



# INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : [www.joiv.org/index.php/joiv](http://www.joiv.org/index.php/joiv)



## An Experimental Study on Deep Learning Technique Implemented on Low Specification OpenMV Cam H7 Device

Rosa Andrie Asmara<sup>a</sup>, Ulla Delfana Rosiani<sup>a</sup>, Mustika Mentari<sup>a</sup>, Arie Rachmad Syulistyo<sup>a</sup>, Milyun Ni'ma Shoumi<sup>a</sup>, Mungki Astiningrum<sup>a</sup>

<sup>a</sup> Information Technology Department, State Polytechnic of Malang, Lowokwaru, Malang, Indonesia

Corresponding author: \*[rosa.andrie@polinema.ac.id](mailto:rosa.andrie@polinema.ac.id)

**Abstract**— This research aims to identify and recognize the OpenMV Camera H7. In this research, all tests were carried out using Deep Machine Learning and applied to several functions, including Face Recognition, Facial Expression Recognition, Detection and Calculation of the Number of Objects, and Object Depth Estimation. Face Expression Recognition was used in the Convolutional Neural Network to recognize five facial expressions: angry, happy, neutral, sad, and surprised. This allowed the use of a primary dataset with a 48MP resolution camera. Some scenarios are prepared to meet environment variability in the implementation, such as indoor and outdoor environments, with different lighting and distance. Most pre-trained models in each identification or recognition used mobileNetV2 since this model allows low computation cost and matches with low hardware specifications. The object detection and counting module compared two methods: the conventional Haar Cascade and the Deep Learning MobileNetV2 model. The training and validation process is not recommended to be carried out on OpenMV devices but on computers with high specifications. This research was trained and validated using selected primary and secondary data, with 1500 image data. The computing time required is around 5 minutes for ten epochs. On average, recognition results on OpenMV devices take around 0.3 - 2 seconds for each frame. The accuracy of the recognition results varies depending on the pre-trained model and the dataset used, but overall, the accuracy levels achieved tend to be very high, exceeding 96.6%.

**Keywords**—Deep learning; face recognition; face expression recognition; depth estimation; object detection; object counting; CNN.

Manuscript received 29 Oct. 2023; revised 5 Dec. 2023; accepted 13 Jan. 2023. Date of publication 31 May 2024. International Journal on Informatics Visualization is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.



### I. INTRODUCTION

Minicomputers have gained popularity recently, offering a range of options for enthusiasts, students, and professionals to delve into embedded applications. The top choices in this category are the Raspberry Pi, Arduino, and OpenMV Cam. The Raspberry Pi, initially introduced by the UK-based Raspberry Pi Foundation in 2012, was conceived as an affordable and user-friendly computer for programming and electronics education. Over time, it has evolved, boasting features like a robust quad-core processor, up to 8GB of RAM, and 4K video support. In contrast, Arduino, created by an Italian team in 2005, is designed to simplify interactive projects and prototype development. It comprises a variety of microcontrollers, such as the well-known Arduino Uno and Arduino Nano, with an intuitive development environment.

OpenMV Cam, introduced in 2015 by founders Kwabena Agyeman and Ibrahim Abdelkader, has emerged as a compelling player in the minicomputer landscape. It offers an

affordable machine vision platform for the swift and straightforward development of embedded computer vision applications. OpenMV Cam, equipped with a potent microcontroller and a high-quality image sensor, facilitates coding and debugging in a Python-based development environment. It additionally supports a wide array of popular machine vision algorithms and libraries. One of OpenMV Cam's standout features is its user-friendliness. Its intuitive development environment and pre-built libraries expedite the initiation of computer vision projects with minimal hassle.

Moreover, the platform provides many examples and tutorials to help users harness its potential effectively. OpenMV Cam excels in real-time machine vision processing, making it a favored choice for robotics, drones, and industrial automation applications. Its cost-effectiveness further extends its appeal, rendering it accessible to a broad user base. Furthermore, its compact physical dimensions make it particularly suitable for wearable devices. The OpenMV Cam H7 has an OV7725 image sensor that can capture images at a

resolution of 640x480 in 8-bit Grayscale or 16-bit RGB565 format. It achieves a frame rate of 75 FPS for resolutions above 320x240 and 150 FPS for resolutions below 320x240. This limitation is entirely handled for implementing deep learning applications in computer vision since they only need a small image size resolution.

The physical size of the OpenMV Cam depends on the specific model. The current iteration, the OpenMV Cam H7, measures approximately 45mm x 36mm x 25mm (1.8in x 1.4in x 1in), making it a compact and portable device. It is small enough to fit in the palm and can be easily mounted on various robotic or other embedded platforms. Regarding technical specifications, the OpenMV Cam H7 is powered by an STM32H7 microcontroller running up to 480MHz. It features 512KB of RAM and 2MB of flash memory, which provides ample space for storing machine vision algorithms and other data.

OpenMV Cam offers distinct advantages in computer vision over Raspberry Pi and Arduino. It features built-in hardware support for image processing, simplifying the development of computer vision applications. Additionally, the platform includes a range of pre-built machine vision algorithms, such as edge detection and color tracking, which can be easily employed and tailored to specific applications. However, it is worth noting that OpenMV Cam employs a lower-power microcontroller and offers less memory than Raspberry Pi, potentially limiting its performance for complex computer vision tasks. A comparative study also favors OpenMV Cam regarding technical specifications and features [1].

Some of the previous research in Computer Vision tasks for Raspberry Pi has been conducted. Research on using the Raspberry Pi 3 as an Object Detector on a Robot Boat was carried out and showed quite good responsiveness on boat devices [2]. In a study by Rosa Andrie et al. [3], an experiment was carried out to predict traffic density using a Raspberry Pi and added parallel computing capabilities using Intel NCS2. The Intel Neural Compute Stick 2 (NCS2) is a powerful, compact device designed for deep learning inference at the edge. In this study, an experiment was carried out using Jetson Nano, which was used for Deep Object Detection [4]. In the next study, an experiment was carried out using Face Recognition on the Raspberry Pi, but the recognition process was carried out on the Google Vision API [5]. Research related to portable sensors using the CC2650 sensor tag has been carried out to analyze the properties and behavior of portable sensors related to data communication, durability, and required power consumption [6], [7].

The implementation of this experimental study can become the basis for further research, especially research related to the design of wearable device prototypes for individuals with visual impairments. Several related studies include research [8] that encompasses the development of smart glasses for visually impaired users and the utilization of various methods, such as Support Vector Machine (SVM), Local Binary Patterns (LBP), Convolutional Neural Networks (CNN), and Deep Convolutional Neural Networks (DCNN), for facial expression recognition. LBP can improve classification accuracy, but in practice, it is challenging to implement in real time due to the associated time complexity [9]. The SVM method has a high level of accuracy; the more accurate the

decision boundary used to classify data classes, the higher the accuracy. However, it is not easy to classify all data accurately, so some outliers are ignored [10]. The CNN method provides better accuracy than other facial expression recognition methods [11].

This study aims to conduct experiments using several Computer Vision Techniques on OpenMV cameras and see the possibility of their implementation in Assistive Devices for visually impaired Persons. The trial will show the performance of the computer vision technique on the OpenMV Camera and the camera's performance with these techniques. Some techniques that will be implemented include Face Expression Recognition, Deep Face Recognition, Deep Object Detection and Counting, Deep Depth Object Estimation, and Object Detection and Counting using CNN and HaarCascade [12]. The Face Expression Recognition used in the test will use the Convolutional Neural Network to recognize five facial expressions: angry, happy, neutral, sad, and surprised. The dataset was self-made using a 48MP resolution camera. The Face Recognition used in the test will utilize MobileNetV2 [13] architecture as the pre-trained model. The data used as a trial is data on 30 different face IDs. Object detection in testing uses two methods, namely Haarcascade and MobileNetV2, as the pre-trained model and uses CIFAR-100 [14] as the dataset. The final test is the estimation of object depth using CNN DenseNet-169 as the pre-trained model and tested on the NYU Depth V2 Dataset. This Experimental study will also show the results of object detection and counting tests using Haar Cascade Classifier.

This paper proposed some contributions as detailed:

- Experiment results of Deep Learning techniques implemented in OpenMV Cam H7 device.
- Analysis of the processing time and recommended technique used for OpenMV Cam.
- Primary dataset development with some specific and unique conditions and taken using OpenMV Cam.

## II. MATERIALS AND METHOD

Numerous studies on facial expressions have been conducted, including one by [15], which employed the DCNN method to enhance facial expression recognition accuracy. The study utilized the Kaggle FER2013 dataset for training and experimenting with the DCNN model. Similarly, [16] proposed using DCNN modeling to create a highly accurate facial expression recognition system. This approach involved deep DCNN modeling through Transfer Learning (TL), where a pre-trained DCNN model was adapted for facial emotion data by replacing its upper layer. The proposed facial expression recognition system was evaluated using eight different pre-trained CNN models (VGG-16, VGG-19, ResNet-18, ResNet-34, ResNet-50, ResNet-152, Inception-v3, and DenseNet-161) and the KDEF and JAFFE datasets. Subsequent research by Pranav et al. [17] introduced a convolutional network model with two layers for facial emotion recognition, incorporating dropouts after each convolution layer. The objective was to develop a Deep Convolutional Neural Network (DCNN) model to classify five distinct human facial emotions. Moreover, research conducted by [18] utilized DCNN for facial emotion recognition, employing two datasets—the Amsterdam

Dynamic Facial Expression Set-Bath Intensity Variations (ADFES-BIV) and the WSEFEP datasets. Notably, OpenMV Cam was also employed for model testing. Several studies have delved into object and face recognition using OpenMV Cam, with one such study [19] focusing on mask detection using OpenMV Cam H7.

These studies serve as reference points in the realm of CNN-based facial recognition. For instance, a Smart Attendance Management System based on Facial Recognition utilized a custom CNN model to facilitate real-time student attendance using facial recognition. The proposed CNN consisted of 20 layers, encompassing a Two-Dimensional Convolutional Layer (Conv2D), Batch Normalization Layer, Max Pooling Layer, and Dense Layer, among others [20]. Another study explored an Android system for guest authorization, leveraging Google Cloud Service for the recognition process. This research incorporated ArcFace and FaceNet, CNN architectures used for facial recognition. It employed encoding via OpenCV, MTCNN, and RetinaFace, with ArcFace demonstrating superior facial recognition accuracy compared to Additionally, Rajput et al. introduced CNN-based low-resolution Facial Recognition Classification, highlighting the use of CNNs to overcome challenges in low-light conditions and achieve accurate face recognition. The study employed Resnet18, consisting of 18 CNN-based layers with Rectified Linear Unit (ReLU) activation functions. This CNN classification concept enhanced recognition accuracy compared to traditional K Nearest Neighbor (KNN) based classifiers [21].

Numerous studies have investigated object detection and counting using CNN, yielding promising results [22], [23], [24], [25], [26], [27], [28], [29]. Detection and recognition of particular objects in the room using CNN is implemented in a support system for blind people. These objects include windows, notice boards, elevators, doors, electrical contacts, trash cans, stairs, tables, smoke detectors, and other objects. The accuracy achieved precision results reaching up to 73.19% [22] Romario et.al detected and counted human objects in crowds and conducted 10 tests with a total of 113 detections achieving an accuracy value of 63% using CNN [23]. This result lacks accuracy but considering that detection in a crowded atmosphere is full of challenges, it is understandable if the accuracy is not that high. Aich et al. employed Global Sum Pooling to enhance accuracy in detecting vehicle objects, crowd objects, and wheat plants, asserting the approach's superiority over existing methods [24]. The multilayer regression network (MRNet) is used as a counter object in crowds and was published in 2019 by Xin Tan et al. [25]. MRNet shows more reliability and higher accuracy in different crowd scenarios. Across the four crowd-counting datasets, MRNet provided competitive performance compared to other recent methods. Several other studies have used a different CNN architecture with varied claim results, such as Scale Pyramid Network [26], MCENet [27], and CentroidNetV2 [28].

Swaraja et al. [30] conducted experiments using 400 homogeneous and 4600 heterogeneous images in object depth detection. The results indicated that EfficientNet exhibited lower object depth detection accuracy at higher input resolutions than ResNet50. Another research venture

explored Recurrent CNN with sparse depth for implementation in SLAM, demonstrating its accuracy, real-time capabilities, and suitability for precise localization and mapping [31]. Further studies showcased fast monocular depth estimation through sparse MobileNet and ASPP-implemented CNNs on FPGAs, achieving high accuracy and real-time speed [32]. Additionally, Ibraheem et al. proposed a CNN for depth map estimation from a single RGB image, highlighting the potential of well-constructed encoders like DenseNet-169 and DenseNet-201 to produce higher-quality depth maps capturing object boundaries more precisely [33]. Five different methods were studied experimentally and applied to OpenMV Camera. The entire experiment uses the existing Net architecture. For data training, the experiments use primary and secondary datasets available.

#### A. OpenMV Camera

OpenMV stands out as a remarkably cost-effective and energy-efficient embedded intelligent camera platform that is purpose-built to cater to various applications within computer vision and wireless sensor networks. It is distinguished by its compatibility with Python 3 and the wealth of resources it brings. These include an expansive and versatile computer vision library, a user-friendly integrated development environment (IDE), and comprehensive preconfigured script examples. The central board is at the heart of the OpenMV ecosystem, compactly measuring 1.4' x 1.2'. This board serves as the core hub, housing crucial components such as a high-quality image sensor MT9M114, a powerful microcontroller unit (MCU), an efficient power supply system, a micro-SD card slot for data storage, and expansion headers to facilitate hardware extensions and customizations. The MT9M114 camera sensor initiates data processing by capturing light through its image sensor core, converting it into electrical signals, and then performing analog-to-digital conversion (ADC) to quantize these signals into digital values. Subsequently, digital signal processing (DSP) operations are applied for image enhancement, including tasks like white balance adjustments and noise reduction. The sensor formats the processed data into a suitable output format, often in the RGB or YUV color space, and transmits it through interfaces like MIPI CSI-2 or parallel connections to the host system. In the host system, further processing, such as compression or analysis, occurs before displaying, storing, or transmitting the final image, allowing control and configuration of sensor parameters as needed. OpenMV Cam H7 Device and the block diagram for MT9M114 are shown in Fig. 1 below.

The OpenMV library, the backbone of its functionality, is meticulously developed in the C programming language, providing the foundation for seamless integration with Python scripts. Users gain access to a broad spectrum of fundamental image manipulation capabilities within this library. These include image loading and saving, precise cropping, flexible resizing, and versatile blending. Furthermore, the library extends its prowess to advanced image enhancement functions, including median filtering for noise reduction, midpoint adjustments for dynamic range enhancement, Gaussian smoothing for artifact removal, and histogram equalization for improved image quality.

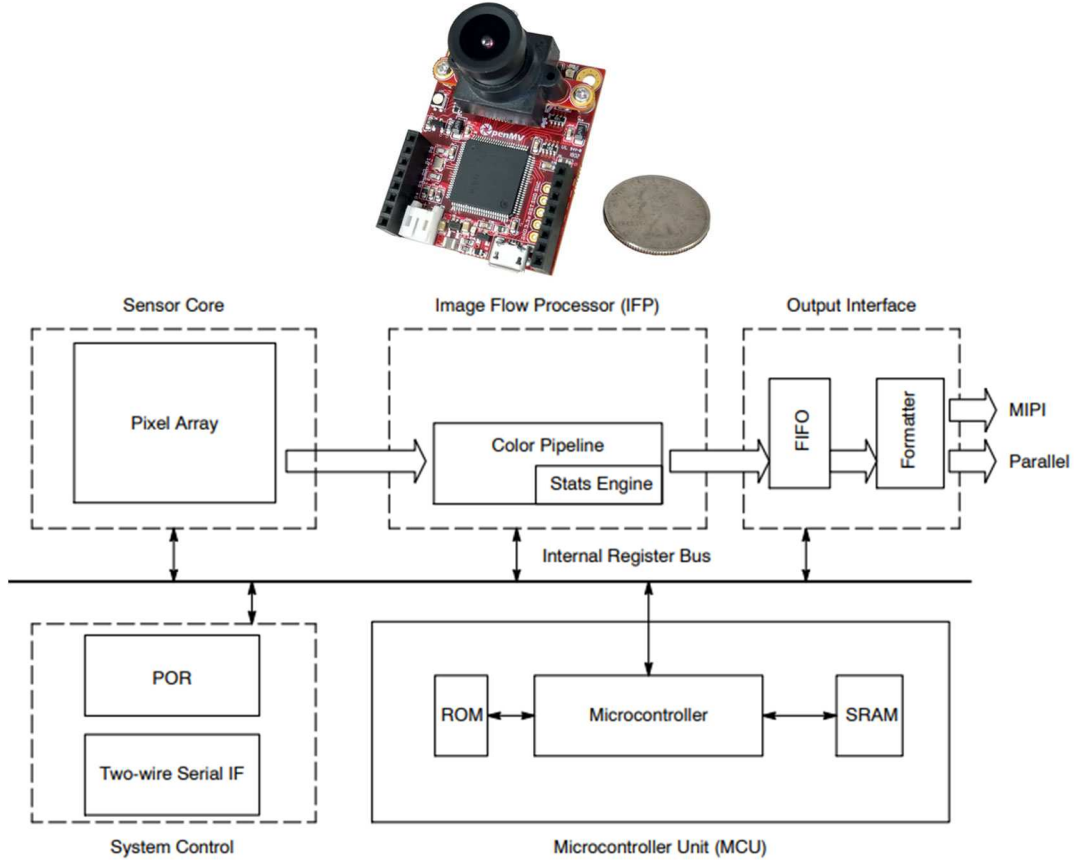


Fig. 1 (Top) OpenMV Cam H7 Device; (Bottom) MT9M114 Schematic diagram

Notably, OpenMV goes beyond the basics by offering a range of sophisticated image-processing libraries. These advanced capabilities empower users to tackle complex tasks with ease. OpenMV can proficiently handle tasks such as QR code recognition and decoding, ApriTags for precise localization and tracking, and various forms of facial and ocular recognition, enabling applications in fields as diverse as robotics, automation, and surveillance. Additionally, OpenMV excels in iris detection, allowing for biometric applications and enhanced security measures [1].

### B. MobileNetV2

MobileNetV2, purpose-built for efficient deep learning inference on mobile and embedded platforms, represents a cutting-edge neural network architecture. It builds upon the original MobileNet architecture by prioritizing resource efficiency while maintaining high performance. The key innovation lies in its use of depthwise separable convolutions, a combination of depthwise and pointwise convolutions that dramatically reduces parameters and computations while retaining the ability to capture complex features.

For the regular convolutions using input  $L_i: height_i \times width_i \times depth_i$ , implement filter kernel  $K \in \mathbb{R}^{k \times k \times d_i \times d_j}$  and giving output result  $L_j: height_j \times width_j \times depth_j$  and having computational cost:

$$height_i \cdot width_i \cdot depth_i \cdot depth_j \cdot k \cdot k \quad (1)$$

while for depthwise separable convolution, the computational cost is:

$$height_i \cdot width_i \cdot depth_i (k^2 + depth_j) \quad (2)$$

The concept of inverted residuals enhances representation without inflating the model size. Furthermore, it balances computational efficiency and expressive power through linear bottlenecks, strategically applying nonlinear activations only where needed. MobileNetV2 offers architectural flexibility through width and resolution multipliers, enabling customization of model size and computational costs. This architecture finds applications in mobile and edge computing [34]. Depthwise separable convolutions, a combination of depthwise and pointwise convolutions, characterize the convolutional block. These significantly reduce model parameters and computational complexity while preserving the ability to capture intricate features. It also incorporates the concept of inverted residuals to improve feature representation without increasing the overall model size. Additionally, the block balances computational efficiency and expressive power through linear bottlenecks, applying nonlinear activations judiciously. Fig. 2 below shows the mobilenetV2 convolutional block in detail.

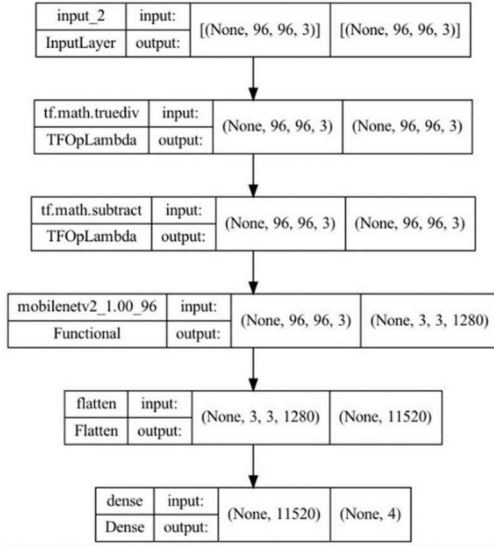


Fig. 2 MobileNetV2 Convolutional Block

The significance of MobileNetV2 lies in its capability to deploy advanced computer vision models on resource-constrained devices. It adapts seamlessly to various tasks, including object recognition, semantic segmentation, and image classification, consistently delivering competitive accuracy within stringent resource constraints. It underscores

the critical role of efficient neural network architecture in extending the boundaries of deep learning to edge devices, making it an indispensable tool for real-time on-device AI applications.

### C. DenseNet-169

DenseNet-169, a highly regarded convolutional neural network (CNN) architecture renowned for its exceptional performance in tasks such as image classification and feature extraction, is a notable member of the DenseNet family. Its dense connectivity patterns set it apart, enabling unprecedented information transmission rates.

In contrast to conventional CNNs, where each layer receives input solely from the previous layer, DenseNet-169 fosters feature reuse by tightly connecting each layer to all subsequent levels. This innovation accelerates the training process and significantly reduces the number of parameters compared to networks of similar depth, thereby enhancing model efficiency. DenseNet-169 incorporates bottleneck layers and transition blocks to streamline the model further while preserving or improving accuracy, making it the preferred choice for precision-critical applications like computer vision, object detection, and image segmentation. Fig. 3 below shows the DenseNet-169 Architecture model and the comparison with another popular DenseNet model.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112		7 × 7 conv	7 × 7 conv	7 × 7 conv
Pooling	56 × 56		3 × 3 max pool, stride 2	3 × 3 max pool, stride 2	3 × 3 max pool, stride 2
Dense Block (1)	56 × 56	1 × 1 conv 3 × 3 conv × 6	1 × 1 conv 3 × 3 conv × 6	1 × 1 conv 3 × 3 conv × 6	1 × 1 conv 3 × 3 conv × 6
Transition Layer (1)	56 × 56 28 × 28		1 × 1 conv 2 × 2 average pool, stride 2	1 × 1 conv 2 × 2 average pool, stride 2	1 × 1 conv 2 × 2 average pool, stride 2
Dense Block (2)	28 × 28	1 × 1 conv 3 × 3 conv × 12	1 × 1 conv 3 × 3 conv × 12	1 × 1 conv 3 × 3 conv × 12	1 × 1 conv 3 × 3 conv × 12
Transition Layer (2)	28 × 28 14 × 14		1 × 1 conv 2 × 2 average pool, stride 2	1 × 1 conv 2 × 2 average pool, stride 2	1 × 1 conv 2 × 2 average pool, stride 2
Dense Block (3)	14 × 14	1 × 1 conv 3 × 3 conv × 24	1 × 1 conv 3 × 3 conv × 32	1 × 1 conv 3 × 3 conv × 48	1 × 1 conv 3 × 3 conv × 64
Transition Layer (3)	14 × 14 7 × 7		1 × 1 conv 2 × 2 average pool, stride 2	1 × 1 conv 2 × 2 average pool, stride 2	1 × 1 conv 2 × 2 average pool, stride 2
Dense Block (4)	7 × 7	1 × 1 conv 3 × 3 conv × 16	1 × 1 conv 3 × 3 conv × 32	1 × 1 conv 3 × 3 conv × 32	1 × 1 conv 3 × 3 conv × 48
Classification Layer	1 × 1		7 × 7 global average pool 1000D fully-connected, softmax	7 × 7 global average pool 1000D fully-connected, softmax	7 × 7 global average pool 1000D fully-connected, softmax

Fig. 3 DenseNet-169 Compared to other DenseNet Layer Structure

Networks consist of  $L$  layers, in each implement  $H_l(\cdot)$  nonlinear transformation, which  $l$  is the layer indexes. Assumed that the output of the  $l^{th}$  layer is  $x_l$ , will give the layer transition:

$$x_l = h_l(x_{l-1}) \quad (3)$$

ResNet, on the other hand, will bypass the non-linear transformation:

$$x_l = h_l(x_{l-1}) + x_{l-1} \quad (4)$$

The densenet will improve information flow from layer to layer using direct connection to all layers, where  $l$ th layer handles all feature maps from all previous layer  $x_0, \dots, x_{l-1}$ :

$$x_l = h_l([x_0, x_1, \dots, x_{l-1}]) + x_{l-1} \quad (5)$$

where  $[x_0, x_1, \dots, x_{l-1}]$  is the combination of feature maps from layers  $0, \dots, l-1$

The figure below shows the Layers of DenseNet-169 compared to other DenseNet pre-trained models. The impact of DenseNet-169 extends beyond image classification, finding utility in diverse fields such as medical image analysis, where it has demonstrated advantages in tasks like organ segmentation and disease detection. The model converges faster and achieves superior performance thanks to its architectural features that encourage feature reuse, mitigate the vanishing gradient problem, and facilitate superior



gradient flow during training. While DenseNet-169 remains a remarkable architectural achievement, it is essential to acknowledge that deep learning has continued to evolve. Researchers are now exploring novel architectural concepts, regularization techniques, and optimization methods to push the boundaries of model effectiveness and performance [35], [36], [37]. A comparative study of various CNN architectures, including DenseNet, ResNet, Inception, and InceptionResNet, found that the DenseNet architecture outperformed the others with an impressive accuracy of 98.76% [38].

#### D. Face Expression and Face Recognition

The data consists of facial images taken using an OpenMV device with Haarcascade as a method for face detection. 1560 images comprised three categories: happy, sad, and neutral. Each label has 500 facial images with a .jpg extension as training data and 150 with a .jpg extension as testing data. Tests were carried out in eight scenarios: location (indoor/outdoor), lighting intensity, and the distance between the camera and the target's face. Four labels/classes were used for face recognition; each label consisted of 500 face images with the .jpg extension as training data, and 10 face images with the .jpg extension as test data. The same scenario is also used for face recognition experiments. Image augmentation used for the datasets is random contrast and random rotation. Table I below shows scenarios for the experiments and Fig. 4 below shows some image examples in the face expression and face recognition primary dataset.

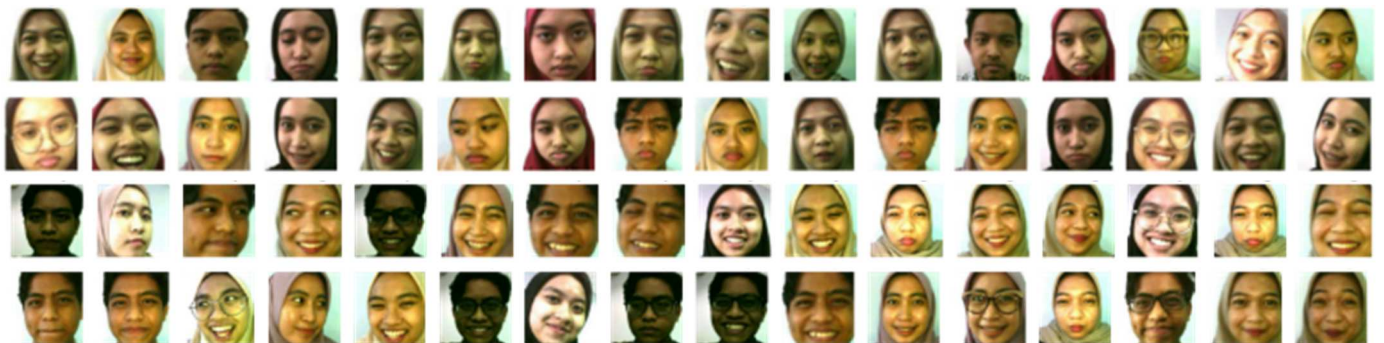


Fig. 4 Image Sample on Primary Dataset Created

- Images encompass the entire human body, from feet to head.
- Images focus on the upper human body, from the waist to the head.
- Images concentrate on the lower human body, ranging from the feet to the waist.

After selecting the most influential dataset, the human body component dataset was employed for the human object detection process. The data underwent initial processing by converting RGB colors to grayscale. Subsequently, it progressed to classification using the Haar Cascade Classifier, involving accuracy assessment and cumulative evaluation of test results with consistent labeling. Once the grayscale conversion was accomplished, a cascade classifier was employed for image data processing. This classifier generated multiple weak classifiers, typically stored at nodes. These weak classifiers were combined into a filter chain to construct a robust classifier that recognizes the trained objects. Each filter possessed a dynamic threshold, facilitating the

TABLE I  
EIGHT DIFFERENT SCENARIOS FOR FACE EXPRESSION AND RECOGNITION SYSTEM

No	Scenarios		
	Environment	Lightness	Distance
1	Indoor	30 Lux	30 cm
2	Indoor	30 Lux	100 cm
3	Indoor	100 Lux	30 cm
4	Indoor	100 Lux	100 cm
5	Outdoor	150 Lux	30 cm
6	Outdoor	150 Lux	100 cm
7	Outdoor	250 Lux	30 cm
8	Outdoor	150 Lux	100 cm

#### E. Object Detection and Counting Using HaarCascade and MobileNetV2

Detecting and counting objects was a central focus of this experiment, involving applying different methods and datasets. The first method, Haarcascade, utilized primary and secondary research data. Primary data comprised digital images captured using the camera module of the openMV device, encompassing various locations such as the campus and laboratory. In contrast, secondary data was derived from images taken with cellphone cameras, DSLRs, and the Penn-Fudan dataset. The secondary dataset assessed detection effectiveness, particularly for specific human body parts. The study categorized three distinct human body part datasets for detection purposes:

estimation of its weight and placement in the classification process. If any filter failed to determine that a particular component did not match the trained object, the object was classified as a non-object. The generic XML file containing these filters proved instrumental in identifying human objects within the system.

The Haar cascade classifier computed results based on feature extraction outcomes, such as edges, lines, or square features. The filters trained in the earlier stages were utilized to ascertain the Haar-cascade detected area, utilizing the image integral. After classification completion, tests were conducted, focusing on two primary aspects: achieving the most accurate detection results in human object detection trials based on human body parts, using OpenCV, and assessing detection outcomes according to the type of human body parts, using training data stored in the XML file. Additionally, training and testing processes were carried out using openMV Cam. The dataset is obtained from CIFAR-

100, and only some classes are taken. Images were taken with complex background and plain background images.

#### F. Depth Object Estimation

In the initial stage, after separating the video from the image input into the system, the input data is resized with dimensions of 640x480 to fit the size of the dataset used, which is dimensions of 640x480. The model will be loaded after the input image has been divided into numerous frames, and it will then be analyzed to determine an object's depth. This system uses a pre-trained model from Alhashim et al. [30], which was developed using the NYU v2 dataset and the DenseNet-169 architectural encoder.

### III. RESULTS AND DISCUSSION

#### A. Face Expression Recognition using CNN

Finding optimal hyperparameters (hyperparameter tuning or hypertuning) is needed to get a model with maximum performance. Hypertuning helps improve performance and reduce model complexity by removing unnecessary parameters. Table II below shows several experiments using different hyperparameters, epochs, and learning rates. In the first experiment, model training for machine learning used epochs of 10 with a learning rate 0.001. In the first experiment, the model learned image data on training data of 0.99 (99%) and showed a learning error (loss) on training data of 0.0415, while the error on data validation was 0.1216.

TABLE II  
HYPERTUNING EXPERIMENT RESULT

No.	Epoch	Learning Rate	Accuracy		Loss		Gap Training - Validation
			Training	Validation	Training	Validation	
1.	10	0.001	0.9992	0.9767	0.0415	0.1216	0.0801
2.	10	0.0001	0.9350	0.8867	0.2932	0.3487	0.0555
3.	10	0.0005	0.9883	0.9700	0.0898	0.1454	0.0556
4.	30	0.001	1.0000	0.9600	0.0092	0.0900	0.0808
5.	30	0.0001	0.9858	0.9400	0.1137	0.1966	0.0829
6.	30	0.0005	1.0000	0.9533	0.0177	0.1035	0.0858
7.	50	0.001	1.0000	0.9667	0.0037	0.0756	0.0719
8.	50	0.0001	0.9992	0.1391	0.0547	0.9567	0.9002
9.	50	0.0005	1.0000	0.9667	0.0078	0.0808	0.0703

Fig. 5 below shows a graph of good Accuracy and Error identified by training loss and validation loss, which decreases to a point of stability with a minimum gap between the two final loss values so that experiment number 2 is the best result of the 9 experiments that have been carried out with a gap value of 0.0555. Table III below shows the results of the hypertuning experiment on MobileNetV2. Experiments were carried out using differences in epochs. A good model is identified by loss training and validation, which decreases to a point of stability with a minimal gap between the two final loss values. The loss of the model is almost always lower on the training dataset than on the validation dataset, which means there must be some gap between the learning curve loss between training and validation. This gap is referred to as the "generalization gap". The test results in Table III for MobileNetV2 show that with the addition of layers, the best model uses epoch 30 with a gap value of 0.0702.

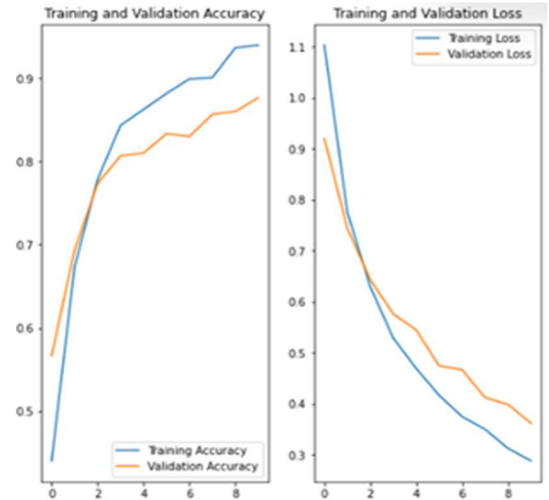


Fig. 5 Training and Validation Accuracy and Loss Experiment Results for Face Expression Recognition

TABLE III  
MOBILENETV2 EXPERIMENT RESULT WITH DIFFERENT EPOCH TRAINING

No	Epoch	Training		Validation		Generalization Gap	Time
		Loss	Accuracy	Loss	Accuracy		
1.	10	0.9462	0.5533	0.8635	0.6087	0.0827	3min 16s
2.	30	0.0083	1	0.0785	0.9767	0.0702	5min 85s
3.	50	0.003	1	0.0755	0.9733	0.0725	17min 19s

Tests were carried out on facial expressions using the CNN models, which have been converted and implemented on OpenMV Cam H7 Plus, to determine whether the model can run when performing classification using machine vision as shown on the Fig. 6 below. An example of the results of testing facial expressions using the CNN method can be seen in Figure 6. In the pre-trained model used, 8 tests were also

carried out, of which 4 were carried out indoors, and the rest were carried out outdoors where each test had a lighting parameter (in LUX units) and the distance between the camera and the correspondent (in CM / M units). The testing results for all scenarios can be seen in Table IV, where the values for accuracy, precision, recall, and F1 Score displayed

are the average of the values obtained for happy, neutral, and sad expressions.

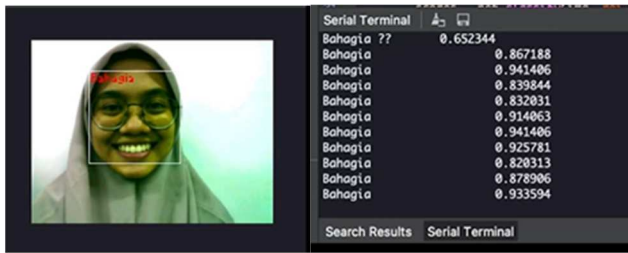


Fig. 6 Face Expression Experiment Result using OpenMV

There were 4 indoor and outdoor experiments where each test had a lighting parameter (in LUX units) and the distance between the camera and the correspondent (in CM / M units). The testing results for all scenarios can be seen in Table IV, where the values for accuracy, precision, recall, and F1 Score displayed are the average of the values obtained for happy, neutral, and sad expressions. From the results of Table IV below, the highest accuracy value is in the 6th scenario (outdoor environment, 150 Lux, and 100cm distance) achieving 72% classification accuracy. These results show

that the model used can work with good accuracy on the OpenMV camera with computational costs that can also be accommodated well. The running time for each frame using this model is around 0.3 seconds on average.

TABLE IV  
ACCURACY, PRECISION, RECALL, AND F-1 SCORE FOR 8 DIFFERENT SCENARIOS

Scenario	Accuracy	Precision	Recall	F-1 Score
1	0.60	0.40	0.51	0.39
2	0.62	0.43	0.61	0.39
3	0.62	0.43	0.50	0.42
4	0.62	0.43	0.45	0.36
5	0.57	0.36	0.49	0.35
6	0.72	0.58	0.75	0.55
7	0.54	0.32	0.31	0.26
8	0.46	0.18	0.12	0

### B. Face Recognition Using CNN

To get a model with maximum performance, the process of finding optimal hyperparameters (hyperparameter tuning or hypertuning) is needed in machine learning. Hypertuning helps improve performance and reduce model complexity by removing unnecessary parameters. Table V below shows some experiments using different hypertuning parameters.

TABLE V  
HYPERTUNING EXPERIMENT TABLE

No.	Epoch	Learning Rate	Accuracy		Loss		Gap Train - Val
			Training	Validation	Training	Validation	
1	10	0.001	1.0000	0.8500	0.0016	0.6082	0.6066
2	10	0.0005	1.0000	0.8000	0.0084	0.4512	0.4428
3	30	0.001	1.0000	0.8750	1.6517e-04	0.3897	0.389535
4	30	0.0005	1.0000	0.8750	0.0014	0.4855	0.4841
5	50	0.001	1.0000	0.8750	1.7187e-04	0.4121	0.411928
6	50	0.0005	1.0000	0.8750	0.0011	0.4376	0.4365
7	100	0.001	1.0000	0.9250	1.0895e-04	0.4137	0.413591
8	100	0.0005	1.0000	0.8750	4.0342e-04	0.3550	0.354597

A good model is identified by training loss and validation loss, which decreases to a point of stability with a minimum gap between the two final loss values so that experiment number 7 is the best result of the 8 experiments that have been carried out with a gap value of 0.413. The experiment results on Fig. 7 show excellent recognition, achieving 92.5% accuracy at around 0.354 seconds for each frame process. These results show that the model used can work with extremely good accuracy on the OpenMV camera.

### C. Depth Object Estimation using CNN

Tests were carried out to find out whether the system worked properly or not when detecting the depth of objects in indoor and outdoor conditions with different lighting. The system's output is in the form of video with object depth detection, where the closer the object is detected, the darker it will be, and the deeper the object is detected, the brighter it will be. In the accuracy test, two frames are taken from each system output with different conditions and lighting, and each frame is given a bounding box randomly

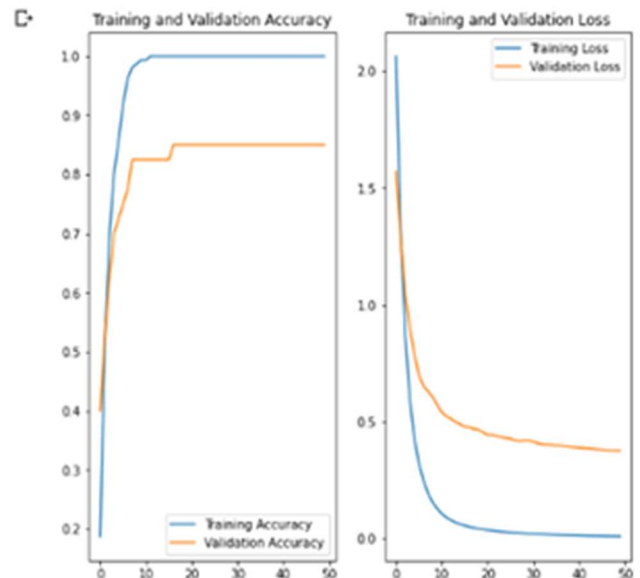


Fig. 7 Training and Validation Accuracy and Loss Experiment Results for Face Recognition

Then, an assessment is made of whether the actual depth detection is appropriate. Another test was carried out to



determine how fast the system performs inference processing for each entered frame. Some examples of experimental

results of object depth estimation accuracy can be seen in Table VI.

TABLE VI  
OBJECT DEPTH ESTIMATION ACCURACY EXAMPLE

No.	Experiment images	Depth detection	
		Correct	Incorrect
1	Indoor image with appropriate lighting 	28	2
2	Indoor image with dim lighting 	29	1
3	Outdoor image with appropriate lighting 	19	11




From all the experiments, the calculation of the results of the appropriate depth detection accuracy for each condition was carried out. The first condition produces an accuracy of 96.6%, while for the second and third condition produces an accuracy of 73.3 and 63.3%. In addition to accuracy calculations, tests were also carried out to find out how fast the system was to process each frame on the DensNet 169 model, and the results obtained were:

- In the object depth detection process, every frame takes 1.4 seconds for every frame or 0.6 FPS in every frame.
- From the input process to combining frames or output, it takes 2.08 seconds or 0.5 FPS.

#### D. Object Detection and Counting using CNN and Haar Cascade

The object detection and calculation system were tested using the CNN method on 13 images that had been taken with OpenMV, where the 13 images consisted of 12 images with slightly complex background conditions and one image with a plain background. These images are taken under different conditions, for example, the number of more than one object, the distance between objects that are close together and distant, and background conditions that are plain and not plain. An example of the system testing results using the CNN method can be seen in Table VII.

TABLE VII  
SYSTEM EXPERIMENT RESULTS PREVIEWS

No.	Experiment images	Detection		Real count result	Count detection result
		Correct	Incorrect		
1		2	1	Cup : 1 Person : 2	Person : 2
2		1	1	Chair : 1 Bowl : 1	Chair : 1
3		1	1	Cup : 1 Person : 1	Person : 1

Based on these tests, it is difficult for OpenMV to detect objects in images with busy backgrounds, with the result that the number of objects detected is 13 out of 25 bounding boxes. Moreover, if the number of correct classifications is calculated based on the correctly detected objects, then of the 13 detected objects, it can be classified correctly. Then, for counting detected objects, the correct count is 7 of 23 object counts. The overall system test results can be seen in Table VIII below.

Based on Table VIII below, it was discovered that data with plain backgrounds had higher precision and accuracy values than data with complex backgrounds. Contrarily, complex backdrops performed better in categorization accuracy, recall, and precision than simple backgrounds. This is due to the identified item needing to be completely accurate; therefore, some background information is also classified in addition to classifying the object. In addition, the system uses the Otsu segmentation method and cannot alter image data.

In addition to using the CNN method, a human object detection and calculation system was also developed using the Haar Cascade method. During the system testing process, various conditions and object positions, such as the effect of the distance between the object and the camera, the effect of the difference in color between the object and the background, the effect of object conditions, and the effect of lighting, were all taken into consideration.

TABLE VIII  
COMPARISON OF PLAIN BACKGROUND AND COMPLEX BACKGROUND TEST RESULTS

	Detection			Classification			
	Prec	Rec	Acc	Prec	Rec	Acc	Accuracy
Non green screen background	0.53	0.66	0.42	0.94	0.97	0.99	0.538
Green screen Background	0.55	0.46	0.44	0.92	0.82	0.97	0.687

The experiment seeks to evaluate the effectiveness of human object detection under various image processing scenarios when utilizing the OpenMV tool. According to the tests, the maximum number of items that OpenMV can detect is four meters away from the camera, and the separation between the human object and the background has no bearing on the detection outcomes. Due to the limits of the Haar Cascade method, which can only detect stark color contrasts between objects and their surroundings, objects that are identical or nearly equal to the backdrop color are likely not detected by the system.

The system can recognize and calculate human things by recognizing upper body components. So that even if other things cover the lower body, the system can still identify and calculate human objects. The device can also detect people in

three different lighting scenarios: bright inside lighting, bright lighting near a window, and bright lighting far from a window. In contrast, in strong light outdoors, the system cannot recognize and calculate human items. Table IX below displays the total test outcomes. The system can detect and calculate human objects by detecting upper body parts. So that the system can still detect and calculate human objects even if other objects cover the lower body. In addition, the system can detect human objects in three different light conditions: bright light indoors, bright light near a window, and bright light far from a window. Meanwhile, the system fails to detect and calculate human objects outdoors in bright light conditions.

TABLE IX  
EXTERNAL CONDITION AFFECTION ON ACCURACY

No.	Experiment	Image Number	Correctly Classified
1	Distance affection	10	6
2	Object color difference	10	9
3	Object Condition	10	6
4	Lighting	4	3
<b>Total</b>	34	24	
<b>Accuracy (correctly classified: image amount)</b>			71%

In testing the detection and calculation of human objects using the Haar Cascade method with OpenMV processing, the correct percentage was obtained at 71%, while using CNN, the correct percentage was obtained at 66%. Test results found that the system tends not to detect and count objects with a low color difference or contrast with the background color. The system also fails to detect and count objects if other objects beyond the chest area block the human object. The test results show that implementing the haarcascade method in OpenMV is less accurate for detecting and calculating human objects in natural conditions. This is because the Haar Cascade algorithm works, which mainly uses the calculation of the difference in black and white pixel values in the image. So, optimal environmental conditions are needed to detect and count objects well. The optimal conditions required are:

- The distance between human objects and the camera is not more than 3 meters and not less than 1 meter,
- The distance between human objects is not too close,
- The objects do not overlap,
- Color the human object contrasts with the background color so that the object's body parts are precise,
- The object's body in the image is not cut off,
- The upper part of the human object's body, from the waist to the head, is not obstructed by other objects beyond the chest,
- The human object's body does not face sideways from the camera's direction.

Apart from that, based on the results of tests that have been carried out, the percentage rate of correct detection and calculation of human objects on OpenMV devices is lower than using OpenCV on PC devices. OpenMV devices have lower specifications than PC devices, using a 32-bit ARM cortex processor. The lower computing capability of OpenMV devices compared to PC devices is reinforced by the condition that OpenMV devices only support Micropython without OS (Operation System) support. With these specifications, the OpenMV device can only process images with a maximum resolution of 240p. If the OpenMV device is

given an image with a resolution above 240p, OpenMV will crash. Apart from that, the camera module of the OpenMV device is only capable of capturing images with a half-quarter VGA resolution of 240x160 pixels, which needs attention. The images produced by OpenCV devices are less sharp than those produced by smartphones or DSLR cameras. Inaccuracy in positioning the camera and adjusting the object's distance from the camera will affect the results of object detection and calculation. The experiment results from Table IX shows good classification, achieving 71% accuracy on HaarCascade and 66% accuracy on CNN, giving a testing time of around 0.4 second per frame. These results show that the model used can work with good accuracy on the OpenMV camera but with some limitations on the time process.

#### IV. CONCLUSION

The research proposed in this study is an experimental study using pre-trained Deep Learning models on the OpenMV Camera H7 device. As has been written, this camera focuses on developing systems that implement computer vision and wireless sensor networks. The pre-training used in this research is MobileNetV2 and DenseNet-169. Meanwhile, several computer vision implementations being tested include Face Recognition, Facial Expression Recognition, Detection and Calculation of the Number of Objects, and Object Depth Estimation. DenseNet-169 is used in Object Depth Estimation, while the others use MobileNetV2. This research tries to train and validate using selected primary and secondary data, with 1500 image data. The computing time required is around 5-7 minutes for every 10 epochs. The hyper tuning test results on facial expressions, as shown in Table II, give optimum results in the 10 epoch 0.001 learning rate with 97.67% validation accuracy, and the face recognition shown in Table V shows optimum results in the 100 epoch 0.001 learning rate with 92.5% validation accuracy. This test shows that in facial expression recognition, overfitting of the training process occurs more quickly, and in the range of 10 epochs, it is recommended to stop the training process. Meanwhile, in face recognition, overfitting conditions occur around 100 epochs, so it is recommended to stop the training process at that epoch with a recommended learning rate of 0.001.

For the face expression recognition result, as shown in Table IV, the highest accuracy value for the best hypertuning parameter applied is in the sixth scenario (outdoor environment, 150 Lux, and 100cm distance), achieving 72% classification accuracy and running time for each frame using this model is around 0.3 seconds on average. The experiment on the face recognition results shows excellent recognition, achieving 92.5% accuracy at around 0.354 seconds for each frame process. The experiment on depth object recognition using DenseNet169 achieved 96,6% accuracy in an indoor environment with proper lighting conditions, with a testing computation cost of around 1.4 seconds per frame. The experiment results from object detection and counting on Table IX show good classification, achieving 71% accuracy on HaarCascade and 66% on CNN and giving a testing time of around 0.4 seconds per frame. These results show that the model can work with reasonable accuracy on the OpenMV camera but with some limitations on the time process. From the results above, for several implementations such as Face Expression Recognition and Face Recognition, OpenMV

Camera is still applicable because the processing time is fast enough. It is quite suitable for Depth Object Estimation, Object Detection, and Counting in terms of accuracy. However, implementing it on an OpenMV Camera needs to be considered because the processing time of each frame is too high.

## REFERENCES

- [1] I. Abdelkader, Y. El-Sonbaty, and M. El-Habrouk, "Openmv: A Python powered, extensible machine vision camera." 2017.
- [2] A. Aqthobillobbany, A. N. Handayani, D. Lestari, Muladi, R. A. Asmara, and O. Fukuda, "HSV Based Robot Boat Navigation System," 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM), Nov. 2020, doi:10.1109/cenim51130.2020.9297915.
- [3] R. A. Asmara, B. Syahputro, D. Supriyanto, and A. N. Handayani, "Prediction of Traffic Density Using YOLO Object Detection and Implemented in Raspberry Pi 3b + and Intel NCS 2," 2020 4th International Conference on Vocational Education and Training (ICOVET), Sep. 2020, doi: 10.1109/icovet50258.2020.9230145.
- [4] A. N. Handayani, D. Lestari, Muladi, W. Ahmad, R. A. Asmara, and O. Fukuda, "Cognitive Function Tools/Robot Design for Elderly Using Image-Processing," 2022 International Conference on Electrical and Information Technology (IEIT), Sep. 2022, doi:10.1109/ieit56384.2022.9967856.
- [5] R. A. Asmara et al., "Face Recognition Using ArcFace and FaceNet in Google Cloud Platform For Attendance System Mobile Application," Proceedings of the 2022 Annual Technology, Applied Science and Engineering Conference (ATASEC 2022), pp. 134–144, 2022, doi:10.2991/978-94-6463-106-7\_13.
- [6] R. A. Asmara, I. Siradjuddin, and R. D. Romadhona, "Android based Wireless Sensor Network (WSN) mobile application on humidity and temperature environmental monitor using CC2650 sensor tag," IOP Conference Series: Materials Science and Engineering, vol. 1073, no. 1, p. 012046, Feb. 2021, doi: 10.1088/1757-899x/1073/1/012046.
- [7] Rafi Hanif Rahmadhani, Imam Fahrur Rozi, and Rosa Andrie Asmara, "Live K-Means Clustering Pada Wireless Sensor Network Menggunakan Google Maps API," in *Seminar Informatika Aplikatif Polinema (SIAP) –2021*, M. A. Hendrawan, H. E. Dien, S. E. Sukmana, and M. Z. Abdullah, Eds., Malang: Information Technology Department, State Polytechnic of Malang, Nov. 2021, pp. 46–52.
- [8] O.-A. Schipor and A. Aiordachioae, "Engineering Details of a Smartglasses Application for Users with Visual Impairments," 2020 International Conference on Development and Application Systems (DAS), May 2020, doi: 10.1109/das49615.2020.9108920.
- [9] B. Niu, Z. Gao, and B. Guo, "Facial Expression Recognition with LBP and ORB Features," Computational Intelligence and Neuroscience, vol. 2021, pp. 1–10, Jan. 2021, doi: 10.1155/2021/8828245.
- [10] S.-H. Sung et al., "A Study on Facial Expression Change Detection Using Machine Learning Methods with Feature Selection Technique," Mathematics, vol. 9, no. 17, p. 2062, Aug. 2021, doi:10.3390/math9172062.
- [11] I. M. Revina and W. R. S. Emmanuel, "A Survey on Human Face Expression Recognition Techniques," Journal of King Saud University - Computer and Information Sciences, vol. 33, no. 6, pp. 619–628, Jul. 2021, doi: 10.1016/j.jksuci.2018.09.002.
- [12] M. Mentari, R. Andrie Asmara, K. Arai, and H. Sakti Oktafiansyah, "Detecting Objects Using Haar Cascade for Human Counting Implemented in OpenMV," Register, vol. 9, no. 2, pp. 122–133, Sep. 2023, doi: 10.26594/register.v9i2.3175.
- [13] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017.
- [14] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," pp. 32–33, 2009, [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [15] M. I. U. Haque and D. Valles, "A Facial Expression Recognition Approach Using DCNN for Autistic Children to Identify Emotions," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Nov. 2018, doi:10.1109/iemcon.2018.8614802.
- [16] M. A. H. Akhand, S. Roy, N. Siddique, M. A. S. Kamal, and T. Shimamura, "Facial Emotion Recognition Using Transfer Learning in the Deep CNN," Electronics, vol. 10, no. 9, p. 1036, Apr. 2021, doi:10.3390/electronics10091036.
- [17] E. Pranav, S. Kamal, C. Satheesh Chandran, and M. H. Supriya, "Facial Emotion Recognition Using Deep Convolutional Neural Network," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Mar. 2020, doi:10.1109/icaccs48705.2020.9074302.
- [18] W. H. Abdulsalam, R. S. Alhamdani, and M. N. Abdullah, "Facial Emotion Recognition from Videos Using Deep Convolutional Neural Networks," International Journal of Machine Learning and Computing, vol. 9, no. 1, pp. 14–19, 2019, doi: 10.18178/ijmlc.2019.9.1.759.
- [19] P. Mohan, A. J. Paul, and A. Chirania, "A Tiny CNN Architecture for Medical Face Mask Detection for Resource-Constrained Endpoints," Innovations in Electrical and Electronic Engineering, pp. 657–670, 2021, doi: 10.1007/978-981-16-0749-3\_52.
- [20] S. Kakarla, P. Gangula, M. S. Rahul, C. S. C. Singh, and T. H. Sarma, "Smart Attendance Management System Based on Face Recognition Using CNN," 2020 IEEE-HYDCON, Sep. 2020, doi:10.1109/hydcon48903.2020.9242847.
- [21] S. S. Rajput and K. V. Arya, "CNN Classifier based Low-resolution Face Recognition Algorithm," 2020 International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), Jul. 2020, doi: 10.1109/icefeet49149.2020.9187001.
- [22] M. Afif, R. Ayachi, E. Pissaloux, Y. Said, and M. Atri, "Indoor objects detection and recognition for an ICT mobility assistance of visually impaired people," Multimedia Tools and Applications, vol. 79, no. 41–42, pp. 31645–31662, Aug. 2020, doi: 10.1007/s11042-020-09662-3.
- [23] M. H. Romario, E. Ihsanto, and T. M. Kadarina, "Sistem Hitung dan Klasifikasi Objek dengan Metode Convolutional Neural Network," Jurnal Teknologi Elektro, vol. 11, no. 2, p. 108, Jun. 2020, doi:10.22441/jte.2020.v11i2.007.
- [24] S. Aich and I. Stavness, "Global Sum Pooling: A Generalization Trick for Object Counting with Small Datasets of Large Images," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2019-June, pp. 73–82, May 2018, Accessed: May 18, 2023. [Online]. Available: <https://arxiv.org/abs/1805.11123v2>
- [25] X. Tan, C. Tao, T. Ren, J. Tang, and G. Wu, "Crowd Counting via Multi-layer Regression," Proceedings of the 27th ACM International Conference on Multimedia, Oct. 2019, doi: 10.1145/3343031.3350914.
- [26] X. Chen, Y. Bin, N. Sang, and C. Gao, "Scale Pyramid Network for Crowd Counting," 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Jan. 2019, doi: 10.1109/wacv.2019.00211.
- [27] W. Kong, H. Li, X. Zhang, and G. Zhao, "A multi-context representation approach with multi-task learning for object counting," Knowledge-Based Systems, vol. 197, p. 105927, Jun. 2020, doi:10.1016/j.knosys.2020.105927.
- [28] K. Dijkstra, J. van de Loosdrecht, W. A. Atsma, L. R. B. Schomaker, and M. A. Wiering, "CentroidNetV2: A hybrid deep neural network for small-object segmentation and counting," Neurocomputing, vol. 423, pp. 490–505, Jan. 2021, doi: 10.1016/j.neucom.2020.10.075.
- [29] H. Li, W. Kong, and S. Zhang, "Deeply scale aggregation network for object counting," Knowledge-Based Systems, vol. 210, p. 106485, Dec. 2020, doi: 10.1016/j.knosys.2020.106485.
- [30] K. Swaraja et al., "CNN Based Monocular Depth Estimation," E3S Web of Conferences, vol. 309, p. 01070, 2021, doi:10.1051/e3sconf/202130901070.
- [31] S. J. Lee, H. Choi, and S. S. Hwang, "Real-time Depth Estimation Using Recurrent CNN with Sparse Depth Cues for SLAM System," International Journal of Control, Automation and Systems, vol. 18, no. 1, pp. 206–216, Sep. 2019, doi: 10.1007/s12555-019-0350-8.
- [32] Y. Sada, N. Soga, M. Shimoda, A. Jinguji, S. Sato, and H. Nakahara, "Fast Monocular Depth Estimation on an FPGA," 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), May 2020, doi:10.1109/ipdpsw50202.2020.00032.
- [33] I. A. Kaust and P. Wonka, "High Quality Monocular Depth Estimation via Transfer Learning," Dec. 2018, Accessed: May 18, 2023. [Online]. Available: <https://arxiv.org/abs/1812.11941v2>
- [34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, doi: 10.1109/cvpr.2018.00474.
- [35] P. Kumar and A. Shankar Hati, "Convolutional neural network with batch normalisation for fault detection in squirrel cage induction motor," IET Electric Power Applications, vol. 15, no. 1, pp. 39–50, Dec. 2020, doi: 10.1049/elp2.12005.



- [36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, doi: 10.1109/cvpr.2017.243.
- [37] I. Kouretas and V. Paliouras, "Simplified Hardware Implementation of the Softmax Activation Function," 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), May 2019, doi: 10.1109/mocast.2019.8741677.
- [38] Q. Zhang, Y. Liu, C. Gong, Y. Chen, and H. Yu, "Applications of Deep Learning for Dense Scenes Analysis in Agriculture: A Review," Sensors, vol. 20, no. 5, p. 1520, Mar. 2020, doi: 10.3390/s20051520.