In-Network Convolution in Grid Shaped Sensor Networks

Niki Hrovatin[1,2,*], Aleksandar Tošić[1,2] and Jernej Vičič[1,3]

[1]*University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies, Koper, Slovenia*
[2]*InnoRenew CoE, Izola, Slovenia*
[3]*Research Centre of the Slovenian Academy of Sciences and Arts, The Fran Ramovš Institute, Ljubljana, Slovenia*
E-mail: niki.hrovatin@famnit.upr.si; aleksandar.tosic@upr.si; jernej.vicic@upr.si
[*]*Corresponding Author*

Received 16 July 2021; Accepted 20 September 2021;
Publication 23 November 2021

Abstract

Gathering information is the primary purpose of a Sensor Network. The task is performed by spatially distributed nodes equipped with sensing, processing, and communication capabilities. However, data gathered from a sensor network must be processed, and often the collective computation capability of nodes forming the sensor network is neglected in favor of data processing on cloud systems. Nowadays, Edge Computing has emerged as a new paradigm aiming to migrate data processing close to data sources. In this contribution, we focus on the development of a sensor network designed to detect a person's fall. We named this sensor network the smart floor. Fall detection is tackled with a Convolutional Neural Network, and we propose an approach for in-network processing of convolution layers on grid-shaped sensor networks. The proposed approach could lead to the development of a sensor network that detects falls by performing CNN inference processing on the edge. We complement our work with a simulation using the simulator ns-3. The simulation is designed to emulate the communication overhead of the proposed approach applied to a wired sensor network that resembles the smart

Journal of Web Engineering, Vol. 21_1, 75–96.
doi: 10.13052/jwe1540-9589.2114

floor. Simulation results provide evidence on the feasibility of the proposed concept applied to wired grid shaped sensor networks.

Keywords: Sensor networks, edge computing, fall detection, convolutional neural networks, network simulator ns-3.

1 Introduction

A sensor network consists of spatially distributed nodes deployed in a dynamic environment for specific monitoring purposes. Several current and potential applications exist ranging from the military domain, environmental monitoring, healthcare, industrial manufacturing monitoring etc. [21]. Typical sensor networks count hundred or thousand of densely deployed nodes. Nodes are devices constrained in processing and communication capabilities, equipped with one or multiple sensors. Such large networks of sensors provide a detailed view of the environment in which they are deployed.

In this contribution, we consider a sensor network deployed to detect events using a Convolutional Neural Network (CNN) that feeds on input from multiple sensor nodes. Since the CNN inference process requires data from multiple sensor nodes, typical solutions convey all the nodes' data to a central processing point usually in the cloud. Therefore computing capabilities of nodes forming the network are not used. Furthermore, transferring data to remote systems like cloud services could introduce security concerns [24] and a substantial latency between event occurrence and detection [18].

Nevertheless, we have recently witnessed the emergence of edge computing [1, 29], a paradigm aiming to move computation as close as possible to data sources. In the present research, we tackle the problem of distributing CNN processing load across sensor nodes' actual data sources of the sensor network. Precisely we propose a technique applicable only to grid-shaped sensor networks, in which neighboring sensor nodes are interchanging sensor readings to conjointly compute Convolution Layers of a CNN on sensor nodes. Additionally, we discuss applying the proposed technique on our non-intrusive fall detection sensor network dubbed smart floor. The smart floor is a grid-shaped sensor network in which each sensor node is sensing the local force applied on the floor. A CNN is used to recognize if the activity occurring over the smart floor is a person that fell on the floor or just activities of daily living like walking, moving objects, etc.

Although the proposed concept takes advantage of the processing power of sensor nodes, substantial communication overhead is generated since each

sensor node is interchanging sensor readings with neighbors. In this study, we address concerns regarding the communication overhead with a simulation based on the ns-3 simulator [11]. The simulation is designed to emulate the communication overhead of the proposed concept applied to a wired grid shaped sensor network that resembles the smart floor. Simulation results show that the communication overhead only leads to severe network congestion if low bit-rate links are used with a large convolution kernel. Furthermore, the examination considers two network topologies, showing a significant difference in the number of packet drops and packet traveling time.

Section 2 gives an overview of the related literature about in-network neural network inference in sensor networks. In Section 3 we discuss fall detection and we give an overview of the current development of the smart floor. In Section 4 we present a technique to in-network compute 3D discrete convolution on sensor nodes of a grid shaped sensor network. Section 5 presents the experimental setup and results of the simulation. Conclusion and future work is given in Section 6.

2 Related Work

Several studies in the Internet of Things (IoT) field are addressing deep learning inference on low-power mobile devices. DeepX [13] is a software accelerator based on runtime layer compression and deep architecture decomposition able to allocate data and model operations optimally across diverse processor types. The Big-Little approach [3] uses small deep learning models that are located on embedded devices (the Little role) to perform inference on a restricted set of events. However, for some critical situations that require a reliable inference, the collected data is also sent to processors in the cloud (the Big role) to perform the inference process on a larger deep learning model.

An interesting approach aiming at the reduction of the communication overhead in IoT networks was proposed in [5]. The proposed architecture moves the deep learning model from the base station towards the data source nodes. However, deep learning models are only partially moved. Models located on data source nodes are small and resource-efficient, designed to output a limited set of codes. The codes are then used to perform the final inference on powerful base-station nodes using large inference models. Although the proposed approach could reduce the communication overhead, it is designed for inference over data sourcing from one device and not from multiple devices, as required in the CNN implemented in the smart floor.

In-network computing involves the processing of data as it travels via the sensor network to the sink node. The sink node in a sensor network is the node that acts as a gateway between the sensor network and external systems and usually has greater processing and storage capabilities than other nodes in the sensor network. In-network computing was already applied to the deep learning concept in [12]. They proposed a distributed deep learning architecture that assigns processing roles to sensor network nodes. Data is gathered on those sensor nodes, processed, and sent to the next node in the chain. Each node in the chain is computing a layer of the convolutional neural network. The sink node is computing the end fully connected layers. The proposed approach effectively distributes a convolutional neural network's computation load among multiple nodes of a sensor network. However, this causes a high load on nodes in the computation chain, especially on the first node that must gather data from all sensing nodes.

Fukushima et al. [8] addressed this problem with a framework for distributed deep learning in wireless sensor networks based on node coordination. They assume that a Wireless Sensor Network (WSN) can resemble a grid structure, and they map CNN neurons to physical nodes of a WSN. Therefore the whole CNN is statically built on the WSN and is trained to work only with that precise deployment of sensor nodes. They demonstrated this approach's feasibility with a simulation and two experiments. Although the research conducted by Fukushima et al. includes a sensor network able to detect fall events, the research does not consider event localization.

In the present contribution, we propose a slightly different approach motivated by the need to detect fall events and the smart floor properties of being grid-shaped, wired, and having a high node density. In our proposed solution, we map to sensor nodes only the convolution layers. A procedure will then extract processed activities from the sensor network and send them to a sink node that will complete the inference using the fully connected layers of the CNN. Therefore, event location can be determined from the location of data reporting nodes. Moreover, the solution differs from the approach proposed by Fukushima et al. since it allows the application of a once trained and well-tested CNN in different network deployments. Training a CNN with neurons built into nodes of a sensor network could lead to slightly different predictions at sensor networks having diverse shapes even if the same training data is used. The root cause of this alteration of predictions is the shape of the building in which the sensor network is placed, physically constraining the shape of the network and the number of senor nodes. Therefore by applying the technique presented by Fukushima et al. the CNNs deployed into different

sensor networks will have a different number of neurons and disposition of holes. (Fukushima et al. were referring to holes, as locations of the monitored environment without sensor nodes.) In the fall detection scenario holes might be numerous due to the irregular shapes of rooms and disposition of walls inside buildings. Inference alteration due to changes in the number of neurons in Artificial Neural Networks is well known [2, 28]. And inference alteration due to changes in the number of holes is documented by Fukushima et al.

A fall detection system must ensure that each fall event is detected and correctly reported. Since undetected fall events can have repercussions on the health of the fallen person, each wrongly predicted fall event must be investigated to find the root cause and prevent similar occurrences. Having the exact inference mechanism on different deployments allows the adjustment of every system instance if eventual flaws are discovered.

3 Non-Intrusive Fall Detection Using Smart Floor

Falls are unpredictable accidental events. Common in childhood, rare in adulthood, but a significant problem among the elderly. The report *Fall Prevention in older Age* [20] carried out by the World Health Organization state that approximately 30% of people aged 65 fall each year, and the odds increase for those over 70 years of age. Falls are critical events in the elderly, which can result in severe injuries or fatalities. More than 50% of injury-related hospitalization among people over 65 years is a fall consequence. The research [10] pointed out the need to provide immediate help after a fall event to prevent severe or fatal consequences.

While fall prevention is enhanced by behavioral, environmental, socioeconomic, and biological risk reduction [20], the fall recognition problem is widely addressed with the use of technological solutions. The literature review conducted by Singh et al. [23] investigates different fall detection systems categorizing them into *wearable*, *ambiance*, and *hybrid* systems. Wearable systems are based on accelerometer or gyroscope sensor technology embedded in items or smartphones. Wearable solutions are low cost solutions that can detect a fall effectively; however, the user must actively wear and maintain these systems. Ambiance fall detection systems are embedded in the monitored environment. Image sensors, acoustic sensors, pressure sensors, infrared sensors, radar sensors or a combination of them is used to monitor user activity and detect falls. Ambiance systems provide good fall detection performance, they usually incur higher costs, but they eliminate the active interaction between user and system. Technology acceptance by

Figure 1 The picture was taken during the data collection event, where fall events were simulated following safety precautions. The smart floor is the white surface surrounded by landing mats.

older adults is a recognized problem, drastically emphasized in monitoring systems [14]. Fall detection technology must be constantly present in the user's life, to assure benefit; hence solutions based on wearable devices or image sensors are considered highly intrusive [6, 23] since users are always aware of the system. A systematic review conducted by Yusif et al. [30], suggests that the main adoption barriers between older people and assistive technologies are: privacy, trust, added value, cost, ease of use, perception of no need, stigma etc. (sorted by importance). The mentioned factors are raising the need to design a fall detection system able to effectively detect falls while being non-intrusive, privacy-aware, and cost-effective.

The literature review conducted by Singh et al. [23] provides many insights about different sensor technologies used in fall detection systems. Analyses suggest that systems based on *passive Infra-red* (PIR) radiation sensors are the only ones achieving a high fall detection rate while being non-intrusive, privacy-aware, and cost-effective. These systems are usually structured as vertical arrays of many PIR sensors positioned on a room wall, designed to detect the fall vertical motion. A PIR fall detection system was developed by Fukushima et al. [8], relying on a 6×6 grid-shaped wireless

sensor network able to detect falls using a CNN. The system achieved good fall detection performance. However, such systems are subject to the risk of obstruction of the sensing range if an object is located between the sensor and the monitored user. An interesting alternative to PIR systems highlighted by Singh et al. are floor pressure sensor systems, which can achieve high fall detection rates while being privacy-aware and non-intrusive. The reviewed solutions rely on complex sensing technologies resulting in high implementation prices, like floor pressure sensing using optical fibers [7]. However, a commercial solution based on capacitive sensor technology embedded in the flooring material has already been released on the market under the name SensFloor [25].

In this contribution, we present recent development of our fall detection solution based on the smart floor implementation presented in [27], which relies on widely available and cheap Force-Sensing Resistor (FSR) technology; adequate to be embedded in a vast choice of flooring materials. The smart floor was recently the object of study in a master thesis [19], intending to establish solid foundations for developing a non-intrusive privacy-preserving fall detection system. A dataset [26] consisting of 420 simulated fall events was collected using the smart floor. In Figure 1, the smart floor during the data collection event. The collected data was used to train different machine learning models and results shown the notable accuracy of CNN in distinguishing activities of daily living from simulated fall events. The effectiveness of CNNs applied to floor pressure sensing systems was highlighted by the contribution [22], where a CNN was used to identify the person's unique walking gait over a smart mat monitoring system. The system was also proposed for fall detection, but further analyses must be conducted.

However, our system's uniqueness does not reside in the machine learning component but rather in the end goal of developing a modular tile system, where each tile will be the sensor node of a distributed sensor network designed to detect fall events while being non-intrusive and privacy-preserving.

4 In-Network Convolution In Grid Shaped Sensor Networks

In this section, we first present the sensor network model taken into account to develop our solution. Then we show how to perform the in-network 3D discrete convolution on grid-shaped sensor networks. Furthermore, we discuss applying the presented technique to in-network compute multiple

convolution layers and pooling layers on the smart floor. We end with a brief description of our in-network fall detection solution.

4.1 The Sensor Network Model

We consider a sensor network resembling the smart floor described in [19, 26, 27]. The network consists of two types of nodes, the majority are sensor nodes, which are nodes equipped with sensor technology sensing a physical quantity in time. These nodes are limited in computational and memory capabilities as they are designed to be cheap. The other type of node is named sink node, which purpose is to gather data sensed by sensor nodes, store it, process it, and interact with external systems. The sink node also has greater computational and storage capabilities than sensor nodes.

Sensor nodes are deployed in a plane following a grid structure; each sensor node is equidistant from the closest sensor nodes in cardinal directions. Sensor nodes are linked via a point-to-point link with each sensor node in their neighborhood. We refer to the neighborhood of a sensor node as the eight closest sensor nodes. In each cardinal and intercardinal direction relative to a sensor node, lies one and only one sensor node from its neighborhood. Sensor nodes can directly interchange sensor readings with nodes in their neighborhood. However, sensor nodes can also retrieve sensor readings from nodes outside their neighborhood and communicate with the sink node using a multi-hop routing strategy based on grid coordinates similar to [16]. A sensor node can be uniquely identified in the network using the IP-address or gird coordinates (e.g. (x, y)). For convenience, we assume that the sink node acts like a central coordination authority that knows grid coordinates and IP-addresses of each sensor node in the network. A sensor node can request the sink node to reveal other sensor node's grid coordinates based on the node IP-address or vice-versa.

4.2 In-network 3D Convolution on Grid-shaped Sensor Network

The 2D discrete convolution technique is widely used in image processing [9] to apply smoothing filters, image sharpening, identify edges, and classify images in conjunction with machine learning methods [15]. Images are represented as matrices of $n \times m$ pixels with variable intensities. The convolution operation will slide a kernel of size $k \times k, k < m$ over each pixel of the image, performing an elementwise multiplication between the kernel and the covered part, summing up results into one single output value. The kernel repeats this process for every pixel it slides over, generating a new 2D matrix.

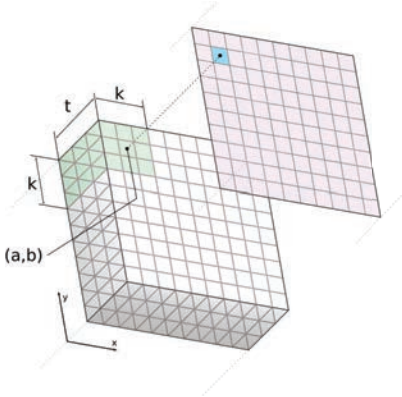


Figure 2 The figure presents the 3D convolution operation. Each square in the white plane represents a sensor node. The green area indicates sensor readings needed to compute a fragment of the convolution operation on the node a, b using the kernel of size $k \times k \times t$. The red grid is the result of the convolution operation over the grid of sensor nodes, highlighted with blue color the convolution result of sensor node on coordinates a, b.

Our proposed distributed solution is based on the assumption of the sensor network's physical grid structure introduced in Section 4.1. The grid-shaped sensor network resembles a matrix on which to perform the convolution operation with a kernel. However, we will not slide the kernel over the matrix of sensor nodes. Instead, each sensor node will compute a fragment of the entire convolution operation.

Since fall events occur over a surface during a time interval, the application of 3D convolution to the smart floor will certainly enhance prediction accuracy as 3D CNNs [17] can operate on planar and temporal dimensions.

To perform the 3D discrete convolution operation on the grid shaped sensor network, we take into account the two planar dimensions and a time dimension detailing the change in sensor values over time. Sensor nodes share the same kernel with all other sensor nodes in the network. For convenience, we describe the procedure only for kernels of size $k \times k \times t$. $k \times k$ the planar dimensions, and t the time dimension as it is shown on Figure 2.

To compute a fragment of the convolution operation, a sensor node located in the grid-shaped sensor network at coordinates (x, y) will first gather t sensor readings from each sensor node located at coordinates $(x + i, y + j)$, where $-\lfloor \frac{k}{2} \rfloor \leq i, j \leq \lfloor \frac{k}{2} \rfloor + v$. Figure 3 shows such operation. If the kernel dimension k is an odd number, the variable $v = 0$, otherwise $v = -1$. If sensor node coordinates are outside the physical network, zero padding is applied.

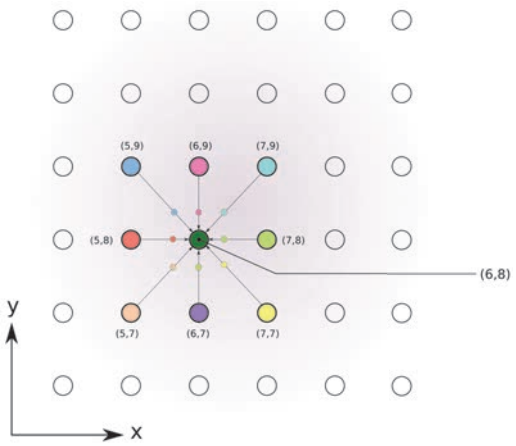


Figure 3 The figure illustrates how a sensor node located at grid coordinates $(6, 8)$ gathers data from other sensor nodes to compute a fragment of the convolution operation with a kernel of size $3 \times 3 \times t$. Sensor nodes are drawn with circles, and data from diverse sensor nodes is represented with a different color.

Then the sensor node will perform the elementwise multiplication between the kernel, its sensor value, and sensor values collected from local sensor nodes, results are summed up in one single output value. We refer to this value as a fragment, and to local sensor nodes as the sensor nodes whose values were used to compute the fragment. Since each sensor node knows its and local sensor nodes' location as coordinates in a grid-shaped sensor network, fragments are computed respecting the same kernel orientation on all sensor nodes. Therefore, each sensor node holds a fragment of the resulting convolution operation between the grid-shaped sensor network and the selected kernel.

4.2.1 Multi-layer convolutional neural network

The proposed technique can be applied to multi-layer CNN such that the first convolution layer is computed on sensor readings as described in the upper section. Outputs from the first layer reside on sensor nodes, maintaining the grid structure. Hence it is possible to compute the next convolution layer following the same principle but on previous layer outputs instead of sensor readings. This concept was already proposed by Fukushima et al. [8], where a multi-layer CNN was built on a wireless sensor network.

They proposed the application of this concept also to in-network compute the pooling layer. **Pooling layers** are used to reduce the dimension of data by

combining multiple values into one single value using the average or max function. However, considering our the smart floor needs, the pooling layer will be implemented only on the time dimension, as we want to maintain as much planar information as possible. Hence the pooling layer can be computed on sensor nodes without the need for data from other sensor nodes.

4.2.2 Communication overhead

The in-network processing of convolution layers requires a noticeable communication overhead. Each sensor node computes a fragment of the convolution layer by gathering data from other sensor nodes. Hence the communication overhead is increased locally to sensor nodes involved in the computation. Fukushima et al. demonstrated the manageability of this concept with two experiments, and they provided a simulation to determine how kernel size affects the communication overhead. The simulation revealed that kernels $k \times k \times t$ of size $k \leq 5$ does not generate a concerning communication overhead.

Our research considers the communication overhead generated in wired sensor networks with a simulation using the network simulator ns-3. Simulation results are presented in Section 5.

4.3 Fall Detection Using In-network Convolution

Our proposed fall detection solution is based on the grid-shaped sensor network model introduced in Section 4.1, a CNN designed to process part of the inference on sensor nodes, and a procedure to extract the field of interest for further processing at sink nodes. Since the sensor network will be deployed in many rooms or an entire building, the whole system must be modular and adaptable to fit any room shape and deployment extensions. Therefore, we will not train a CNN on the whole sensor network, like were done by Fukushima et al. [8], as this will generate different CNNs at each deployment which might produce slightly different inference outcomes.

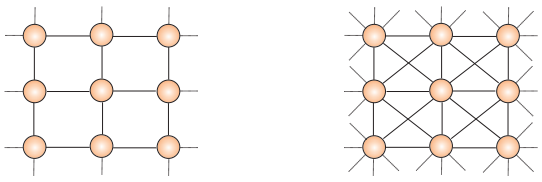
Our system is designed to work on a CNN trained on the smart floor described in Section 3, composed of 16 pressure sensors. This limits the CNN input to readings from only 16 sensor nodes structured like a 4×4 grid. Hence, we need a mechanism to extract activities happening on the sensor network and feed them in the CNN for inference.

The activity extraction starts from sensor nodes which are constantly sensing the pressure applied on them. However, they will proceed with further processing only if they identify a significant change in the sensed

pressure, which signals an activity on them. Triggered sensor nodes will in-network compute convolution layers of the CNN using the technique explained in Section 4.2. Adjacent convolution layer results are grouped via node coordination, and the grouped result is sent to a sink node. The sink node will complete the inference by processing convolution layer results with the CNN's fully connected layers. The exact event location can be determined from the location of nodes reporting the grouped result.

5 Evaluation

This section presents results obtained from a simulation using the ns-3 simulator [11]. Via a simulation, we aim to evaluate whether the proposed concept of in-network computing convolution layers in wired grid shaped sensor networks generates a communication overhead that leads to a severe network congestion. Precisely in the simulation, we try to model the smart floor. Additionally, we compare two network topologies, the *plain grid* in which sensor nodes are linked with four neighboring nodes, one link in each cardinal direction, and the *diagonal grid* in which each sensor nodes are linked with eight neighboring nodes, one link in each cardinal and intercardinal direction. Topologies are illustrated in Figure 4. Although the two network topologies have numerous redundant links, we selected them since our end goal is the development of the smart floor as a modular tile system, where each tile will be a sensor node able to link with adjacent sensor nodes through wired point to point links embedded in the tiles. To perform this investigation, we examine how kernel size and link bitrate affects the traveling time (TT) of datagrams in the network. We define the TT of packets in the network as the elapsed time between the issuing of the packet by a sensor node, and the receipment of the packet from the destination node.



(a) (b)

Figure 4 Network topologies considered in the simulation: *plain grid* in (a), *diagonal grid* in (b).

5.1 Experimental Setup

The simulated network is designed to resemble the smart floor, a grid shaped sensor network in which each sensor node is sensing the force applied on it (see Section 3). Sensor nodes are measuring the applied force 100 times per second [26], converting the force into an integer value. Therefore, a smart floor sensor node is generating 400B/s of data at fixed-length encoding using 32b integers. In the simulated network, we approximate this bitrate using packets of 512B. Each sensor node in the simulation generating a batch of packets each second, and sending one packet of the batch to each of its local sensor nodes. We refer to local sensor nodes of a node at coordinates (x, y) as the sensor nodes whose coordinates are in the range $(x \pm \lfloor \frac{k}{2} \rfloor, y \pm \lfloor \frac{k}{2} \rfloor)$, (x, y) excluded, k one dimension of a $k \times k$ kernel. Clocks of all nodes are synced, and a batch identifier is applied to packets. Packets are not sent all at the same time, but each packet is sent at a randomly chosen time in a $1 - \Delta t$ second time interval (Δt computed from the kernel size and link bitrate to allow the packets to reach the destination before the next batch of packets commences issuing). Packets that do not reach the destination within one second after their batch starts issuing are counted as lost packets.

The simulated network consists of a 15×15 grid of sensor nodes. Nodes are linked following two network topologies the *plain grid*, and the *diagonal grid*. Links between nodes are full-duplex point-to-point links of bitrate dr. dr being an independent variable of values $dr = \{0.125, 0.25, 0.5, 1, 5, 10\}$– Mbps. Bitrate values of dr were chosen to approximate bitrate values of commonly used communication standards in IoT and sensor networks (e.g., RS-232, UART, USB, and Ethernet). Each sensor node acts also as a router. Routing tables of nodes are statically computed at network deployment using the Dijkstra Shortest Path First algorithm [4]. A simulation was ran for each combination of independent variables network topology, link bitrate dr, and kernel size k. Kernel size values of $k = \{3, 5, 7\}$ should be appropriate since, on the smart floor, a larger kernel size will cover an area much larger than the interested detection area of a fall event. (The smart floor includes 16 pressure sensors deployed on a surface of 1.2 m^2.)

Data were collected only from one sensor node during the simulation, the one located in the middle of the grid, precisely at coordinates $(7, 7)$. Only this node was monitored since it is located in the central portion of the network, where the communication overhead is condensed. The recorded data include TT of packets reaching the monitored node and packet loss throughout the whole network. Figures 5 and 6 display scatter plots of

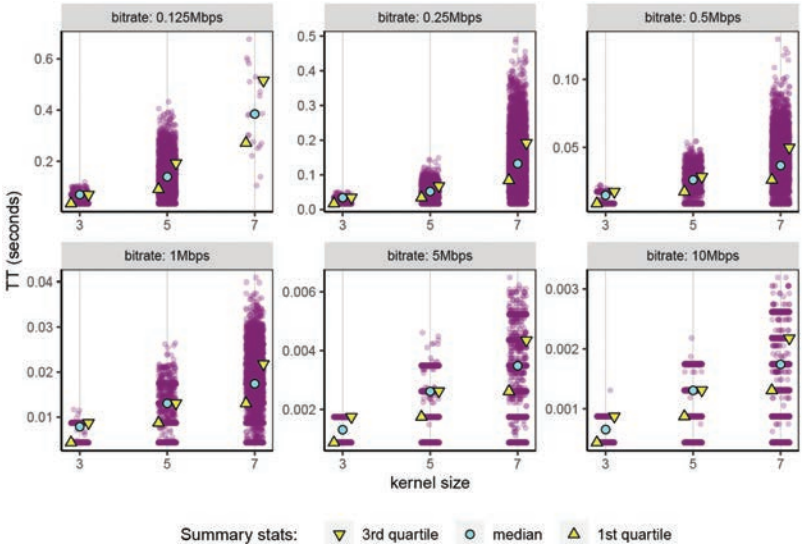


Figure 5 Scatter plots of packets TT, at combinations of independent variables: topology *plain grid*, bitrate $dr = \{0.125, 0.25, 0.5, 1, 5, 10\}$–Mbps and kernel size $k = \{3, 5, 7\}$. Data from the node at coordinates $(7, 7)$.

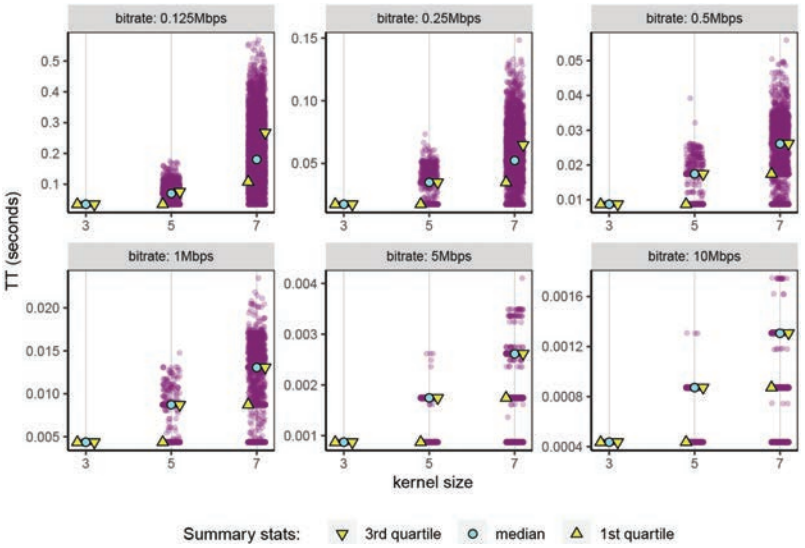


Figure 6 Scatter plots of packets TT, at combinations of independent variables: topology *diagonal grid*, bitrate $dr = \{0.125, 0.25, 0.5, 1, 5, 10\}$–Mbps and kernel size $k = \{3, 5, 7\}$. Data from the node at coordinates $(7, 7)$.

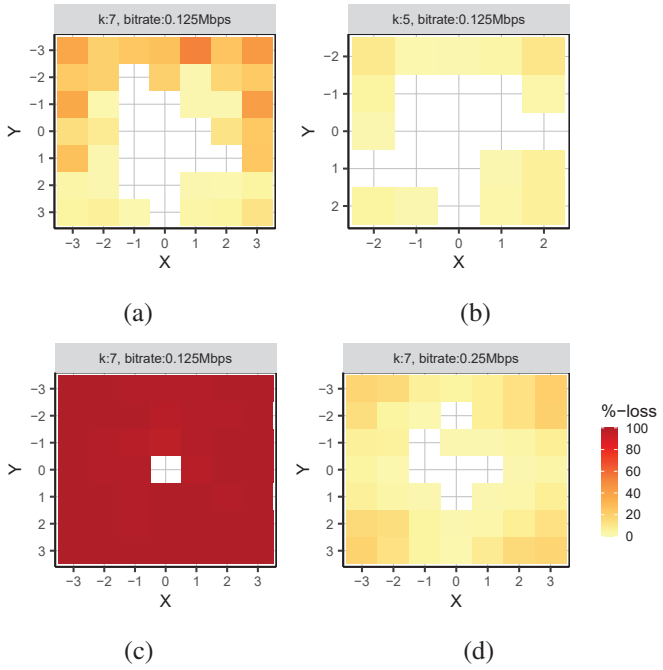


Figure 7 Percentage of packets sourcing from sensor nodes at coordinates $(7 + X, 7 + Y)$ that do not reach the monitored node within one second after their batch starts issuing. The monitored node is at coordinates $(7, 7)$. Figure (a) *diagonal grid* topology. Figures (b),(c),(d) *plain grid* topology. We have not observed packet loss on the monitored node in other combinations of considered independent variables topology, bitrate, and kernel size.

packet TT. In the former, the network is deployed following the *plain grid* topology, and in the latter, the network is deployed following the *diagonal grid* topology. Table 1 presents summary statistics of packet TT and packet loss. Figure 7 display the percentage of packets sourcing from sensor nodes local to the monitored node that do not reach the monitored node.

5.2 Experimental Results

From plots in Figures 5 and 6 can be seen, that the data is not normally distributed. We believe that the reason behind the non-normal distribution of packet TT is that packets are traveling through multiple nodes to reach the destination, some packets traveling through more nodes than other packets due to kernel size. Moreover, we notice that at lower datarates ($dr \leq 0.5$), packet TT begin to spread. A possible cause for the spread of TT is traffic

Table 1 Summary statistics of packet TT recorded at the node at coordinates $(7, 7)$. The column **%** packet loss expresses the packet loss of the whole network at selected independent variables

Network Topology	Kernel Size	Link Bitrate (Mbps)	Min TT (s)	Avg TT (s)	Max TT (s)	Std TT (s)	1st Quartile TT (s)	Median TT (s)	3rd Quartile TT (s)	% Packet Loss
Plain grid	3	0.125	0.03482	0.05443	0.11952	0.01915	0.03488	0.06963	0.06976	
		0.25	0.01741	0.0266	0.05191	0.00906	0.01744	0.03432	0.03488	
		0.5	0.0087	0.01313	0.02246	0.00439	0.00872	0.01478	0.01744	
		1	0.00435	0.00656	0.0117	0.00219	0.00436	0.00789	0.00872	
		5	0.00087	0.00131	0.00175	0.00044	0.00087	0.00131	0.00174	
		10	0.00044	0.00065	0.00131	0.00022	0.00044	0.00065	0.00087	
	5	0.125	0.03482	0.14719	0.43254	0.07272	0.09082	0.13952	0.19282	1.7
		0.25	0.01741	0.05089	0.14889	0.02183	0.03488	0.05232	0.06825	
		0.5	0.0087	0.02324	0.05463	0.0093	0.01744	0.02611	0.02843	
		1	0.00435	0.01122	0.02645	0.00443	0.00872	0.01306	0.0131	
		5	0.00087	0.00219	0.00461	0.00085	0.00174	0.00261	0.00262	
		10	0.00044	0.00109	0.00218	0.00042	0.00087	0.00131	0.00131	
	7	0.125	0.10553	0.39668	0.67717	0.15938	0.27151	0.38481	0.51607	86.9
		0.25	0.01741	0.14409	0.49108	0.0792	0.08454	0.13234	0.19204	5
		0.5	0.0087	0.03858	0.12956	0.0174	0.02616	0.03668	0.04988	
		1	0.00435	0.01658	0.04086	0.00686	0.01306	0.01741	0.0218	
		5	0.00087	0.0031	0.00646	0.00123	0.00261	0.00348	0.00435	
		10	0.00044	0.00154	0.00318	0.00061	0.00131	0.00174	0.00218	
Diagonal grid	3	0.125	0.03482	0.03487	0.03494	4e-05	0.03482	0.03488	0.03488	
		0.25	0.01741	0.01743	0.01747	2e-05	0.01741	0.01744	0.01744	
		0.5	0.0087	0.00872	0.00874	1e-05	0.0087	0.00872	0.00872	
		1	0.00435	0.00436	0.00437	1e-06	0.00435	0.00436	0.00436	
		5	0.00087	0.00087	0.00087	1e-06	0.00087	0.00087	0.00087	
		10	0.00044	0.00044	0.00044	1e-07	0.00044	0.00044	0.00044	
	5	0.125	0.03482	0.0667	0.17552	0.02454	0.03494	0.06976	0.07564	
		0.25	0.01741	0.03059	0.07342	0.00979	0.01744	0.03488	0.03494	
		0.5	0.0087	0.01483	0.03921	0.00442	0.00872	0.01744	0.01744	
		1	0.00435	0.00736	0.01476	0.00214	0.00436	0.00872	0.00872	
		5	0.00087	0.00146	0.00262	0.00041	0.00087	0.00174	0.00174	
		10	0.00044	0.00073	0.00131	0.00021	0.00044	0.00087	0.00087	
	7	0.125	0.03482	0.19328	0.56775	0.10669	0.10756	0.18028	0.26753	8
		0.25	0.01741	0.05129	0.14832	0.02147	0.03488	0.05232	0.06473	
		0.5	0.0087	0.02204	0.05592	0.00785	0.01744	0.02611	0.02621	
		1	0.00435	0.01054	0.02346	0.00353	0.00872	0.01306	0.01308	
		5	0.00087	0.00205	0.0041	0.00066	0.00174	0.00261	0.00262	
		10	0.00044	0.00102	0.00175	0.00033	0.00087	0.00131	0.00131	

congestion; packets are waiting for links to be freed. In Figure 7 can be seen, that at kernel size $k = 7$ and bitrate $dr = 0.125$ Mbps, there is a considerable difference in packet loss between *plain grid* and *diagonal grid* topology. The difference can be noticed also in Table 1. Furthermore, in Table 1 can be

seen that severe network congestion occurs only at plain grid topology at parameters: $k = 7$ and bitrate $dr = 0.125$ Mbps with packet loss ratio at 87%. A two-tailed Mann-Whitney U test showed that there is a significant difference in packet TT between a network deployed following the *plain grid* topology and one deplyoed following the *diagonal grid* topology. We ran the test at bitrate $dr = 10$ Mbps and kernel size $k = \{3, 5, 7\}$. Test results: $k = 3$ (W = 176082, p-value $< 2.2e^{-16}$), $k = 5$ (W = 1516573, p-value $< 2.2e^{-16}$),$k = 7$ (W = 5879408, p-value $< 2.2e^{-16}$).

6 Conclusion and Future Work

In this contribution, we depicted the future development of our smart floor as a **non-intrusive fall detection system**. We gave a brief overview of the fall detection problem and how it is solved using technology solutions, highlighting our system's potential.

We proposed a technique to partially process the CNN inference on sensor nodes of a grid-shaped sensor network, and we discussed the application of this technique to our system. The proposed solution that will fit the smart floor needs is able to adapt to changes in the sensor network topology, does not require a new CNN training for each deployment, effectively reduces computation load on the end system, and the activity extraction procedure cuts the data transfer overhead in the sensor network.

Our research is complemented by a simulation designed to emulate the communication overhead of the proposed technique in grid-shaped wired sensor networks. Simulation results show that severe network congestion occur at *plain grid* topology when large convolution kernels are used ($k > 5$) in networks with low bitrate links ($dr \leq 0,125$ Mbps). Interestingly, at the same configuration parameters, the severe network congestion was not observed at *diagonal grid* topology.

Considerably more research will need to be done on the activity extraction procedure since the end objective is to develop a privacy-preserving system, and the activity extraction procedure could potentially provide location privacy by removing any linkage to the source of the data through the grouping of data in anonymous ready to process chunks.

An advanced activity extraction procedure could be applied to coordinate sensor nodes and in-network perform the whole CNN inference processing. Therefore, the sensor network will communicate with external systems only to report falls.

Acknowledgements

The authors gratefully acknowledge the European Commission for funding the InnoRenew project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Regional Development Fund). They also acknowledge the Slovenian Research Agency ARRS for funding the project J2-2504.

References

[1] Ejaz Ahmed, Arif Ahmed, Ibrar Yaqoob, Junaid Shuja, Abdullah Gani, Muhammad Imran, and Muhammad Shoaib. Bringing computation closer toward the user network: Is edge computing the solution? *IEEE Communications Magazine*, 55(11):138–144, 2017.

[2] F Arifin, H Robbani, T Annisa, and NNMI Ma'Arof. Variations in the number of layers and the number of neurons in artificial neural networks: Case study of pattern recognition. In *Journal of Physics: Conference Series*, volume 1413, page 012016. IOP Publishing, 2019.

[3] Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Steven Bohez, Pieter Simoens, Piet Demeester, and Bart Dhoedt. Distributed neural networks for internet of things: The big-little approach. In *Internet of Things. IoT Infrastructures*, pages 484–492. Springer International Publishing, 2016.

[4] Stuart E Dreyfus. An appraisal of some shortest-path algorithms. *Operations research*, 17(3):395–412, 1969.

[5] Rong Du, Sindri Magnusson, and Carlo Fischione. The internet of things as a deep neural network. *IEEE Communications Magazine*, 58(9):20–25, 2020.

[6] Le Fang, Yu Wu, Chuan Wu, and Yizhou Yu. A non-intrusive elderly home monitoring system. *IEEE Internet of Things Journal*, 2020.

[7] Guodong Feng, Jiechao Mai, Zhen Ban, Xuemei Guo, and Guoli Wang. Floor pressure imaging for fall detection with fiber-optic sensors. *IEEE Pervasive Computing*, 15:40–47, 03 2016.

[8] Yuta Fukushima, Daiki Miura, Takashi Hamatani, Hirozumi Yamaguchi, and Teruo Higashino. Microdeep: In-network deep learning by micro-sensor coordination for pervasive computing. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 163–170. IEEE, 2018.

[9] Rafael C Gonzalez, Richard E Woods, and Barry R Masters. Digital image processing third edition. *Pearson Prentice Hall*, pages 743–747, 2008.

[10] R Jan Gurley, Nancy Lum, Merle Sande, Bernard Lo, and Mitchell H Katz. Persons found in their homes helpless or dead. *New England Journal of Medicine*, 334(26):1710–1716, 1996.

[11] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14(14):527, 2008.

[12] Nada B Jarah. Deep learning in wireless sensor network. *Journal of Al-Qadisiyah for computer science and mathematics*, 13(1):Page–11, 2021.

[13] Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–12. IEEE, 2016.

[14] F Le Deist and M Latouille. Acceptability conditions for telemonitoring gerontechnology in the elderly: optimising the development and use of this new technology. *Irbm*, 37(5-6):284–288, 2016.

[15] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. A survey of convolutional neural networks: analysis, applications, and prospects. *arXiv preprint arXiv:2004.02806*, 2020.

[16] Wen-Hwa Liao, Jang-Ping Sheu, and Yu-Chee Tseng. Grid: A fully location-aware routing protocol for mobile ad hoc networks. *Telecommunication systems*, 18(1):37–60, 2001.

[17] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[18] Mithun Mukherjee, Rakesh Matam, Constandinos X Mavromoustakis, Hao Jiang, George Mastorakis, and Mian Guo. Intelligent edge computing: Security and privacy challenges. *IEEE Communications Magazine*, 58(9):26–31, 2020.

[19] Hrovatin Niki. *Neintruzivna identikacija padcev s pomočjo pametnih tal: magistrsko delo*. PhD thesis, Univerza na Primorskem, Fakulteta za matematiko, naravoslovje in informacijske tehnologije, 2020.

[20] World Health Organization, World Health Organization. Ageing, and Life Course Unit. *WHO global report on falls prevention in older age*. World Health Organization, 2008.

[21] Kazi Chandrima Rahman. A survey on sensor network. *Journal of Computer and Information Technology*, 1(1):76–87, 2010.

[22] Qiongfeng Shi, Zixuan Zhang, Tianyiyi He, Zhongda Sun, Bingjie Wang, Yuqin Feng, Xuechuan Shan, Budiman Salam, and Chengkuo Lee. Deep learning enabled smart mats as a scalable floor monitoring system. *Nature communications*, 11(1):1–11, 2020.

[23] Anuradha Singh, Saeed Ur Rehman, Sira Yongchareon, and Peter Han Joo Chong. Sensor technologies for fall detection systems: A review. *IEEE Sensors Journal*, 20(13):6889–6919, 2020.

[24] Ashish Singh and Kakali Chatterjee. Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115, 2017.

[25] Axel Steinhage and Christl Lauterbach. Sensfloor® and navifloor®: Robotics applications for a large-area sensor system. *International Journal of Intelligent Mechatronics and Robotics (IJIMR)*, 3(3):43–59, 2013.

[26] Aleksandar Tošić, Niki Hrovatin, and Jernej Vičič. Data about fall events and ordinary daily activities from a sensorized smart floor. *Data in Brief*, 37:107253, 2021.

[27] Aleksandar Tošić, Jernej Vičič, and Michael David Burnard. Privacy preserving indoor location and fall detection system. In *Human-Computer Interaction in Information Society : proceedings of the 22nd International Multiconference Information Society – IS 2019*, pages 9–12, 2019.

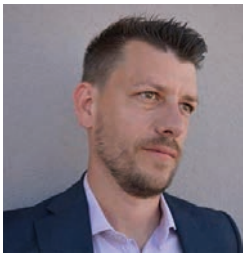
[28] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer, 2003.

[29] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE access*, 6:6900–6919, 2017.

[30] Salifu Yusif, Jeffrey Soar, and Abdul Hafeez-Baig. Older people, assistive technologies, and the barriers to adoption: A systematic review. *International journal of medical informatics*, 94:112–116, 2016.

Biographies

Niki Hrovatin was bor in Trieste, Italy, in 1994. He received the M.S. degree in Computer Science from the University of Primorska, Koper, Slovenija, in 2020. He is currently a Teaching Assistant and Ph.D. student at University of Primorska, and a Research Assistant at InnoRenewCoE, Izola, Slovenija. His current research interests include sensor networks, distributed systems, and blockchain.



Aleksandar Tošić. PhD candidate, teaching, and research assistant at University of Primorska, and Innorenew CoE. His fields of research are distributed, and decentralized systems, Peer to Peer networks, and distributed ledger technologies.

Jernej Vičič. Associate professor and research associate at the University of Primorska and Research Centre of the Slovenian Academy of Sciences and Arts. His research interests are quite broad ranging from language technologies to distributed systems.