pathways to the stimulus to speed up decision-making as inputs are not equally complex and thus do not often require the network's full capacity. This dynamicity leads to reducing the number of operations and, by extension, inference time as well as energy consumption.

The most straightforward implementation of dynamic neural networks (DynNNs) is through Early Exit (Scardapane, Scarpiniti, Baccarelli, & Uncini, 2020). For instance, in a classification task, it mounts small internal classifiers onto the backbone to make quick decisions for easy inputs, without resorting to the full-fledged network. A response is returned if the internal classifier is sufficiently confident; otherwise, the example is passed on to subsequent layers.

Many other ways exist to incorporate dynamic structures in (deep) neural networks. In addition to the early-exit approach described above, other studies made input dependence possible through **1) attention mechanisms** which allow a focus on the most important parts of the input (Hu, Shen, & Sun, 2018); **2) gating functions** that remove the least salient components (e.g. channels of feature maps)(Gao, Zhao, Dudziak, Mullins, & Xu, 2018); **3) runtime parameter adaptation** that aims at adaptively generating or updating the architecture's intrinsic characteristics (e.g., network width or depth) given the input's features (Xia, Yin, Dai, & Jha, 2021); **4) dynamic activation functions** that activate neurons according to the relevance of the input stimulus thus increasing the representation power of models (Chen, et al., 2020); **5) mixture-of-experts ensemble** that selects the best expert when the type of operations differs from one input to another (Shazeer, et al., 2017). For a review of DynNNs see (Han, et al., 2021).

In conclusion, by conditioning on inputs, a considerable amount of computation can be saved; even when hardware or application constraints change over time, tuneable parameters assist in controlling the accuracy-speed trade-off accordingly. Furthermore, DynNNs are complementary

to other approaches to accelerate neural networks (pruning, quantization) since the dynamic decision blocks are designed to be off-the-shelf items.

2.3. Spiking neural networks

Spiking Neural Networks (SNNs) are networks of spiking neurons inspired by biology. Spiking neurons emit action potentials, so-called spikes, which are binary events that are localized in time. Unlike units in artificial neural networks (ANNs), spiking neurons are intrinsically stateful and thus useful for temporal processing. Moreover, the brain's almost exclusive reliance on spiking neurons is a testimony to their theoretical abilities in terms of power efficiency. The human brain consumes, on average, 20-30W. This energy efficiency is a key motivation behind considering SNNs within the scope of CONVOLVE.

While supervised learning combined with end-to-end, gradient-based training remains the gold standard for most deep learning applications, gradient-based methods are not directly applicable to SNNs due to their binary all-or-nothing spiking mechanism reflecting the biological reality of action potentials. To overcome this limitation, SNNs can either be created by training and then *converting* a conventional ANN into an SNN or by directly training the SNN using surrogate gradient methods. However, it is implausible that a biological network uses either of these approaches so other training mechanisms should be possible.

2.3.1. ANN-to-SNN conversion

The derivation of a spiking neural network (SNN) from a trained artificial neural network (ANN) is an established but niche area of research in the neural network community. As noted above, if SNNs existed inside a mature framework for computation and learning then the conversion process would be unnecessary. The lack of established, general purpose training methods has been a barrier to *ab initio* creation of SNNs until recently (see Section 2.3.2).

ANN-to-SNN conversion has been an attractive area and relatively low-risk approach for applying SNNs to applications. Incorporating a conversion step to the end of a typical machine learning workflow is desirable as it allows the energy and effort embodied in trained ANNs to be utilised instead of having to start afresh. Early work in this area was done by (Diehl, et al., 2015) and normalisation of spike rates to optimise use of the representational space available was introduced by (Rueckauer, Lungu, Hu, Pfeiffer, & Liu, 2017). More work has been carried out that minimises discrepancies between SNN and ANN results (Sengupta, Ye, Wang, Liu, & Roy, 2019; Deng & Gu, 2021). Li provides a description of the sources of noise and used this as rationale to add negative spikes to further reduce the error (Li, Ma, & Furber, 2022).

The appeal of the conversion of ANNs to SNNs comes from the success of ANNs (including deep neural networks and transformer networks) in a range of applications. On the assumption that an SNN version of an existing ANN may offer some advantage in terms of energy efficiency or implementation efficiency, the lure of tapping into the existing library of successful ANNs to create even more efficient SNNs is a powerful motivator, but this vision comes with caveats and pitfalls.

Existing conversion methodologies assume that:

1. There is a one-to-one mapping from each neuron in the ANN to a single neuron in the SNN.
2. Connectivity between neurons and the parameters of each synapse are preserved during conversion.
3. The output activation of each neuron is mapped to a spike rate of the resulting spiking neuron.
4. Neurons do not possess some of more biologically realistic characteristics seen in other SNNs, such as the leaky membrane voltage and shaped post-synaptic currents. The neuron acts as a simple accumulator into which weighted activations are summed.

Consequently, one can identify limitations of the resulting SNNs:

1. Any potential gains in processing efficiency due to the temporal nature of processing in the SNN is unexplored since the source ANN does not model time in any meaningful way and the SNN is limited by the compute model of the ANN.
2. Since each neuron's output is rate coded, spike timing conveys no information and so a potential source of information is lost (related to point 1, above)
3. In the implementation of the two networks, the basic operation in the ANN (a multiply-accumulate) is replaced by one or more additions (one per spike processed) in the SNN. This results in multiple memory accesses in the SNN where only one was required in the ANN. The energy efficiency of rate-coded SNNs is therefore far from certain.

In CONVOLVE, we have already implemented the Li conversion method on the SpiNNaker platform. This method uses very low spike rates to reduce energy consumption at a cost of reduced output precision. One aspect of the training in the ANN domain is to mitigate the performance accuracy impact that this reduction in precision would normally entail. The results are interesting, but do not offer the two orders of magnitude energy savings envisaged for CONVOLVE.

Conventional methods for ANN-to-SNN conversion use the idea of a one-to-one mapping between ANN neurons and SNN neurons. This is a straightforward approach but is the worst of both worlds when it comes to conceptually bridging between ANNs and biological neural networks; in almost all implementations, rate-coded SNNs are less energy efficient than ANN models implemented on GPUs but we do not see the associated benefits in fault tolerance to neuron death and the relatively low latency of inference that is seen in biology. Directly trained SNNs sidestep the issue of activation value encoding by concentrating on optimising for the function of the network — e.g. getting the correct inference — rather than optimising the function of the components — e.g. working to get the spike rate of a specific neuron to match one in an ANN but doesn't allow the reuse and leverage of previously trained models.

In later work on CONVOLVE, alternative representations of ANN activity in SNNs will be explored including temporal and population codes that take advantage of specific timing information and larger populations of neurons. This will raise the possibility of new conversion and training methods and has potential for more efficient use of a SNN's resources, reducing energy costs.

2.3.2. Training of spiking neural networks with surrogate gradient methods

Surrogate gradients constitute a continuous relaxation of the other otherwise ill-conditioned gradients of an SNN, which is either zero or infinite in most cases. Although surrogate gradients can be thought of as a gradient of a smoothed-out loss function, their computation does not usually require defining such a loss. Instead, they can be computed directly by inducing reasonable approximations into the gradient computation of an otherwise non-differentiable loss (Neftci, Mostafa, & Zenke, 2019). In practice, surrogate gradients are introduced into auto-differentiation frameworks such as Tensorflow or Pytorch, but defining a custom surrogate derivative for non-differentiable activation functions such as the Heaviside function, which has derivative zero everywhere except at threshold where its derivate is not defined. Although a rigorous theory of surrogate gradient learning in SNNs is still missing, empirically, the method has proved incredibly successful, paving the way for the widespread use of SNNs. A distinct advantage of direct training with surrogate gradients is that the method is compatible with arbitrary input coding. The practitioner does not have to specify whether the network uses a timing or a rate code internally. Surrogate gradients efficiently find a suitable strategy on-the-fly through end-to-end training (Figure 2).

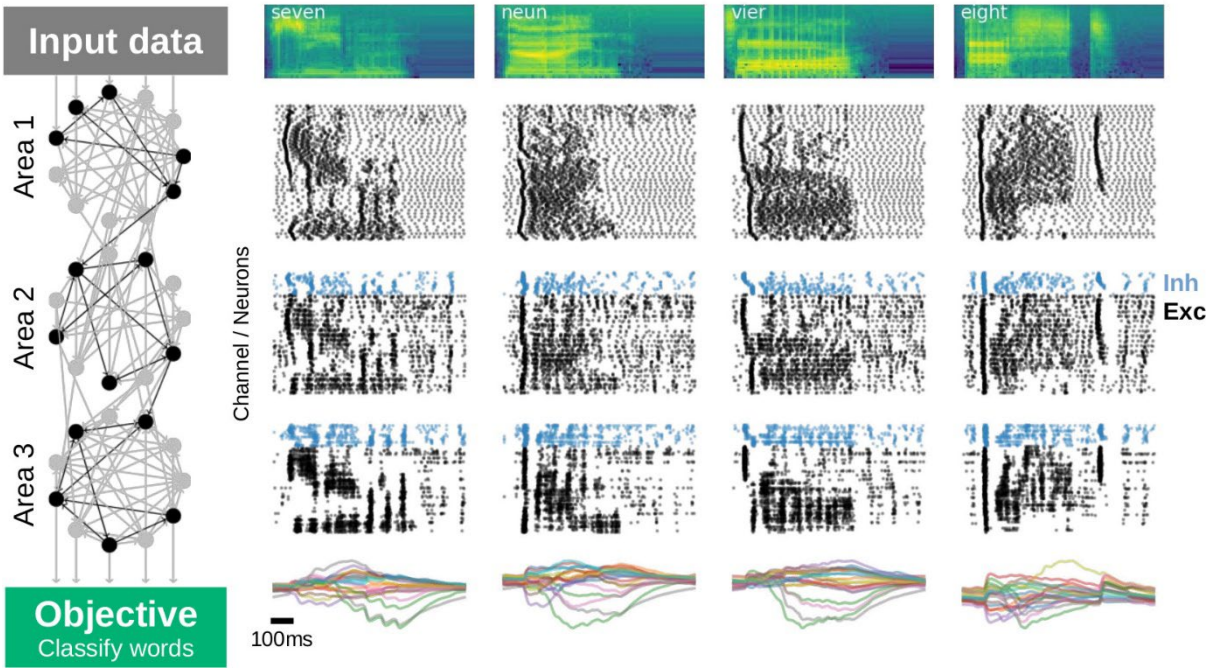

FIGURE 2: EXAMPLE OF AN SNN TRAINED WITH SURROGATE GRADIENTS ON THE HEIDELBERG DIGITS SPEECH RECOGNITION TASK. LEFT: SCHEMATIC OF THE NETWORK ARCHITECTURE. TOP SPECTROGRAMS OF AUDIO INPUT. MIDDLE SPIKE RASTER PLOTS IN THE DIFFERENT LAYERS/AREAS. BOTTOM LINEAR READOUT NEURON ACTIVITY CORRESPONDING TO THE TWENTY CLASSES OF THE TASK.

2.4. Learning strategies

Improving computational efficiency of deep-learning systems does not only hinge on the inference phase but also on data and energy efficiency during training. Large deep neural networks have grown incredibly data and power-hungry as evidenced by recent deep learning models. While neuromorphic substrates may offer a remedy to the energy efficiency, training of

neuromorphic substrates often comes with its own set of challenges largely related to the extensive memory demands of backpropagation through time over extensive time horizons (Zenke & Neftci., Brain-Inspired Learning on Neuromorphic Substrates, 2021). To address these challenges, CONVOLVE will focus on developing learning strategies for SNNs that reduce the need for backpropagation and can operate without labelled data and self-supervised learning approaches are essential. Further we will investigate hardware-friendly continual learning algorithms that reduce the memory requirements of current continual learning approaches.

One approach to continual learning is to take inspiration from the multiple learning processes at work in biological synapses, each operating over different timescales. Spike-timing dependent plasticity (STDP) is the most well-known mechanism for learning in neocortex but other processes, from the probability of neurotransmitter vesicle release to the widening of the synaptic junction and ultimately to the construction of entirely new synapses, are involved in the storage of new memory engrams. A potentially valuable line of research is to assume that these different mechanisms function as substrates for memory at different timescales from short term, through medium to long term memory. Through this framework, new but temporary knowledge can be acquired, used and then discarded without modifying knowledge stored in longer-term memory, protecting it from slow erosion. This *memory hierarchy* approach, managed at the level of individual synapses is an area of learning in spiking networks that we can explore in CONVOLVE.

3. Roadblocks

After having highlighted the state of the art in Section 2, we discuss the roadblocks that prevent a direct application of the discussed methods within the scope of CONVOLVE to ultimately reach the objectives presented in Section 1.

3.1. Model compression

Both model compression techniques, quantization as well as pruning, need to be evaluated in light of the considered use-case. In this process, existing methods could be refined, and novel approaches can be developed.

3.1.1. Quantization

The approaches presented in Section 2.1.1 depend on the considered model and data. Hence, these methods require being evaluated in the context of the use-cases and the proposed models. In this process, suitable quantization targets will be defined. Here, the quantization strategies potentially balance the predictive performance of a given model and the associated computational footprint. This assessment has to be done by re-evaluating the requirements of each use-case individually. Also, the availability of the data set impacts the selection of a suitable quantization strategy.

3.1.2. Pruning

Most pruning strategies target pruning after training and use either weight magnitude or gradient-flow as pruning criterion, but not task-relevance. This calls for the development of pruning-at-initialization strategies that close the gap with post-hoc-pruning strategies by extending methods on gradient-flow and neural-tangent-kernel based pruning and developing new parameter importance-based methods.

3.2. Dynamic neural networks

In the literature, dynamism has been implemented in deep neural networks using a variety of approaches, depending on the type of data (e.g., text, image, video), tasks at hand and deployment resources. This dynamism is usually achieved by adding small neural networks to the model's backbone in order to implement adaptive decision-making (e.g., deciding whether to exit at a given stage). Furthermore, the presence of discrete variables (e.g., the number of exits) or sparse matrices (e.g., dynamic sparse convolutions) in some of these models impedes the training process. The challenge for us is to develop new methods of training these architectures and controlling the extra computations during inference while ensuring a tangible benefit from their use.

3.3. Spiking neural networks

SNNs need to be developed for the target applications. In that scope, either conversion or surrogate gradient methods can be applied.

3.3.1. Conversion of artificial neural networks to spiking neural networks

For extant conversion approaches there is a defined relationship between an encoded activation value and a pattern of spikes, for instance in simple rate coding the rate of firing of a spiking neuron corresponds to a number, the activation value, seen in the parent ANN. Rate coding, though inefficient in most hardware implementations because of the expense of repeated memory accesses, does have some potential advantages that should be borne in mind when considering other coding schemes, notably its noise tolerance that comes from temporal redundancy. Further work is necessary to go beyond this one-to-one mapping of ANN neuron to SNN neuron.

3.3.2. Training of spiking neural networks with surrogate gradient methods

A major roadblock for surrogate gradient learning in SNNs is the extensive memory requirements of Back-Propagation Through Time for large networks simulated over an extended number of timesteps. To mitigate this problem, we will investigate online (forward-in-time) learning strategies as well as reducing the need for back-propagation, by exploring local learning rules derived from self-supervised learning principles, such as Latent Predictive Learning (Halvagal & Zenke, 2022).

3.4. Continual learning strategies

Current continual learning algorithms usually require extra memory for each model parameter during training which increase the memory demand of deep learning models substantially. We will investigate the viability neuronal-centric continual learning approaches with a reduced memory footprint.

3.5. Learning strategies

When training both spiking and conventional neural networks on temporal data tasks, backpropagation through time (BPTT) is the algorithm of choice. One downside of BPTT is that it requires storing the entire activation history of all units in the network during the whole input sequence to evaluate the gradient. This requirement has three distinct disadvantages in the streaming data setting:

1. It is memory intensive for tasks requiring long time horizons and a satisfactory temporal resolution. These memory demands challenge their utility even on conventional hardware, especially neuromorphic systems, where memory is often a limiting factor.
2. It requires truncating backpropagation over finite temporal windows whereby the time horizon must be chosen ad-hoc, often significantly affecting final learning performance.
3. The algorithm induces backward locking whereby weights can only be updated after completely processing a finite truncation window.

While forward-in-time strategies such as real-time recurrent learning do not have these limitations, their computational complexity is prohibitively higher than BPTT. Although approximations such as SuperSpike (Zenke & Ganguli, SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks, 2018) or e-Prop (Bellec, et al., 2020) exist to mitigate this problem in SNNs, their approximate character often results in reduced task performance. Together these limitations pose severe roadblocks for efficiently training neural networks in the streaming data setting and call for novel learning strategies that either constitute effective albeit better approximations to RTRL or dispense with the need for temporal backpropagation altogether, for instance, by introducing negative-sample-free greedy self-supervised loss functions.

The major roadblock to the memory hierarchy framework outlined earlier is that it is a risky, blue-sky line of research. Although based on knowledge of real synapses, there is no established body of theory underpinning it. Furthermore, individual synapses contain a significant amount of state – information about multiple synaptic units that represent short-, medium- and long-term memory – that goes against the trend to reduce the data per synapse. Thus, even if the memory mechanism is successful in its function, it is far from clear that it represents an efficient way of organising information for recall and computation.

4. Research

Since each industry-relevant use-case comes with its own requirements and solutions, the applicable methods used to optimize their computational footprint might differ. In the following, the use-cases are briefly described followed by their targeted algorithmic principles to approach ultra-low power neural network processing within WP4. This section is closed by a discussion on data representations for SNNs and learning strategies.

4.1. Use-case 1: Deep noise suppression

The objective of Deep Noise Suppression (DNS) or Speech Enhancement is to enhance the quality and intelligibility of speech signals in both the uplink (Tx) and downlink (Rx) by minimizing background noise. This is a difficult undertaking because of the diverse and intricate acoustic scenarios that can occur in real-life situations, including the proximity of an unwanted speaker (known as a "jammer") to the primary user's microphone. This is an active area of research and the topic of the high-profile Intel Neuromorphic DNS Challenge (Timchek, et al., 2023).

4.1.1. Proposed solution

We aim to reduce the amount of computation required to perform speech denoising by introducing early exit layers into the model. To explore this, we start by comparing the nsNet2 model (Braun, Gamper, Reddy, & Tashev, 2021; Braun & Tashev, Data augmentation and loss normalization for deep noise suppression, 2020) used in the DNS Challenge with its dynamic version featuring the same layer topology. We will then explore how other models used in denoising, like CRUSE, can be converted into a dynamic neural network version.

Another area of possible gains in memory storage efficiency (and hence area and power savings) is the use of stochastic computation during learning. In this approach the precision of synaptic weights stored in memory is reduced, reducing the memory footprint of the weight matrix. During computation an intermediate state retains high bit precision, but when it comes time to write back the state to memory (usually the new synaptic weight) the precision is again reduced by rounding using a dithering methodology that requires a random number for each rounding. This can have an impact on the accuracy of the final network, but this impact is typically small compared to the savings in memory. We will investigate stochastic computation with the GNA applications within WP4 and develop hardware within WP2 to support this methodology.

A third area of investigation is the development of a framework for representation and computation using sparsely active, temporally-coded SNNs, incorporating continual learning to adapt to the changing background noise. Such a framework would be based around fixed weight coding, also called N-of-M coding, where the activity in a group of neurons has a homeostatic mechanism to maintain approximately constant activity. The representative power of such codes have been demonstrated by a number of groups. This work will also allow an investigation into combining short-, medium- and long-term storage into a single synapse, a more biologically plausible mechanism for the management of stored knowledge that allows more flexibility in the assimilation of information than is used in other state of the art SNNs.

Models: For this task, a number of baseline models have been identified, which are state-of-the-art in the field of deep noise suppression. These are also suitable candidates for further optimization efforts, such as those described above. These models include nsNet2, CRN, CRUSE, and DEMUCS.

nsNet2 is a well-established model following a rather straightforward architecture based on fully-connected layers and gated recurrent units (GRUs). The model accepts a real-valued log-magnitude spectrogram as input and computes a gain mask which is then applied to the input spectrogram in order to suppress noise.

CRN, CRUSE, and DEMUCS are convolutional models based on the UNet architecture, which has proved effective in the field of image segmentation. They feature encoder and decoder subnetworks performing down-sampling and up-sampling, respectively. Each subnet is comprised of stack of convolutional blocks, and analogous feature maps on encoder and decoder blocks are connected by skip layers. Furthermore, these architectures feature a sequence modeling block in the bottleneck, composed of recurrent units such as GRU or LSTM. While CRN and CRUSE behave similarly to nsNet2 in that they accept spectrograms and generate suppression gain masks, DEMUCS operates directly on the time-domain signal (i.e. the waveform) using 1D-dilated convolution, and as such, can generate audio directly.

On top of these aforementioned models, which have already been used successfully in the field of noise suppression, we are interested in exploring the viability of alternative architectures that have not been applied to this task, such at UNet++, lightweight transformer (based on depth-wise separable convolution or long-short range attention), or variations thereof.

As presented in section 2.2, although dynamic neural networks are designed to ensure computational complexity reduction, they can still benefit from static compression techniques for more compactness. In this use-case, 8-bit weight quantization methods and post-training pruning could help the proposed models to achieve higher levels of computational savings with minimal loss in performance. Therefore, model deployment on resource-constrained microcontrollers can be tested through potential experiments.

Data: For training, the data from the DNS2020 Challenge (Reddy, et al., 2020) can be used. It is a synthetic dataset composed of several other datasets (clean speech, noise, RIRs) that are processed using a script provided by Microsoft to generate an arbitrary number of noisy-clean training examples. For evaluation purposes, the DNS2020 dataset features a test set that could be used. However, since most of that data is somewhat challenging and not representative of general real-world scenarios, it may be necessary to extend the evaluation set with "cleaner" samples.

Evaluation: Evaluation will be performed by comparing the predicted speech signals with the corresponding clean ones from the aforementioned DNS2020 test set. The metrics to be considered are those related to speech quality and intelligibility such as PESQ, STOI, or ViSQOL (Michael Chinen, 2020). Alternatively, or additionally, non-intrusive speech quality predictors could be used, such as DNSMOS (Reddy, et al., 2020) or NISQA.

When it comes to dynamic models, during inference, a suitable exit point must be chosen according to an exit strategy. To disentangle the performances of denoising and exit-strategy models, two evaluation schemes are suggested:

- Oracle: the exit strategy is based on a label corresponding to the SNR, log-spectral distortion, or similar quality score. This can be used to determine if the model will manage to preserve computational resources when facing easy samples. Due to the dependency on input labels, this method is not representative of real-world usage.
- End-to-end: the exit strategy is based on an estimated parameter, which is supposed to correlate with the perceptual quality (or SNR etc.) of the denoised sample. Ideally, this value would be computed using a very shallow sub-network feeding on the generated mask. The predictor sub-network must be trained accordingly. This evaluation method lifts the dependency on labels obtained intrusively and is therefore representative of real-world scenarios.

4.1.2. Architecture of how the research done in the WP fits within WP

As stated before, *dynamism* refers to the ability of the system to adapt its computational resources and power consumption in response to varying workloads, while *quantization* involves reducing the precision of numerical values used in computations to save power. *Binarization* serves as the extreme end of quantization – only making two states available to convey information in neural networks. By employing these techniques and developing effective strategies for switching between different dynamic models, the system can achieve a balance between power consumption and performance, ultimately leading to improved power efficiency.

SNNs can be trained using a variety of learning strategies, including supervised, unsupervised, and reinforcement learning. These learning strategies can be adapted to the dynamic nature of SNNs, for example, by using STDP rules to adjust the strength of connections between neurons based on the timing of spikes. In terms of dynamism, SNNs can be further optimized by using dynamic input and output encodings that adapt to changing input patterns, and by introducing dynamic sparsity that allows the network to selectively activate only the necessary neurons for a given task, thus potentially reducing energy consumption.

4.1.3. Dependencies with other WPs

We mainly see dependencies with WP5 (Compiler), WP6 (SoC generation) and WP2 (Accelerator blocks). For WP2 and WP5, one main point of discussion would be how *profiling* of the network can be achieved both on compiler and model level, identifying memory/processing bottlenecks and based on that, optimized memory-tiling. Furthermore, tensor-operations might be optimized/ parallelized in hardware to speed up inference processing. WP6, on the other hand, could provide meaningful insights in optimized processing/memory design which centres around the question: Given a certain amount of required memory (RAM), memory bandwidth and processing power, how can you optimally place them on a SoC regarding power and latency constraints?

4.1.4. Use-case requirement addressed by the WP

This use-case addresses real-time, "low"-latency, ultra-low power processing and smart sensors.

4.1.5. Contribution to the demos

We will provide a real-time implementation of the noise suppression models developed hitherto, running on an embedded hardware target. The current main candidate for the latter is the NXP I.MX 8M Plus microprocessor. In the case of dynamic models, we will provide the possibility of manually changing the exit point as well as using an automatic early exiting system developed as part of the next use-case.

4.2. Use-case 2: Speech quality prediction

Speech quality is an extremely important metric to consider when designing speech processing solutions. However, performing user listening tests during the development stage would be extremely unpractical and strenuous.

While there exist several full-reference metrics based on conventional DSP or perceptual models (PESQ, POLQA, ViSQOL, etc) that correlate nicely with a human-attributed mean opinion scores (MOS), these are often computationally expensive and would require a clean reference speech signal that is unattainable in real-world scenarios. Therefore, there is an interest in developing speech quality prediction (SQP) models that can approximate user preferences and run efficiently at minimal computational cost. Furthermore, a very-efficient SQP model can be used as loss function for training of DNS models, or even executed side-by-side at runtime, to determine the optimal exit stage of dynamic DNS models.

4.2.1. Proposed solution

The proposed solution is to develop network architectures for unintrusive SQP that minimize their computational footprint and optimizes them for always-on applications. This will be done in close collaboration between GNA, FMI, and MAN.

Existing SQP models are often based on convolutional or recurrent network architectures. We will follow a two-pronged approach whereby we will modify existing models and develop new SNN models. Specifically, we will start with existing baseline SQP models (e.g., DNSMOS or QualityNet, see below) and modify them to use binary activation functions, and reduced bit-width weights. Further, we will explore the impact of weight pruning to reduce their memory demand. We will train the performance-optimized models on data from the DNS Challenge (see below) using surrogate gradients to overcome differentiability issues. Finally, we will measure their correlation with MOS predicted by the full baseline model. Similarly, we will develop SNN models that solve the same task and compare their prediction accuracy. In all cases, we will quantify the required number of multiply-adds, while specifying binary multiplication and required memory as

appropriate. The latter measurements will be used to in conjunction with experts on compilers and hardware accelerators to identify optimal accuracy-efficiency trade-offs.

Models: Most SQP models comprise the same building blocks: a spectral transform, a framewise feature extractor, a time-dependency model, and a pooling mechanism. An overview of the overall architecture can be found in (Mittag, 2022)(sections 3.2 to 3.6). Two common such models are DNSMOS (Reddy, et al., 2020) and QualityNet (Fu, Tsao, Hwang, & Wang, 2018). DNSMOS has been developed by Microsoft and features a simple architecture based on 2D convolution and dense layers. It has been trained to predict MOS collected from listening tests conducted using the ITU-T P.808 subjective testing framework, where users are asked to rate variants of a given speech sample that have been processed through a variety of methods. QualityNet features a perceptually motivated architecture comprising a bidirectional-LSTM block followed by fully-connected layers predicting framewise scores. These are then averaged together to provide a global score. The model is trained using PESQ as a target, but the authors suggest that it could be trained on other metrics or MOS. DNSMOS assumes a constant-length spectrogram excerpt as input, while QualityNet features the presence of bidirectional-LSTMs; this makes both models unsuited for real-time inference, which should also be addressed by this work package.

Data: The task in this project can be formulated as a supervised learning problem where we attempt to predict a MOS or surrogate metric thereof from a speech audio sample. The training data for this project is the same as for the noise suppression use-case and can be found here. Each audio sample is 30 seconds long. The 6k dataset comes with precomputed full-reference labels that could be used as targets, namely PESQ, STOI, SI-SDR, and SI-SNR. These labels are computed for overlapping segments (10 seconds long with 5 seconds overlap, resulting in 5 sets of labels for each speech sample). Note that other labels could be computed and used, including those derived from larger and more powerful SQP models, such as ViSQOL, NISQA, or DNSMOS.

Evaluation: The reported evaluation metrics for most speech quality prediction works consist in some measure of statistical correlation with the target MOS or intrusive speech quality measure. For instance, the literature on DNSMOS shows the Pearson Correlation Coefficient (PCC) and Spearman Rank Correlation Coefficient (SRCC) between human ratings and estimated metrics, including DNSMOS:

	PESQ	**SDR**	**POLQA**	**DNSMOS (M_0)**
PCC	0.78	0.23	0.79	0.93
SRCC	0.82	0.25	0.84	0.94

For QualityNet, the reported metrics are PCC (here called Linear Correlation Coefficient - LCC), SRCC, and Mean Squared Error (MSE) between PESQ and the model's estimate:

	MSE	LCC	SRCC
Autoencoder +NN [22]	0.1529	0.8434	0.8675
Quality-Net	**0.1266**	**0.8749**	**0.8807**

On top of metrics of performance, we would be interested in assessing the computational saving afforded by the optimization efforts. Such metric might include inference time (in seconds/milliseconds) from which it will be possible to derive a "real-time factor", number of

operations per inference (FLOPS/MACS), and energy consumption per inference (in nJ/pJ). To collect these metrics, a target hardware platform must be chosen.

4.2.2. Architecture of how the research done in the WP fits within WP

The proposed work will be carried out in collaboration between GNA and FMI with input from UIR and MAN and touches T4.2. At a later project stage, we will work toward a SNN implementation of the SQP network (T4.3) and an integrated approach of an efficient SQP network controlling the dynamism of the DNS network.

4.2.3. Dependencies with other WPs

The final architectural choice depends on the target hardware platform and the compiler suite used. Thus, this WP depends on input from the hardware accelerator (WP2) and compiler levels (WP5) which will be provided during regular consortium meetings.

4.2.4. Use-case requirement addressed by the WP

The use-case (specific) requirements addressed here is to provide high *Speech-quality* for speech-enhancement applications in communication devices.

4.2.5. Contribution to the demos

We will provide a real-time implementation of the speech quality estimation models developed hitherto, running on an embedded hardware target. The current main candidate for the latter is the NXP I.MX 8M Plus microprocessor. As part of the demo, the device should listen to incoming audio and provide an instantaneous measure of SQ, either frame-wise or over short segments.

4.3. Use-case 2: Siren detection and tracking

This use-case focuses on the analysis of acoustic scenes. More specifically, typical traffic scenes are targeted within two distinct sub tasks: First, we aim to detect the presence of emergency vehicles in the acoustic scene based on siren sounds and, second, these sources should be tracked in 2D space over time. The underlying acoustic scenes constitute challenging conditions since the included acoustic signals potentially show a high degree of spatio-temporal overlap and a fast range of different signal-to-noise ratios as well as source speeds and amplitudes need to be tackled in real-time.

For both tasks, the processing can be subdivided into two parts. First, the feature extraction that is centred around the calculation of Mel-Spectrograms based on (multi-channel) raw audio signals in the current implementation and second, the actual neural network processing. The latter is done by drawing on recurrent neural architectures like GRUs or LSTMs. Their natural

temporal processing capabilities promise high efficiency for this specific use-case. Both processing parts contribute to the overall performance and efficiency.

While the detection network is trained on a publicly available dataset (Asif, et al., 2022), data augmentation strategies are used to generate an audio data set mimicking road conditions providing detailed spatial information of the underlying sound sources (Damiano & Waterschoot, 2022).

4.3.1. Proposed solution

Within the scope of WP4, we would like to improve the computational footprint of our models by drawing on different approaches. First, we would like to utilize compression methods like quantization and pruning. For quantization, we target a 8-bit integer representation to boost the efficiency of the existing solutions that feature a 32-bit floating-point precision in the current implementation. Depending on the results, binarization of the activation functions as the extreme case can be explored. It is noteworthy, that this approach represents an intermediate step when it comes to the investigation of SNNs. Here, surrogate gradient methods can be applied in potential post-quantization optimization steps to restore performance. By this, spatio-temporal sparsity could be anchored in the activation of the proposed solution. In addition, pruning methods promise to further reduce the computational footprint. Depending on the hardware support, structured as well as unstructured pruning will be considered. Since an always-on mode is intended for the tracking network, the discussed methods promise to foster highly efficient solutions by exploiting sparsity at various levels.

As highlighted in D1.1, feature extraction is a critical part art of this use-case and hence a target for optimization approaches. The calculation of spectrograms comes with additional overhead and constrains the latency depending on its parametrization. In this context, recent advances in the field of raw audio processing could be visited (Gong & Poellabauer, 2018; Dai, Dai, Qu, Li, & Das, 2017). Again, their applicability should be investigated in respect of supported hardware acceleration to find an optimal solution in the context of the CONVOLVE project.

Additionally, dynamism could be exploited to improve the efficiency of the overall proposed solution. For the feature extraction model, recurrent connections from the core network could be used to dynamically adapt the model's parametrization depending on the current input through dynamic updates of the hidden states. Moreover, since sirens constitute relatively sparse events in time, an always-on operation of the tracking network (that is trained exclusively to track emergency vehicles) is undesirable. In this context, not only does the size of the network matter for deployment but also does the targeted latency as the detection network is intended to provide prediction with a rate of 10 Hz, while the tracking network could output spatial information at higher rates depending on the distance and velocity of the sound source. Therefore, we rely on the resource-friendly detection network to trigger the heavy tracking network whenever an emergency vehicle is detected. This could be made possible through dynamically skipping (or, jumping) unimportant sequences (i.e., absence of target signal patterns). Furthermore, since emergency vehicle sirens become more strident over distance, it is more efficient to allow the system (i.e., car) to adjust its processing capacities and data feed over time. Thus, input adaptive sampling (e.g., adaptive timestep) could also be a promising research avenue to reduce the computational and energy footprints of the proposed neural architectures.

4.3.2. Architecture of how the research done in the WP fits within WP

Sparsity and dynamic neural networks are topic of T4.2. Here, static, and dynamic approaches to reduce the overall energy requirements will be considered. This encompasses quantization as well as pruning strategies that have the potential to anchor sparsity deeply into the design, both in terms of connectivity as well as activation. Moreover, the instantiation of dynamic neural networks for the use-case is also part of T4.2 in which different modes of dynamisms are investigated.

As already stated above, binarization of activation functions represents an intermediate step to the development of SNNs for the target application. The latter is subject of T4.3. Further steps like surrogate gradient-based training based on the results achieved with static methods also correspond to this task.

4.3.3. Dependencies with other WPs

Since the algorithmic developments target the full design stack, a close interaction with all other WPs is intended. In this context, any sparsity consideration – at the level of connectivity or activity – must keep the targeted hardware accelerators in mind. The same holds true for the quantization. Hence, the visited approaches closely rely on the input of WP2 to fully draw on the benefits of the proposed algorithmic principles. Further, a close interaction with WP5 and WP6 is targeted in terms of profiling the proposed solutions as well as the interaction of individual model components like feature extraction and neural network processing. Since any extension of the use-case and/or model comes with additional security risks, WP3 is also involved.

4.3.4. Use-case requirement addressed by the WP

The use-case of acoustic scene analysis targets real-time and low-latency, ultra-low power neural network processing. Considerations regarding other types of feature extraction as well as anchoring dynamism into the design could also promote smarter sensors.

4.3.5. Contribution to the demos

The benefits of the aforementioned algorithmic approaches could be highlighted in the demos. Intermediate point-demos can be used to test the real-time applicability of proposed solutions. Here, even implementations on conventional hardware could serve as demonstrator highlighting potential benefits as well as bottlenecks.

4.4. Use-case 3: Video-based traffic analysis

4.4.1. Proposed solution

This use case comprises a video-based sensor solution with embedded AI for real-time traffic analysis. The system is capable of detection and tracking of all traffic participants including vehicles, pedestrians and bicycles. It measures their speed, trajectory and behaviour. The system collects the data in real-time to allow traffic control. The embedded analysis has the following advantages:

- It omits a bottleneck for centralized cloud processing,
- it allows immediate anonymization to guarantee the privacy of the traffic participants, and
- it reduces the bandwidth of communication between the sensor and the traffic controller.

Figure 3 shows a state-of-the-art edge device with AI processing. This solution consumes typical 25 W and requires special design for heat dissipation. The results of the CONVOLVE project allows a more cost-efficient solution with more intelligence.

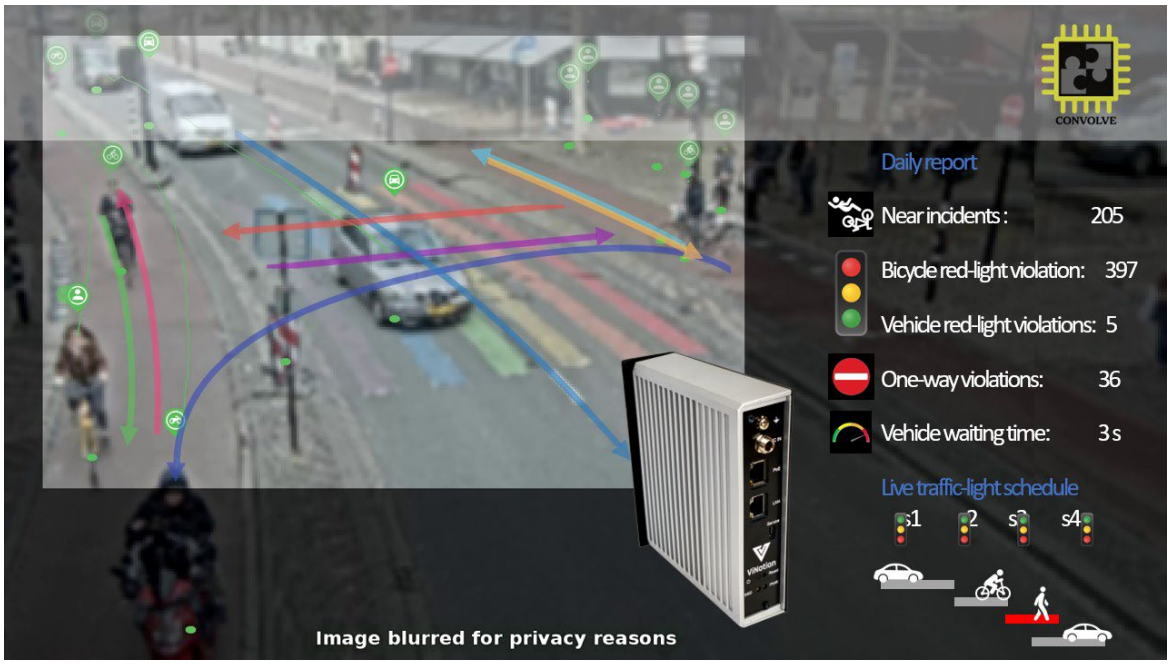

FIGURE 3: STATE-OF-THE-ART EDGE DEVICE WITH AI PROCESSING.

The system pipeline consists of the following main processing blocks: video decoding, object detection and classification, object tracking, post processing. Only object detection and classification are currently implemented by an ANN, however we are open to alternative implementations where object detection, classification and tracking are combined in a single ANN or executed in parallel ANNs.

4.4.2. Architecture of how the research done in the WP fits within WP

Optimization of ANN (Quantization / pruning) and DNN research is of interest, since both can be utilized to decrease the memory and computational requirements of the model and thereby improve the real-time behaviour and lower the latency of the proposed system.

This mainly concerns the current ANN object detection and tracking algorithm. However, optimization of the complete pipeline is of interest for the final solution. Hence, this also includes video decoding, pre-processing and post-processing.

4.4.3. Dependencies with other WPs

Conform Figure 17 of the project plan (Figure 4), WP1 defines the use-case. The output of this WP4 will be the input for WP5 and WP7. We require application software development in a high-level description like PyTorch, so that the compiler (WP5) performs an optimized mapping onto the the ULP hardware in WP7.

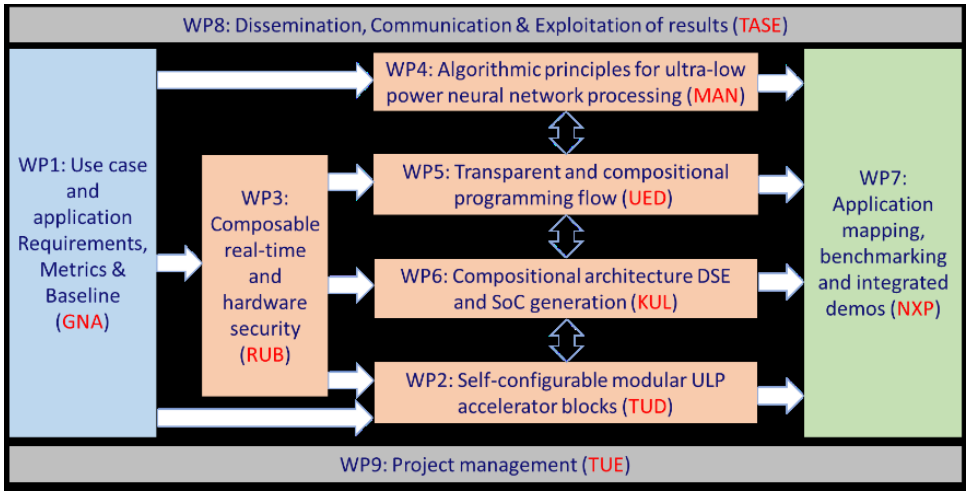

FIGURE 4: SCHEMATIC ILLUSTRATION OF THE PROJECT PLAN.

4.4.4. Use-case requirement addressed by the WP

The following high-level requirements apply for the smart traffic sensor solution with embedded AI:

- Flexibility for large variety of video sensor types
  - Frame-rates (25 fps)
  - Resolutions (HD video), 8 – 16 bits, number of colour channels
- Allow a diverse set of processing blocks in a pipeline
  - Video coding, colour conversion, detection, tracking, projective transformations, etc.
  - Heterogenous processing with highly efficient inter-communication
- Future proof
  - New algorithms including AI are advancing in a rapid pace

- Low power per operation
  - For small product design without dissipation concerns
  - Allowing complex applications
- Tooling for rapid application development in software

4.4.5. Contribution to the demos

We provide the currently available implementation of the smart camera solution with embedded AI on traditional hardware like the Jetson Xavier NX for benchmarking of the project results.

We provide (open)source application software as a baseline of the use case to allow the consortium partners to design, test and validate the ULP hardware for AI processing. We provide support for mapping of the use case application software onto the new hardware and help in the benchmarking of the project results.

4.5. Use-case 4: On-board computer vision

The on-board processing of Earth Observation satellite images reduces congestion in the communications channel while allowing rapid decision-making and preventing security breaches on confidential imagery or scenes. The deployment of modern architectures for classic computer vision tasks are prohibitive as used on ground environment due the reduced resources available on-board.

4.5.1. Proposed solution

As stated before, different techniques can be used to reduce the computational and memory footprint of the models. The over-parameterization of general-purpose deep models let enough room for improvement with quantization and sparsification/pruning techniques.

As CNN is the predominant architecture in computer vision tasks, it would be interesting to explore filter-wise pruning or channel-wise pruning approaches in the convolutional layers as they concentrate much of the computational cost of the model.

Regarding quantization, in most scenarios, pretrained models are fine-tuned for a specific task and sometimes the full datasets needed for training the model from scratch are not available at time of deployment. Therefore, for these scenarios, post-quantization training done in the quantization-aware training approach would be only possible with the reduced dataset used for fine-tuning. From this fine-tuning dataset, a small set of calibration data could be extracted for choosing the quantization parameters in a post-training quantization approach.

In the same line, although dynamic neural networks look promising, we may not be able to train from scratch new or modified backbones due to the lack of base datasets.

Although current used architectures are CNN, SNN could be a nice option to reduce the power budget. Direct conversion of CNNs to SNNs seems to be the simplest way to reuse knowledge

acquired on long CNN trainings without requiring training the models from scratch. In the case of a huge performance degradation, fine-tuning datasets could be used for retraining the resulting SNN.

4.5.2. Architecture of how the research done in the WP fits within WP

Both pruning and quantization strategies are part of T4.2. The implementation of SNNs for the target applications falls within the scope of T4.3.

4.5.3. Dependencies with other WPs

The requirements of this use case, as in the rest, start from WP1, to which the WPs 2 to 6 are directly related. Moreover, in the specific case of satellite vision, the security requirements are fundamental (WP3) and the HW architecture blocks (WP2) have restrictions, so there could be a strong interaction with both work packages.

4.5.4. Use-case requirement addressed by the WP

The main requirement addressed by the WP is low power neural network processing.

4.5.5. Contribution to the demos

We will provide the current implementation, trained for a specific computer vision task that can be used for comparing and validating the results. Also, the fine-tuning dataset will be shared as fine-tuning will probably be required to improve the results of some of the strategies.

4.6. Data representations for Spiking Neural Networks

As mentioned previously (Section 2.3.1) existing artificial SNNs tend to be a hybrid of ANN understanding mapped onto biologically-inspired components (Eliasmith, 2013). This does not exploit the strengths of either parent: numeric values are translated into spike *rates*, requiring multiple spikes per value and incurring significant latency. In a biological system (i.e. an animal) macroscopic behaviour alone demonstrates that computation must happen within a few — possibly *one* per serially-connected neuron — spike time.

Information encoding must therefore be done differently. Spikes are digital *events* so, other than the presence or absence of a spike, information must be coded in the temporal *order* of different spike arrivals, almost certainly combined with the timing *differences* involved. It is plausible, even likely, that this coding is used and it gives access to a vast code space. It is also reasonable to map such computations onto biological processes where each incoming stimulus causes a state change, the effects of which wane over time; simple models can demonstrate the detection of both 'coincidence' — within a determined window — and the order of arrival of stimuli using no

more than the expected input 'weights' and an exponential (or similar) decay of recorded state over time.

If this principle can be extracted and exploited, it opens up many possibilities in SNNs. Of primary interest to CONVOLVE would be the reduction in spike communications with a consequent reduction in power. Set against this may be an added complexity of state holding *within* the neurons; it will be part of the research to discover the practicability of making this storage adequately in digital logic at a reasonable cost.

A secondary motivation is to accommodate continuous learning. Unlike ANNs, SNNs exploit passing time, which makes backpropagation in time into time travel; assuming biological systems have not mastered this they must train continuously both to learn and to adapt to changing circumstances. This must use feedback and, at the neuron level this is, presumably, some 'memory' of recent-past behaviour which can be reinforced or diminished. This could use a similar mechanism to the 'firing' system, albeit over a longer time-scale.

Within WP4 we are seeking to produce and simplify such models, retaining their utility whilst developing a mechanism appropriate for a digital implementation for WP2.

4.6.1. Proposed solution

Figure 5 shows the excitation and firing outcome of just two incoming spikes to a synthetic neuron with the same interval but in reversed order. The calculations involved are simply addition and exponential decay, with a threshold comparison to determine if the neuron produces an output as a result of each incoming stimulus. All these functions are biologically credible.

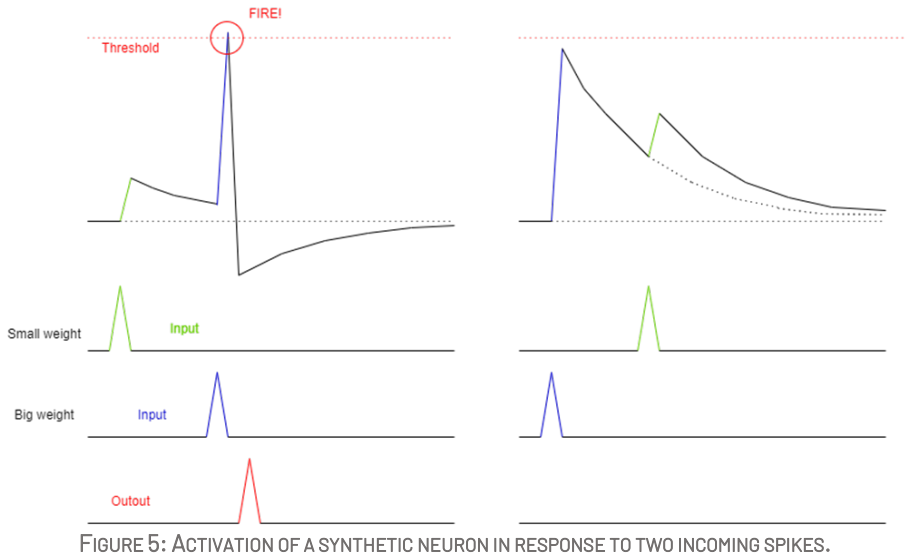

FIGURE 5: ACTIVATION OF A SYNTHETIC NEURON IN RESPONSE TO TWO INCOMING SPIKES.

In a digital implementation the expensive calculation is the exponential. This can be approximated with a cheap calculation by multiplying values with a constant at intervals or a single but more expensive response to each incoming event: either of these approaches can benefit from additional arithmetic support and it is not clear which might prove cheaper overall.

The former approach could be amenable to some DMA-like accelerator working autonomously on programmed areas of RAM; the latter approach would be a suitable candidate for augmenting the ISA of a RISC-V processor.

In addition to spike firing, maintaining exponentially decaying variable applicable to some learning processes where records of 'recent' history are used with some feedback from the (overall network) outcome to adjust incoming spike weights. This process may amplify the problem since some models suggest that (biologically) 'records' are kept on a pre-synapse basis and there are many more synapses than neurons. Compromises in this space are of research interest but the facility to maintain such variables economically is significantly interesting.

4.6.2. Architecture of how the research done in the WP fits within WP

This SNN work is intended to produce an implementable model of what are presumed to be biological spiking neurons. If this model can be supported successfully it can vastly reduce the communications overhead from current (rate-based) SNNs. Neural 'spikes' already carry their information in their timing, so are individually small; the distribution of spikes in a network is expensive though, so reducing the traffic by an order of magnitude can represent a significant energy saving. The longer-term goal would be to exploit a similar mechanism ——supported by the same hardware accelerator(s)— to facilitate concurrent, continuous learning in the network.

4.6.3. Dependencies with other WPs

The work here is not *directly* attempting to address applications; instead, it is searching for a practical alternative mechanism with the capacity to address such problems. Therefore, there is not a strong input dependency although material from WP1 will be integrated where possible. Due to their implicitly timed behaviour, some of the audio-processing applications are probably the most pertinent. There is a strong output dependency with WP2 since any requirements must be feasible for hardware implementation. Most of this should be satisfied since investigators in the relevant parts of these WPs are already working together.

4.6.4. Use-case requirement addressed by the WP

Use-cases are not addressed directly here but will be used as inputs to guide the work.

4.7. Learning strategies I

In addition to the use-cases above, FMI will develop new continual and online learning algorithms for spiking and non-spiking networks to provide the theoretical basis for future ultra-low-power on-chip learning in T4.4. Specifically, we will focus on three directions. First, we will improve our theoretical understanding of surrogate gradient techniques for training SNNs by relating them to theoretically grounded theories of learning stochastic networks. Second, we will work toward reducing the memory footprint of current continual learning techniques that keep track of per-parameter importance (Zenke, Poole, & Ganguli, Continual Learning Through Synaptic

Intelligence, 2017) by exploring a neuronal-centric approach, and finally, we will extend our existing work of online learning through temporal prediction with local learning rules (Halvagal & Zenke, 2022) to the temporal tasks and spiking networks.

4.8. Learning strategies II – Memory Hierarchies

Manchester will develop theory for the *memory hierarchy* framework described earlier, embedding short-, medium- and long-term storage into each synaptic connection in an SNN. The aim of this is to reduce the data flow between neurons and to manage the assimilation of new knowledge into an existing network with minimal disruption, facilitating continual learning.

4.8.1. Proposed solution

Investigate learning rules for a feedforward network with sets of sparse synaptic connections each of which contains elements of short-, medium- and long-term memory. Devise rules for the capture data held in one set of memory sub-units to be recoded and stored through modifications of the next level of the memory hierarchy. Establish metrics to quantify the performance of such a network, including memory capacity, rate of assimilation, distortion of previously stored knowledge, etc.

4.8.2. Architecture of how the research done in the WP fits within WP

This work will provide a substrate for energy efficient neural computation, which is a key aim of WP4. Conceptually similar to the conventional notion of the cache, the philosophy of this work is to keep data close to where it is used most, in this case distributed within each synapse. This is intended to reduce the energy cost of moving data round. It is possible that the number of neurons required to implement a given function may be reduced by this methodology, leading to further savings in area and power. However, this claim is speculative at this point in the research.

4.8.3. Dependencies with other WPs

No dependences.

4.8.4. Use-case requirement addressed by the WP

This work addresses learning issues in audio processing use-cases and has application in image processing use-cases.

4.8.5. Contribution to the demos

This work will be too preliminary at the time that the demos are implemented. The results will appear towards the end of the project (M33).

5. Plans

The table below (Table 1) summarizes final and intermediate subprojects. More specifically, all collaborations are listed as well as the involved partners and their topic.

Application	Partners	Topics	Links to other Applications	Section in Document
Image-based Object detection and classification	AXE, VIN/TUE,	ANN / DynNN		4.4
Object segmentation	VIN	Merging ANN's		4.4
Deep speech denoising	GNA, UIR	DynNN	Speech quality prediction might be a submodule	4.1
Deep speech denoising	GNA, MAN	Quantisation in SNN		4.1
Deep speech denoising	GNA, FMI	Quantisation, pruning, knowledge distillation, sparsity		4.1
Speech quality prediction	GNA, FMI	Binarization/ sparsity		4.2
Speech quality prediction	GNA, MAN	SNN		4.2
Siren detection and tracking	BOS, FMI	Quantization, pruning		4.3
Siren detection and tracking	BOS, FMI	Binarization/ sparsity		4.3
Siren detection and tracking	BOS, UIR	DynNN		4.3
Siren detection and tracking	BOS, FMI, MAN	SNN		4.3
Deep speech denoising	GNA, MAN	Continual learning in SNNs	Links to all apps that support continual learning	4.8, 4.1
DynNN early exit predictor	UIR, MAN	Accelerate DynNNs by early exit detection	All applications using Dynamic NNs	

Object detection	TAS, FMI	Quantization/Pruning		4.5
Object detection	TAS, FMI, MAN	SNN		4.5

TABLE 1 SUBPROJECTS IN WP4.

Bibliography

Liu, T., & Zenke, F. (2020). Finding Trainable Sparse Networks through Neural Tangent Transfer. *International Conference on Machine Learning*, 6336–47.

Frankle, J., & Carbin, M. (2018). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *ArXiv*, 1803.03635 [Cs].

Halvagal, M. S., & Zenke, F. (2022). The Combination of Hebbian and Predictive Plasticity Learns Invariant Object Representations in Deep Sensory Networks. *bioRxiv*, 2022.03.17.484712.

Lee, J. L.-H. (2021). *Resource-efficient deep learning: A survey on model-, arithmetic-, and implementation-level techniques*. arXiv.

Michael Chinen, F. S. (2020). *ViSQOL v3: An Open Source Production Ready Objective Speech and Audio Metric*. arXiv.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv preprint arXiv:2103.13630*.

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).

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications, 11*(1).

Bellec, G., Kappel, D., Maass, W., & Legenstein, R. (2017). Deep Rewiring: Training Very Sparse Deep Networks. *arXiv preprint arXiv:1711.05136*.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., . . . Keutzer, K. (2019). *FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Reddy, C. K., Gopal, V., Cutler, R., Beyrami, E., Cheng, R., Dubey, H., . . . Johannes, G. (2020). The INTERSPEECH 2020 Deep Noise Suppression Challenge: Datasets, Subjective Testing Framework, and Challenge Results. *arXiv preprint arXiv:2005.13981*.

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020). Dynamic relu. *Computer Vision--ECCV 2020: 16th European Conference, Proceedings, XIX*(16), 23-28.

Dai, W., Dai, C., Qu, S., Li, J., & Das, S. (2017). *Very deep convolutional neural networks for raw waveforms*. 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP).

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

Frankle, J., Dziugaite, G. K., Roy, D. M., & Carbin, M. (2021). Pruning Neural Networks at Initialization: Why Are We Missing the Mark? *arXiv preprint arXiv:2009.08576*.

Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., & Xu, C.-z. (2018). Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*.

Gong, Y., & Poellabauer, C. (2018). *How do deep convolutional neural networks learn from raw audio waveforms?* openreview.

Hu, J., Shen, L., & Sun, G. (2018). *Squeeze-and-excitation networks.* In Proceedings of the IEEE conference on computer vision and pattern recognition.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). *HAQ: Hardware-Aware Automated Quantization With Mixed Precision.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Lee, N., Ajanthan, T., & Torr, P. H. (2019). SNIP: Single-Shot Network Pruning Based on Connection Sensitivity. *arXiv preprint arXiv:1810.02340*.

Liberis, E., Dudziak, Ł., & Lane, N. D. (2021). *μNAS: Constrained Neural Architecture Search for Microcontrollers.* Proceedings of the 1st Workshop on Machine Learning and Systems.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., Van Baalen, M., & Blankevoort, T. (2021). A White Paper on Neural Network Quantization. *arXiv preprint arXiv:2106.08295*.

Mittag, G. (2022). *Deep Learning Based Speech Quality Prediction.* Springer.

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine 36 (6), 36*(6), 51–63.

Neill, J. O. (2020). An Overview of Neural Network Compression. *arXiv preprint arXiv:2006.03669*.

Scardapane, S., Scarpiniti, M., Baccarelli, E., & Uncini, A. (2020). Why should we add early exits to neural networks? *Cognitive Computation, 12*(5), 954-966.

Braun, S., Gamper, H., Reddy, C. K., & Tashev, I. (2021). *Towards efficient models for real-time deep noise suppression.* ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Braun, S., & Tashev, I. (2020). Data augmentation and loss normalization for deep noise suppression. *Speech and Computer: 22nd International Conference, SPECOM, Proceedings, 22*, 7-9.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., . . . Keutzer, K. (2020). Q-BERT: Hessian based ultra low precision quantization of bert. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(5), 8815-8821.

Gupta, S., Agrawal, A., Gopalakrishnan, K., & Narayanan, P. (2015). *Deep Learning with Limited Numerical Precision.* Proceedings of the 32nd International Conference on Machine Learning.

Fu, S.-W., Tsao, Y., Hwang, H.-T., & Wang, H.-M. (2018). Quality-Net: An End-to-End Non-intrusive Speech Quality Assessment Model based on BLSTM. *arXiv preprint arXiv:1808.05344*.

Tanaka, H., Kunin, D., Yamins, D. L., & Ganguli, S. (2020). Pruning Neural Networks without Any Data by Iteratively Conserving Synaptic Flow. *Advances in neural information processing systems, 33*, 6377-6389.

Xia, W., Yin, H., Dai, X., & Jha, N. K. (2021). Fully dynamic inference with deep neural networks. *IEEE Transactions on Emerging Topics in Computing, 10*(2), 962-972.

Zenke, F., & Ganguli, S. (2018). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation, 30*(6), 1514-1541.

Zenke, F., & Neftci., E. O. (2021). Brain-Inspired Learning on Neuromorphic Substrates. *Proceedings of the IEEE, 109*(5), 935-950.

Zenke, F., Poole, B., & Ganguli, S. (2017). *Continual Learning Through Synaptic Intelligence.* International conference on machine learning.

Zhang, D., Yang, J., Ye, D., & Hua, G. (2018). *LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks.* Proceedings of the European Conference on Computer Vision (ECCV).

Timchek, J., Shrestha, S. B., Rubin, D. B., Kupryjanow, A., Orchard, G., Pindor, L., . . . Davies, M. (2023). The Intel Neuromorphic DNS Challenge. *arXiv preprint arXiv:2303.09503*.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., & Pfeiffer, M. (2015). *Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing.* 2015 International joint conference on neural networks (IJCNN).

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., & Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience, 11*.

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience, 13*, 95.

Deng, S., & Gu, S. (2021). Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*.

Li, C., Ma, L., & Furber, S. B. (2022). Quantization Framework for Fast Spiking Neural Networks. *Frontiers in Neuroscience, 16*, 1055.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., & Wang, Y. (2021). Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906, 44*(11).

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition.* Oxford University Press.