Neurocomputing 491 (2022) 494–506

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Enabling automation and edge intelligence over resource constraint IoT devices for smart home

Mansoor Nasir [a], Khan Muhammad [b,*], Amin Ullah [c], Jamil Ahmad [a], Sung Wook Baik [d], Muhammad Sajjad [a,e,*]

[a] Digital Image Processing Laboratory, Department of Computer Science, Islamia College Peshawar, Peshawar 25000, KP, Pakistan
[b] Visual Analytics for Knowledge Laboratory (VIS2KNOW Lab), School of Convergence, College of Computing and Informatics, Sungkyunkwan University, Seoul 03063, Republic of Korea
[c] CORIS Institute, Oregon State University, Corvallis, OR 97331, USA
[d] Sejong University, Seoul 143-747, Republic of Korea
[e] Color and Visual Computing Lab, Department of Computer Science, Norwegian University of Science and Technology (NTNU), Gjøvik 2815, Norway

ARTICLE INFO

Article history:
Received 17 August 2020
Revised 11 March 2021
Accepted 14 April 2021
Available online 3 November 2021
Communicated by Chennai Guest Editor

Keywords:
Artificial intelligence
Edge intelligence
IoT
Smart home
Deep learning
Human fall detection

ABSTRACT

Smart home applications are pervasive and have gained popularity due to the overwhelming use of Internet of Things (IoT). The revolution in IoT technologies made homes more convenient, efficient and perhaps more secure. The need to advance smart home technology is necessary at this stage as IoT is abundantly used in automation industry. However, most of the proposed solutions are lacking in certain key areas of the system i.e., high interoperability, data independence, privacy, and optimization in general. The use of machine learning algorithms requires high-end hardware and are usually deployed on servers, where computation is convenient, but at the cost of bandwidth. However, more recently edge AI enabled systems are being proposed to shift the computation burden from the server side to the client side enabling smart devices. In this paper, we take advantage of the edge AI enabled technology to propose a fully featured cohesive system for smart home based on IoT and edge computing. The proposed system makes use of industry standards adopted for fog computing as well as providing robust responses from connected IoT sensors in a typical smart home. The proposed system employs edge devices as a computational platform in terms of reducing energy costs and provides security, while remotely controlling all appliances behind a secure gateway. A case study of human fall detection is evaluated by a custom lightweight deep neural network architecture implemented over the edge device of the proposed framework. The case study was validated using the Le2i dataset. During the training, the early stopping threshold was achieved with 98% accuracy for training set and 94% for validation set. The model size of the network was 6.4 MB which is significantly lower than other networks with similar performance.

1. Introduction

Over the last few decades, many researchers have focused on connecting everyday objects and weaving a web of interconnected devices. These can be physical devices, automobiles, embedded systems or everyday home appliances. The connectivity of these devices is enabled by the underlying technology known as Internet of Things (IoT) [1]. These devices can communicate with one another and seamlessly collect and transfer data in between to adapt and reciprocate to dynamic situations. The IoT is gaining tremendous traction from the IT industry as well as individual enthusiast alike. This is due to the fact that microcontroller technologies have evolved that enable these complex infrastructures and are more accessible to average developers. The use of single board computers also refers to as System on Chip (SoC) in IoT based systems, which enables rapid interconnection of environmental sensors and decent computational capabilities along with vision sensors that make it easy to develop fairly in large applications. Advancements in machine learning algorithms for mobile and low-power devices also pave way for intelligent devices in IoT systems.

A typical home automation system connects heterogeneous devices, collects data, and decides based on the observed data from these sensors. The means of communication between these devices is typically a Wi-Fi or Bluetooth, and in some cases a cellular network (3G/4G/5G) [2]. The first part of any smart home solution is

* Corresponding authors.
E-mail addresses: khan.muhammad@ieee.org (K. Muhammad), muhammad.sajjad@icp.edu.pk (M. Sajjad).

https://doi.org/10.1016/j.neucom.2021.04.138
0925-2312/

focused on the automation of the devices and providing an easy to use interface [3,4] that enables the end-user to interact with the system using graphical user interface (GUI). This enables the user to control different appliances i.e., fans, lights, entrance gates, etc., The other half of the system is focused on implementing smart AI enabled applications on edge devices thereby enabling smart devices that learn from the environment and help to reduce the cost of high-end deep learning servers. With the advancements in computer vision, the smart devices can detect intruders [5], enabling virtual fencing for boundary wall crossings [6], detect stationary abandoned objects [7], recognize faces for law enforcement [8] and many more. The heavy use of computer vision solutions in smart home implementations makes us curious to investigate further and improve upon the currently employed techniques. The main problem that these systems face is the computational cost which is nearly impossible for a low-power, resource constrained device to present real-time response to the end-user. One solution to this problem is to off-load computation burden to the cloud and transfer back the results in real-time. However, single cloud solutions often suffer from latency and may result in varied response time and hence are not reliable in many circumstances [9].

A typical scenario of the smart home is shown in Fig. 1. The smart home solution is composed of sensor layer that senses the environment and retrieves real-time data. In the case of a smart home, the sensors can be for light control, alarms, water management, surveillance cams, and temperature control, etc., These devices can be controlled and monitored via a smartphone app that is accessible through a secure and convenient Internet gateway. The gateway provides support to be connected through the local Internet using Wi-Fi or it can use cellular network for long range connectivity.

Most computer vision applications use Deep Neural Networks (DNNs) that enable computers to recognize objects and detect complex scenarios with almost human like accuracy. Modern mobile application-based solutions and embedded vision enabled devices are supported by DNNs and are becoming increasingly popular. DNNs not only provide high accuracy with reliable and consistent inference, they can also be applied to a broad spectrum of domains i.e., natural language processing [10], generative tasks [11], video processing [12,13], and speech recognition [14]. Despite the popularity and high accuracy of the DNNs, proposing a reliable solution with reasonably lower latency and lower energy consumption is very difficult mainly because of the computational requirements of the DNN. The only retreat from this problem is to look for a centralized cloud-based datacenter with presumably unlimited resources in terms of computation, storage and bandwidth. Connected devices generate the data, that is sent to the cloud and a DNN based model is trained over the cloud and results are relayed back to the device which it can use to make inferences about any given task.

However, this is a very rigid and cloud centric solution which can face latency issues as connected devices are growing each year and the amount of data each device produces is overwhelmingly larger due to the size and quality of the images and videos produced by cell phones and surveillance cameras [15]. This also produces another challenge, as such a large amount of data needs to be sent to the remote cloud via a cellular connection, which can be very expensive and consumes a tremendous amount of energy and bandwidth especially on mobile devices. To alleviate these issues and bottlenecks, and to avoid a cloud centric solution, an emerging field of edge computing can be exploited to present a very balanced solution, where the computational burden is shifted from the cloud to the edge of the network i.e., access points, base stations, routers and switches, etc., Due to its close proximity to the actual IoT devices, it enables an energy efficient, less computationally demanding, lower bandwidth consumption, and highly energy efficient solution with the same accuracy of the DNN. The edge-enabled solution for smart home is ideal because the nodes of the network are mostly stationary or confined to a restrictive premise. The use of DNN inference heavily relies on the available bandwidth between the server and the device. In case of a mobile device, the latency varies and depends on the distance and interference between nodes [16]. However, the restrictive nature of the nodes in smart home enables us to reliably infer the desired results without considering latency between the inference server and IoT devices.

The proposed work provides a solution to the above challenges by presenting a novel edge AI enabled DNN based solution for smart home. The framework is divided into two portions. In the first phase we use IoT infrastructure to automate and connect

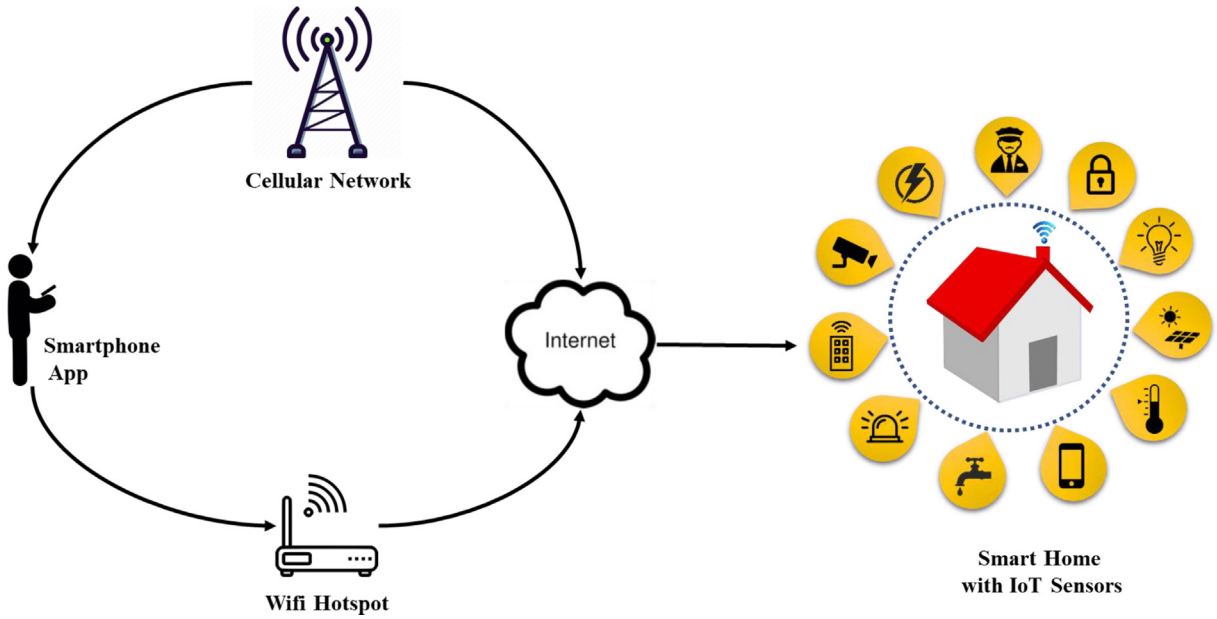


Fig. 1. Smart home with IoT sensors and a control app for end user.

control devices using a smartphone app. In the next phase, we use DNN to present a framework which exploits edge AI to enable deep embedded vision specifically optimized for resource constrained devices with minimum consumption of bandwidth and energy. A case study of human fall detection is implemented and evaluated using deep learning running on Jetson Nano with Message Queueing Telemetry Transport (MQTT) being the publish subscribe mechanism between devices. In summary, the contributions of this work are as follows:

- The proposed system uses IoT infrastructure via a smartphone app and resource constrained device as computational hub to automate smart home devices with minimum cost and energy, thereby reducing human interaction.
- A customized DNN architecture is designed and optimized to achieve transfer learning and computational offloading in smart home to gather actionable intelligence over a resource constrained edge device.
- A case study of human fall detection is implemented and evaluated to enable edge intelligence over a fully featured cohesive system for smart home based on IoT and edge computing.

The remainder of this paper is organized as follows: Section 2 reviews the state-of-the-art solutions presented in the literature in the recent past. In Section 3 we discuss the proposed solution in detail and Section 4 presents the results. Section 5 concludes the paper with potential future direction in related areas.

2. Literature review

IoT is a term used to represent different nodes, products, objects and literally *things* that are connected to each other via some medium, typically a wireless connection. These devices communicate to one another and pass information in between nodes. The typical communication method used in IoT infrastructure is MQTT, which is a machine-to-machine communication protocol that establishes a lightweight connection between resource constrained IoT devices [17]. Using MQTT protocol, the number of messages sent/receive is relatively low or where the bandwidth needs to be conserved for some other important tasks. The smartphone application can make use of MQTT protocol as it consumes less energy, provides high reliability, requires less data packets, and is ideal for broadcasting messages to multiple devices [18]. MQTT protocol is so versatile and bandwidth efficient, that it has been used in many long-range applications between satellite and servers using a dialup connection as well as for smaller home automation devices [19].

Home automation typically refers to an automated system being used to monitor the conditions within the house and perform some actuation based on a pre-defined criterion i.e., turning on the air condition based on the values of temperature sensors. However, with the advancements being made in the field of IoT and machine learning, these automation tasks are much better at regressing the process more efficiently. Using the same example of air conditioner, the machine learning approach can not only further automate the process, but can also predict the temperature for tomorrow and also provide energy consumption details of each item in the house [20] along with estimated monthly cost [21]. The automation process is mainly achieved by a centralized processing unit with lower energy requirements and ideally a cheaper device with less computational requirements as a whole. George et al. [22] proposed a secure smart home concept by incorporating the MQTT protocol as a gateway and encrypting the network traffic using an access control list. The proposed solution also provided visualization for data collected from sensors and conveniently displayed it on the webpage. The system makes use of single board computer Raspberry Pi where all sensors are connected to this device and the Raspberry Pi itself is connected to the Internet.

The concept of an automated and smart home is idealized by Alam et al., [23] where the authors proposed a smart home solution by connecting all the IoT devices with a single computational platform to perform inference on all data. A single board computer alone is not capable of handling such data in real-time, so in order to cope with this problem, the authors proposed a cloud computing based and web services-based solution. They concluded that by automating data and infrastructure, we can make lives easier for the household.

For any smart home to be acceptable in a smart city, it has to be secure. Focusing on this point, some authors including [8,24] proposed a blockchain based approach to encrypt the messages being transfered between devices. Rapid prototype applications are being developed by electronic market entrepreneurs for average consumers using low cost and low energy sensors and actuators. These solutions are not only affordable but also have practical value for the customers. The major obstacle for such automation solutions is the tradeoff between offline and online communication between devices in real-time. The data is acquired and sensed by the sensors while the server side is used to process the data. Constant communication channel between the actuators and server side is necessary and needs be reliable for viable communication. This is where the MQTT protocol comes in handy and allows the communication to be fully asynchronous [25].

The IoT infrastructure is based on very large number of physical, heterogeneous devices where one or many applications are driving it to achieve a higher goal. Some of the common applications of IoT includes intelligent transportation [26], smart grid [27,28], and intelligent irrigation systems [29]. Such systems are capable of providing tremendous amount of insight into the system and are able to prolific value to their organizations. However, these systems require efficient, high processing data servers with scalable data storage and real-time remote-control mechanisms with visualization capabilities [30]. One of the upcoming and truly innovative technology to achieve such demanding requirements is the Cloud of Things (CoT) [31]. The CoT not only provide presumably infinite computation and storage capabilities but also provide means of orchestration for seemingly heterogeneous devices. The use of CoT enables researchers to develop and evolve real-time systems capable of responding with huge number of connected IoT devices [32]. The CoT acts as a building block for any IoT solution with huge number of heterogeneous devices transparently connecting various applications. One such system is ClouT [33], where the aim is to connect IoT devices to the cloud thereby reducing the computational time and reduce the load on low-spec IoT devices incapable of processing huge streams of data, i.e., multimedia data from surveillance cameras in a smart city. The CoT is an elegant solution; however, it requires smart gateways to efficiently manage and route data to act as a presentational layer for the end user [34].

In view of the above arguments, CoT is a suitable solution for most of the IoT applications, however, the sheer number of devices connecting to the cloud has increased many folds in the past few years. A single cloud cannot comply with the demands of the IoT applications and it is more intensified in the real-time applications where a small delay ends up being the failure of the system [35]. For this purpose, the research is shifted from traditional cloud-based solutions to Fog/Edge Computing-based systems. Fog Computing is a recently introduced architecture and a paradigm in which the computing capabilities of a traditional cloud-based network is shifted from a centralized data centers to local end-user devices and networks. Fog computing principally extends the cloud computing architecture to the edge of the network which enables an innovative variety of silent services and applications for end-users. Edge computing is getting traction as it has more versatile

application potential than traditional cloud. Lightweight algorithms running on the edge of the network on IoT devices can conserve less bandwidth and provide ready-to-use computed data to the end user without consulting the cloud every single time [36]. Moreover, the edge can be equipped with AI to compute and analyze data in real-time thereby reducing the cost of cloud and bandwidth.

Smart applications are being developed using the AI integrated with IoT devices for healthcare, environmental monitoring, surveillance cameras, and banking. Leveraging the advancements in AI and machine learning, several pre-trained models are capable of recognizing everyday objects in surveillance videos. These systems can classify images but can also provide the location of the object within the image; the procedure known as object localization in computer vision. Typically, the data generated from IoT sensors are sent to the cloud for training and then training weights or pre-processed inference are sent back to the device. However, sending such a huge amount of data to the cloud is impractical and, in some situations, very expensive and time consuming due to the cloud centric nature of the CoT. To assuage the energy requirements and latency bottlenecks of the typical cloud-based solution, the edge AI [37] capabilities of the new and emerging paradigm is exploited in this work.

Computer vision technologies with advancements in DNNs made tremendous progress in the past decade and computers are more capable of recognizing faces and other everyday objects with human like accuracy. Cloud based vision APIs can provide relatively faster inferencing with using ordinary hardware. However, the size and inference time of these models is an issue and more robust algorithms and models are being proposed for mobile devices that can provide inference in real-time with single board computers i.e., Raspberry Pi or Jetson Nano. Models. For instance, MobileNet [38] and SqueezeNet [39] can swiftly recognize objects in real-time without computational overhead. In this work, we have used SqueezeNet model for human fall detection in smart home leveraging the edge AI technology, reducing reliance on cloud centric solution, providing proof-of-concept to deploy AI on Fog/Edge computing within the context of IoT to develop more diverse and evolving smart applications.

3. Proposed framework

Home automation, security, safety, privacy, and energy consumption is an emerging field of technology, and can be implemented with different microcontrollers and processors i.e., Desktop PC, laptop, Raspberry Pi, Arduino, etc., All these microcontroller and processors have their own pros and cons, but Raspberry Pi is more versatile and efficient option compared to other units. The smart home concepts can have many components including 1) Sensors, 2) Gateways, 3) Protocols, 4) Firmware, and 5) Cloud based databases.

The main focus of bringing AI to the smart home is to automate certain processes that need more than thresholds and pre-defined configurations. For example, one such task is fall detection. The process of integrating AI in smart home is more natural and requires significant amount of computational overhead. Today, most of the data generated by IoT devices resides outside the scope of the cloud so it is more natural to bring the machine learning capabilities closer to the source of the data and reduce the amount of bandwidth it consumes by transferring the entire dataset for each training session. Therefore, we leverage an approach called mobile edge computing [37], where the collaborative nature of the architecture makes it easy to train data locally on the device, and then send data to the higher-level layers of the DNN to train it on edge server. The remaining data is relayed to the cloud where the resource intensive tasks are offloaded and the remining mode is trained on the cloud. Fig. 2 shows the architecture derived from federated learning proposed by Konečný et al., [40]. The authors proposed a method called on demand deep learning co-inference with edge synergy. The authors argue that using the right sizing approach for DNN layers, the early exit point of the layer must be maintained to offload computationally intense tasks to the cloud from the comparatively resource constraint devices.

We make use of the same technique discussed here and build upon our own framework to work with SqueezeNet for fall detection using Jetson Nano as edge processing node. In this architecture, the data acquisition and processing layer acquires data and is processed locally on the Raspberry Pi as it is the main processing device for IoT devices. The Raspberry Pi itself is not capable of training the model as it lacks GPU capabilities that Jetson Nano possesses. The Jetson Nano receives the data and trains a preliminary model for text and multimedia data acquired from the surveillance camera. The multimedia data from the surveillance cameras are much larger in size compared to other devices, therefore the data is trained on the edge node, where the deep learning model is trained using Jetson Nano capable of performing certain level of computation. However, we assume that the number of multimedia devices may grow in the future and the edge device mentioned here may not respond to all data for training. For this purpose, we split the model and lower-level layers are placed on edge server, while the rest of the higher-level layers of the DNN are placed on the cloud with presumably an infinite amount of resources. The cloud model can receive weights from the edge server, it can also update parameters sent from the Jetson Nano or it can also be trained by the data directly sent to the cloud by the IoT devices itself. However, the data sent by the devices directly can compromise the privacy and may require additional security. The aggregation module of the cloud layer provides means of aggregation for all parameters sent to the cloud and it helps update all the weights of a single model that can send predictions and estimations along with trends to the IoT devices. In order to deploy AI services through cloud, we make use of cloud services provided by Amazon, called Amazon Web Services. It provides off the shelf technologies to integrate and deploy AI in IoT applications without too many complications. Moreover, it has built in dashboard to support Jetson Nano and is compatible with most IoT applications in general.

Details of the working of the edge device and device-based learning for the IoT devices are presented in Fig. 3. The edge layer is equipped with a slightly more powerful device Jetson Nano that acts as a Fog Node and performs machine learning tasks for text and multimedia data. The device receives data and sends it to a feature extraction model. Then the features are sent to train the model itself, where they can be assessed for offloading the cloud or evaluated for training accuracy. The final predictions are sent back to the device and thereafter to the cloud for storage and updating of the cloud-based learning model. The cloud sends updated parameters to the Raspberry Pi which controls and overlooks the connected IoT devices. In this case, the Raspberry Pi changes the values of certain IoT devices like temperature sensor, and surveillance cameras to change parameters. For instance, the temperature sensor can raise the threshold based on the values sent by the cloud model for the next day. In case of water management, the estimation module can predict the water requirements for the day based on values estimated by the model in previous days of the week. The estimation module uses a REST API and MQTT protocol to connect to the devices and updates the relevant sensor with appropriate results received from the model.

The use of deep learning architecture for human fall detection in a smart home is achieved through a model called SqueezeNet. This architecture is specifically designed to be implemented on

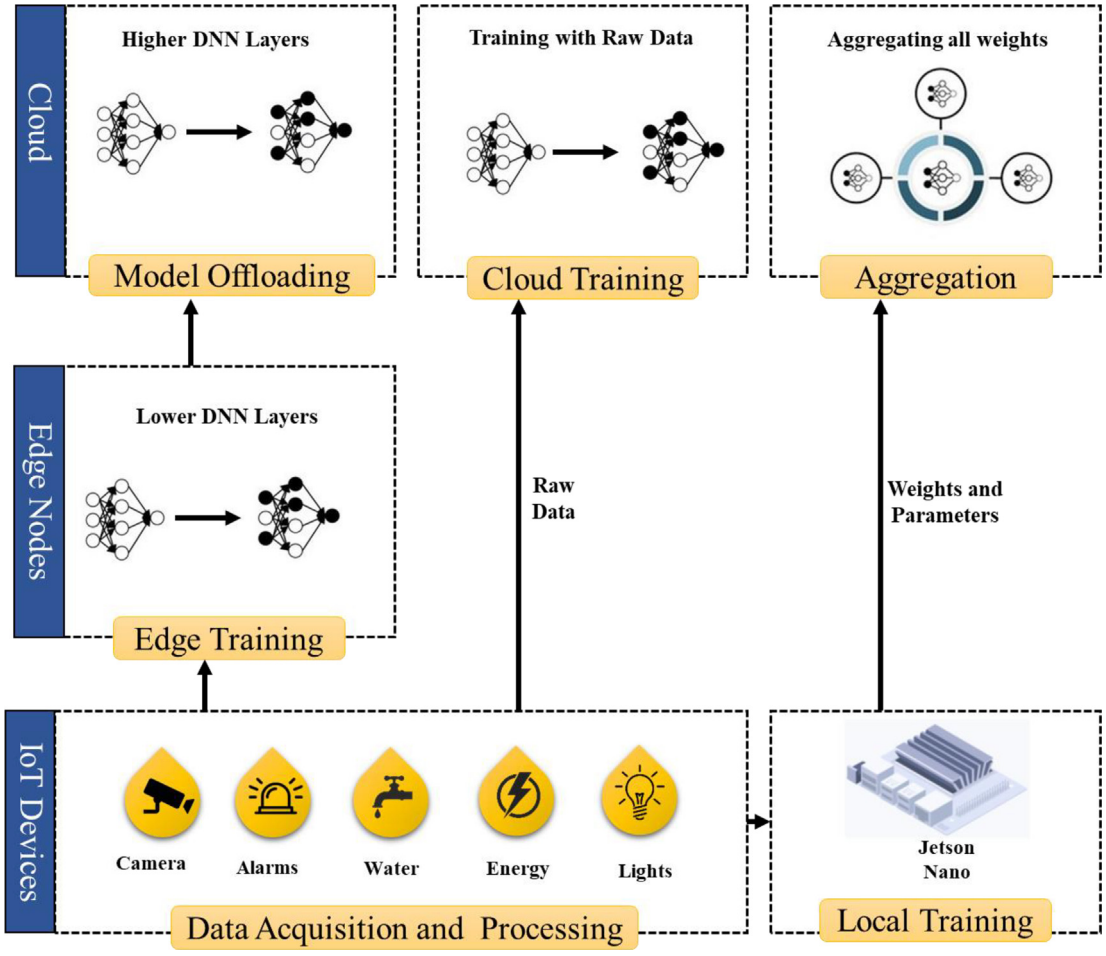


Fig. 2. Three ways deep learning architecture for IoT devices.

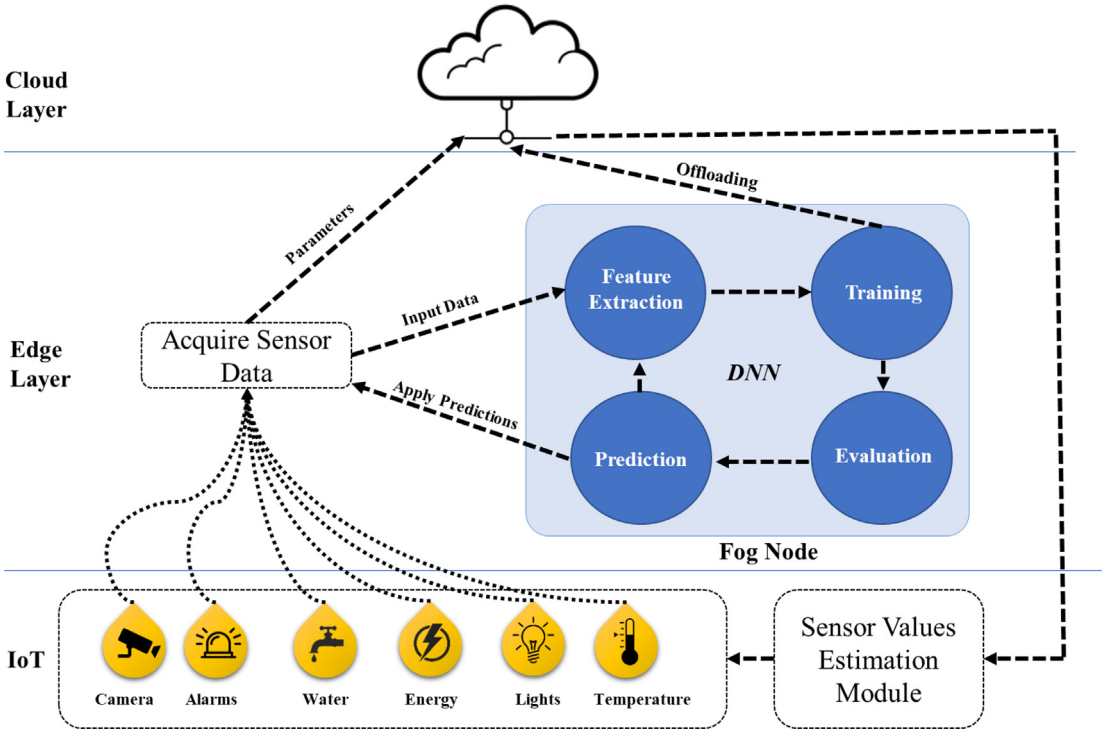


Fig. 3. Working of local learning using Jetson Nano as an edge device.

low power mobile devices with less computational resources. Compare to AlexNet [41], SqueezeNet is 500 times smaller in size and more than 3 times faster, which makes it ideal for our proposed architecture. The main architecture of SqueezeNet is presented in Fig. 4 (a). The main component of the SqueezeNet is called fire module which is presented in Fig. 4 (b). This module contains of two layers: one is called Squeeze and the other is called Expand. The SqueezeNet architecture is based on fire modules, convolution, and pooling layers. The fire module's squeeze layer tries to maintain the size of the feature map while the expand layer increases it. In order to get a high level of abstraction, a more common approach is to increase the depth of the architecture while maintaining the feature map size to minimum.

The convolutions presented in Fig. 4 (b) are 1x1 in fire module and work as fully connected layers while working on the same position of a feature point. All 1x1 convolutions are used to reduce the depth of the feature map. This implies that the 3x3 convolutions following the 1x1 have lesser computations to perform. This is specifically helpful as the 3x3 convolutions need 9 times more computations than the 1x1 convolutional operation.

Fig. 5 shows the proposed method for training the SqueezeNet model for fall detection. The model starts by augmenting the dataset and generating as much synthetic data as possible. Ee used [42] approach to augment the dataset. The augmented dataset is divided into three sets: training, testing, and validation. The training dataset is passed to the layers, the validation error is calculated at the end, and hyperparameters are updated using Gradient Descent with Momentum (SGDM) Optimization [43] technique. The model is trained for 87 epochs and the final model is obtained with a training accuracy of 99% and a validation accuracy of 96%. The lower layers of the model are passed for edge learning to the Jetson Nano while the higher layers are trained on the cloud where a GPU enabled PC is setup for the task. To better understand the bottlenecks of the SqueezeNet architecture, we evaluate the runtime and data yield by each layer of the architecture illustrated in Fig. 6. The heterogeneity of the layers depict that they produce non-consistent data at different layers, however, it is certain that the later layers produce more data and consume more time as it has a larger number of parameters. So, we decided to partition the layers and train the layers from conv_1 to fire_9 on edge device while from conv_10 to the soft_max are offloaded to the cloud for training.

Since, the availability of storage space on edge devices is scarce and a new model would take a very long time to train, we decided to make use of transfer learning in order to minimize the computational complexity and execution time of the network. By using transfer learning, we enabled the models to be trained on already labeled data for pre-defined tasks and models can be updated with relatively smaller number of labeled data with fine tuning of weights in existing layers. Moreover, by using transfer learning we can maximize the accuracy of the model by not initializing it with random weights. We make use of a pre-trained model that helps learn the actions more quickly and provides more accurate future classifications.

SqueezeNet is very efficient because of its smaller model size and computational parameters compared to AlexNet and other models. SqueezeNet has 18 layers and the number of parameters is 50 times less than AlexNet and the model size is usually less than 10 MB. The customized SqueezeNet architecture is shown in Fig. 7. The customized CNN architecture uses 16bit data type in contrast to the original 32bit resulting in lesser file size than the original. The original model size was 9.33 MB, however, after this change, the file size reduced to only 6.4 MB. Moreover, the customized model has now 360 times less parameters compared to the AlexNet and 16 times less parameters than the original. The transfer learning in this architecture is achieved by modifying the fully connected and classification layers of the pre-trained networks leaving the preliminary layers and their weights unchanged.

The fully connected layer is initialized with random weights and for optimization the Stochastic Gradient Descent with Momentum (SGDM) is used. The SGDM is faster than GDM in terms of convergence as the updated weight parameter Δw is collectively fused with the gradient in each iteration in a linear fashion. The SGDM optimizer is depicted in equation 1.

$$w := w - \alpha\Delta\Theta(w_i) + \eta\Delta w \tag{1}$$

The $\Theta(w_i)$ is the loss function that needs to be optimized. It is also known as cost function or objective function at the i^{th} instance of the data observed in each iteration. α is the learning rate and w is the weight parameter for activations and biases. η is the momentum parameter for temporal elements to update hyperparameters and Δw is the change observed in w in the last instance. In order to prevent network from overfitting, the initial layers of the model are

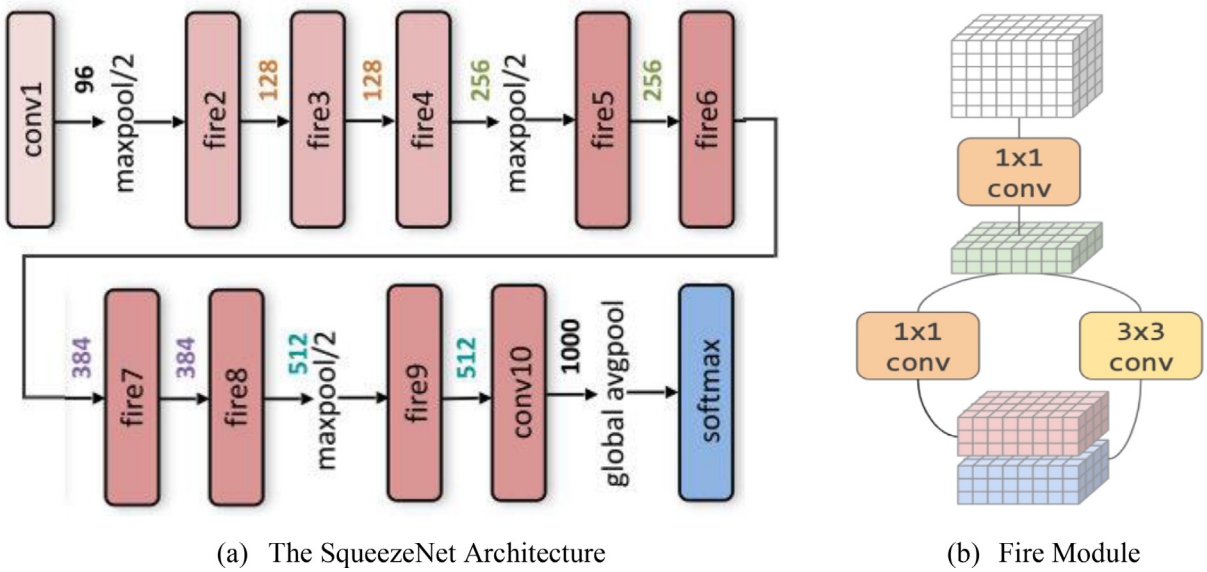


Fig. 4. (a) The overall SqueezeNet architecture (b) Single fire module.

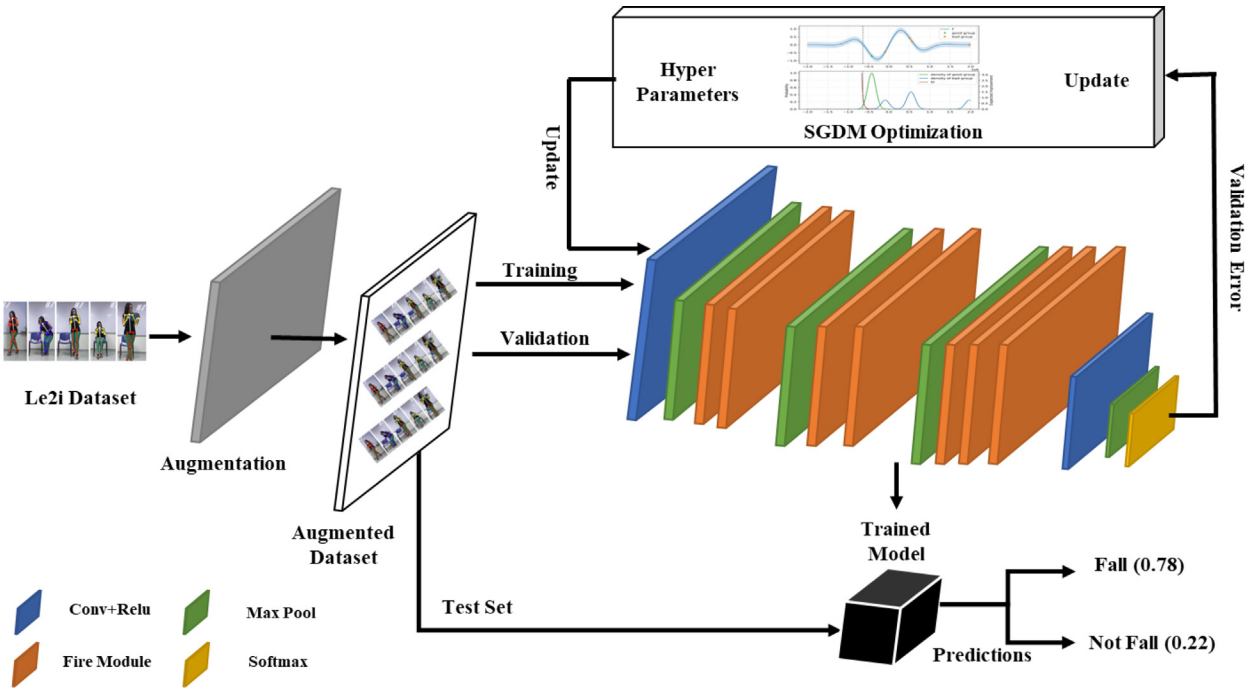


Fig. 5. The proposed SqueezeNet based fall detection system for smart home.

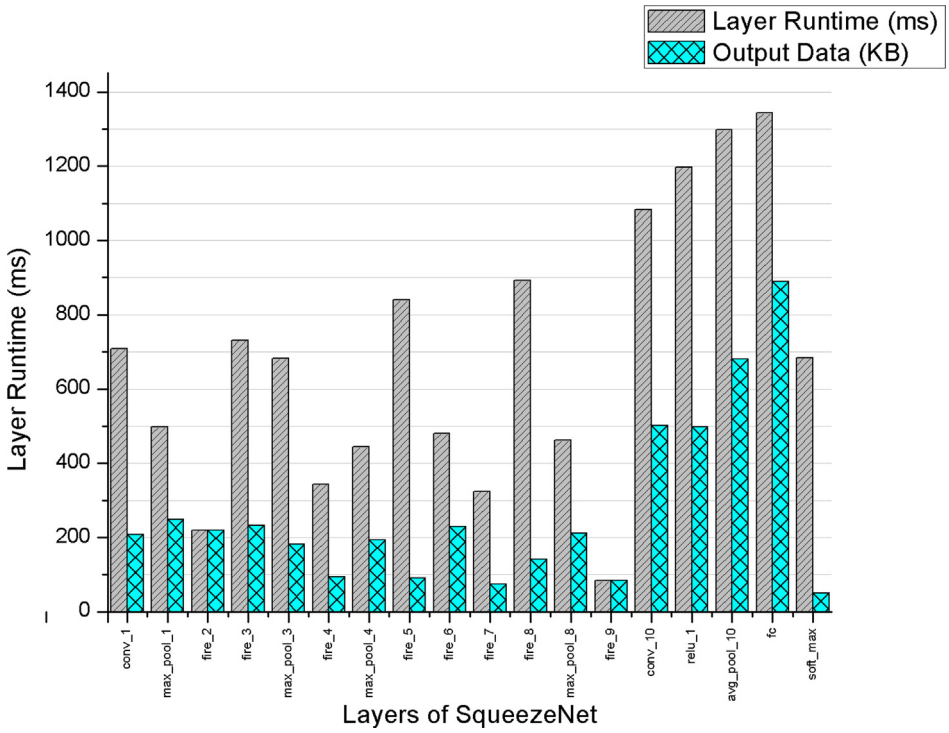


Fig. 6. The runtime and output data of SqueezeNet on Jetson Nano.

freezed and only the non-freezed layers get updated during the initial training phase. Furthermore, the data is passed from the augmentation phase where multiple operations including zooming, rotation, and scaling are applied to get meaningful features from the training dataset. It also helps minimize the possibility of overfitting during training. Furthermore, a distinct layer is added at the end of the network to remove some activations which also helps reducing the possibility of overfitting. The resulting network is robust, optimized and have far fewer computational parameters to be trained on edge device. Computational intense tasks are pushed to the cloud where the rest of the network is deployed to run the remaining higher layers and produce the output. Experimental results suggest the applicability and significance of the proposed architecture which is discussed in the next section.

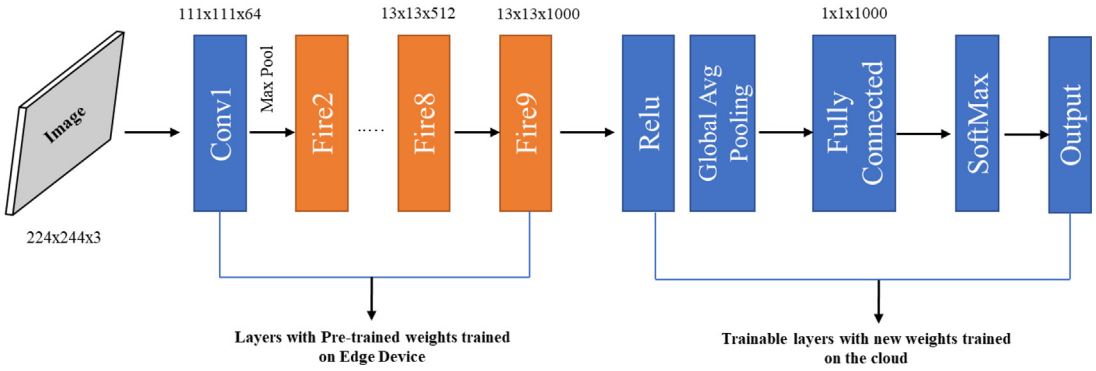


Fig. 7. SqueezeNet customized for training on the Edge device for fall detection.

4. Result and discussion

In this section, we describe the experimental setup and explain the automation and model evaluation process in detail. The automation of the smart home is achieved by using Raspberry Pi as central processing and interfacing hub for all IoT devices. The system makes use of other devices such as H-bridge, relays, power supply to control several devices including fingerprint reader to open door motors. Fig. 8 demonstrates the interfacing of various devices with Raspberry Pi in which the red wires show voltage supply, black wires show the ground voltage connectivity, and the blue wires indicate the connection of I/O pins of different sensors with General-Purpose Input Output (GPIO)s. All of these devices are connected to GPIO pins except the camera which is connected to default SCSI camera port. The optical fingerprint device is connected to the USB port through TTL connector.

Raspberry Pi is a single board computer equipped with all the necessary equipment to facilitate the project. A number of model

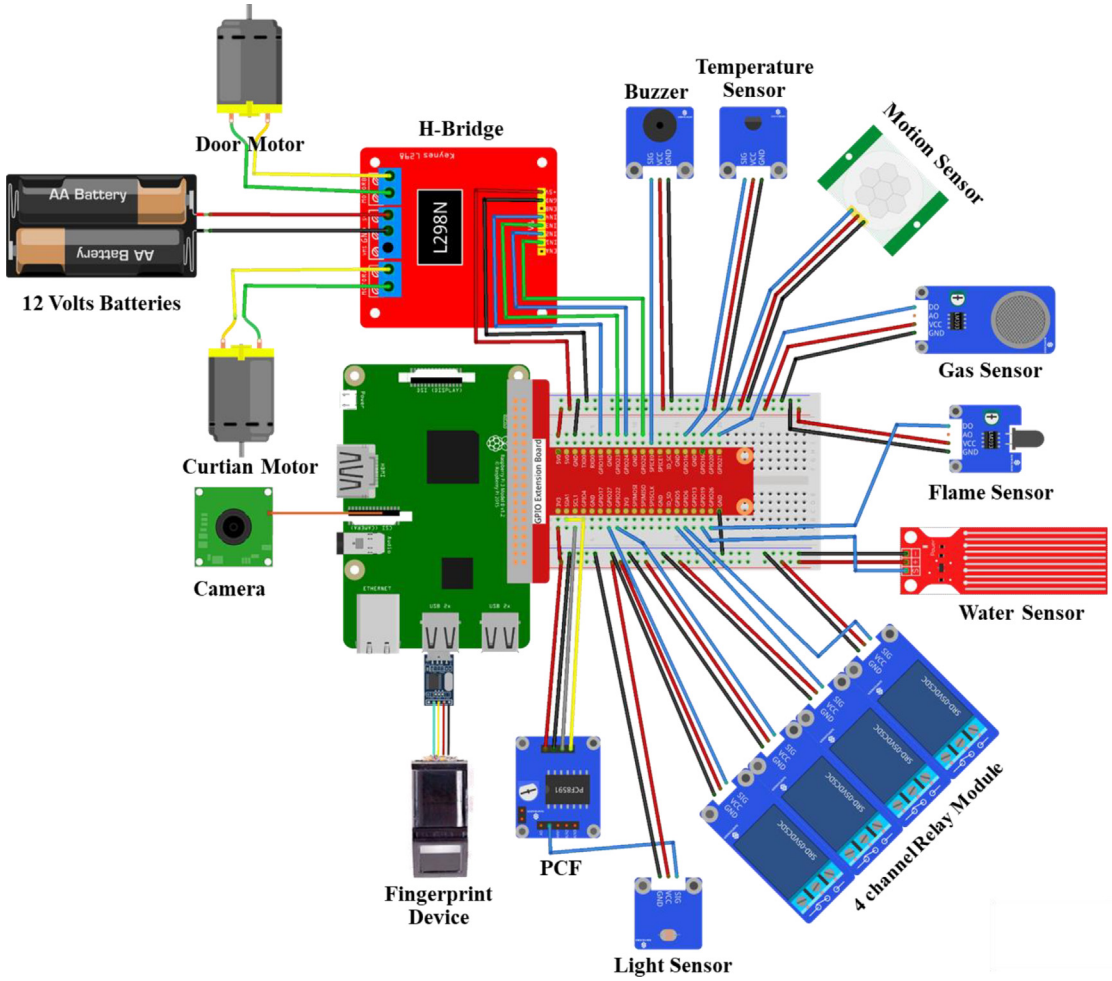


Fig. 8. Interfacing IoT sensors to the Raspberry Pi for smart home automation.

choices are available and choosing the right one can make the design process of the smart home project easier. In order to choose the right one, we analyzed all the available models and weigh the advantages and disadvantages of each model. The summary of the available models is presented in Table 1. As we can see, the Raspberry Pi model 3B + has most advantages with more available memory and expandable storage options. This specific model has the most GPIO pins and USB ports, which help expand the connectivity of different devices natively. This model has Universal Asynchronous Receiver/Transmitter (UART), I2C, and analog and digital connections along with SPI availability, helping to connect more diverse peripherals simultaneously.

Raspberry Pi is not the only option available to automate the smart home application, and there are a number of compelling options available for this task. Choosing the right wireless sensor node can make a huge impact on developing the prototype more rapidly. Table 2 summarizes and compares a number of available prototyping platforms for wireless sensor networks. We evaluate and assess the cost, size, weight, operating system, available memory, and storage of each platform. Based on these comparisons we concluded that Raspberry Pi is an affordable, suitable, and versatile platform for smart home automation, that supports higher level languages with sizable memory and expandable storage that supports IoT devices and has huge online community.

The automation of smart home involves interfacing of different IoT devices and acquiring and pre-processing the values retrieved from these scalar sensors. Since the data generated by these sensors are mainly discrete values and almost non-existent multimedia data, the processing of such information on Raspberry Pi is doable with very little latency. However, when it comes to multimedia data generated by camera attached to Raspberry Pi, the amount of data is staggering. Raspberry Pi alone cannot simply handle the data and cannot make inference in real-time. In order to process and infer from the multimedia data, we needed to choose a platform that supports real-time processing capabilities of multimedia data with significantly better performance in AI. For this purpose, we evaluated a number of platforms and compared the memory, cost, energy requirements, language support, and most importantly AI performance. The summary of these platforms is available in Table 3. The availability of 120 CUDA cores in Jetson Nano makes it an ideal candidate for the task of fall detection using DNNs. The cost is twice as high from the Raspberry Pi, however, in terms of AI performance, there is no better platform available for multimedia data processing than Jetson Nano. It supports all high-level languages and provides 40 pin interfaces for IoT devices with support for all Linux based distributions.

5. Dataset

To evaluate the performance of the proposed methodology we used videos from the Le2i dataset. This dataset contains 191 videos with varying difficulties including occlusion, illumination changes, and textured backgrounds. Each video is at 25 frames per second with constant resolution of 320×240 pixels. The videos contain different scenarios where the actors perform falling and standing

Table 1
Different models and specifications of Raspberry Pi.

Specifications	Model A	Model B	Model 3B+
Storage	SD Card	SD Card	MicroSD Card Slot up to 64 GB
System on Chip (SoC)	BCM2835 Broadcom	BCM2835 Broadcom	BCM2835 Broadcom
Memory	SD RAM 256 MB @ 400Mhz	SD RAM 512 MB @ 400Mhz	SD RAM 1 GB @ 400Mhz
Power Draw	5 V with 1.2A-600 mA	5 V with 1.2A-750 mA	5 V with 1.8A-600 mA
GPIO Pins	26	26	40
USB 2.0	1x Port	2x Ports	4x Ports

Table 2
Comparison of different wireless sensor nodes along with their specifications and price.

Name	RAM	Processor	OS	External Storage	Weight (g)	Cost ($)	Size (mm)
Raspberry Pi	256 MB to 1 GB	BCM2835 Broadcom SoC Processor	Raspbian (Debian)	Up to 64 GB	45	15–35	85x54x17
Lotus	64 k	LPC1758	Rtos	512 k Flash	18	300	76x34x7
TelosB	10 k	MSP430	TinyOS	48 k Flash	23	99	65x31x6
Cricket	4 k	ATMEL128L	TinyOS	512 k Flash	18	225	58x32x7
Iris	8 k	ATMEGA1281	Mote	128 k Flash	18	115	58x32x7

Table 3
Comparison of different prototyping platforms along with their specifications.

Board	Chip	RAM	Energy	OS	Language Support	Analog Pins	GPIO	USB	AI Performance (GFLOPs)	Cost	Size	Weight
Jetson Nano	1.4Ghz Quad Core Cortex A57	4 GB	5 V	All Linux Distro	Java, Python, C, C++, CUDA	–	40	4x USB 3.0	∼472	99	100x79	60
Raspberry Pi 3B+	BCM2835 Processor	256 MB-1 GB	5 V	Raspbian	Python, Java, C, C++	0	20–40	1–4 2.0	∼21.4	25–35	85x54x17	42–45
Beagle Bone	ARM AM335 @ 1Ghz	512 MB	5 V	Linux Angstrom	C, C++	6	14	1	∼19.1	45	86x53	40
Phidgets	SBC	64 MB	6 V to 15 V	Linux	VB, C#, Java, MATLAB, Python	8	80	6	∼14.6	50–200	81x53	60
Udoo	ARM Cortex A9	1 GB	6 V to 15 V	Debian, Android	Arduino, C, C++, Java	14	62 + 14	5	∼20.2	99–135	110x85	120–170

Table 4
Snippets of the Le2i fall detection dataset for indoor scenarios.

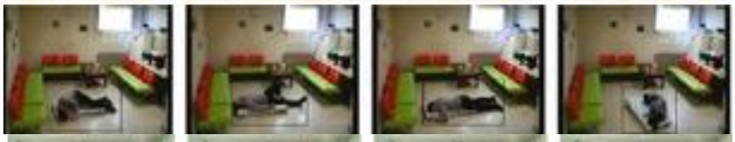
Scenario	Samples
Home	
Office	
Lecture Room	
Coffee Room	

Table 5
Number of labeled instances available in Le2i dataset along test and train split.

Scenario	Training	Frames	Fall Events	Non-Fall Events	Testing
Home	45	10,201	3705	6496	15
Office	31	12,182	7232	4950	16
Lecture Room	13	9239	1247	7992	9
Coffee Room	53	16,213	4891	11,322	17

including, lecture rooms, home, office, and coffee room. A sample of the dataset images are presented in Table 4 showing all the indoor scenarios.

The Le2i dataset contains 191 videos that are split in training and testing where 70% (142 videos) of the dataset were used in training and 30% (57) videos were used in validation. Videos were converted to frames where fall and non-fall events were separated and are presented in Table 5 along with test train split.

6. Training and evaluation

As discussed in the previous section, we used a customized SqueezeNet DNN architecture to train a model for fall detection in smart home using deep embedded vision. The purpose of choosing the SqueezeNet was to train a model that not only performs better but also has less model size compared to other available networks. In the initial phase, the network was trained with a GPU

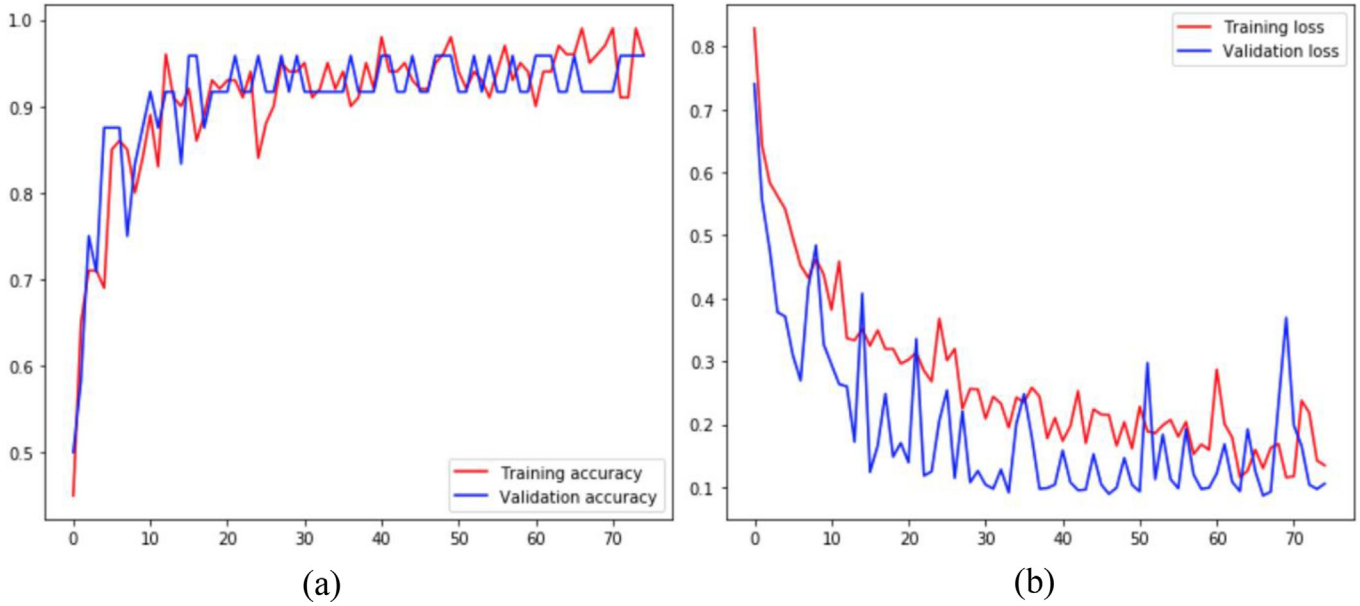


Fig. 9. (a) Training and validation accuracy (b) Training and validation loss.

with all the training data, and the number of iterations was set to 2000 with batch size of 64. The learning rate for the training was set to 0.001 with the step size of 500. The gamma was set to 0.1 and the momentum value was set to 0.9. During the training, the early stopping threshold was achieved on 8th epoch achieving 98% accuracy for training dataset and 94% for testing dataset. Training loss was calculated at 0.086 while validation loss was recorded at 0.099. The validation and training loss along with validation and training accuracy is presented in Fig. 9. The model size of the network was 6.4 MB which is significantly lower than other networks with similar performance.

In order to further evaluate the performance of the model, we split the dataset into smaller chunks with observed and cumulative values from the dataset shown in Table 6. In order to plot the Receiver operating characteristics (ROC) curve we needed to evaluate the False Positive Rate (FPR) and True Positive Rate (TPR) along with Area Under the Curve (AUC). True Positive (TP) is a scenario where the model classifies a frame as Fall and it actually is a Fall. Non-Fall frame classified as Fall is considered as False Positive (FP). In frames where the Fall was present and they were classified as Non-Fall, are False Negative (FN) cases. The frames where the fall was not present and the classifier also marked them as NonFall, is considered as True Negative (TN). Based on these values, the sensitivity and specificity are calculated using equation 2 and 3, respectively.

$$\text{Sensitivity} = \frac{I_{TP}}{I_{TP} + I_{FN}} \tag{2}$$

$$\text{Specificity} = \frac{I_{TN}}{I_{FP} + I_{TN}} \tag{3}$$

Based on a number of evaluations and experiments with tuning hyperparameters we come up with the highest accuracy of 95% on the validation set. The ROC curve of the fall detection is plotted for the Le2i dataset in Fig. 10. The AUC is used a reference data to observe the accuracy and convergence of the model. The ROC curve is constructed against sensitivity and TPR or 1-Specificity of the model.

There are a number of comparisons criteria that we need to consider before comparing the proposed work with other techniques present in the literature. There are two distinct methods of fall detection: one is by using wearable sensors while the other is vision based. Since, the proposed method is also vision based, so we are comparing the methods based on vision sensors only.

In order to evaluate the accuracy of the trained model, a conclusive comparison with other fall detection techniques based on computer vision is conducted. The proposed system outperforms most of the existing systems and provide good results with sensitivity of 95.35%, specificity of 94.34%, and a total overall accuracy of 94.79%. The results of our experiments show that a relatively better accuracy is achieved without using high-end hardware. The comparison of the proposed system along with other systems is presented in Table 7.

The trained model was evaluated on three different platforms and architectures for different number of images from the dataset.

Table 6
Split-up dataset for in-depth evaluation of the model.

Scenario	Precision	Recall	F-measure
home_1	0.92	0.74	0.92
coffee_room_1	0.90	0.95	0.93
office_1	0.98	0.75	0.88
lecture_room_1	0.97	0.97	0.97
home_2	0.94	0.97	0.97
coffee_room_2	0.89	0.88	0.88
office_2	0.87	0.92	0.88
home_3	0.88	0.98	0.98
lecture_room_2	0.96	0.94	0.94
office_3	0.98	0.98	0.98

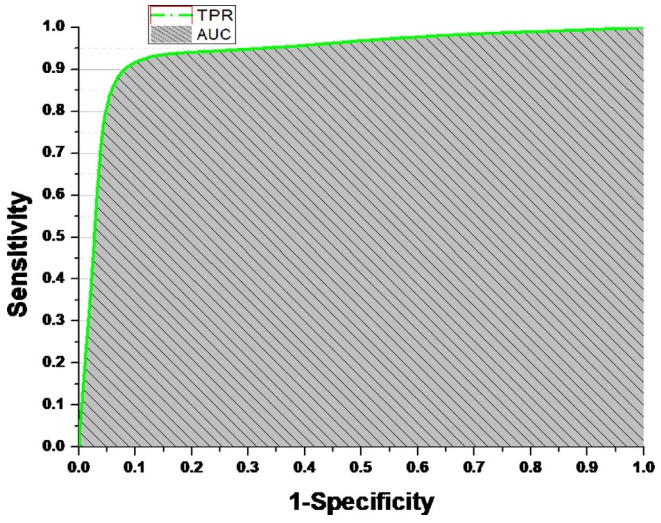


Fig. 10. ROC curve of the Le2i dataset with reference testing data.

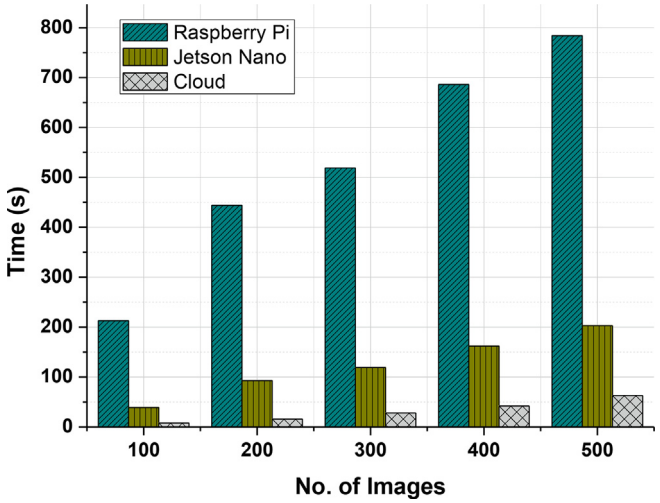


Fig. 11. Classification time (s) for different datasets and platforms.

Table 7
Comparison of the different vision-based fall detection techniques with the proposed system conducted on videos from Le2i dataset.

Method	Accuracy (%)	F-Score	FPR	FNR	Precision	Sensitivity	Specificity
Proposed	94.79	0.9425	0.0566	0.0465	0.9318	0.9535	0.9434
M. Chamle [44]	79.31	0.8182	0.2692	0.1563	0.7941	0.8438	0.7308
Poonsri [45]	86.21	0.9111	0.3571	0.0682	0.8913	0.9318	0.6429
Marcos [46]	94.21	0.9370	0.1087	0.0326	0.9468	0.9674	0.8913
Kwolek [47]	90.00	0.9271	0.2083	0.0430	0.8990	0.9570	0.7917
Wang [48]	89.20	0.9175	0.2182	0.0430	0.8812	0.9470	0.7818

We specifically observed the inference time of each platform for a batch of images randomly sampled from the dataset. Although the accuracy of the model remains the same in general, the execution time on each platform varied widely.

Fig. 11 summarizes the executing time in seconds for all platforms used in the proposed architecture. The Raspberry Pi has comparatively lower GPU capacity and remains the worst performing platform. Jetson Nano having 120 CUDA cores, has significantly higher frame classification rate than Raspberry Pi and Cloud being the best of all of these three plateforms.

7. Conclusion

In this paper, we presented a fully featured IoT based smart home automation with computer vision applications deployed over the edge devices. The proposed framework takes benefit of the fog/edge computing technology for computing high demanding tasks such as training and inferencing deep neural network. A case study of human fall detection was implemented and evaluated using SqueezeNet architecture from which we concluded that such demanding task cannot be achieved by all resource constrained devices and only some of them are capable of providing real-time computational capabilities. The proposed framework provides a proof-of-concept design to extend the existing technology to a new level and introduces fog-based multimedia data processing using deep learning which enables the low-cost computation over the edge of the network. This not only reduces the computational cost but also the network bandwidth and storage over the costly cloud solutions. In the future, we are planning to test and evaluate newer deep learning architectures to provide a comprehensive survey of different networks over resource constrained devices.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the faculty research fund of Sejong University in 2019.

This work was also supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2019-0-00136, Development of AI-Convergence Technologies for Smart City Industry Productivity Innovation).

References

[1] K. Ashton, That 'internet of things' thing, RFID journal 22 (7) (2009) 97–114.
[2] T.-H. Kim, C. Ramos, S. Mohammed, Smart city and IoT, Elsevier, 2017.
[3] R.T. Fielding, R.N. Taylor, Architectural styles and the design of network-based software architectures Irvine Doctoral dissertation, University of California, 2000.
[4] D. Romero, G. Hermosillo, A. Taherkordi, R. Nzekwa, R. Rouvoy, F. Eliassen, RESTful integration of heterogeneous devices in pervasive environments, in: IFIP International Conference on Distributed Applications and Interoperable Systems, Springer, 2010, pp. 1–14.
[5] A. Shahid et al., Computer vision based intruder detection framework (CV-IDF), in: 2017 2nd International Conference on Computer and Communication Systems (ICCCS), IEEE, 2017, pp. 41–45.
[6] B. T. Morris, M. M. J. I. t. o. c. Trivedi, and s. f. v. technology, "A survey of vision-based trajectory learning and analysis for surveillance," vol. 18, no. 8, pp. 1114-1127, 2008.
[7] N. Bird, S. Atev, N. Caramelli, R. Martin, O. Masoud, and N. Papanikolopoulos, "Real time, online detection of abandoned objects in public areas," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 3775-3780: IEEE.
[8] M. Sajjad *et al.*, "Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities," vol. 108, pp. 995-1007, 2020.
[9] A. Mtibaa, A. Fahim, K. A. Harras, and M. H. J. A. S. C. C. R. Ammar, "Towards resource sharing in mobile device clouds: Power balancing across mobile devices," vol. 43, no. 4, pp. 51-56, 2013.
[10] T. Young, D. Hazarika, S. Poria, and E. J. i. C. i. m. Cambria, "Recent trends in deep learning based natural language processing," vol. 13, no. 3, pp. 55-75, 2018.
[11] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672-2680.
[12] M. Nasir, K. Muhammad, J. Lloret, A. K. Sangaiah, M. J. J. o. P. Sajjad, and D. Computing, "Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities," vol. 126, pp. 161-170, 2019.
[13] Muhammad Sajjad, Sana Zahir, Amin Ullah, Zahid Akhtar, Khan Muhammad, Human behavior understanding in big multimedia data using CNN based facial expression recognition, Mobile Networks Appl. 25 (4) (2020) 1611–1621.
[14] D. Amodei et al., Deep speech 2: End-to-end speech recognition in english and mandarin, in: International Conference On Machine Learning, 2016, pp. 173–182.
[15] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, Q. Li, Lavea: Latency-aware video analytics on edge computing platform, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, 2017, pp. 1–13.
[16] P. Liu, B. Qi, S. Banerjee, Edgeeye: An edge service framework for real-time intelligent video analytics, in: Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking, 2018, pp. 1–6.
[17] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, C.K.-Y. Tan, Performance evaluation of MQTT and CoAP via a common middleware, in: 2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP), IEEE, 2014, pp. 1–6.
[18] R. A. J. J. o. O. S. S. Light, "Mosquitto: server and client implementation of the MQTT protocol," vol. 2, no. 13, p. 265, 2017.
[19] R. J. Nunes and J. C. Delgado, "An Internet application for home automation," in *2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No. 00CH37099)*, 2000, vol. 1, pp. 298-301: IEEE.
[20] Fath U Min Ullah, Amin Ullah, Ijaz Ul Haq, Seungmin Rho, Sung Wook Baik, Short-term prediction of residential power energy consumption via CNN and multilayer bi-directional LSTM networks, IEEE Access 8 (2020) 123369–123380.
[21] W. H. Kim, S. Lee, and J. J. P. C. S. Hwang, "Real-time energy monitoring and controlling system based on Zigbee sensor networks," vol. 5, pp. 794-797, 2011.
[22] S. G. Varghese, C. P. Kurian, V. George, A. John, V. Nayak, and A. J. I. W. S. S. Upadhyay, "Comparative study of ZigBee topologies for IoT-based lighting automation," vol. 9, no. 4, pp. 201-207, 2019.
[23] M. R. Alam, M. B. I. Reaz, M. A. M. J. I. t. o. s. Ali, man,, and p. C. cybernetics, "A review of smart homes—Past, present, and future," vol. 42, no. 6, pp. 1190-1203, 2012.
[24] A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, Blockchain for IoT security and privacy: The case study of a smart home, in: 2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops), IEEE, 2017, pp. 618–623.
[25] M.B. Yassein, M.Q. Shatnawi, S. Aljwarneh, R. Al-Hatmi, Internet of Things: Survey and open issues of MQTT protocol, in: 2017 International Conference on Engineering & MIS (ICEMIS), IEEE, 2017, pp. 1–6.
[26] L. F. Herrera-Quintero, J. C. Vega-Alfonso, K. B. A. Banse, and E. C. J. I. I. T. S. M. Zambrano, "Smart its sensor for the transportation planning based on iot approaches using serverless and microservices architecture," vol. 10, no. 2, pp. 17-27, 2018.
[27] F. Leccese, M. Cagnetti, and D. J. S. Trinca, "A smart city application: A fully controlled street lighting isle based on Raspberry-Pi card, a ZigBee sensor network and WiMAX," vol. 14, no. 12, pp. 24408-24424, 2014.
[28] Z. A. Khan, T. Hussain, A. Ullah, S. Rho, M. Lee, and S. W. J. S. Baik, "Towards Efficient Electricity Forecasting in Residential and Commercial Buildings: A Novel Hybrid CNN with a LSTM-AE based Framework," vol. 20, no. 5, p. 1399, 2020.
[29] M.N. Rajkumar, S. Abinaya, V.V. Kumar, Intelligent irrigation system—An IOT based approach, in: 2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT), IEEE, 2017, pp. 1–5.
[30] M.M. Trompouki, L. Kosmidis, Towards general purpose computations on low-end mobile GPUs, in: 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2016, pp. 539–542.
[31] R. Petrolo, R. Morabito, V. Loscrì, and N. J. A. o. T. Mitton, "The design of the gateway for the cloud of things," vol. 72, no. 1-2, pp. 31-40, 2017.
[32] P. Wright and A. Manieri, "Internet of Things in the Cloud," in *Proc. of the 4th Int. Conf. on Cloud Computing and Services Science*, 2014.
[33] K. Tei, L. Gürgen, ClouT: Cloud of things for empowering the citizen clout in smart cities, in: 2014 IEEE World Forum on Internet of Things (WF-IoT), IEEE, 2014, pp. 369–370.
[34] R. Petrolo, V. Loscri, and N. J. T. o. E. T. T. Mitton, "Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms," vol. 28, no. 1, p. e2931, 2017.
[35] K. Jang, J. Sherry, H. Ballani, T. Moncaster, Silo: Predictable message latency in the cloud, in: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, 2015, pp. 435–448.
[36] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012, pp. 13–16.
[37] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. J. P. o. t. I. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," vol. 107, no. 8, pp. 1738-1762, 2019.
[38] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[39] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. J. a. p. a. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," 2016.
[40] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. J. a. p. a. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016.
[41] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.
[42] T. DeVries and G. W. J. a. p. a. Taylor, "Dataset augmentation in feature space," 2017.
[43] N. Loizou and P. J. a. p. a. Richtárik, "Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods," 2017.
[44] M. Chamle, K. Gunale, K. Warhade, Automated unusual event detection in video surveillance, 2016 International Conference on Inventive Computation Technologies (ICICT) vol. 2 (2016) 1–4.
[45] A. Poonsri, W. Chiracharit, Fall detection using Gaussian mixture model and principle component analysis, in: 2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE), IEEE, 2017, pp. 1–4.
[46] A. Núñez-Marcos, G. Azkune, I. J. W. c. Arganda-Carreras, and m. computing, "Vision-based fall detection with convolutional neural networks," vol. 2017, 2017.
[47] B. Kwolek and M. J. N. Kepski, "Improving fall detection by the use of depth sensor and accelerometer," vol. 168, pp. 637-645, 2015.
[48] S. Wang, L. Chen, Z. Zhou, X. Sun, J. J. M. t. Dong, and applications, "Human fall detection in surveillance video based on PCANet," vol. 75, no. 19, pp. 11603-11613, 2016.

Mansoor Nasir received his M.S. degree in Computer Science from Institute of Management Sciences, Peshawar, Pakistan. He is currently pursuing Ph.D. from Islamia College, Peshawar, Pakistan. His research interest includes Network Security, Distributed Systems, and Trusted Computing.

Khan Muhammad received the Ph.D. degree in digital contents from Sejong University, Seoul, South Korea, in 2019. He is currently an Assistant Professor with the School of Convergence, College of Computing and Informatics, Sungkyunkwan University, Seoul. He is a Professional Reviewer for over 100 well-reputed journals and conferences. He has registered eight patents and published over 170 articles in peer-reviewed international journals and conferences in his research areas. His research interests include medical image analysis, information security, video summarization, computer vision, fire/smoke scene analysis, IoT/IoMT, and video surveillance.

Amin Ullah received Ph.D. degree in digital contents from Sejong University, Seoul, South Korea. He is currently working as a Postdoc Researcher at the CoRIS Institute, Oregon State University, Corvallis, Oregon, USA. His major research focus is on human action and activity recognition, sequence learning, image and video analytics, content-based indexing and retrieval, 3D point clouds, IoT and smart cities, and deep learning for multimedia understanding. He has published several papers in reputed peer reviewed international journals and conferences including IEEE Transactions on Industrial Electronics, IEEE Transactions on Industrial Informatics, IEEE Transactions on Intelligent Transportation Systems, IEEE Internet of Things Journal, IEEE Access, Elsevier Future Generation Computer Systems, Elsevier Applied Soft Computing, International Journal of Intelligent Systems, Springer Multimedia Tools and Applications, Springer Mobile Networks and Applications, and IEEE Joint Conference on Neural Networks.

Jamil Ahmad received the B.C.S. degree (with Distinction) in computer science from the University of Peshawar, Peshawar, Pakistan, in 2008, the master's degree with specialization in image processing from Islamia College Peshawar, Peshawar, in 2014, and Ph.D. degree in digital contents from Sejong University, Seoul, South Korea in 2018. He is currently a regular Faculty Member with the Department of Computer Science, Islamia College Peshawar. His current research interests include deep learning, medical image analysis, content-based multimedia retrieval, and computer vision. He has published several journal articles in the above areas in reputed journals, including the Journal of Real-Time Image Processing, Multimedia Tools and Applications, Journal of Visual Communication and Image Representation, PLoS One, Journal of Medical Systems, Computers and Electrical Engineering, SpringerPlus, Journal of Sensors, and KSII Transactions on Internet and Information Systems.

Sung Wook Baik is a Full Professor at Department of Digital Contents and Head of Intelligent Media Laboratory (IM Lab), Sejong University, Seoul, Korea. He received the Ph.D. degree in Information Technology Engineering from George Mason University, Fairfax, VA, in 1999. He worked at Datamat Systems Research Inc. as a senior scientist of the Intelligent Systems Group from 1997 to 2002. Since 2002, he is serving as a faculty member at department of Digital Contents, Sejong University. His research interests include image processing, pattern recognition, video analytics, Big Data analysis, multimedia data processing, energy load forecasting, IoT, IIoT, and smart cities. His specific areas in image processing include image indexing and retrieval for various applications and in video analytics the major focus is video summarization, action and activity recognition, anomaly detection, and CCTV data analysis. He has published over 100 papers in peer-reviewed international journals in the mentioned areas with main target on top venues of these domains including IEEE IoTJ, TSMC-Systems, COMMAG, TIE, TII, Access, Elsevier Neurocomputing, FGCS, PRL, Springer MTAP, JOMS, RTIP, and MDPI Sensors, etc. He is serving as a professional reviewer for several well-reputed journals and conferences including IEEE Transactions on Industrial Informatics, IEEE Transactions on Cybernetics, IEEE Access, and MDPI Sensors. He is a member of IEEE. He is involved in several projects including AI-Convergence Technologies and Multi-view Video Data Analysis for Smart Cities, Effective Energy Management Methods, Experts' Education for Industrial Unmanned Aerial Vehicles, Big Data Models Analysis etc. supported by Korea Institute for Advancement of Technology and Korea Research Foundation. He has several Korean and an internationally accepted patent with main theme of disaster management, image retrieval, and speaker reliability measurement.

Muhammad Sajjad received the master's degree from the Department of Computer Science, College of Signals, National University of Sciences and Technology, Rawalpindi, Pakistan, in 2012, and the Ph.D. degree in digital contents from Sejong University, Seoul, South Korea, in 2015. He is currently working as an ERCIM Research Fellow at NTNU, Norway. He is also the Head of the Digital Image Processing Laboratory and an Associate Professor with the Department of Computer Science, Islamia College University Peshawar, Pakistan. He has published more than 65 papers in peer-reviewed international journals and conferences. His primary research interests include computer vision, image understanding, pattern recognition, robotic vision, and multimedia applications. He is also a professional reviewer of several journals, an Associate Editor of IEEE Access, and acting as a Guest Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.