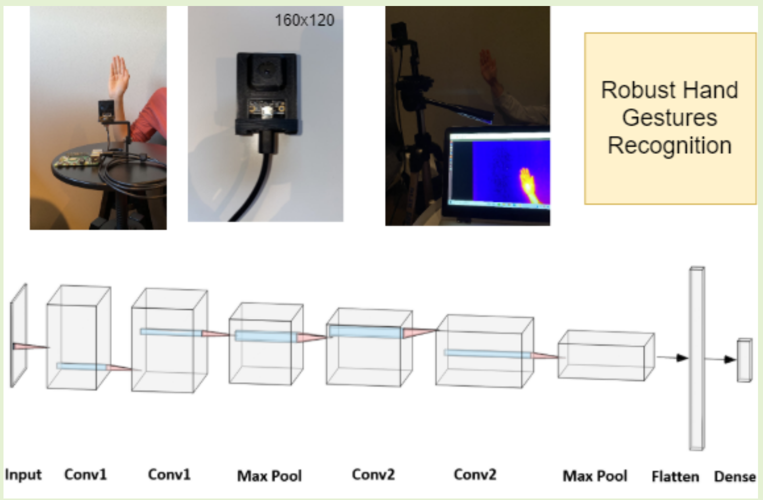
Robust Hand Gestures Recognition using a Deep CNN and Thermal Images

Daniel S. Breland, Aveen Dayal, Ajit Jha, Phaneendra K. Yalavarthy, *Senior Member, IEEE*, Om Jee Pandey, *Senior Member, IEEE* and Linga Reddy Cenkeramaddi, *Senior Member, IEEE*

***Abstract*—Medical systems and assistive technologies, human-computer interaction, human-robot interaction, industrial automation, virtual environment control, sign language translation, crisis and disaster management, entertainment and computer games, and so on all use RGB cameras for hand gesture recognition. However, their performance is limited especially in low-light conditions. In this paper, we propose a robust hand gesture recognition system based on high-resolution thermal imaging that is light-independent. A dataset of 14,400 thermal hand gestures is constructed, separated into two color tones. We also propose using a deep CNN to classify high-resolution hand gestures accurately. The proposed models were also tested on Raspberry Pi 4 and Nvidia AGX edge computing devices, and the results were compared to the benchmark models. The model also achieves an accuracy of 98.81% and an inference time of 75.138 ms on Nvidia Jetson AGX. In contrast to hand gesture recognition systems based on RGB cameras, which have limited performance in the dark-light conditions, the proposed system based on reliable high resolution thermal images is well-suited to a wide range of applications.**



***Index Terms*—High resolution thermal imaging, Hand gesture recognition, Thermal sensors, Human-computer interaction, Human-robot interaction, machine learning, Deep CNN.**

I. Introduction

Hand gestures are used in a variety of applications. As stated in [1], applications include gesture controlling robots, in various medical systems, control visualization devices, and many more. Other use areas could be more clearly directed, for example, to entertainment or to assisting with daily tasks. Hand gestures help users in many ways, for example human-computer interaction (HCI) for disabled and elderly people, human-robot interaction in many indoor and outdoor applications. Other applications include hand gesture recognition for turning on and off some machines, controlling the speed of the fan, controlling content in television. Many applications can also be customized to meet the needs of the user and aid in communication and control of appliances. A hand movement with a gesture, rather than using voice or electronic controls, would be very stable and robust in many challenging environments. Dynamic gestures, which use the movement of a controller, have been around for a while. Robust hand gesture recognition is more difficult to develop and will require more detailed images with all possible practical variations. The accuracy and speed of a hand gesture recognition system are determined by the equipment's quality and algorithms. A faster computer processes information faster, and a more detailed/pixelated image usually necessitates more processing resources but provides greater accuracy. It must be tested and applied in real-world scenarios to determine the best cost-benefit ratio.

A smart and sustainable product should be able to evolve. That is, it should be capable of learning new variations in the

Compiled on October 7, 2021. This work was supported by the Indo-Norwegian collaboration in Autonomous Cyber-Physical Systems (INCAPS) project: 287918 of International Partnerships for Excellent Education, Research and Innovation (INTPART) program from the Research Council of Norway.

D. S. Breland, A. Dayal and L. R. Cenkeramaddi (corresponding author) are with the ACPS research group, Department of Information and Communication Technology, University of Agder, Grimstad, Norway (e-mail: danisb15@student.uia.no, aveendayal97@gmail.com, linga.cenkeramaddi@uia.no).

A. Jha is with the Department of Engineering Science, University of Agder, Grimstad, Norway (e-mail: ajit.jha@uia.no).

P. K. Yalavarthy is with the Department of Computational and Data Sciences, Indian Institute of Science, Bangalore 560 012, India (e-mail: yalavarthy@iisc.ac.in).

O. J. Pandey is with the Electronics and Communication Engineering Department, SRM University, Andhra Pradesh, India (e-mail: omjee.iitk@gmail.com).

hand gestures as well as adapting to the current user's gestures. This is because cognitive movement differs from person to person [1].

There are several works based on RGB cameras and machine learning for hand gestures recognition reported in [2]. A simple and intelligent system to recognize the expression of speech-disabled people is proposed in [3]. An efficient interpretation of hand gestures to control smart interactive television is proposed in [4]. An efficient approach for the recognition of hand gestures from very low resolution images is proposed in [5]. Hand gesture recognition and human pose recognition approaches are similar and are based on RGB images. Human action recognition based on spatial distribution, direction pixel and transform has been proposed in [6]. Abnormal human activity recognition using the changes in orientation of silhouette in key frames has been proposed in [7]. Human action recognition using decisive pose has been proposed in [8]. A multi-resolution descriptor for human action recognition has been proposed in [9]. A real-time approach for static hand gesture recognition based on RGB camera is proposed in [10]. A Robust hand gesture recognition using multiple shape-oriented visual cues has been proposed in [11]. However, the performance of RGB cameras-based hand gesture recognition is limited especially in low-light and dark-light conditions. A hand gesture recognition sensor that employs ultra-wideband (UWB) impulse signals reflected from a hand [12]. A CNN is used to classify six gestures. It is proposed to use an UWB impulse radar sensor and an 8x8 pixel thermal sensor to recognize hand gestures [13]. A deep learning based approach is used in this work. Surrounding noise may limit the performance of such systems. A machine learning based multi-features capacitive sensor has been proposed for hand gesture recognition [14]. The K-Nearest Neighbour (KNN) classifiers are used in this work. A three-axis accelerometer and gyroscope sensor-based continuous hand gesture recognition technique in a smart device is proposed [15]. It is proposed to use simultaneous pressure sensors to recognize hand gestures [16]. An extreme learning method is used in this work and 11 gestures were classified. A deep learning technique called long short-term memory (LSTM) is used to classify hand gestures using data from an inertial measurement unit (IMU), electromyographic (EMG), and finger and palm pressure data [17]. However, most of these do not fit for many practical applications.

The visibility of objects in an image is controlled by the background lighting, which varies depending on the working environment. This impediment has been removed by using thermal cameras. However, hand gesture recognition has been attempted using low resolution RGB images under variable illumination settings [18] and [19], as well as depth images in low-intensity environments [20]. When the imaging scene is completely dark, however, RGB cameras fail to record any item in the imaging scene. The distance at which something can be detected varies depending on the camera's quality and resolution. A thermal camera, on the other hand, can detect objects in complete darkness as long as the object is not the same temperature as the background. Thermal cameras rely on infrared radiation emitted by objects, which does not require energy from external sources such as the sunlight or artificial light. This means that a thermal camera detects radiation and return temperature values in human-readable colors (typically red/purple/yellow). Thermal cameras also have the advantage of being able to exclude any external disturbances. A thermal camera doesn't detect and include objects in RGB images with many details. They'll be blended into the background, highlighting only the warmer/colder objects [21], [22].

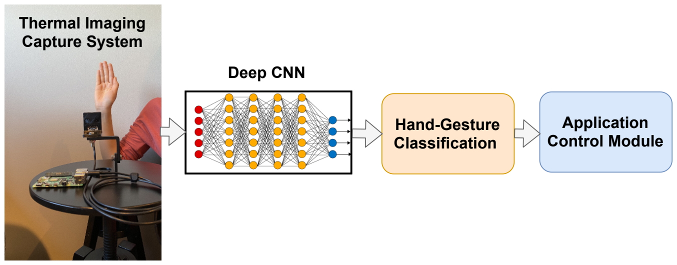

Thermal Imaging Capture System
Deep CNN
Hand-Gesture Classification
Application Control Module

Fig. 1: A top level system diagram.

Thermal imaging based hand gestures are robust compared to the RGB based hand gestures [2], [23], [24]. A thermal images of 32x32 pixels and deep learning is proposed for hand gestures recognition for sign language digits [2]. However, the camera used is of very low resolution and performance is limited for many long-range applications. In addition, such low-resolution cameras poses several challenges to be able to design a robust hand gesture recognition systems. To overcome these challenges, we design a robust hand gesture recognition system using high resolution thermal imaging along with deep learning. To create a high resolution thermal imaging based dataset of hand gestures, FLIRs thermal camera module Lepton 3.5 [25] with the breakoutboard Purethermal 2 [26] is used. It has a variation in hand positions in front of the camera to increase sample variation as well as hand posture. The Lepton camera has a resolution of 160 x 120 pixels. The dataset was created by photographing 240 images in grayscale and plasma colors. Making the entire dataset of 480 images per person, for a total of 14,400 images.

To the best of our knowledge, this is the first study to describe high-resolution thermal imaging-based hand gestures which are independent of background lighting, including dark light conditions. We also propose a light-weight deep learning model for classification of hand gestures using images obtained from a high resolution thermal camera.

II. System Setup and Thermal Camera Details

A top level system diagram in shown in Fig. 1. Hand gestures are captured using a thermal imaging system, as can be seen. For classification, the captured images are fed into a machine learning model. The control module receives the classified hand gestures for further actions. Parameters such as the number of captured images per second and how frequently the system should be in the idle state can be tailored to meet the needs of the application. The details of individual modules are further elaborated in the following sections.

The thermal image dataset was created using a FLIR Lepton 3.5 thermal camera and a Purethermal 2 breakout board. The

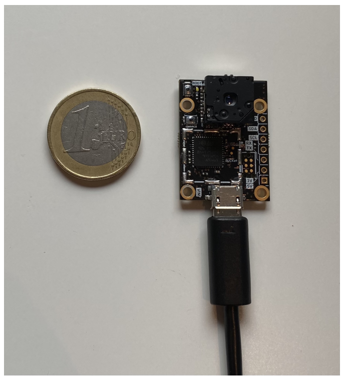


Fig. 2: FLIR Lepton 3.5 in Purethermal 2 breakout board, with size compared to a 1 Euro coin.

hand gestures are captured using this thermal camera. When connected to a Raspberry Pi 4, it connects to the camera module via a USB interface. The breakout board connects to a micro-USB port and transfers data via inter inter integrated circuit (I2C) over USB. The Lepton module will be powered by the Raspberry Pi as well. All connections are made through this port because Purethermal 2 requires 5V and only has one connectivity point.

The FLIR Lepton 3.5 module is mounted in the breakout board's camera slot, where it is connected to a FLIR Lepton socket. Thermal camera along with breakout board can be seen in Fig. 2. The breakout board includes pre-configured firmware for controlling the Lepton module [27].

Using a USB interface makes it simple to use the module with different platforms, such a PC, a Raspberry Pi or any edge computing device with USB port. Before the Raspberry Pi can be used as a webcamera, the I2C interface and the serial peripheral interface (SPI) bus must be activated. A custom software program is made to retrieve images from Lepton when using a Raspberry Pi. The complete system setup is shown in Fig. 6, while taking images.

A micro-USB to USB cable is used to connect the thermal camera to computing unit. This allows for connection to the vast majority of computing devices while also making reading and writing the module easier. The software program used to capture and store images is depicted in Fig. 3. A software program that will capture the desired number of images with only two inputs. It saves the images in both a fusion (default) color version, as well as in a ice fire version as named in the engineering datasheet [25]. As a result, for each image taken, the dataset grows by two. If there are any problems, the software program is also designed to allow for only one scale, as well as checking the total number of images in the folders by providing input 5.

Details on the FLIR Lepton 3.5 module and Purethermal 2 breakout boards are presented in the following sections. It also includes some important features for comprehending the Lepton module.

A. FLIR Lepton 3.5

FLIR Lepton 3.5 is the thermal camera from FLIR with the highest resolution Long Wavelength Infrared (LWIR) micro

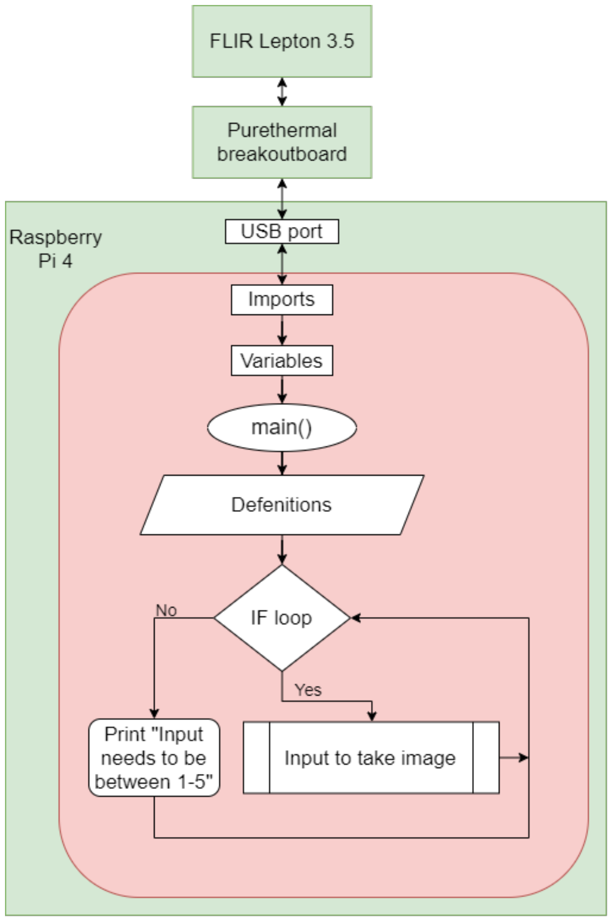


Fig. 3: Flowchart showing psudo code for capturing the thermal images with Raspberry Pi. ”X” corresponds to the number of images the program will take when input is 2.

camera. It has a 160×120 pixel resolution and a radiometric calibrated array of 19 200 pixels [28]

Lepton 3.5 requires an external power supply ranging from 2.5V to 3.1V and has a nominal power consumption of 160mW. It can reach a peak power of 650mW for about one second during shutter events, while the low power mode consumes only 5mW. As a result, it could be used in IoT or small battery-powered objects. The Field Of View (FOV) is 57° (horizontaly). The thermal sensitivity is 50mK and each pixel is measured individually with a pixel pitch at 12 μm. Sensing temperature range is between -10°C to +140°C, while operating temperature is from -10°C to +65°C. To name a few integrated functions, there is a digital thermal image processing function, automatic environment compensation, noise filters, and gain control. Images are exported at<9Hz, with video transmitted via SPI and the module controlled via I2C [25].

The Flat-Field Correction (FFC) is used by the camera module to display the heated areas as accurately and distinguishably as possible. The FFC will correct the temperatures that the camera detects in order to produce a more clear and uniform image. The FFC will re-calibrate to provide the best image quality [25]. When taking photos, the difference in image quality is visible, as will be discussed later. An FFC is performed during the startup sequence and, if in automatic mode, at regular intervals (every 3 min as default). This timer can be adjusted to run more frequently or less frequently as needed. When turned on, it has some default settings, such as auto FFC mode with an interval of 180 000 ms, delta temperature of 150°C, gain mode set to high, color Fusion with Raw 14 video output, and radiometry control enabled in addition to the other settings.

B. Purethermal 2

The Purethermal 2 breakout board is a FLIR Lepton smart I/O module with pre-configured plug-and-play functionality via a USB port. The default settings are compatible with standard webcam applications running on Linux, Windows, Mac, and Android. If the module is configured to be used for other purposes, the firmware is open source [29]. For more information, see Purethermal breakoutboard firmware github [30]. STM32 ST-LINK Utillity, a software program for operating STM32 processors, is recommended for flashing the Purethermal 2 breakout board. A FLIR Lepton device is required to operate the Purethermal 2 breakout board [31].

Purethermal 2 output is obtained via USB, with data transferred via I2C, UART, GPIO, or JTAG. It supports both Lepton 2 and 3 versions, with Lepton 3.5 being used to collect data for this dataset. It employs an STM32F412 processor to process data and perform calibration. This processor enables image processing prior to receiving any images, which relieves the load on the third-party device used to view and store images. The maximum input voltage is 5.5V, and the nominal power consumption is 61.1mA, with a peak of 230mA during the shutter event. The schematics and the firmware are open source making it easier to customize its use or to create supporting objects [27].

III. Dataset details

A. Thermal images capturing

The dataset is captured in both full color and grayscale because doing so will double the size of the dataset. Despite the fact that the images are similar, the colors are different. The hands in images are moved around within the image to capture several possibilities of gestures. Some of the movements are caused by people being unable to keep their hands still, and some are caused by them being told to move. The natural movement will result from the position in which people are forced to keep their hands when taking photographs. Some variations of hand movements are shown in Fig. 4. Fig. 4a depicts the starting position, Fig. 4b (forced angle) depicts when the hand is intentionally bent, and Fig. 4c (naturally moved) depicts when the hand falls towards the body after being static for a long time. This will result in natural variation, which increases the number of differences in the dataset for the same hand gesture.

When connected to the Raspberry Pi, the flowchart in Fig. 3 describes the python script used to capture images for the dataset. It is a very simple program that makes use of the Lepton library for Python programming. This library enables Lepton 3.5 to capture images, which are then stored on the Raspberry Pi using the OpenCV (cv2) and matplotlib libraries. The main program is a loop that awaits input in the form of numbers ranging from one to five. When input equals two, a new input is required. This second input is the number of rounds or images it will take before asking for the first input again.

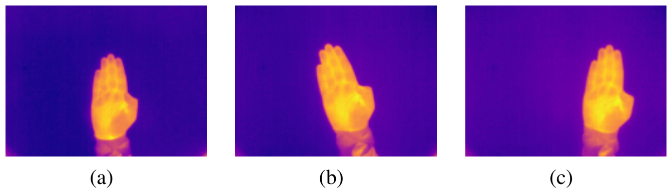


Fig. 4: Example of hand movement within frame: (a) Hand position 1; (b) Hand position 2; (c) Hand position 3;

People's hand mobility varies in the same way that movement and positioning will increase variation. Fig. 5 shows a brief example of differences. Where some people can keep their hands together, as shown in Fig. 5a and 5b, others are not able to keep their hands completely enclosed as Fig. 5c. The natural variation in this also helps to broaden the dataset's diversity by including people with different hand shapes and mobility. When every gesture has more variations, then the algorithm will have more things to consider. If the algorithm has not been trained with enough diverse samples, it becomes more difficult to read an image in a practical scenario. The different nationalities of the participants also increase the likelihood of receiving different gesture shapes. Their upbringing and occupations may also have an impact on the shape and mobility of their hands.

The dataset is created in the uniform background. The uniform background will draw attention to the hand and emphasize the unique characteristics of each hand. People are naturally warmer than the inside room where the images were taken, though the quality of the images may vary depending on the circumstances. Fig. 5 shows how hand gestures of some people are shown as cold. Fig. 5d, 5e and 5f show the problem when the camera is reading the hand as cold. In fact, their hands/fingers are not cold. The blue fingers may come from several things, as the images were taken from January to March and the outside temperature was lower than in the spring/summer time. If people were walking outside prior to taking images, their skin might have been colder than the reference wall behind. Some people might also have a lower body temperature, depending on location in the world. Although this should not have any big consequences, a few °C will be shown clearly in thermal images. Personal features may also play its part, as wet hands will more likely be colder or closer to the room temperature. Other factors may also have an influence but it will all create a variation in the dataset, that makes it harder for computers to learn. That also makes it better for testing, to check the quality and performance of the algorithms proposed using such diverse dataset.

While taking the thermal images, people had to hold their arms to their sides directly in front of the camera in order to capture images hands properly. Ideally, the people

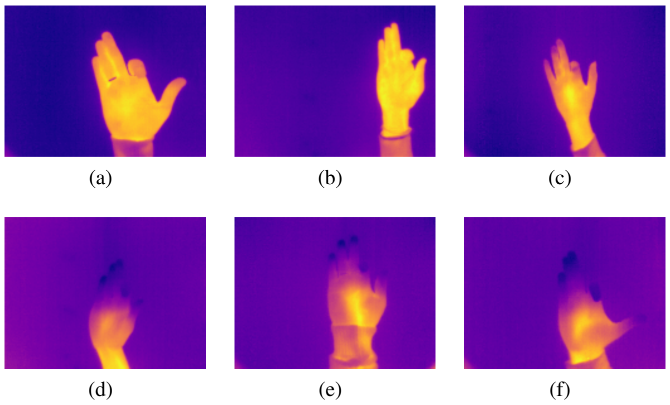


Fig. 5: Example of difference between images, with position, gesture and temperature differences: (a) Good temperature difference, good hand position; (b) Good temperature difference, good hand gesture; (c) Good temperature difference, medium hand gesture; (d) Poor temperature difference, good hand gesture; (e) Poor temperature difference, medium hand gesture; (f) Poor temperature difference, poor hand gesture;

photographed had their arms to the side, with a 90° angle upwards.The position in which people have been holding their arms is depicted in Fig. 6. Because the camera tripod is not very tall, it must be placed at a different height than their elbows. In some cases, the tripod has been placed on top of other objects, or it has been raised by placing it on a variable-height object. Keeping their arms in such static positions may be too taxing if not supported. As a result, a table or stool was placed for the people to support their elbows. It was sometimes necessary to change angles completely and face the same direction as their hand was pointing with gestures h, i and j. The Lepton 3.5's Horizontal FOV (HFOV) is 57°, indicating that it was designed to capture more details in images rather than objects.

All images are taken at roughly the same distance and angle from the camera. The distance maintained between the camera and the hand is 40 to 60 cm. It can capture images with a high level of detail at that distance. However, it is technically possible to capture images at greater distances as well. Figure 5a is an example of a high-detail image in which different skin temperatures are clearly visible and all fingers are visible. Images will begin to lose detail as they are moved further away, as shown in Fig. 5c. The camera tripod can be adjusted up and down as well as sideways to point the camera straight at the hands. Because the tripod is so small, adjusting to a different setting is simple because the setup is mobile and portable. Making it easier to meet people close to them and not being confined to a single location.

The dataset is made up of 14 400 images divided into two color tones, which are then divided into 10 gestures of 24 images each per person. Fig. 7 and Fig. 8 are both one set of ten gestures from the same person. All images should look as close to these gestures as possible, with only the natural/personal differences for each person. When sorted by gesture, the dataset contains 720 images of each gesture in

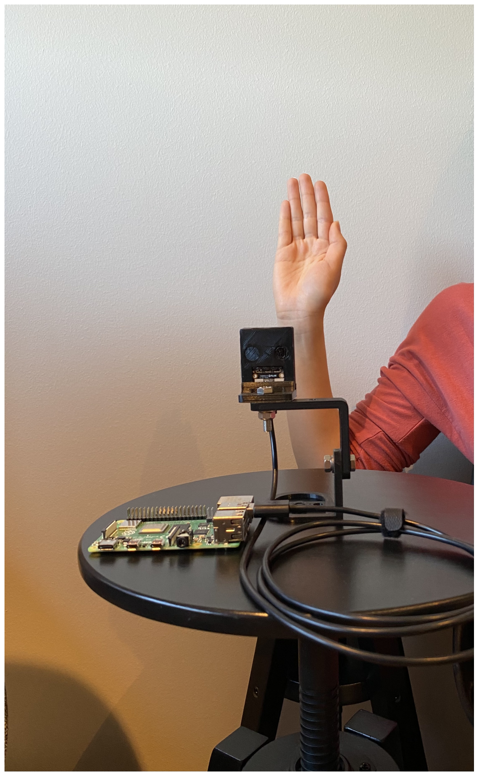


Fig. 6: Image example of how the images were taken. By placing hand in front of camera with a plain background. The hand is also supported by another barstool to keep the hand as stable as possible.

both fusion and grayscale. The size of the images varies from fusion to grayscale. Fusion colors will be larger, with each image taking up 25kB and totaling around 18MB. Grayscale images are smaller, weighing in at 17kB per image and totaling 12.2MB for a single gesture. With 30 people, the total size of the dataset is around 300MB of data, resulting in some detailed thermal images suitable for image recognition. The algorithm will be more accurate with a larger dataset and more data, but it may slow down in terms of how long it takes to train. The small amount of memory required to integrate and store thermal images is advantageous because it will make integration into any machine learning algorithm and computing platform easier.

Because the images do not represent any numbers or letters, the gestures are thought to be similar. Where the gestures use some of the same fingers but in different positions and placement. As seen in the first four images of Fig. 7 and Fig. 8. The palm of the hand is in the center of all of these images, but the fingers are in various positions. Depending on the image's quality and temperature, a small change in position can be difficult to detect. As a result, having a large number of images reduces the likelihood of selecting the incorrect label for the image. The dataset is doubled when images are taken in two different colors, and the algorithm learns to detect more images in different colors as well.

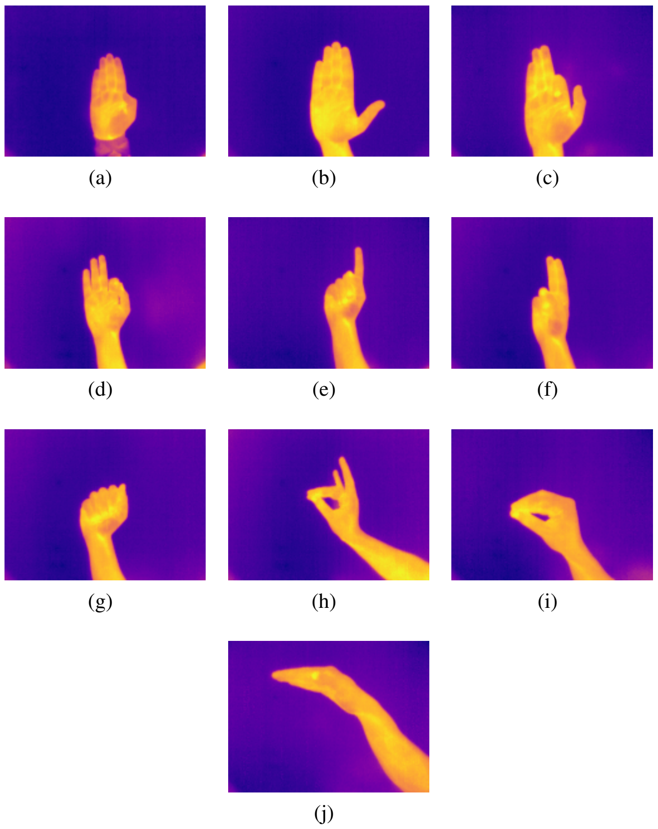


Fig. 7: A complete set of fusion colored thermal images: (a) Fusion image a; (b) Fusion image b; (c) Fusion image c; (d) Fusion image d; (e) Fusion image e; (f) Fusion image f; (g) Fusion image g; (h) Fusion image h; (i) Fusion image i; and, (j) Fusion image j.

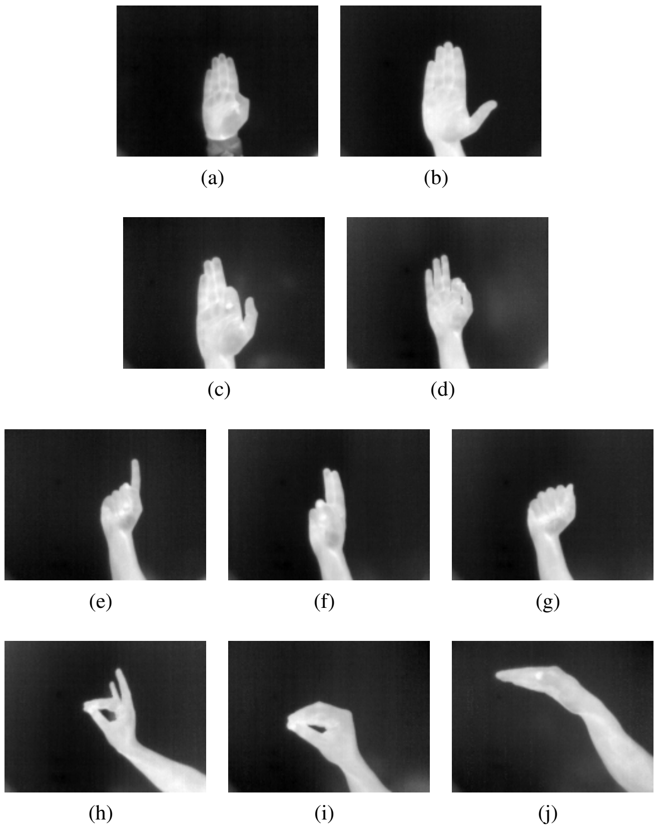


Fig. 8: A complete set of grayscale thermal images: (a) Grayscale image a; (b) Grayscale image b; (c) Grayscale image c; (d) Grayscale image d; (e) Grayscale image e; (f) Grayscale image f; (g) Grayscale image g; (h) Grayscale image h; (i) Grayscale image i; and, (j) Grayscale image j.

B. Thermal imaging in low lighting conditions

Images are captured in various lighting conditions to demonstrate the robustness of the FLIR Lepton 3.5 thermal camera. This section shows images of how the camera works in various lighting conditions, ranging from dim to completely dark. The Fig. 9 depicts three different light shades in the room. The images were taken in the same session, and the lighting was reduced from normal, as shown in Fig. 6. The hand was kept in the same position to see any differences, as shown in the Lepton image at the bottom right of Fig. 9.

It can be seen in Fig. 9b, this scenario has less thermal leakage than the others, which can occur when the camera is live streaming and constantly calculating the temperature within the frame. This could be due to a number of factors, such as a new temperature sensor calibration or a bias calculation of the background temperature when the light went off. The last image taken as shown in Fig. 9c, which can explain the temperature difference between the hand and the right side of the image. As the body move closer and the hand warms up its surroundings.

The fact that all images are captured using the 3D printed casing shown in Fig. 10 also demonstrates that the camera is unaffected by external or surrounding light. Because all other sensors are covered by the casing in this image, only the lens will be able to detect any surrounding light. The casing will provide more stable conditions that will not be affected by small external temperature changes.

IV. Machine learning

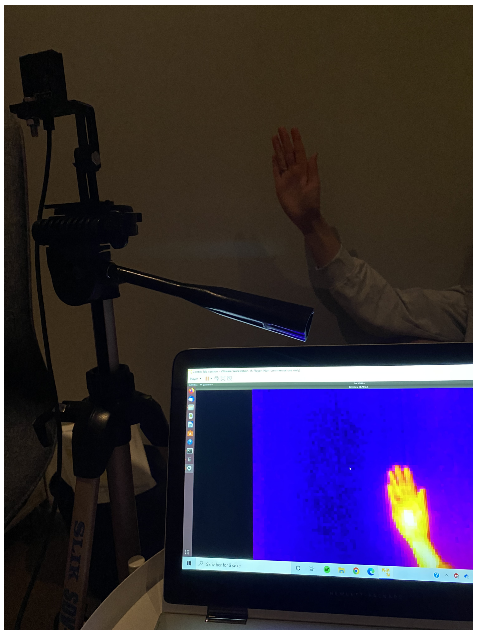
A. Preprocessing

The dataset was created as shown in the above sections. After that we divide the entire dataset into two parts, Train dataset and Test dataset. The test dataset consists of 20% of the entire dataset. We further divide the train dataset into two parts namely, Train dataset and Validation dataset. All the division of the dataset is done in a manner such that equal number of samples are taken from each class, so as to avoid an imbalance distribution. Table I shows the details of the three datasets.

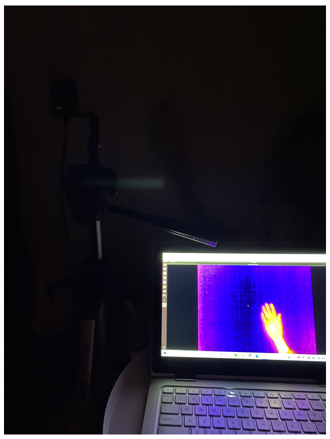
B. Model

The proposed CNN model consists of the following layers

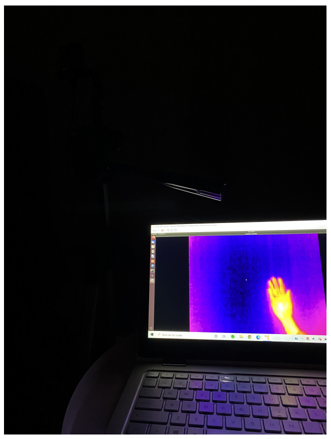
1) Convolution Layer: In this layer the kernel maps ('k') performs the convolution operation on the input feature map. These layers consists of the following main parameters, dimensions of the kernel map i.e $(H \times W \times D)$. 'H' is the height of kernel map, 'W' is the width of the kernel map, 'D' is the depth of the kernel map and 'N' number of kernel maps.



(a)



(b)



(c)

Fig. 9: Image of how the reflection is in a low light scenario: (a) Lower light than normal; (b) Almost completely dark; (c) Without light;

TABLE I: Number of samples per class in Train, Validation and Test Datasets.

Class	Train Dataset	Validation Dataset	Test Dataset
0	519	57	144
1	519	57	144
2	519	57	144
3	519	57	144
4	519	57	144
5	519	57	144
6	519	57	144
7	519	57	144
8	519	57	144
9	519	57	144

In this work $H \neq W$ i.e a rectangular kernel map. We also use a stride of 1 for each convolution layer. The convolution

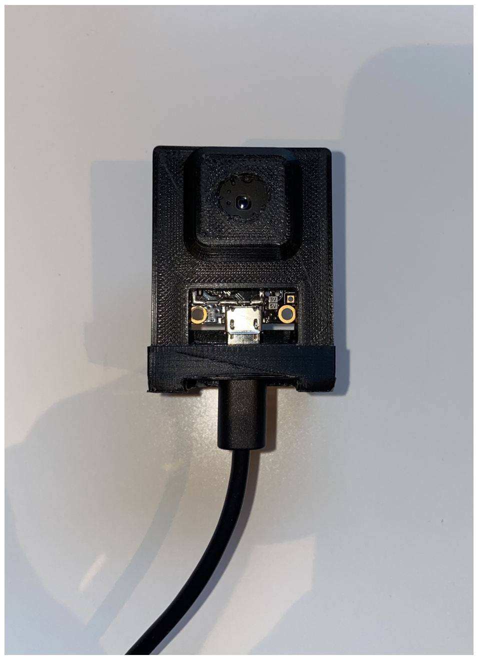


Fig. 10: FLIR Lepton 3.5 in casing.

operation is shown according to equation (1) [32].

$$y_{i^{l+1},j^{l+1},n} = \sum_{i=0}^{H}\sum_{j=0}^{W}\sum_{d=0}^{D} k_{i,j,d,n} \times {x^{l}}_{i^{l+1}+i,j^{l+1}+j,d} \tag{1}$$

In equation (1), x^l is the output of the previous layer $(l-1)$ which becomes the input feature map to the current layer (l). '$y_{i^{l+1},j^{l+1},n}$' is the intermediate output after performing the convolution operation. We repeat this for for all 'N' maps to obtain 'Y'. We then input this 'Y' to an non linear activation function, ReLU. The function ReLU is shown in the equation (2) [32].

$$f(x) = \max(0, x) \tag{2}$$

Thus the final output from the convolution layer is obtained as shown in the equation (3) [32].

$$X^{l+1} = f(Y) \tag{3}$$

2) Max Pool Layer: Pooling layer is used to reduce the dimensions of the feature map while extracting efficient representations. There are different variants available for Pooling layer namely, Max Pool, Average Pool,etc. In this work we use Max Pool Layer, whose operation is shown in the equation (4) [32].

$$y_{i^{l+1},j^{l+1},n} = \max_{0\leq i\leq H, 0\leq j\leq W} {x^{l}}_{i^{l+1}\times H+i,j^{l+1}\times W+j,d} \tag{4}$$

3) Dilated Convolution Layer: This layer is a type of convolution layer whose kernel map's field of view is ≥ 1 [33]. This variable length field of view is parameterised by the variable 'dilation rate'. In this work we use a 'dilation rate ' of 2. Fig. 11, shows the difference between a convolution layer with dilation rate of 1 (regular convolution layer) and a convolution layer with dilation rate of 2.

In this work we use 4 convolution layers, 2 with dilation rate of 1 and 2 with dilation rate of 2. Each of the convolution layer

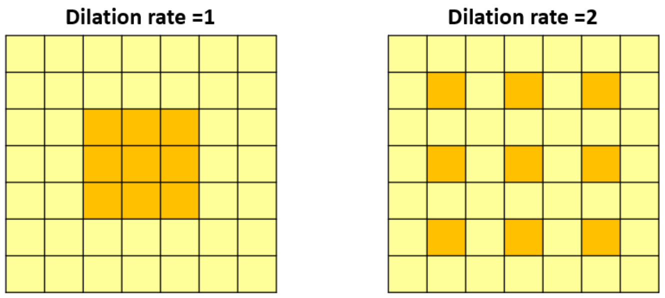


Fig. 11: Kernel map's field of view for convolution layer with dilation rate of 1 and 2.

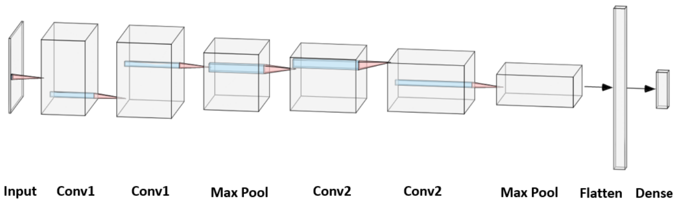


Fig. 12: Proposed CNN model architecture. Conv1 and Conv2 corresponds to convolution layers with dilation rate of 1 and 2 respectively.

is followed by a batch normalization operation to improve training speed and decrease overfitting [34]. We also use 2 Max Pool layers. The entire model architecture is shown in Table II and Fig. 12.

TABLE II: Architecture details of the proposed CNN model. Conv1 and Conv2 are convolutioon layers with dilation rate 1 and 2 respectively.

Layer	Output Shape	Kernel Map Dimension
Input	(None,120,160,1)	None
Conv1	(None, 118,159,16)	$(3 \times 2 \times 16)$
Conv1	(None, 116,158,32)	$(3 \times 2 \times 32)$
Max Pool	(None, 38,79,32)	(3×2)
Conv2	(None, 30, 73, 64)	$(5 \times 4 \times 64)$
Conv2	(None, 22, 67,128)	$(5 \times 4 \times 128)$
Max Pool	(None, 7, 33, 128)	(3×2)
Flatten	(None, 29568)	None
Dense	(None,10)	None

C. Training

We train the proposed CNN model using 'Adam Optimizer' with a learning rate of 0.001 [35] and a batch size of 32. All the weights were initialized using 'Kaiming initializer' [36]. The model was trained using 10 fold cross validation i.e the entire train dataset was divided into 10 parts and for each iteration 9 parts were used for training and one part was used as validation dataset. This procedure was repeated for 10 times hence the name 10 fold cross validation.

D. Benchmark Model

We compare the proposed CNN model with MobileNetV3 model as benchmark. MobileNetV3 is the '3^{rd}' version among the MobileNet family of architectures. MobileNet models are designed for optimised performance on mobile and edge computing devices. These models are specifically trained to have low latency while maintaining the accuracy of the model. There are two variants of MobileNetV3 that was proposed in [37], MobileNetV3 Small and MobileNetV3 Large. The difference between the two is the total number of parameters used to train the model. In this work we use both the variants of the model pre trained on ImageNet dataset. Before training the model we first have to slightly modify the benchmark models to adapt for the given task. We remove the classification layer (output layer) of the benchmark model and add a new classification layer with 10 classes for the given task. We also use a global average operator layer to flatten the output of the benchmark model. This layer is then to the new classification layer. We train the benchmark model using transfer learning technique called as 'Fine Tuning' method.

1) Fine Tuning: In this method few layers of the benchmark model along with the new classification layer are trained on the given task. This methodology of training can be very useful as compared to training from scratch. This is because the benchmark model's pre-trained weights act as a good parameter initializer and can optimize better on the given task.

We train both the benchmark models via 10 fold cross validation method. We also use an RMS Prop optimiser [38] with a batch size of 32 for training. All the models including the proposed model are trained on Google Colab i.e on Nvidia's T4 GPU with 12 GB GPU RAM, using Keras Deep Learning Library [39].

V. Hardware Deployment

We deploy the all the models, Proposed CNN model and the benchmark models on edge computing devices such as Raspberry Pi 4 Model B and Nvidia Jetson AGX Xavier.

A. Raspberry Pi 4B

Raspberry Pi is a hardware computer made for development use, for people to get an affordable and powerful computer that can be adapted to all situations. The hardware is designed to implement many types of software, vary much like a full PC. The operating systems can be chosen depending on the use case, such that the developer can use whatever operating system that gains most advantages. For example, Raspberry Pi foundation has an official operating system called Raspberry Pi Operating System [40], which is supported and updated for all versions of Raspberry Pi. Raspberry Pi comes in different versions, whereas the newest model is 4B, which is shown in Fig. 13a. It has older versions and smaller computers for places demanding of taking less space [41].

To use in this machine learning timing, it was used a Raspberry Pi 4B [42]. This is the newest and most powerful Raspberry Pi computer, which has an upgraded processor and the option to configure RAM between 2, 4 or 8 GB. The computer is powered through a USB-C port with 5.1V and 3A current in order to get full effect. The new processor allows for two 4K resolution monitors to be connected, using micro HDMI ports. As its predecessors it has four USB ports, split in

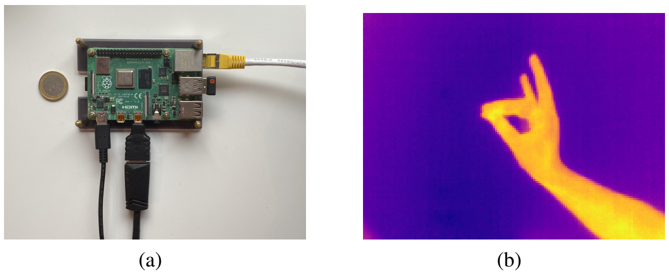


Fig. 13: Images of Raspberry Pi 4B compared to a 1 Euro coin: (a) Top view of RPi 4B; (b) Plasma colored thermal image example, gesture 7.

pairs of two. Where one pair is USB v2 and the other two are USB v3 for even faster transfers. In order to receive Internet connection, it has WiFi and a Gigabit Ethernet port connection built in [42].

The main technical specifications tells us that the processor is a 64-bit Quad core Cortex-A72 (ARMv8) System on Chip with a clock speed of 1.5GHz. The customizable RAM is LPDDR4-3200 SDRAM. It has Bluetooth v5 and BLE (Bluetooth Low Energy), along with IEEE 802.11ac WiFi with 2.4GHz and 5GHz connections. It is supported with OpenGL ES 3.0 graphics, to help render graphics on the Raspberry Pi. On the socket, it has a micro-SD card slot for where the operating system is loaded as well as storage. All these specs and more documentation can be found on the specification web page [43].

Depending on the machine learning model complexity, the operating system required to be run on the Raspberry Pi may vary. The Raspberry Pi 4B was installed with a 64-bit Ubuntu 18.04.5 LTS [44]. The 64-bit Ubuntu 18.04 is found on Ubunut's homepage under alternative downloads, as the most common downloads are 32-bit [45].

When installing Tensorflow, it is important ro think of the version that is being installed. As it needs to be trained and used with the same versions of Tensorflow. It will therefore be a problem if the model algorithm is trained on Tensorflow version 2.2 and then later used with version 2.4. There is probably some workarounds, but the easiest and sustainable is to install an operating system with 64-bit when needed. As is explained by Q-engineering in this article [46].

B. Nvidia Jetson AGX Xavier DK

Nvidia has developed a high-performance, low-powered computing hardware platform that is designed to use for deep learning and computer vision, which is the black box in Fig. 14. The hardware platform needs to have a SDK installed, called JetPack [47]. JetPack will provide an operating system including CUDA, VisionWorks, OpenCV, TensorRT and many more, built on top of a LTS Linux kernel [47]. Making it a very usable Linux machine, that is customizable for individual use. JetPack along with other development software is available at [48]. When JetPack is installed, the Jetson AGX will be available to use as a normal Linux computer with Ubunut.

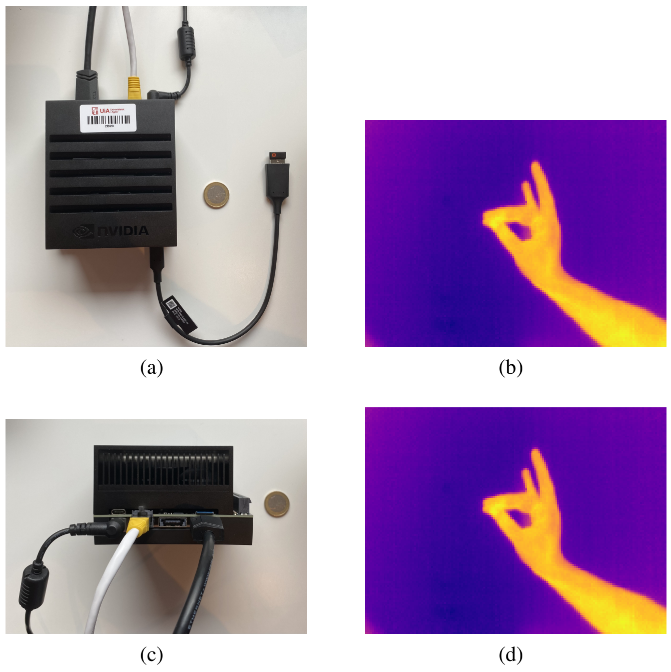


Fig. 14: Images of Jetson AGX Xavier compared to a 1 Euro coin: (a) Top view of AGX; (b) Plasma colored thermal image, gesture 7; (c) Sideways view of connections and micro-SD card; (d) Plasma colored thermal image, gesture 7.

This is very useful, to be able to use the computer for different use-cases without any modifications.

The Jetson AGX is built on a 512-core Volta GPU with Tensor Cores, along with 8-core ARM v8.2 64-bit CPU, with 8MB L2 and 4MB L3. It has a large memory of 32GB RAM, which is 256-bit LPDDR4x and internal storage of 32GB, type eMMC 5.1. It is installed with a 7-way VLIW Vision Processor Accelerator as well as two NVDLA Engines for DL Acceleration. The physical measurments are 105mm x 105mm x 65mm for the entire module [49].

Before deploying the model on the above mentioned edge computing devices we convert the models into Tensorflow Lite version. Tensorflow Lite is a variant of Tensorflow that is built to provide optimal performance on mobile and edge computing devices. The Tensorflow Lite version optimises the model's size and latency such that deep learning models can easily be deployed for real world applications. Thus we convert all the models into their corresponding Tensorflow Lite versions.

VI. Results and Discussion

We show the 10 fold cross validation results of all the models in Fig. 15. The average 10 fold validation accuracy for the proposed model, MobileNetV3 Large and MobileNetV3 Small is 98.42%, 99.42%, 99.86% respectively. We also show the accuracy plot of the proposed CNN model to show the convergence after 50 epochs in Fig. 16.

We next compare the models based on their test accuracy. After 10 fold cross validation we have 10 models available for each model. We can get the test accuracy by either combining the results from all the 10 models [50] [51] or obtain a single model by training the model on the entire train dataset with

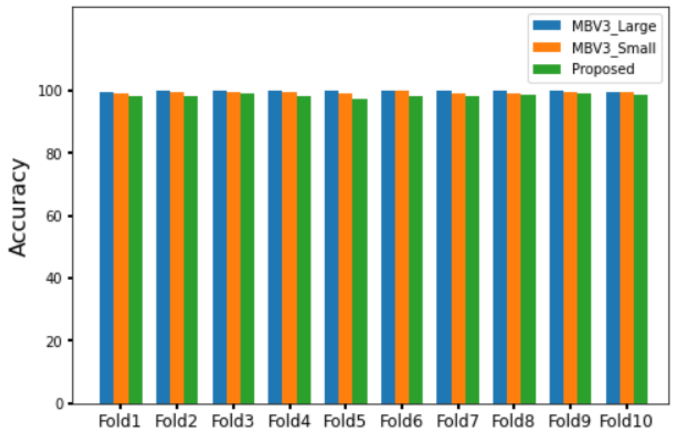


Fig. 15: 10 fold cross validation accuracy of the proposed model and the benchmark models.

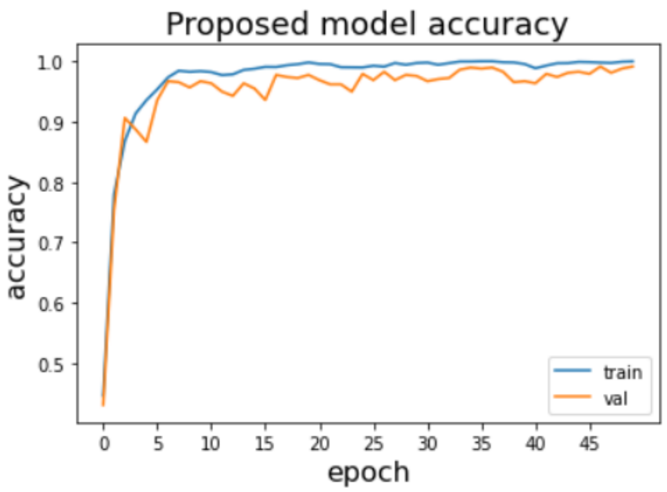


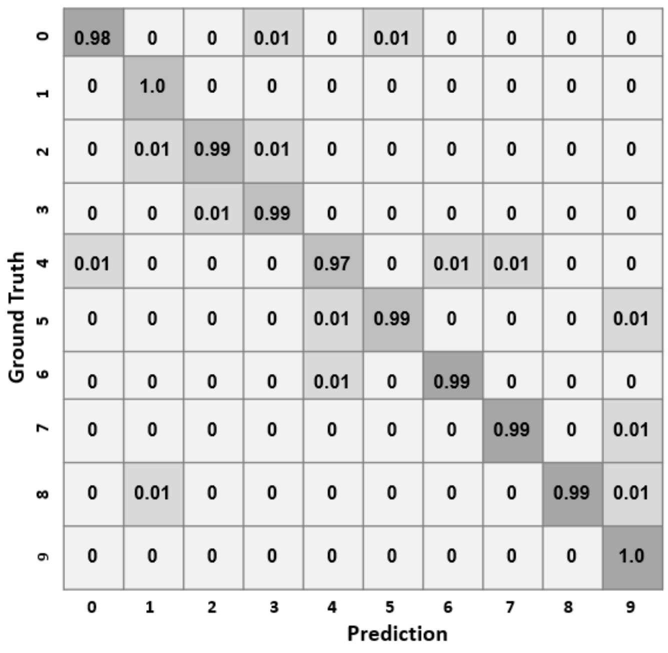
Fig. 16: Training and Validation accuracy of the proposed CNN model for fold 10.

optimal hyperparamter values obtained during 10 fold cross validation [52]. We choose the latter method for simplicity sake, we obtain 3 models corresponding to the Proposed CNN model, MobileNetV3 Small and MobileNetV3 Large respectively. We then evaluate these models on the test dataset to get the test accuracy values. These values are summarised in Table III.

TABLE III: Test accuracy values of the Proposed CNN model, MobileNetV3 Small and MobileNetV3 Large models.

Model	Test Accuracy
MobileNetV3 Small	99.72%
MobileNetV3 Large	99.98%
Proposed CNN model	98.81%

After this we compare the proposed CNN model with the benchmark models in terms of model's size both tensorflow and tensorflow lite versions and also compare them in terms of the number of parameters. These results are summarized in Table IV and V. As seen from Table V, the Proposed CNN model's TFLite version is 3 times smaller than the MobileNetV3 Small model and 8 times smaller than the MobileNetV3 Large model.



	0	1	2	3	4	5	6	7	8	9
0	0.98	0	0	0.01	0	0.01	0	0	0	0
1	0	1.0	0	0	0	0	0	0	0	0
2	0	0.01	0.99	0.01	0	0	0	0	0	0
3	0	0	0.01	0.99	0	0	0	0	0	0
4	0.01	0	0	0	0.97	0	0.01	0.01	0	0
5	0	0	0	0	0.01	0.99	0	0	0	0.01
6	0	0	0	0	0.01	0	0.99	0	0	0
7	0	0	0	0	0	0	0	0.99	0	0.01
8	0	0.01	0	0	0	0	0	0	0.99	0.01
9	0	0	0	0	0	0	0	0	0	1.0

Fig. 17: Confusion matrix of the proposed model.

TABLE IV: Test accuracy values of the Proposed CNN model, MobileNetV3 Small and MobileNetV3 Large models.

Model	Total Parameters	Trainable Parameters
MobileNetV3 Small	1,540,218	1,430,058
MobileNetV3 Large	4,239,242	3,725,818
Proposed CNN model	504,858	504,378

TABLE V: Model's Size of Tensorflow (TF) and Tensorflow Lite version (TFLite) of the Proposed CNN model, MobileNetV3 Small and MobileNetV3 Large models. Model size is measured in MB.

Model	TF Model Size (MB)	TFLite Model Size (MB)
MobileNetV3 Small	12	6
MobileNetV3 Large	31	16
Proposed CNN model	6	2

We next plot the various performance metrics such as 'Precision', 'Recall', 'F1 score' for the proposed CNN model [53]. The confusion matrix of the proposed CNN model is shown in the Fig. 17. The performance metric values are shown in the Table VI.

We lastly show the inference time of the tensorflow lite version models deployed on Raspberry Pi 4 Model B and Nvidia Jetson AGX Xavier. These values are summarized in Table VII and VIII.

The performance of the proposed model is compared to the recently reported models for hand gesture recognition using CNNs. All of these works except [2] are based on RGB images. The performance of the proposed model is comparable to the high performance models, [54], [55] and [2].

It's challenging to recognize hand gestures in complex backgrounds even with RGB images. For RGB images with

TABLE VI: Precision, Recall and F1 Score values of the Proposed CNN model for each class.

Class	Precision	Recall	F1 Score
0	0.99	0.98	0.99
1	0.99	1.0	0.99
2	0.99	0.99	0.99
3	0.98	0.99	0.98
4	0.99	0.97	0.98
5	0.99	0.99	0.99
6	0.99	0.99	0.99
7	0.99	0.99	0.99
8	1.0	0.99	0.99
9	0.98	1.0	0.99

TABLE VII: Inference time of all the models on Raspberry Pi 4B. TFLite is the Tensorflow Lite version of the models.

TFLite model	Inference Timing
MobileNetV3 Small	0.033844s
MobileNetV3 Large	0.079605s
Proposed CNN Model	0.140968s

TABLE VIII: Inference time of all the models on Jetson AGX Xavier. TFLite is the Tensorflow Lite version of the models.

TFLite model	Inference time
MobileNetV3 Small	0.013664s
MobileNetV3 Large	0.035300s
Proposed CNN model	0.075138s

complicated backgrounds, some attempts have been made [18], [20] and [70]. Hand gesture recognition has been done using low resolution RGB images under variable illumination settings [18] as well as depth images in low-intensity environments [20]. When the imaging scene is completely dark, however, RGB cameras fail to record any item in the imaging scene. It will also be challenging for hand gesture recognition in thermal images in complex backgrounds. We have future plans to work on this.

VII. Conclusion

A complete end-to-end system with a robust hand recognition model is presented in this paper. The system is designed to be highly portable, and a thermal dataset is created. The dataset includes 30 people and 14,400 thermal images of hand gestures, with 7200 in fusion color and 7200 in grayscale. The images were then classified into ten different categories. It was tested with three different machine learning models in this work. The proposed lightwight CNN model is of size 6MB only and its tensorflow lite version is only 2MB. The model also achieves an accuracy of 98.81% and an inference time of 0.075138s on Nvidia Jetson AGX. Because of reliable thermal imaging, the proposed hand gesture recognition is robust and unaffected by external light sources.

Acknowledgment

This work was supported by the Indo-Norwegian collaboration in Autonomous Cyber-Physical Systems (INCAPS) project: 287918 of International Partnerships for Excellent Education, Research and Innovation (INTPART) program from the Research Council of Norway.

TABLE IX: Performance comparison of proposed model and other models.

S.No	Model	Accuracy	Ref.
1	Proposed Model	98.81%	This work
2	3D CNN	77.5%	[56]
3	CNN and RNN	85.46%	[57]
4	Recurrent 3D CNN	88.4%	[58]
5	CNN and RNN	89.5%	[59]
6	Deep CNN	90.7%	[60]
7	3D CNN	94.4%	[61]
8	Deep CNN	94.6%	[62]
9	Deep CNN and Image processing	95.61%	[63]
10	CNN w/morphological filters	96.83%	[64]
11	Deep CNN	97.1%	[65]
12	CNN w/data augmentation	97. 2%	[66]
13	CNN with 3D Receptive fields	97.5%	[67]
14	3D CNN and LSTM	97.8%	[68]
15	DC CNN	98.02%	[69]
16	2D CNN	98.2%	[54]
17	Compact CNN	98.81%	[55]
18	Deep learning based CNN	99.52%	[2]

References

[1] Juan Pablo Wachs et al., “Vision-based hand-gesture applications,” *International Journal of Humanoid Robotics*, vol. 54, no. 02, pp. 60–71, 2011.

[2] D. S. Breland, S. B. Skriubakken, A. Dayal, A. Jha, P. K. Yalavarthy, and L. R. Cenkeramaddi, “Deep learning-based sign language digits recognition from thermal images with edge computing system,” *IEEE Sensors Journal*, vol. 21, no. 9, pp. 10 445–10 453, 2021.

[3] D. K. Vishwakarma and R. Kapoor, “Simple and intelligent system to recognize the expression of speech-disabled person,” in *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, 2012, pp. 1–6.

[4] “An efficient interpretation of hand gestures to control smart interactive television,” *Int. J. Comput. Vision Robot.*, vol. 7, no. 4, p. 454–471, Jan. 2017. [Online]. Available: https://doi.org/10.1504/IJCVR.2017.084991

[5] D. Vishwakarma, R. Maheshwari, and R. Kapoor, “An efficient approach for the recognition of hand gestures from very low resolution images,” in *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 467–471.

[6] D. Vishwakarma and R. Kapoor, “Integrated approach for human action recognition using edge spatial distribution, direction pixel and -transform,” *Advanced Robotics*, vol. 29, no. 23, pp. 1553–1562, 2015. [Online]. Available: https://doi.org/10.1080/01691864.2015.1061701

[7] D. K. Vishwakarma, R. Kapoor, R. Maheshwari, V. Kapoor, and S. Raman, “Recognition of abnormal human activity using the changes in orientation of silhouette in key frames,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2015, pp. 336–341.

[8] D. K. Vishwakarma, “A two-fold transformation model for human action recognition using decisive pose,” *Cognitive Systems Research*, vol. 61, pp. 1–13, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389041719305224

[9] D. K. Vishwakarma and T. Singh, “A visual cognizance based multi-resolution descriptor for human action recognition using key pose,” *AEU - International Journal of Electronics and Communications*, vol.

107, pp. 157–169, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1434841119303942

[10] B. B, D. R, V. T. Selvam, V. V. Kumar, and R. R, "Improved real-time approach to static hand gesture recognition," in *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 2021, pp. 416–422.

[11] S. Bakheet and A. Al-Hamadi, "Robust hand gesture recognition using multiple shape-oriented visual cues," *EURASIP Journal on Image and Video Processing*, vol. 2021, no. 1, p. 26, Jul 2021. [Online]. Available: https://doi.org/10.1186/s13640-021-00567-1

[12] S. Y. Kim, H. G. Han, J. W. Kim, S. Lee, and T. W. Kim, "A hand gesture recognition sensor using reflected impulses," *IEEE Sensors Journal*, vol. 17, no. 10, pp. 2975–2976, 2017.

[13] S. Skaria, D. Huang, A. Al-Hourani, R. J. Evans, and M. Lech, "Deep-learning for hand-gesture recognition with simultaneous thermal and radar sensors," in *2020 IEEE SENSORS*, 2020, pp. 1–4.

[14] W. K. Wong, F. H. Juwono, and B. T. T. Khoo, "Multi-features capacitive hand gesture recognition sensor: A machine learning approach," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8441–8450, 2021.

[15] H. P. Gupta, H. S. Chudgar, S. Mukherjee, T. Dutta, and K. Sharma, "A continuous hand gestures recognition technique for human-machine interaction using accelerometer and gyroscope sensors," *IEEE Sensors Journal*, vol. 16, no. 16, pp. 6425–6432, 2016.

[16] B. B. Atitallah, M. Bilal Abbasi, R. Barioul, D. Bouchaala, N. Derbel, and O. Kanoun, "Simultaneous pressure sensors monitoring system for hand gestures recognition," in *2020 IEEE SENSORS*, 2020, pp. 1–4.

[17] X. Zhang, Z. Yang, T. Chen, D. Chen, and M.-C. Huang, "Cooperative sensing and wearable computing for sequential hand gesture recognition," *IEEE Sensors Journal*, vol. 19, no. 14, pp. 5775–5783, 2019.

[18] D. K. Vishwakarma, R. Maheshwari, and R. Kapoor, "An efficient approach for the recognition of hand gestures from very low resolution images," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 467–471.

[19] D. K. Vishwakarma and R. Kapoor, "Simple and intelligent system to recognize the expression of speech-disabled person," in *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, 2012, pp. 1–6.

[20] D. K. Vishwakarma and V. Grover, "Hand gesture recognition in low-intensity environment using depth images," in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 2017, pp. 429–433.

[21] Rikke Gade, Thomas B. Moeslund, "Thermal Cameras and Applications," *Machine Vision & Applications*, vol. 25, no. 01, pp. 246–262, 2014.

[22] L. Reddy Cenkeramaddi, J. Bhatia, A. Jha, S. Kumar Vishkarma, and J. Soumya, "A survey on sensors for autonomous systems," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020, pp. 1182–1187.

[23] Pramod Kumar et al. Nus dataset. [Online]. Available: https://www.ece.nus.edu.sg/stfpage/elepv/NUS-HandSet/

[24] P. K. et al., "Hand posture and face recognition using a fuzzy-rough approach," *International Journal of Humanoid Robotics*, vol. 07, no. 03, pp. 331–356, 2010.

[25] FLIR Systems. Flir lepton engineering datasheet. [Online]. Available: https://www.flir.com/globalassets/imported-assets/document/flir-lepton-engineering-datasheet.pdf

[26] Digi-Key Electronics. Purethermal-2. [Online]. Available: https://www.digikey.com/en/products/detail/flir-lepton/500-0771-01/7606616

[27] GroupGets. Groupgets purethermal 2 flir lepton® smart i/o module. [Online]. Available: https://groupgets-files.s3.amazonaws.com/PT2/PureThermal%202%20-%20Datasheet%20-%201.7.pdf

[28] FLIR Systems. Flir lepton 3 & 3.5. [Online]. Available: https://lepton.flir.com/wp-content/uploads/2015/06/Lepton3_3.5-Data-Sheet.pdf

[29] Kurt Kiefer. Purethermal 1/2/mini reference firmware. [Online]. Available: https://github.com/groupgets/purethermal1-firmware

[30] STMicorelectronics. Stm32 st-link utility. [Online]. Available: https://www.st.com/en/development-tools/stsw-link004.html#overview

[31] GroupGets. Purethermal 2 - flir lepton smart i/o module. [Online]. Available: https://groupgets.com/manufacturers/getlab/products/purethermal-2-flir-lepton-smart-i-o-module

[32] J. Wu, "Introduction to convolutional neural networks," 2017.

[33] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2016.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[35] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[37] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," *CoRR*, vol. abs/1905.02244, 2019. [Online]. Available: http://arxiv.org/abs/1905.02244

[38] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[39] F. Chollet *et al.*, "Keras: The python deep learning library," *ascl*, pp. ascl–1806, 2018.

[40] Raspberry Pi Foundation. Operating system images. [Online]. Available: https://www.raspberrypi.org/software/operating-systems/#raspberry-pi-os-32-bit

[41] Raspberry Pi Foundation. Products. [Online]. Available: https://www.raspberrypi.org/products/

[42] Raspberry Pi Foundation. Raspberry pi 4. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[43] Raspberry Pi Foundation. Raspberry pi 4 tech specs. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[44] Canonical Ltd. List of releases. [Online]. Available: https://wiki.ubuntu.com/Releases

[45] Canonical Ltd. Alternative downloads. [Online]. Available: https://ubuntu.com/download/alternative-downloads

[46] Q-engineering. Install tensorflow 2.4 on raspberry 64 os. [Online]. Available: https://qengineering.eu/install-tensorflow-2.4.0-on-raspberry-64-os.html

[47] NVIDIA Corporation. Jetson faq. [Online]. Available: https://developer.nvidia.com/embedded/faq#xavier-faq

[48] NVIDIA Corporation. Jetson download center. [Online]. Available: https://developer.nvidia.com/embedded/downloads#?search=Jetpack

[49] NVIDIA Corporation. Jetson agx xavier developer kit. [Online]. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[50] D. Ruta and B. Gabrys, "Classifier selection for majority voting," *Information Fusion*, vol. 6, no. 1, pp. 63–81, 2005, diversity in Multiple Classifier Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253504000417

[51] L. Brewster, J. Dale, A. Hansell, N. Whitney, T. Guttridge, S. Gruber, M. Elliott, I. Cowx, and A. Gleiss, "Development and application of a machine learning algorithm for classifcation of elasmobranch behaviour from accelerometry data," *Marine Biology*, vol. 165, 03 2018.

[52] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," 2020.

[53] K. M. Ghori, R. A. Abbasi, M. Awais, M. Imran, A. Ullah, and L. Szathmary, "Performance analysis of different types of machine learning classifiers for non-technical loss detection," *IEEE Access*, vol. 8, pp. 16 033–16 048, 2020.

[54] F. Zhan, "Hand gesture recognition with convolution neural networks," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, 2019, pp. 295–298.

[55] L. Chen, J. Fu, Y. Wu, H. Li, and B. Zheng, "Hand gesture recognition using compact cnn via surface electromyography signals," *Sensors*, vol. 20, no. 3, p. 672, Jan 2020. [Online]. Available: http://dx.doi.org/10.3390/s20030672

[56] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015, pp. 1–7.

[57] K. Lai and S. N. Yanushkevich, "Cnn+rnn depth and skeleton based dynamic hand gesture recognition," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 3451–3456.

[58] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4207–4215.

[59] A. Tewari, B. Taetz, F. Grandidier, and D. Stricker, "[poster] a probabilistic combination of cnn and rnn estimates for hand gesture based interaction in car," in *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, 2017, pp. 1–6.

[60] P. S. Neethu, R. Suguna, and D. Sathish, “An efficient method for human hand gesture detection and recognition using deep learning convolutional neural networks,” *Soft Computing*, vol. 24, no. 20, pp. 15 239–15 248, Oct 2020. [Online]. Available: https://doi.org/10.1007/s00500-020-04860-5

[61] N. N. Hoang, G.-S. Lee, S.-H. Kim, and H.-J. Yang, “A real-time multimodal hand gesture recognition via 3d convolutional neural network and key frame extraction,” in *Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence*, ser. MLMI2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 32–37. [Online]. Available: https://doi.org/10.1145/3278312.3278314

[62] A. V. and R. R., “A deep convolutional neural network approach for static hand gesture recognition,” *Procedia Computer Science*, vol. 171, pp. 2353 – 2361, 2020, third International Conference on Computing and Network Communications (CoCoNet’19). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050920312473

[63] Y.-L. Chung, H.-Y. Chung, and W.-F. Tsai, “Hand gesture recognition via image processing techniques and deep cnn,” *Journal of Intelligent & Fuzzy Systems*, vol. 39, pp. 4405–4418, 2020, 3. [Online]. Available: https://doi.org/10.3233/JIFS-200385

[64] R. F. Pinto, C. D. B. Borges, A. M. A. Almeida, and I. C. Paula, “Static hand gesture recognition based on convolutional neural networks,” *Journal of Electrical and Computer Engineering*, vol. 2019, p. 4167890, Oct 2019. [Online]. Available: https://doi.org/10.1155/2019/4167890

[65] P. Bao, A. I. Maqueda, C. R. del-Blanco, and N. García, “Tiny hand gesture recognition without localization via a deep convolutional network,” *IEEE Transactions on Consumer Electronics*, vol. 63, no. 3, pp. 251–257, 2017.

[66] M. Z. Islam, M. S. Hossain, R. ul Islam, and K. Andersson, “Static hand gesture recognition using convolutional neural network with data augmentation,” in *2019 Joint 8th International Conference on Informatics, Electronics Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision Pattern Recognition (icIVPR)*, 2019, pp. 324–329.

[67] Ho-Joon Kim, J. S. Lee, and J. Park, “Dynamic hand gesture recognition using a cnn model with 3d receptive fields,” in *2008 International Conference on Neural Networks and Signal Processing*, 2008, pp. 14–19.

[68] N. L. Hakim, T. K. Shih, S. P. Kasthuri Arachchi, W. Aditya, Y.-C. Chen, and C.-Y. Lin, “Dynamic hand gesture recognition using 3dcnn and lstm with fsm context-aware model,” *Sensors (Basel, Switzerland)*, vol. 19, no. 24, p. 5429, Dec 2019, 31835404[pmid]. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/31835404

[69] X. Y. Wu, “A hand gesture recognition algorithm based on dc-cnn,” *Multimedia Tools and Applications*, vol. 79, no. 13, pp. 9193–9205, Apr 2020. [Online]. Available: https://doi.org/10.1007/s11042-019-7193-4

[70] D. K. Vishwakarma, “Hand gesture recognition using shape and texture evidences in complex background,” in *2017 International Conference on Inventive Computing and Informatics (ICICI)*, 2017, pp. 278–283.

Daniel Skomedal Breland Was born in Kristiansand, Norway, in 1996. He is currently working on his master thesis in communication technologies at the department of ICT, University of Agder, Grimstad, Norway. He received his bachelor’s degree in electronics engineering in 2019, from the University of Agder. His main interest of research are wireless communication and sensor networks in IoT environments.

Aveen Dayal currently works as a visiting research student at the department of ICT, University of Agder, Grimstad, Norway. He received the bachelor’s degree in Computer Science and Engineering in 2020. His main research interests are in machine learning for autonomous cyber-physical systems.

Ajit Jha was born in Nepal in 1984. He received B.Sc. in Electronics and Communication Engineering (Bangladesh), European Masters in Photonic Networks from Aston University (England) and Scuola Superiore Sant Anna (Italy) and PhD from Technical University of Catalunya (Spain) and Karlsruhe Institute of Technology (Germany) in 2007, 2012 and 2016 respectively. From 2016-2019 he worked he worked at various industries related to Autonomous vehicle working on innovative technologies such as Automotive ethernet, ADAS, Surround view system , camera mirror system, Blind sport warning to name a few. Currently, he is an Associate Professor of Mechatronics at Department of Engineering Sciences at University of Agder, Norway. He is actively involved in research focused on sensors, sensor fusion, image/signal processing, ML, ADAS functionalities towards Autonomous systems, and IoT. He has (co) authored more than 20 articles and two patents. In addition, he has been active reviewer, and member of techincal program committee of numerous international peer-review journals and conferences. Dr. Jha is the recipient of Erasmus Mundus Masters Course (EMMC) and Erasmus Mundus Joint Doctorate (EMJD) both funded by European Union (EU).

Phaneendra K. Yalavarthy received the M.Sc. degree in engineering from the Indian Institute of Science, Bangalore, India, and Ph.D. degree in biomedical computation from Dartmouth College, Hanover, NH, USA, in 2007. He is an Associate Professor with the Department of Computational and Data Sciences, Indian Institute of Science, Bangalore. His research interests include medical image computing, medical image analysis, and biomedical optics. He is a senior member of IEEE, SPIE, and OSA, and serves as an associate editor of IEEE Transactions on Medical Imaging.

Om Jee Pandey received the B.Tech. degree in electronics and communication engineering from Uttar Pradesh Technical University, Lucknow, India, in 2008, the M.Tech. degree in digital communications from the ABV-Indian Institute of Information Technology and Management, Gwalior, India, in 2013, and the Ph.D. degree from the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur, India, in 2019. He worked as a Postdoctoral Fellow with the Communications Theories Research Group, Department of Electrical and Computer Engineering, University of Saskatchewan, Saskatoon, SK, Canada. From 2008 to 2011, he worked with Escorts Ltd., and FANUC India Private Ltd. He was a Senior Lecturer with the Jaipur Engineering College and Research Center, Jaipur, from 2013 to 2014. He is currently working as an Assistant Professor with the Department of Electronics and Communication Engineering, SRM University AP, Amaravati. His research interest includes the signal processing for wireless networks with a specific focus on robust sensor node localization and tracking over wireless ad hoc networks. He also works on related areas, such as low-latency data transmission, data aggregation, and distributed detection and estimation in wireless sensor networks. He is a regular reviewer for various reputed journals of IEEE, including the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE ACCESS, IEEE INTERNET OF THINGS, and the IEEE TRANSACTIONS ON COMMUNICATIONS.

Linga Reddy Cenkeramaddi received the master’s degree in electrical engineering from the Indian Institute of Technology, New Delhi, India, in 2004, and the Ph.D. degree in electrical engineering from the Norwegian University of Science and Technology, Trondheim, Norway, in 2011. He worked for Texas Instruments in mixed signal circuit design before joining the PhD program at NTNU. After finishing his PhD, he worked in radiation imaging for an atmosphere space interaction monitor (ASIM mission to International Space Station) at University of Bergen, Norway from 2010 to 2012. At present, he is the group leader of the autonomous and cyber-physical systems (ACPS) research group and working as an Associate Professor at University of Agder, Campus Grimstad, Norway. His main scientific interests are in Cyber-Physical Systems, Autonomous Systems and Wireless Embedded Systems. He has co-authored over 80 research publications that have been published in prestigious international journals and standard conferences. He is co-principal investigator of many research grants from Norwegian Research Council. He is a senior member of IEEE. He is also a member of the Editorial Boards of various international journals, as well as the technical program committees of several IEEE conferences.