Sensor Fusion Neural Networks for Gesture Recognition on Low-power Edge Devices

Gabor Balazs[1,2,†], Mateusz Chmurski[1,3,†], Walter Stechele[2] and Mariusz Zubert[3]
[1]*Infineon Technologies AG, Am Campeon 1-15, Neubiberg, Germany*
[2]*Technical University of Munich, Munich, Germany*
[3]*Lodz University of Technology, Lodz, Poland*

Keywords: Sensor Fusion, Gesture Recognition, Convolutional Neural Network, Radar, Time of Flight.

Abstract: The goal of hand gesture recognition based on time-of-flight and radar sensors is to enhance the human-machine interface, while taking care of privacy issues of camera sensors. Additionally, the system needs to be deployable on low-power edge devices for applicability in serial-produced vehicles. Recent advances show the capabilities of deep neural networks for gesture classification but they are still limited to high performance hardware. Embedded neural network accelerators are constrained in memory and supported operations. These limitations form an architectural design problem that is addressed in this work. Novel gesture classification networks are optimized for embedded deployment. The new approaches perform equally compared to high-performance neural networks with 3D convolutions, but need only 8.9% of the memory. These lightweight network architectures allow deployment on constrained embedded accelerator devices, thus enhancing human-machine interfaces.

1 INTRODUCTION

1.1 Background

In the upcoming cockpits of cars, the communication with the machine will be different than known today. A trend towards a system without touch sensors is seen, for example, with the voice activated controls of the phone or multimedia. Touch-less controls are also beneficial for safe driving, as the driver does not need to split his concentration between the road and multimedia controls (Young et al., 2003). The driver can swiftly adjust any multimedia settings with his voice or the help of hand gestures without driving blindly.

In accordance with the steadily increasing awareness and need of privacy and personal data protection, there is an interest in a system for driver monitoring without camera sensors. Radar and depth sensors produce data that denies easy identification of individuals, thus are a good choice for this task.

The different nature of data coming from time-of-flight (ToF) and radar sensors leads to different pre-processing schemata and difficulties in combining the information with traditional methods. The superior classification accuracy and generalization capabilities of networks come to the cost of immense computational effort for the processor (Alom et al., 2018).

To meet the requirements of mass-produced vehicles, all computations have to be performed on automotive microcontrollers and embedded convolutional neural network (CNN) accelerators. This fact limits the applicability of heavy, deep networks and favors lightweight models, optimized for embedded inference. The main aspects of this work can be summarized as follows:

- Introduction of a lightweight, yet robust, multimodal system for hand gesture recognition. The system relies on combined radar and ToF sensory data only. It offers variants that meet the different model requirements of state-of-the-art embedded accelerators.
- Ensurance of privacy by not relying on camera sensors, thus addressing the growing concerns about internet of things devices spying on the private life of customers.
- A system design for edge devices with limited compute capabilities. Accordingly, we propose four CNN models with few parameters and standard neural network operations only.

†Both authors contributed equally.

Balazs, G., Chmurski, M., Stechele, W. and Zubert, M.
Sensor Fusion Neural Networks for Gesture Recognition on Low-power Edge Devices.
DOI: 10.5220/0010234101410150
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 141-150
ISBN: 978-989-758-484-8

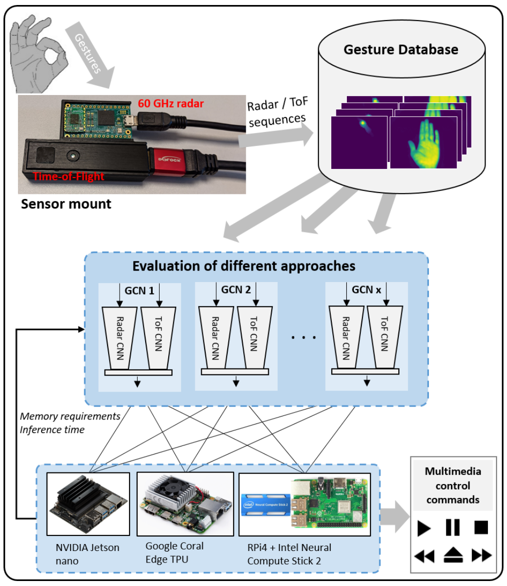

Figure 1: System overview: Gesture classification networks for multi-modal input data coming from radar and ToF sensors. The resulting networks are deployed on various embedded hardware accelerators.

1.2 Related Work

After the initial success of neural networks for static image processing (Girshick et al., 2013) (Krizhevsky et al., 2012), architectures were introduced to also deal with spatio-temporal data such as video sequences. A way of interpreting it is to use the different video frames as feature channels of one input tensor to 2D CNNs (Feichtenhofer et al., 2016).

The authors of (Donahue et al., 2014) use a combination of convolutional layers and long-short term memory (LSTM) cells in order to classify spatio-temporal data. A CNN is applied frame-wise to extract features, which then are passed to the LSTM for classification over time. A similar approach used 3D CNNs to acquire local spatio-temporal features, which are then fed into an LSTM to calculate global, long-term features of the video (Molchanov et al., 2016) (Tran et al., 2014).

These advances opened the door to activity classification and gesture classification frameworks. Most hand gesture sensing systems rely on optical sensors, especially on vision data from camera sensors. A very powerful approach was presented in (Köpüklü et al., 2018). An optical based multi-input system is proposed by (Abavisani et al., 2018), where camera data is fused with its optical flow for improved classification results. Authors of (Concha et al., 2018) make use of optical flow information, too, but feed it along with frame-wise features into a three-stream 2D CNN for action recognition.

While vision based approaches show good results, research also focuses on robust multi-modal systems and systems not relying on cameras. Authors of (Molchanov et al., 2015) present a system for driver's hand-gesture recognition based on permutations of camera, radar and depth sensors. An early fusion is applied within the input layer of the classification CNN by stacking the different sensor modalities in the feature dimension. A robust gesture classification system is introduced in (Hazra and Santra, 2019), where the authors use a self-attention neural network with LSTM cells for range-Doppler map classifications.

Whereas more and more applications focus on mobile hardware platforms rather than on cloud solutions, strict embedded requirements like model size are not investigated by many works . Authors of (Ceolini et al., 2019) propose a sensor fusion network for hand gesture classification using electromyography and vision data, specially designed for mobile usage. The model uses a combination of CNN and support vector machine parts, resulting in a model size of 144 MB. Authors of (Wang et al., 2016) solely rely on a 60 GHz frequency-modulated continuous-wave (FMCW) radar for hand gesture recognition on a wearable. Whereas it is designed for a low-power device, the proposed architecture has a model size of 689 MB and consumes 265 MB of GPU memory. An extensive analysis of efficient neural network design is presented in (Köpüklü et al., 2020), where a two-staged network is used for online gesture classification. With the NVIDIA Jetson TX2 as the inference hardware, the network is based on three-dimensional CNNs, which are supported by this GPU.

Besides the strict memory constraints, not all of the currently available low-power CNN accelerators support widely used operations like LSTM. Thus, many approaches are not feasible for edge deployment on these constrained devices (Donahue et al., 2014) (Molchanov et al., 2016) (Tran et al., 2014) (Hazra and Santra, 2019) (Naguri and Bunescu, 2017) (Wang et al., 2016)(Chai et al., 2016) (Köpüklü et al., 2020). Consequently, the lack of lightweight, privacy-aware networks based on constrained operations is addressed in this work. The proposed method is compared to (Molchanov et al., 2015), as their analysis also includes the fusion of radar and ToF data and focuses on offline classification.

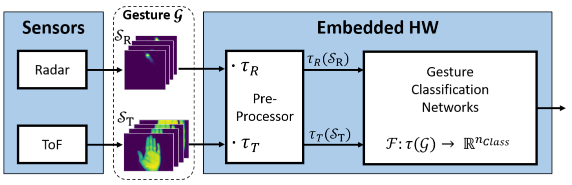

Figure 2: Deployment on embedded HW accelerators.

2 SYSTEM DESIGN

The system is planned to avoid the usage of camera sensors in order to ensure privacy of the users. With this goal in mind, radar and time of flight (ToF) sensors are chosen.

Thus, no sensible camera data is available for possibly malicious use, e.g. technologies based on person identification. An additional advantage of this sensor setup is that the system performance is not correlated to environmental conditions such as poor lighting or reflections of the bright sun.

The system consists of a data pre-processor that prepares the input streams, followed by the gesture classification network *GCN*. The network has the structure of a classification CNN with individual input branches for each sensor modality and a late fusion in the fully connected layers.

2.1 Dataset Description

In this section, we describe the dataset, its gathering, and pre-processing steps that are needed for neural network processing.

Data Gathering. The data was gathered indoors and inside the car with the desired mounting position in vicinity of the gear selection lever. With regards to train an online-classification system, a constant stream of data is recorded as a raw base for further processing. Two sensors are fixed to a custom, 3D-printed mount. They monitor the hand gestures in the proximity of the gear selector lever, pointing up towards the rear mirror (see Fig. 1).

The gestures are expected to be performed close to the sensors in a range of $r = [0\text{m}, 0.3\text{m}]$. A Time-of-Flight sensor[1] is configured to scan this distance and it delivers a three-dimensional point cloud. This point cloud is projected into a two-dimensional image plane, where the depth is denoted by the grayscale intensity. This image spans a field of view of 62° in azimuth and 45° in elevation with a resolution of 224×171 pixels.

[1]pmd CamBoard pico flexx.

The radar sensor[2] uses a FMCW radar. One radar frame is composed of 32 chirps ($N_c = 32$, $T_c = 0.8\text{ms}$) and each chirp consists of 64 samples. The radar is configured to measure 10 frames per second ($T_f = 0.1\text{s}$). The resulting maximum velocity

$$v_{max} = \frac{\lambda}{4T_c} = \frac{5\text{mm}}{3.2\text{ms}} = 1.56 \text{ m/s} \qquad (1)$$

allows the system to detect gestures, even when carried out quickly. The frequency bandwidth is 5 GHz (58 GHz – 63 GHz), resulting in a range resolution of $\Delta r = \frac{c}{2B} = 0.03m$. The velocity resolution is

$$\Delta v = \frac{v_{max}}{\frac{N_c}{2}} = \frac{\lambda}{2N_cT_c} = 9.75 \text{ cm/s}. \qquad (2)$$

Both sensors of the system are synchronized to a rate of 10 Hz.

From the constant stream of information, gestures are extracted for building the training dataset. A thresholding method is employed to the ToF and radar streams to detect active gestures and mark the start and end frames of a gesture. The gesture is saved to a hard disk in raw format along with the corresponding label. The classes form a set of intuitive gestures for multimedia control.

In order to have a dataset with high variance, multiple subjects were recorded performing the hand movements above the sensors. Gestures performed by left and right hands were recorded in order to learn control inputs of both, the driver and the passenger. In total the dataset consists of 2,225 gestures with gesture lengths up to 29 frames, describing 9 gesture classes (Tab. 1, Fig. 3).

Data Pre-processing. The ToF images are processed with a Wiener filter to smooth the noisy depth values. Furthermore, the depth values beyond the desired ranges are filtered out.

The radar data is processed in multiple steps: First, the chirps are brought to zero mean by subtracting the mean value of a chirp from each of the samples. Then, the range is computed with a first stage fast Fourier transform (FFT) over the range samples with an FFT size of 128, from which the positive half is used. The range-Doppler images (RDI) are computed with the second stage FFT with an FFT size of 64, resulting in an RDI dimension of 64×64. Before each of the FFTs, the signal is multiplied with a Hann window function. Subsequently, the absolute values of the RDI are thresholded with an ordered statistic CFAR (OS-CFAR) in order to maximize the signal-to-noise ratio. OS-CFAR was chosen because of better multi-target capabilities in comparison to cell averaging CFAR (CA-CFAR). This is expected to have

[2]Infineon BGT60TR13C.

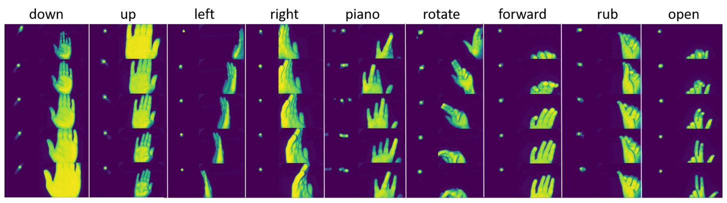

Figure 3: Visualization of the gesture classes in the dataset. Only a few frames per class are taken for clarity, as the classes vary in length.

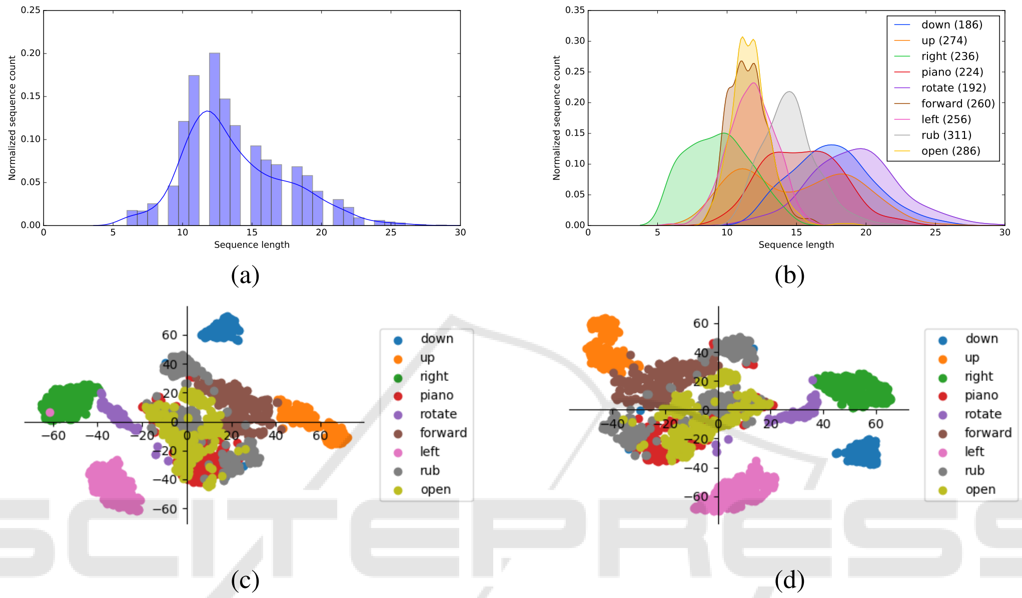

Figure 4: (a) The dataset consists of gestures of various lengths. During training, the gesture lengths are padded to a uniform length in order to have constant length input tensors. (b) Per-class distribution of gesture lengths. The number in brackets denotes the amount of gestures in this class. (c) and (d) show the t-SNE representation of the down-scaled ToF and radar sequences of the dataset. The image dimensions are 32×32 and the sequences are padded to a uniform length. Clearly, the gestures for navigation (*left*, *right*, *up*, *down*) form distinct clusters, whereas the other gesture classes are more difficult to distinguish. Both sensor modalities provide cluster information about the classes so that the networks can benefit from both sensor types.

a positive impact on the classification of the gestures with individual fingers moving (compare RDIs of *piano* gesture in Fig. 3).

Data Format. After data pre-processing, the contents of one training sample are the two input sequences and the corresponding label. A raw radar sequence is a volume $\mathcal{S}_R \in \mathbb{R}^{t \times x \times y \times f}$, where $t \geq 1$ denotes the timesteps in this sequence. Each timestep stores a RDI with $x \times y$ as the range and Doppler dimensions, and f as the number of feature channels. Here, $f = 1$, as only the intensity of the RDI is used. A raw ToF sequence $\mathcal{S}_T \in \mathbb{R}^{t \times x \times y \times f}$ is a similar volume to $\mathcal{S}_R$, but $x \times y$ denote the pixel dimensions of the ToF sensor output and there might be a different value of t. There is only one feature $f = 1$ for $\mathcal{S}_T$, too, as it describes the distance of a target to the sensor. A transformation τ needs to be performed in order to feed these tensors into the network model. τ depends on the network type and the embedded hardware to be deployed on. Further information about τ is described along the neural network models in 2.2.

In the following a gesture $\mathcal{G}$ denotes spatio-temporal data in form of a tuple $\mathcal{G} = (\mathcal{S}_R, \mathcal{S}_T)$. The entries of $\mathcal{G}$ are a radar and a ToF sequence belonging to one specific class.

Temporal Adjustments. The gestures vary in length so that it is not possible to directly feed them into a network of constant input size (Fig. 4a, b). In order to adjust $\mathcal{S}_R$ and $\mathcal{S}_T$ to the same length t_0, the sequences are zero-padded, before and after the original sequence. The fix length t_0 is chosen to be the maximum gesture length of the dataset. Within these t_0 frames, the gesture is put into a random position.

Table 1: Dataset of hand gestures recorded with radar and ToF sensors.

GESTURE	*down*	*up*	*left*	*right*	*forward*	*rotate*	*open*	*piano*	*rub*	TOTAL
AMOUNT	186	275	256	235	260	192	286	224	311	2,225

Data Augmentation. In order to increase robustness of the system, following data augmentation techniques are applied. They are utilized randomly to individual input streams.

Random Shifting of Complete Sequences: ToF sequences $\mathcal{S}_T$ are shifted in both pixel dimensions (x, y) by a random value which can be up to 10% of the respective pixel dimensions ($\pm 0.1x$, $\pm 0.1y$). Radar sequences $\mathcal{S}_R$ are shifted randomly by up to 5% in Doppler, and 10% in range dimension ($\pm 0.05x$, $\pm 0.1y$). The empty space is filled with zeros.

Zeroing Out Regions: Random selection of areas in the images, which are then filled with zeros. The selection can either be the borders of the image for ToF, or patches within the image for both sensor modalities. The border padding simulates different scenarios of gestures that are not completely included within the field of view of the sensor. Random patch zeroing reduces overfitting of the network to certain regions. Patch sizes are from one pixel up to a square of 5×5 pixels.

Adding Constants to the Sequence: To reduce the impact of numerical values, a random integer with a value up to 5% of the maximum pixel value is added to the sequences.

2.2 Gesture Classification Network

The *GCN* are composed of two parts. First, the spatio-temporal data coming from the sensors is processed to an intermediate representation with a transformation τ, followed by an encoder architecture $\mathcal{A}$. After the gestures $\mathcal{G}$ are transformed to the network-specific format by transformation τ, the input tensors are ready to be fed into the network models. Each network has repeated individual encoding cells that consist of a 2D convolution, max-pooling, batch normalization, dropout, and ReLU activation (Fig. 5). Multiple cells are stacked after each other until the processed tensors reach a desired final embedding shape. The embeddings are used for late fusion and classification.

The fusion is done with a fully connected layer that uses the concatenated information of both encoded inputs. n_{class} neurons output the classification result. As a summary, each network $\mathcal{F} = (\tau, \mathcal{A})$ maps the input gesture $\mathcal{G}$ to an n_{class}-dimensional vector ($\mathcal{F} : \mathcal{G} \to \mathbb{R}^{n_{class}}$).

In the following, four network architectures $\mathcal{A}$ with individual transformations τ are described:

Time Distributed, $\mathcal{F}^{TD}$ = (τ^{TD}, $\mathcal{A}^{TD}$). In this network version, each time step is processed with a shared CNN backbone with `TimeDistributed` layers in Keras to retrieve spatial information, followed by temporal integration using LSTM cells. Each input stream has its own n_{LSTM} LSTM cells. For this approach, no transformation is done with τ^{TD} : $\mathbb{R}^{t \times x \times y \times f} \longrightarrow \mathbb{R}^{t \times x \times y \times f}$.

The encoding cells use convolutions with 3×3 filter size, and are wrapped into Keras' `TimeDistributed` layers. Once a spatial dimension (x, y) is below 8, the embedding of the last cell is passed to an LSTM cell. Each input modality is assigned to its individual LSTM unit. Hence, the temporal integration takes part after spatial feature extraction and before the fusion.

3D Conv, $\mathcal{F}^{3D}$ = (τ^{3D}, $\mathcal{A}^{3D}$). Here, 3D convolutions are used directly on the spatio-temporal data of each sensor type. No transformation is needed for this approach, consequently τ^{3D} : $\mathbb{R}^{t \times x \times y \times f} \longrightarrow \mathbb{R}^{t \times x \times y \times f}$.

The encoding cells of the $\mathcal{A}^{3D}$ architecture use 3D convolutions and 3D max pooling. Their filter dimensions are chosen in order to filter separately for spatial and temporal features. For a given spatio-temporal tensor $\mathcal{C} \in \mathbb{R}^{t \times x \times y \times f}$, three-dimensional convolutional filter $f_s \in \mathbb{R}^{1 \times 3 \times 3}$ is applied for spatial, and $f_t \in \mathbb{R}^{3 \times 1 \times 1}$ for temporal filtering. Accordingly, the max pooling sizes are chosen to reduce the respective dimension. Once the spatial dimension (x, y) is below 8, both tensors are flattened to vectors and concatenated for fusion and classification.

Video as Image, $\mathcal{F}^{VI}$ = (τ^{VI}, $\mathcal{A}^{VI}$). Each RDI and ToF image can be reshaped to vectors in order to solve the problem of an additional dimension for the time. Those vectors are stacked to one single image per gesture, which can be processed with a standard network with two-dimensional convolutions. The transformation needed is

$$\tau^{VI} : \mathbb{R}^{t \times x \times y \times f} \longrightarrow \mathbb{R}^{t \times xy \times f}. \quad (3)$$

After the transformation τ^{VI}, the input tensors do not have the 4th dimension. Hence, they are processed with a 2D CNN. As the individual frames in this tensor are vectors, spatial feature extraction is done with 2D convolutions with filter size 1×3 along the vectors dimension. Generally, 2×2 max pooling layers

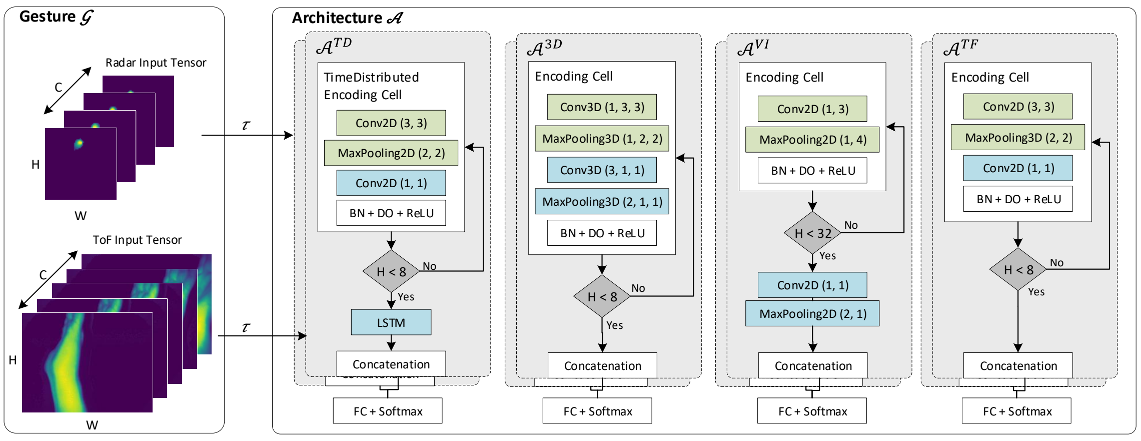

Figure 5: The general architecture of the proposed gesture classification networks *GCN*. The gestures $\mathcal{G}$ from the dataset are fed through the transformation τ to the architectures $\mathcal{A}$. Together they form the network $\mathcal{F} = (\tau, \mathcal{A})$. The individual transformations τ is depicted in Fig. 6. At the end of each encoding cell, batch normalization (BN) and drop out (DO) are applied before ReLU-activation. Green colored nodes denote operations for spatial feature extraction. Blue nodes show the operations that reduce in the time dimension. Both sensor modalities are processed with the same architectural structure. Here, the processing of only one modality is shown for visualization reason. The concatenation fuses the information of both input streams.

reduce the image area by $1/4$. Similarly, the max pooling in this approach reduces by the same amount, but in the single direction of the image vectors by using 1×4 max pooling. These encoding cells are stacked after each other until the image vector dimension z is below 32, resulting in an embedding of shape $\mathbb{R}^{t \times z \times 1}, z \in (0, 32)$.

Time as Feature, $\mathcal{F}^{TF} = (\tau^{TF}, \mathcal{A}^{TF})$. Both inputs have only one feature ($f = 1$), namely the reflectivity in RDI, and distance in ToF images. This allows rearranging the input tensor in a way, so that the time dimension t is placed as the feature channel f of the input $\mathcal{C}$. The transformation

$$\tau^{TF} : \mathbb{R}^{t \times x \times y \times 1} \longrightarrow \mathbb{R}^{x \times y \times t} \tag{4}$$

produces a three-dimensional input tensor $\mathcal{C}^{TF} \in \mathbb{R}^{x \times y \times t}$ to the network.

In contrast to τ^{VI}, where the spatial context of the frames is lost, the transformation τ^{TF} preserves the images, but still reduces the dimensionality to three. The input tensors are processed with encoding cells using 3×3 convolutions until the spatial dimension (x, y) is below 8 for both inputs branches. The embeddings of the tensors are flattened, concatenated and then fed into the fully connected layer for classification.

Each above mentioned approaches $\mathcal{F}$ can process various input tensor sizes, such as the raw input format (RDI: 64×64, ToF: 224×171) or a reduced input size S (RDI: 32×32, ToF: 32×32) and each approach can be tuned with the number of convolutional filters b. The number is

$$b = b_0 + 2^{\delta}, \tag{5}$$

where parameter b_0 is the offset and δ the depth of the encoding cells.

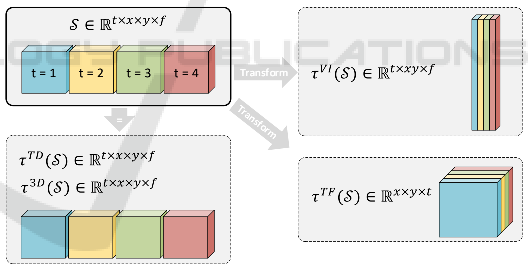
$\mathcal{S} \in \mathbb{R}^{t \times x \times y \times f}$
t = 1
t = 2
t = 3
t = 4
Transform
$\tau^{VI}(\mathcal{S}) \in \mathbb{R}^{t \times xy \times f}$
Transform
$\tau^{TD}(\mathcal{S}) \in \mathbb{R}^{t \times x \times y \times f}$
$\tau^{3D}(\mathcal{S}) \in \mathbb{R}^{t \times x \times y \times f}$
$\tau^{TF}(\mathcal{S}) \in \mathbb{R}^{x \times y \times t}$

Figure 6: The visualization of the two transformations τ^{VI} and τ^{TF}. Both transform 4D data to 3D tensors that can be fed into CNNs without recurrent units. τ^{TD} and τ^{3D} are identity operations.

2.3 Embedded Deployment

With the rising demand for mobile applications, the market for embedded accelerators is emerging (Reuther et al., 2019). Mainly research teams and universities are offering chips in the regime of low-power devices (MIT Eyeriss chip (Chen et al., 2016b), Intel MovidiusX processor (Intel, 2020), Google EdgeTPU (Google, 2020), the DianNao accelerator family (Chen et al., 2016a), NVIDIA Jetson Nano (NVIDIA, 2020) and Rockchip RK3399Pro (Rockchip, 2020)).

Table 2: Performances of the different gesture classification approaches in comparison to † 3DCNN from (Molchanov et al., 2015) given the reduced input S (32 × 32, 32 × 32).

	INPUT S	
APPROACH	ACC	SIZE
$\mathcal{F}_{32}^{TF}$	95.1%	598 KB
$\mathcal{F}_{32}^{3D}$	96.6%	529 KB
$\mathcal{F}_{32}^{VI}$	96.9%	667 KB
$\mathcal{F}_{32}^{TF}$	97.9%	967 KB
3DCNN †	98.6%	10.86 MB

Table 3: Model requirements for the low-power accelerators The unsupported operations for our accelerator versions. 3D: Conv3D, BN: BatchNormalization, DO: Dropout. Basic OPs: Conv2D, ReLU and softmax, pooling and concat. 3DCNN from (Molchanov et al., 2015).

H/W	RNN	3D	BN	DO	BASIC	3DCNN
TPU	-	-	-	-	✓	-
NCS	-	-	✓	✓	✓	-
NANO	✓	✓	✓	✓	✓	✓

Out of this portfolio, only the Intel MovidiusX, used in the Neural Compute Stick 2 (NCS2), the Google EdgeTPU, and the NVIDIA Jetson Nano are commercially available for a competitive price. These three embedded CNN inference accelerators are used for our evaluation.

In order to use the accelerators, the network model constraints of each hardware are to be met. Many state-of-the-art network topologies that claim to aim for edge deployment, lack the applicability to various low-power edge devices. The main disadvantage of using hardware such as the EdgeTPU is the lack of support for 3D convolutional layers. Table 3 shows that not all accelerators support the operations needed to compute the network models introduced in Section 2.2 and models from literature. $\mathcal{A}^{3D}$ uses 3D convolutions, $\mathcal{A}^{TD}$ relies on RNN and TimeDistributed Keras layers. Thus, for the embedded deployment the $\mathcal{A}^{TF}$ and $\mathcal{A}^{VI}$ architectures are used, as they comply with the model requirements.

3 EXPERIMENTAL EVALUATION

3.1 Training Setup

The networks are trained on an NVIDIA TITAN V GPU until convergence, which is achieved after 100 epochs of training with a train/test split of 80%. Each network architecture is evaluated three times and the mean performance values are reported. Weight training is done with an ADAM optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is regulated with a cosine decay with an initial value of 1e-4, a warm-up of 10% of the total iterations and holding the maximum learning rate for 10% of training iterations. ADAM optimizer minimizes the softmax cross-entropy loss of the predicted n_{class} classes. The weights of the convolutional filters decay with an L2-regularizer with a regularization factor of 1e-4 and a dropout of 40% is applied to fully connected layers.

After each epoch, the class-wise accuracy α_s is evaluated on the test set and used for scaling the weights of each class w_s for the next training epoch. The optimizer weights the classes according to

$$w_s = 0.5 + \frac{1 - \alpha_s}{2}. \tag{6}$$

Weights range from 0.75 to 1.25, where higher accuracy leads to lower weights and vice versa.

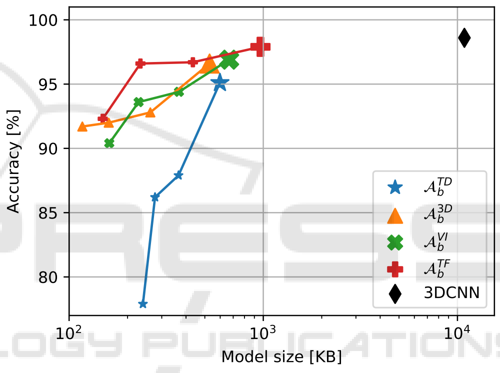

Figure 7: Classification accuracies over the model sizes of the network architectures from Section 2.2, compared to 3DCNN (Molchanov et al., 2015). Each of the investigated architectures is evaluated in four different variants of the parameter $b \in \{0,1,2,3\}$, describing an ever increasing amount of convolutional filters (Eq. 5). The colored Pareto-fronts are described by the best values out of three evaluation runs for each network variant.

3.2 Results and Analysis

All of our approaches are evaluated after the training on the GPU and saved to frozen network model files. These representations are then used to be deployed on the three accelerator devices (Section 2.3) by converting the models to the corresponding intermediate representations of the hardware.

Authors of (Molchanov et al., 2015) propose a network architecture for fusing multi-modal input data for gesture classification. We re-implemented their architecture and evaluated it on our dataset and on the NVIDIA Jetson Nano accelerator. Their model is denoted as 3DCNN in the comparisons. Whereas the

Table 4: Comparative results of proposed *GCN* architectures. Inference times t are averaged over 50 forward passes. M denotes the model sizes. Values are measured for input resolution S $(32 \times 32, 32 \times 32)$. *due to unsupported operations. ‡*uint8*-quantized values. †*float16*-quantized values. [1]3DCNN from (Molchanov et al., 2015).

	GPU					Edge					
						EdgeTPU		NCS		Nano	
Model	M[MB]	Acc	Prec	Rec	F1	t [ms]	M[KB]	t [ms]	M[KB]	t [ms]	M[MB]
$\mathcal{F}_{32}^{VI}$	0.67	96.9%	97.1%	96.9%	97.1%	21.1	169‡	11.4	316†	20.3	0.67
$\mathcal{F}_{32}^{TF}$	0.97	97.9%	98.0%	97.9%	97.9%	38.7	250‡	5.4	459†	16.5	0.97
3DCNN [1]	10.86	98.6%	98.4%	98.3%	98.3%	N/A*	N/A*	N/A*	N/A*	27.4	8.0

proposed network architecture is implemented without quantization of weights, we use a *uint8*-quantized form of the graph for comparison on the CNN accelerator. The model requires 10.86 MB of space in full precision.

Classification Performance. Our four approaches—$\mathcal{F}^{TD}$, $\mathcal{F}^{3D}$, $\mathcal{F}^{VI}$, and $\mathcal{F}^{TF}$ show a good classification performance that is strongly dependent on the architectural parameter b. With the largest number of convolutional filters in our tests ($b = 32$), the best accuracy is achieved by the time-as-a-feature approach 97.9% with a full-precision model size of M_{32}^{TF} = 967 KB. Compared to 3DCNN (Molchanov et al., 2015), our proposed model $\mathcal{F}_{32}^{TF}$ shows similar classification performance: The accuracy is -0.7%, the precision -0.4%, the recall -0.4%, and the F1-score -0.4%.

Decreasing the number of filters b in the models leads to lower classification performance. As seen in Fig. 7, the accuracy is directly correlated to b. The largest drop of accuracy (77.9%–95.1%) is with the model $\mathcal{F}_b^{TD}$, $b \in \{4, 8, 16, 32\}$.

Another impressive architecture is $\mathcal{F}_4^{TF}$, where 92.3% accuracy is achieved with only 149 KB of full precision network parameters. Compared to 3DCNN , this is an improvement of 69 times in the relation between accuracy and model size $\frac{Acc}{Size}$.

Edge Device Deployment. The main advantage of our approach is the application-oriented design, that allows the network to be deployed on multiple embedded accelerators. This is due to the simple model architecture that only relies on supported operations such as 3D-convolutions (Tab. 3). As a consequence of the slim model, fewer weights have to be learned during training, stored and convolved during inference. The full precision model size of our largest proposed approach is only M_{32}^{TF} = 967 KB. The quantization further improves the applicability, because the model size is now compressed to only 250 KB.

Table 5: Comparison of results based on the classification of multi-modal inputs and singular modality, all evaluated for network model $\mathcal{F}_{32}^{TF}$.

	Radar	ToF	Radar+ToF
Acc	68.8%	87.7%	97.9%
Prec	66.9%	96.3%	98.0%
Recall	64.9%	90.4%	97.9%
F1-Score	54.4%	53.4%	97.9%

Ablation Study—Sensor Modalities. In order to underline the importance of a multi-modal system, the classification performance is evaluated with two single modality inputs and compared to the result of the multi-modal input. (Tab. 5). Results show that the classification performance is significantly improved, when using the fused information of both sensors.

Discussion. The benefit of the sensor fusion of ToF and radar can be seen especially for the gesture classes *piano* and *rub*. For small input sizes, the ToF does not deliver much information, as the reduction to 32×32 pixels blurs the depth image quality. In contrast to that, the networks can rely on the micro Doppler signature of the individual fingers moving up and down.

Counter-intuitively, the inference of the larger model $\mathcal{F}^{TF}$ on NCS and Jetson Nano is faster than the execution of the small model $\mathcal{F}^{VI}$ (Tab. 4). The reason is assumed to be the faster execution of standard 3×3 convolutions, instead of the 1×3 filters in the $\mathcal{F}^{VI}$.

4 CONCLUSION

In this work, a privacy-aware gesture recognition system based on a radar and a ToF sensor is proposed.

Current gesture sensing solutions typically rely on camera sensors that may violate privacy standards (Wang et al., 2016) (Ceolini et al., 2019). Moreover, many solutions tend to large model sizes that are infeasible to deploy on resource-constrained embedded accelerators. A further restriction of the embedded hardware is the high number of unsupported operations that limit the applicability of many state-of-the-art networks (Tab. 3).

Our solution employs a lightweight, two-stage algorithm that first transforms the spatio-temporal 4D data of each sensor modality to a 3D tensor with fixed shape. The gesture, now in the form of 3D data, is then classified by a *GCN* with one input branch for each sensor modality. The gesture recognition networks are designed for the deployment on EdgeTPU, NCS2, and Jetson Nano, avoiding unsupported operations, such as recurrent layers or 3D convolutions.

Our largest proposed network achieves equal classification performance as 3DCNN (Molchanov et al., 2015) with only 8.9% of the model size. On the low-end of the model sizes, we propose a gesture classification network that only uses 149 KB of memory while still performing robustly (92.3% accuracy). Thus, our network models can be deployed on resource-constrained embedded accelerators in the performance range of the Google EdgeTPU, Intel NCS2 and NVIDIA Jetson Nano.

In the future, the system is planned to be further optimized in order to be deployed on automotive microcontrollers such as Infineon's AURIX.

REFERENCES

Abavisani, M., Joze, H. R. V., and Patel, V. M. (2018). Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training. *CoRR*, abs/1812.06145.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Hasan, M., Esesn, B. C. V., Awwal, A. A. S., and Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *CoRR*, abs/1803.01164.

Ceolini, E., Taverni, G., Khacef, L., Payvand, M., and Donati, E. (2019). Sensor fusion using emg and vision for hand gesture classification in mobile applications. In *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE.

Chai, X., Liu, Z., Yin, F., Liu, Z., and Chen, X. (2016). Two streams recurrent neural networks for large-scale continuous gesture recognition. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 31–36.

Chen, Y., Chen, T., Xu, Z., Sun, N., and Temam, O. (2016a). DianNao Family: Energy-Efficient Hardware Accelerators for Machine Learning. *Commun. ACM*, 59(11):105–112.

Chen, Y.-H., Krishna, T., Emer, J., and Sze, V. (2016b). Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. In *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, pages 262–263.

Concha, D. T., Maia, H. D. A., Pedrini, H., Tacon, H., Brito, A. D. S., Chaves, H. D. L., and Vieira, M. B. (2018). Multi-stream convolutional neural networks for action recognition in video sequences based on adaptive visual rhythms. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 473–480.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2014). Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389.

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. *CoRR*, abs/1604.06573.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Google (2020). Google EdgeTPU Documentation.

Hazra, S. and Santra, A. (2019). Radar gesture recognition system in presence of interference using self-attention neural network. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1409–1414.

Intel (2020). Intel Neural Compute Stick 2 Documentation.

Köpüklü, O., Köse, N., and Rigoll, G. (2018). Motion fused frames: Data level fusion strategy for hand gesture recognition. *CoRR*, abs/1804.07187.

Köpüklü, O., Gunduz, A., Kose, N., and Rigoll, G. (2020). Online dynamic hand gesture recognition including efficiency analysis. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2(2):85–97.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Molchanov, P., Gupta, S., Kim, K., and Pulli, K. (2015). Multi-sensor system for driver's hand-gesture recognition.

Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., and Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. pages 4207–4215.

Naguri, C. R. and Bunescu, R. C. (2017). Recognition of dynamic hand gestures from 3d motion data using lstm and cnn architectures. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1130–1133.

NVIDIA (2020). NVIDIA Jetson Nano Documentation.

Reuther, A., Michaleas, P., Jones, M., Gadepally, V., Samsi, S., and Kepner, J. (2019). Survey and benchmarking of machine learning accelerators. *2019 IEEE High Performance Extreme Computing Conference (HPEC)*.

Rockchip (2020). Rockchip Documentation.

Tran, D., Bourdev, L. D., Fergus, R., Torresani, L., and Paluri, M. (2014). C3D: generic features for video analysis. *CoRR*, abs/1412.0767.

Wang, S., Song, J., Lien, J., Poupyrev, I., and Hilliges, O. (2016). Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 851–860. ACM.

Young, K., Regan, M., and Hammer, M. (2003). Driver distraction: A review of the literature. *Distracted Driving*.