EAI Endorsed Transactions on
Industrial Networks and Intelligent Systems
Research Article

Resource-Efficient Deep Learning: Fast Hand Gestures on Microcontrollers

Tuan Kiet Tran Mach[1], Khai Nguyen Van[1], Minhhuy Le[1,2,*]

[1]Intelligent Communication System Laboratory, Phenikaa University, Hanoi 12116, Vietnam
[2]Falcuty of Electrical and Electronic Engineering, Phenikaa University, Hanoi 12116, Vietnam

Abstract

Hand gesture recognition using a camera provides an intuitive and promising means of human-computer interaction and allows operators to execute commands and control machines with simple gestures. Research in hand gesture recognition-based control systems has garnered significant attention, yet the deploying of microcontrollers into this domain remains relatively insignificant. In this study, we propose a novel approach utilizing micro-hand gesture recognition built on micro-bottleneck Residual and micro-bottleneck Conv blocks. Our proposed model, comprises only 42K parameters, is optimized for size to facilitate seamless operation on resource-constrained hardware. Benchmarking conducted on STM32 microcontrollers showcases remarkable efficiency, with the model achieving an average prediction time of just 269ms, marking a 7× faster over the state-of-art model. Notably, despite its compact size and enhanced speed, our model maintains competitive performance result, achieving an accuracy of 99.6% on the ASL dataset and 92% on OUHANDS dataset. These findings underscore the potential for deploying advanced control methods on compact, cost-effective devices, presenting promising avenues for future research and industrial applications.

Received on 2 April 2024; accepted on 2 July 2024; published on 3 July 2024

Keywords: Intelligent system, Hand gesture recognition, Embedded, Communications, Human-Computer Interaction, Microcontroller, TinyML.

doi:10.4108/eetinis.v11i3.5616

1. Introduction

With the growing ubiquity of computers in everyday life, bridging the interaction gap between humans and machines has become a critical research area. While conventional input methods like keyboards and mice persist, they fail to represent the most natural form of human-machine interaction. As technologies such as virtual reality, remote control, and augmented reality gain traction, traditional input devices often prove inadaptable. Hand gesture recognition (HGR) emerges as a promising alternative, offering a more intuitive and natural means of interaction. HGR systems interpret and classify separately hand poses and motions in real-time, facilitating remote interaction with digital interfaces and environments [1]. These systems find applications across various domains, from gaming to touchless control systems and accessibility solutions like sign language translation. HGR holds promises to enhance communication for the hearing-impaired, aid in medical tasks, and provide faster, more convenient computer access for individuals with disabilities, the elderly, and children. In an era where devices are increasingly integrated into daily life, the ability to interact effortlessly and intuitively with technology is paramount, with hand gesture recognition leading this revolution by redefining human-computer interaction.

HGR is categorized into two common approaches including sensor-based and vision-based [2]. Sensor-based HGR relies on sensors attached directly to the user's hand or arm to capture various data types, such as shape, position, movement, or trajectory. Although sensor-based excels at collecting accurate data, it may be less practical for everyday use due to its reliance on specialized equipment. In contrast, vision-based HGR utilizes one or more cameras to capture hand motions and appearances. This approach offers greater

*Corresponding author. Email: huy.leminh@phenikaa-uni.edu.vn

flexibility and ease of setup compared to sensor-based systems. Vision-based systems can interpret hand gestures without requiring users to wear specialized equipment, allowing for more natural and intuitive interactions. However, vision-based HGR may face challenges in handling diverse lighting conditions, complex backgrounds, and occlusions, which can affect gesture recognition accuracy [3]. Additionally, vision-based systems may require more computational resources for real-time processing, particularly when dealing with complex hand poses and data type.

Due to the field's lengthy development history, there are several related work in the field of vision-based HGR achieved satisfactory results. The earliest vision-based techniques took advantage of the characteristics of images to apply the most efficient processing methods. For instance, color information is crucial for identifying specific hand gestures against varying backgrounds, while texture analysis aids in discerning subtle differences in hand configurations. The primary application of color detection involves identifying the skin tone present on the hand. Method [4] process a low-cost web-cam image with a fusion of four stage. First, Jayashree et al. used gray threshold method combined with median filter and Gaussian filter to filter noise and convert RGB image to denoised binary image. Then "Sobel" edge detection is applied for extracting region of interest. Finally, they used feature matching method, specifically Euclidean distance calculation, to compare the feature vectors of centroid and area of edge between test set and the training set. The proposed method was evaluated in American Sign Language (ASL) alphabet with 26 static hand gesture related to A-Z and achieved a positive result of 90.19%. However, following the success of state-of-the-art deep learning models in image-related tasks, the field of image processing has increasingly adopted advantages from deep learning [5, 6, 7, 8, 9]. Rather than completely eliminate traditional vision techniques, a hybrid approaches using a Dual–Channel Convolutional Neural Network (DC-CNN), fusion the hand gesture images and hand edge images after preprocessing using Canny edge detection [10]. The output can be classified using the SoftMax classifier, and each of the two-channel CNNs has a unique weight. The proposed system's recognition rate is 98.02%. However, the performance of these methods is limited by how well the handcrafted extractor selected features represent the characteristic of hand. In contrast, end-to-end deep learning models have the ability to automatically learn hierarchical features directly from raw data. Hussain et al. [11] fine-tuned two state-of-the-art CNN architectures - Inception V3 and Efficient B0 - that have achieved noticeable performance on various image-related tasks. Both models were trained with the same input, namely the RGB images of recorded hand gestures. These models were evaluated on the ASL dataset yielding accuracy of 90% and 99%, respectively. Improvements in sensors technologies bring new approaches to leveraging depth image data captured by devices such as Kinect and Intel RealSense. The method [12] uses two VGG19 with identical architectures but distinct input types. Specifically, VGG19-v1 was fed the RGB images to extract colour-based features, while VGG19-v2 took the depth images as input to learn depth-based representations. By combining the two streams of information, the authors were able to achieve a classification accuracy as high as 94.8% on the ASL dataset. More advanced deep learning models aim to combine multi-scale or multi-level features to enhance network learnability. The ExtriDeNet method utilizes two modules - IFFB and IFAB - to extract image features at different scales [13]. The multi-scale features are then merged before being fed into a classifier. The model demonstrates outstanding performance with some highly complex datasets such as HGR-I with 93.56% or NUS-II with 98.75%. Recent works such as [14] and [15] have made changes to the backbone and neck components of the YOLO architecture to develop lightweight detection models. In particular, the approach described in [14], which using ShuffleNet V2 as the backbone in YOLOv3, has achieved impressive results on two challenging datasets with complex backgrounds such as senz3D dataset reaching 99.5% and Microsoft Kinect dataset reaching 99.6%. Significantly, the model size was only 8.9 MB compared to the 123.5 MB size of the original YOLOv3 network. Although many studies have been carried out, these studies focus heavily on improving model accuracy and pay little attention to computational costs. This poses challenges for executing the model on low-cost, constraint hardware such as microcontroller devices with limited memory capacity and computational speed, resulting in significant inference time delays.

The rapidly growing adoption of Internet of Things (IoT) devices powered by microcontrollers, with some reports indicating over 250 billion in 2022, opens up opportunities to deploy applications across many industries however faces challenges [16]. These low-cost, energy-efficient microcontrollers facilitate the domain of Tiny Machine Learning (Tiny-ML) involving direct deployment of deep learning models at the sensor to substantially expand AI applications through localized intelligent tasks. While this approaches provides an avenue to deploy hand gesture recognition systems with improved flexibility and broader application scope by leveraging resource-constrained devices to enable optimizations in energy consumption, costs and operations, integrating deep learning poses difficulties due to microcontrollers' limited memory restricting model sizes, processing power

impacting speed/efficiency, and battery life requiring energy-efficient algorithms [17]. Overcoming hurdles is critical to realize microcontroller-based deep learning's full potential, including for feasible HGR in highly resource-limited environments through continued Tiny-ML technique research. Deploying the HGR system on microcontrollers opens up potential avenues in the field of touchless control in factories with low investment and usage costs.

The main purpose of this research is to develop a compact HGR system tailored for microcontrollers with constrained resources. Considering the above limitations, our focus lies in proposing a lightweight CNN architecture that satisfies the requirements of model size, inference time, and computational cost. Additionally, we explored optimization techniques aimed at compressing model size before implementation on these microcontrollers. This outlines the entire major contributions within this research. The architecture is based on MobileNetV2 [18] and MobileNetV3 [19] with 2D depthwise separable convolution (DSC), bottleneck and Squeeze-and-Excitation block [20]. Our proposal is used as pixel-based feature extractor, which extracts spatial features in the image. After obtaining high-level features using our architecture as backbone, we integrate a classifier or detector as top module depending on intended use. The proposed method exhibits the competitive result with state-of-art model when evaluate in two different datasets American Sign Language (ASL) [21] and OUHANDS [22] datasets. Emphasizing the achievement of accurate results from a single input frame, our model significantly enhances inference time efficiency. The qualified model is benchmarked in STM32 platform.

The following sections of this document are presented in order: Section 2 will detailly describe our proposed method, encompassing the processes of data preprocessing, augmentation, and our proposed model architecture. In Section 3, a comprehensive evaluation of the results, including in-depth analysis and comparative against preceding methods, will be provided. This section will also describe the benchmarking of our model's performance on a variety of microcontrollers from the STM32 family. The conclusions will be summarized and presented in the final Section 4.

2. Proposed method

Figure 1 illustrates the block diagram of the process involved in building proposed method. Before performing model training, data augmentation techniques is used for enhancing training data diversity and then normalize the entire dataset to make sure the pixel values of images are within a consistent range. The model was trained using the classification module with the ASL dataset due to its substantial volume, facilitating enhanced feature learning of hand gestures. Additionally, we employ transfer learning techniques by leveraging obtained pre-trained model with the OUHANDS dataset, utilizing detector module as top layers. This approach enables the model to benefit from prior learning and adapt more effectively to the nuances of hand gesture recognition tasks. After training the model, the quantization algorithm is employed using the TFLite framework. This algorithm facilitates the reduction of the float-point TensorFlow model to 8-bit precision, making it compatible with embedded hardware that exclusively supports 8-bit computations. Final evaluation will be done on some STM32 microcontrollers and deploy to OpenMV H7 to investigate inference time.

2.1. Data preparation

Data augmentation. Data augmentation is a critical technique for training deep learning models, helping to address limited datasets and improve generalization by artificially expanding the data through transformations such as rotation, flipping, adding noise, and more. In hand gesture recognition, the accurate identification of distinct gestures heavily depends on hand shape and structure, requiring models to learn robust representations under diverse real-world capturing conditions involving various angles, positions, and environmental factors. Therefore, geometric transformations like translation, rotation, and random zoom are essential to simulate these capture variations within the dataset and expose the model to a broader range of inputs. A random brightness adjustment from -0.1 to 0.1 is also implemented to reduce sensitivity to lighting changes during usage. Through intelligently applying such augmentations involving affine and color transformations, the dataset is expanded in a way that enhances the model's stability to factors like perspective and lighting, contributing to its overall robustness and generalized capabilities critical for accurate hand gesture identification. Because the purpose is to help the model learn the necessary features, the augmentation technique is only applied to the training data set and not to the testing data. Examples of some images after the augmentation process are shown in Figure 2.

Normalization. Normalizing an image refers to the process of adjusting the pixel values to conform to a standardized scale or distribution. This process aids in improving convergence during optimization, as it minimizes issues related to features at different scales, thereby facilitating smoother convergence and preventing oscillation. Furthermore, normalization acts as a form of regularization by preventing models from being outrageously sensitive to certain features, thus aiding in preventing overfitting and enhancing generalization performance. Given an image I represented as a matrix of pixel values, where

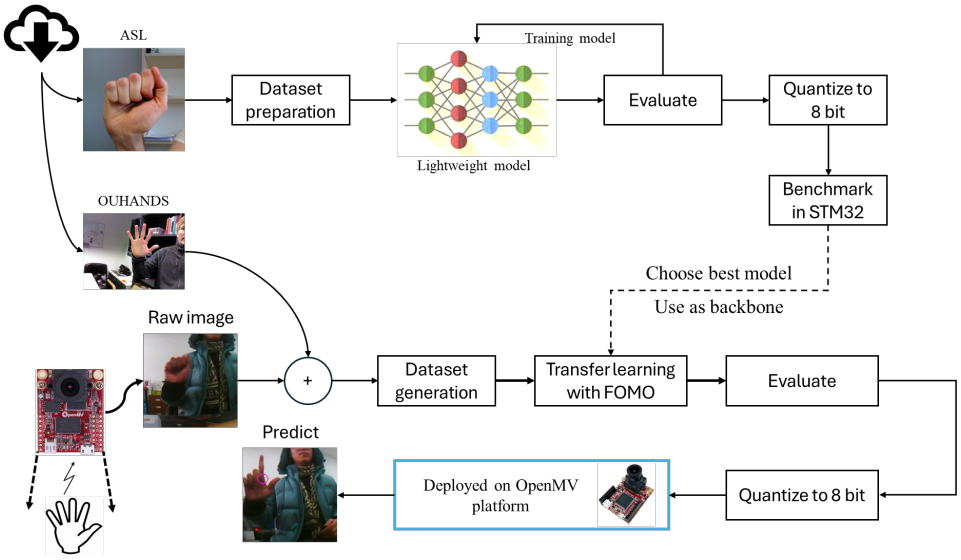

Figure 1. Block diagram of proposed method for hand gesture recognition.

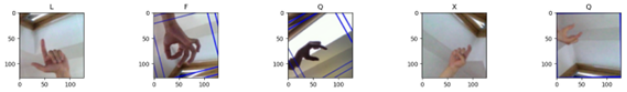

Figure 2. Some samples after data augmentation with Random Zoom, Random Rotation, Random Brightness.

I_{ij} denotes the pixel value at row i and column j, the process of normalizing using Min-Max scaling to rescale the pixel value to range [0, 1] as Equation 1 below:

$$I'_{ij} = \frac{I_{ij} - min(I)}{max(I) - min(I)} \tag{1}$$

2.2. Proposed model

We employed an end-to-end approach using deep learning model to automatically learn the representation of feature from raw data, delivering a complete solution without needing anything from conventional vision-based techniques. The proposed model is constructed based on the MobileNetV3 [29] architecture with some modifications in construction. In detail, we reduced the number of layers, adjusted the bottleneck block and classifier. This approach is aimed at refining the data and effectively decreasing the number of parameters. Unlike previous research that focus heavily on model performance, our method focuses on the trade-off between accuracy and model size. The proposed method helps optimize performance on computational cost to provide higher computational efficiency. As illustrated in Figure 3 and detailed in Table 1, the overall architecture contains a feature extractor as backbone followed by a classifier as head. The backbone first applies a standard CNN block for initial feature learning from the input through a series of operations: 2D CNN with 16 filters using kernel size of 3 × 3, batch normalization for gradient stabilization, hard swish as activation, max pooling for down-sampling, and dropout layer for regularization. In summary, the function $H_1(\bullet)$ transforms the multi-channels input $X_i \in \mathbb{R}^{H\times W\times C}$ into 16 feature maps $z_1 \in \mathbb{R}^{H'\times W'}$, presented as Equation 2 below:

$$\mathbf{z_1} = \boldsymbol{H_1}(\bullet) = \text{ Dropout } (MP^2(\text{ h}(BN(C^3(X_i))))) \tag{2}$$

where $C^3(\bullet)$ is the 2D standard convolution with kernel size of 3, $BN(\bullet)$ represents the batch normalization operation, $MP^2(\bullet)$ denotes the max pooling with stride of 2, h is hard swish activation function.

Following by 3 micro-bottleneck blocks that serve to extract higher-level features, micro-bottleneck is crucial ingredient that helps the model learn features that accurately and succinctly describe the data. Each of them is composed of one SE Conv Block and two SE Residual Block, which includes pointwise 2D, depthwise 2D, batch norm, hard-swish and notably, squeeze and excitation block as described in Figure 4. Incorporating the Squeeze & Excitation block enriches the data representation capabilities of these sub-blocks by dynamically recalibrating features based on channel-wise information. To mitigate output complexity and enhance cost-effectiveness, a dropout layer with a 0.1 rate is appended at the conclusion of each block before transitioning to the subsequent stage. Micro-bottleneck blocks employ a sequence of convolutional layers with 8, 16, and 32 filters, respectively, maintaining a consistent kernel size of 3×3 across all blocks. We analyze the impact of expansion factor (t) on performance and generalization, testing values of 0.25

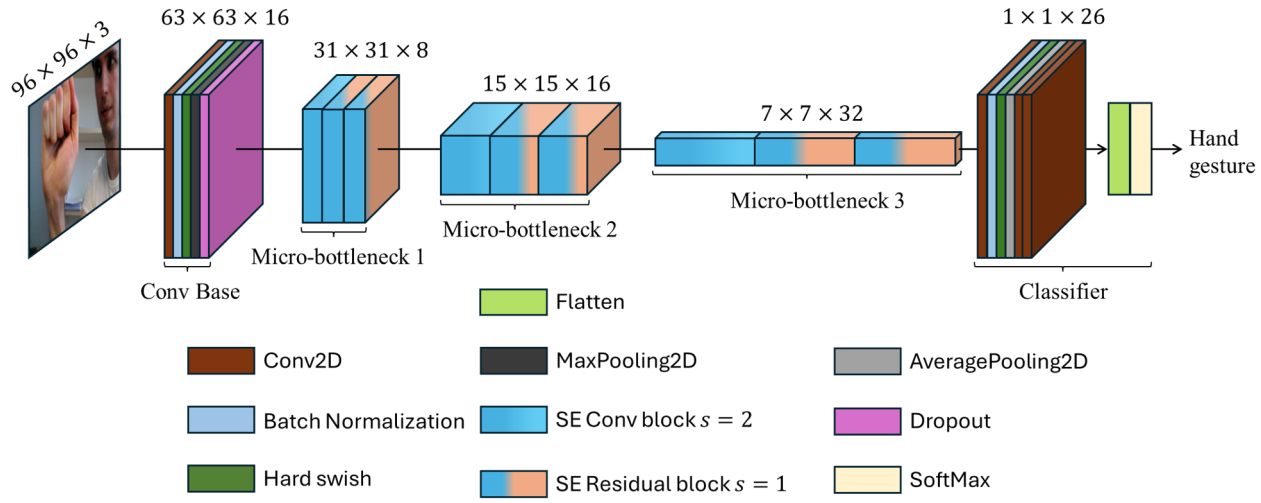

Figure 3. Proposed model architecture as base model for hand gesture recognition. I use the first four stages as feature extractor and final stage as lightweight classifier.

Table 1. Model configuration

Block/Stages	Number of parameters	Sub-block/Layers	Configurations
First stage	512	Conv2D	$F = 16, k = 3 \times 3, s = 1$
		Batch Norm	-
		Hard swish	-
		Max Pooling 2D	$s = 1$
		Dropout	$r = 0.1$
Micro-bottleneck 1	2130	SE Conv Block	$F = 8, k = 3 \times 3, s = 2, ratio = 3$
		SE Residual Block	$F = 8, k = 3 \times 3, s = 1, ratio = 1$
		SE Residual Block	$F = 8, k = 3 \times 3, s = 1, ratio = 1$
Micro-bottleneck 2	5732	SE Conv Block	$F = 16, k = 3 \times 3, s = 2, ratio = 3$
		SE Residual Block	$F = 16, k = 3 \times 3, s = 1, ratio = 1$
		SE Residual Block	$F = 16, k = 3 \times 3, s = 1, ratio = 1$
Micro-bottleneck 3	19656	SE Conv Block	$F = 32, k = 3 \times 3, s = 2, ratio = 3$
		SE Residual Block	$F = 32, k = 3 \times 3, s = 1, ratio = 1$
		SE Residual Block	$F = 32, k = 3 \times 3, s = 1, ratio = 1$
Lightweight classifier	14042	Conv2D	$F = 64, k = 1 \times 1, s = 1$
		Batch Norm	-
		Hard swish	-
		Average Pooling 2D	$s = 7$
		Conv2D	$F = 128, k = 1 \times 1, s = 1$
		ReLU	-
		Conv2D	$F = 26, k = 1 \times 1, s = 1$
Flatten	0	-	-
Output	0	SoftMax	$Classes = 26$
Total	42072		

and 3.0. This assesses behavior under a traditional residual (t < 1) or inverted residual (t > 1) structure.

SE Conv Block is delineated as a series of consecutive operations, as illustrated in Figure 4, and outlined by Equation 3:

$$\begin{aligned} z_i &= H_F^{'i}(\bullet) \\ &= Dropout(BN(P^1(SE(\,\mathrm{h}(BN(P^1(z_{i-1}))))))) \end{aligned} \tag{3}$$

where $z_i \in \mathbb{R}^{H' \times W' \times F'}$ is the feature map at i_block after input feature map $z_i \in \mathbb{R}^{H \times W \times F}$ being processed by $H_{F'}^{i}(\bullet)$ operation, let $P_c^1(\bullet)$, $BN_c(\bullet)$, $D_c^3(\bullet)$ denotes the 2D pointwise convolution, the batch normalization, and 2D depthwise convolution respectively applied to the c-th feature map of input z_{i-1} from previous stage. Within input X obtained from previous layer, each operation

can be computed as below Equations 4 to 6:

$$P^1_{c,l}(\boldsymbol{x}) = \sum_f \{X\}_f \cdot \{\boldsymbol{K}_l\}_f + B_l \tag{4}$$

$$BN_c(\boldsymbol{x}) = \gamma \odot \frac{\boldsymbol{x} - \mu_B}{\hat{\sigma}_B} + \boldsymbol{\beta} \tag{5}$$

$$\left\{D^3_{c,l}(X)\right\}_i = \sum_u \{X\}_u \circ \{\boldsymbol{K}_l\}_{i-u} + \boldsymbol{B}_l \tag{6}$$

where $X_{H\times W\times f}$ represents the $H \times W$ matrix with f-th feature map; $\mathbf{K}_l$ and $\mathbf{B}_l$ denote the 2D convolution kernel weights of the l-th layer transform f-th input feature maps to c-th output feature, and $\{\bullet\}_i$ is the i-th element of a set of channels. Further, μ_B, $\hat{\sigma}_B$, γ and β are sample mean, standard deviation, scale parameter and shift parameter respectively used in batch normalization.

SE Residual Block functions similarly to the SE Conv Block. However, in addition to its primary operations, the SE Residual Block incorporates an extra shortcut employing Conv2D with a kernel size of 1×1. This additional step enhances information retention in the output following the model's extraction process, thereby mitigating the risk of excessive information loss as the model depth increases. Furthermore, it serves to prevent gradient explosion during backpropagation. The main operation of a SE residual block is a series of layers described as Equation 7:

$$\begin{aligned} z_i &= H^i_{F'}(\bullet) \\ &= Dropout(P^1(SE(\,\mathrm{h}(BN(D^3(\,\mathrm{h}(P^1(z_{i-1})))))) \\ &\qquad + C^1(z_{i-1})) \end{aligned} \tag{7}$$

where C^1 is convolution 2D with kernel size 1×1 in shortcut.

Adjusting the top layer of a neural network model to include Conv2D and Pooling layers is a strategic move, especially when working with image data. Incorporating Conv2D 1×1 layers followed by pooling layers can significantly reduce the number of parameters in a neural network compared to using Dense layers. This is because Conv2D 1×1, also known as pointwise convolution, acts as a dimensionality reduction technique, allowing for the mixing of channel information without affecting the spatial dimensions. When combined with pooling layers, which further condense the data by spatially downsizing the output from the convolutional layers, the model becomes more computationally efficient. Dense layers, while powerful, can lead to a vast number of parameters, especially in deep networks, because each neuron is connected to all neurons in the previous layer. This not only increases the computational load but also the risk of overfitting. By using Conv2D 1×1 and pooling layers, the network maintains its ability to learn complex features with fewer parameters, leading to faster training times and potentially better generalization on unseen data. The operation of the lightweight classifier is serialized as Equation 8:

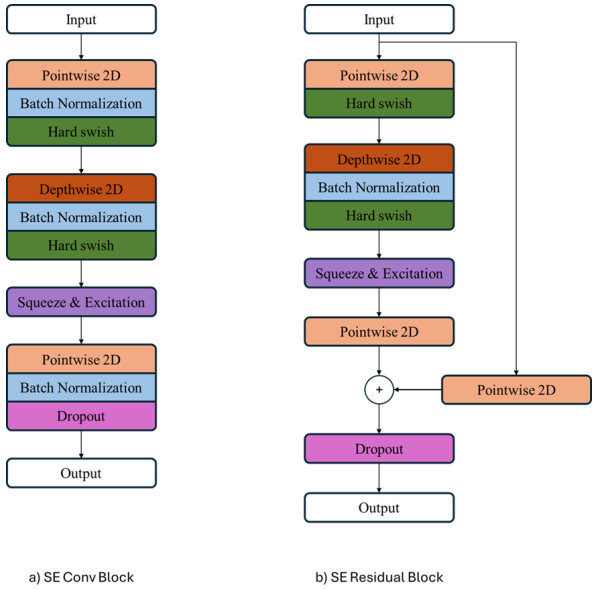

Figure 4. Detailed structure of each SE block used in micro-bottleneck. Block a) is SE Conv Block to help reduce spatial dimension before entering block b) SE Residual Block to better extract features.

$$\begin{aligned} \hat{y} &= H(\bullet) \\ &= \alpha(Flatten(C^1(ReLU(C^1(AP(\,\mathrm{h}(BN(C^1(z_{i-1}))))))))) \end{aligned} \tag{8}$$

where AP denotes Average Pooling with stride of 7 and $\alpha(\bullet)$ is SoftMax activation function.

2.3. Training process

As mentioned above, the training process is divided into two parts. Because each hand gesture dataset usually does not have many samples, we use the ASL set for pre-training because this data set has nearly 100,000 samples. The obtained model was benchmarked on STM32 before being transferred learning with the FOMO technique on the OUHANDS and the self-collected data set.

We use Adam optimizer to minimize categorical cross-entropy loss function for multi-class classification task. The loss quantifies the dissimilarity between the predicted probabilities and the true categorical labels as represented in Equation 9:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} y_{ij} log(p_{ij}) \tag{9}$$

The lower the loss, the more accurate the model. By minimizing the loss, the model learns to assign higher probabilities to correct class, improving accuracy. We initialize learning rate at 0.001 and use learning rate decay strategy if the validation loss does not decrease. The model is trained on an NVIDIA RTX A5000 High-Performance Computer (HPC) for 100 epochs with a batch size of 64 samples.

After training with ASL, we transfer learning the obtained model on OUHANDS dataset. The model was made some modification to be able to integrate the FOMO technique. The model was cut off the last three layer at Average Pooling 2D and replaced by a per-region class probability map. This map, called heat map, is a sized-down version of the input. We use a custom loss function with weight to quantify the class probabilities for each unit in heat map with corresponding patch in the input image. The true label need to be created by segment a input image with bounding box. Every outside the box is background and inside the box assigned to the corresponding label.

3. Experimental result & benchmark

3.1. Experimental Setup & Dataset

Hand gesture recognition has been an active research area for long time, resulting in the availability of numerous open-source datasets that can be leveraged. For training and evaluation, we utilize existing datasets commonly used in previous work. However, newer datasets captured under diverse real-world conditions with indoor and outdoor involving multiple participants, while providing a more realistic simulation, tend to be smaller in size. This poses a challenge as limited data makes it difficult to train models from scratch to achieve optimal performance.

American Sign Language (ASL) dataset [21] is selected for pre-training and the OUHANDS [22] for transfer learning. With ASL, the dataset includes 87,000 images of American Sign Language alphabets, each 200x200 pixels, across 26 classes including letters A-Z. Each image in the dataset is a hand captured in different experimental environments and under different lighting conditions. However, this dataset does not appear people, only hands inside the image. For the OUHANDS dataset, this data is more diverse and better simulates real-life scenario. The data was recorded in many different environments by many volunteers and there were people in the images as a distracting agent. However, this data set only has 3000 images for 10 gestures including A, B, C, D, E, F, H, I, J, K. Figure 5 illustrates some sample from two dataset.

In addition, we also collect a data set in our environment with classes corresponding to OUHANDS. This data set will be mixed with the OUHANDS set to increase the number of samples as well as

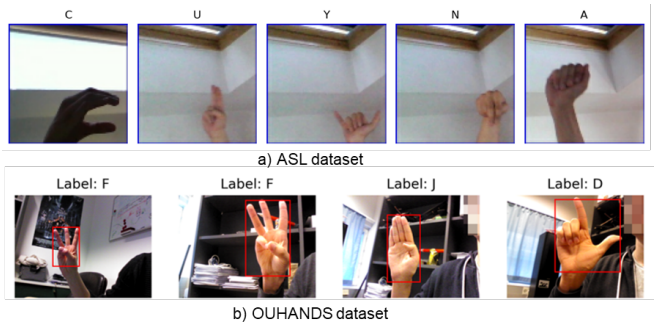

Figure 5. Some samples from two dataset. a) from ASL and b) from OUHANDS dataset

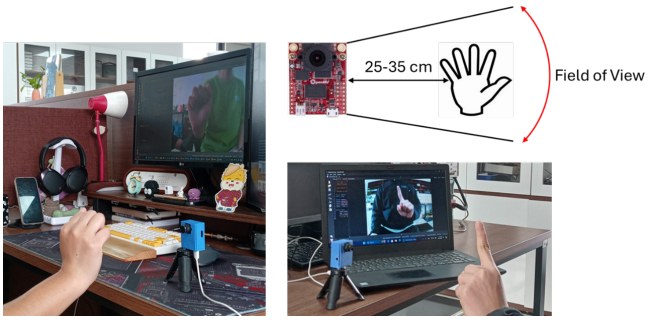

Figure 6. Experimental setup use the same hand gesture as OUHANDS

help the model generalize better. Our dataset includes 300 images for 10 classes, each class will contain 30 images, corresponding to the number of experimental repetitions of a participant in the OUHANDS dataset. Figure 6 describes our experimental setup. Each recording will record a 30-second video, the hand will maintain the same gesture and move to different angles of the camera to capture many different perspectives.

3.2. Experimental Results

We investigate the behavior of the bottleneck blocks by adjust expansion factor (t) in two different directions: traditional residual block and inverted residual block. Observing the results in Figure 7, a comparison between the models employing t values of 0.25 and 3.0 reveals high accuracies of 96.7% and 99.6%, respectively. Although the model using t = 0.25 is 3% lower in accuracy, it is also 1.4 times smaller than the model using t = 3.0. However, when employing transfer learning with the OUHANDS dataset using the FOMO technique, significant differences appear, as depicted in Figure 8. The confusion matrix highlights pronounced disparities between the two models. While the t = 3.0 model achieves high accuracies exceeding 80% across all classes, the t = 0.25 model exhibits instability and variability among different classes. Notably, class Non

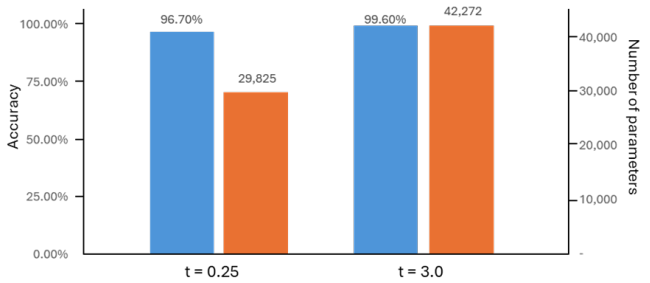

Figure 7. Accuracy and number of parameters when using two different expansion factor, t = 0.25 and t = 3.0

achieves a score exceeding 99%, contrasting sharply with class C, which registers only 43%.

This observation underscores a drawback of lightweight deep learning models based on the TinyML technique. Because it must be optimized for specific datasets and task, these models often sacrifice generalizability. A comparison between the OUHANDS and ASL datasets further elucidates this point. While both datasets relate to HGR, OUHANDS presents distinct challenges, particularly in background differentiation, absent in the ASL dataset. Consequently, despite optimization for ASL, the model utilizes an expansion factor of 0.25, yet still struggles to perform effectively on OUHANDS.

3.3. Ablation Studies

This section aims to showcase the effectiveness of the proposed architecture, characterized by a fusion of SE blocks with a lightweight classifier (LC), through an ablation study. To achieve this, we conduct a comparison among various CNN structures, each constructed by adjusting specific segments of the proposed model. The first modification involves removing the SE block from both the SE Conv and SE Residual blocks while maintaining identical hyperparameters. This allows us to assess the impact of SE blocks on model performance. The second modification entails replacing the lightweight classifier layer with a Flatten layer followed by Fully Connected layers. This alteration enables us to evaluate the model's effectiveness in terms of computational costs. Looking at table 2, removing the SE block reduces model size, but it also reduces the model's performance on the OUHANDS data set as well as the ASL data set. The model only achieved 85.5% when evaluated on OUHANDS and approximately 95% when evaluated on the ASL set. If we eliminate the LC and replace it with a classifier block including Flatten and Dense, the model increases the number of parameters to 99.7K but the model's performance also drops to 96.12% on ASL. This proves that our proposed model is superior when using the proposed modules.

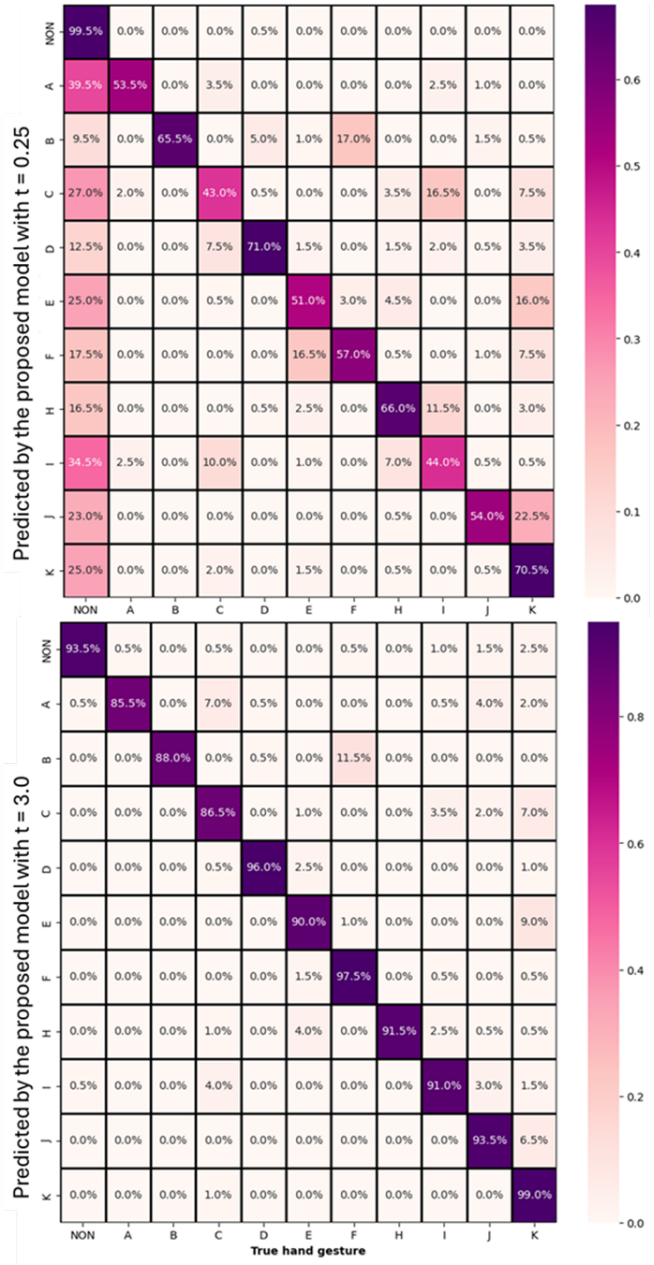

Figure 8. Confusion matrix of proposed method with a) t = 0.25 and b) t = 3.0 when deploying transfer learning in OUHANDS dataset

Table 2. Summary of ablation evaluation of the proposed model in comparing its accuracy and model size with other some modifications

Backbone	ASL	OUHANDS	# of params
Bottleneck without SE	95.23%	85.5%	33.4K
Bottleneck without LC	96.12%	-	99.7K
Bottleneck	99.60%	92.0%	42.1K

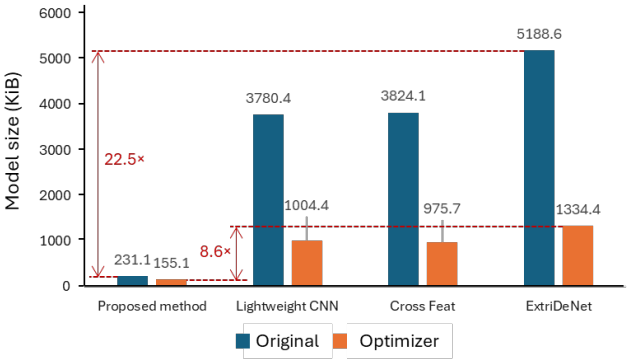

Figure 9. Comparing size (a) before and after implementing the quantization technique for each of the following models: proposed model with micro-bottleneck, Lightweight CNN [24], CrossFeat [23] and ExtriDeNet [13]

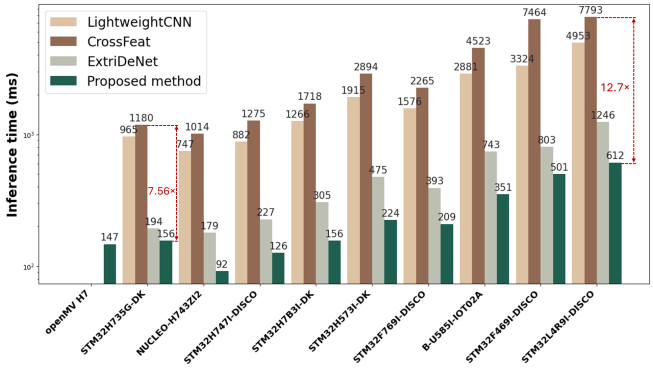

Figure 10. Benchmark inference time when deploying in OpenMV H7 and STM32 microcontrollers

3.4. Benchmark in microcontrollers

To prepare the models for deployment on microcontrollers, we utilize the post-training integer quantization method to convert the existing floating-point model to 8-bit precision. This process is achieved using the TFLite framework's tools, allowing us to logically transform the model without the need for manual retraining, while still preserving its accuracy. The quantization leads to a significant reduction in model size, up to 4 times smaller, enhancing both size and inference speed on the CPU. Looking at Figure 9, it can be seen that our model is much smaller than previous models. With only 155 KiB after optimization, the proposed model is 6× smaller than the Cross Feat [23] model with 975.5 KiB, the smallest of the previous models. Meanwhile comparing with the state-of-art ExtriDeNet, our method is 8.6× smaller. Even so, the model's accuracy as mentioned above is still extremely competitive. The other models on the table are all larger than 1000 KiB. This can be a big barrier to being able to deploy the model on microcontrollers. When considering inference time, our model also has the fastest with average inference time about 269 ms, twice as fast as extriDeNet [13] (about 500ms), and 10× faster than other models (CrossFeat [23] with 3281 ms and Lightweight CNN [24] with 2025 ms). The model achieves fastest prediction in NUCLEO-H743ZI2 series with only 92 ms, about 11× faster than slowest model in benchmarking. Looking at Figure 10, we also deploy proposed model with FOMO in OpenMV H7 to check the feasibility and obtain the inference time at 147 ms.

3.5. Comparision with previous method

The proposed model delivers superior performance, achieving accuracy of over 99% with ASL dataset. Some other methods conducted previously also produced similar results on the ASL dataset. Methods [25] using convolutional neural networks with spatial pyramid pooling, this structure enables neural networks to accept input images of arbitrary sizes and aspect ratios, overcoming the need for resizing or cropping images to fit a fixed input size. Besides, SPP typically utilizes multiple levels of pooling, with each level representing a different grid size. This enables the network to capture both fine-grained details and global context information from the input image. The results obtained from this model have an accuracy of up to 99.99%. However, this model has a number of parameters up to 6.3 million parameters and is completely incapable of embedding in microcontrollers. With two other models that are built lighter, method [24] built light bottleneck blocks in succession gives a result of 98.72% accuracy. However, with a size of up to 848K parameters, this model is still 20× larger than the proposed model. CrossFeat [23] uses convolutional with different kernel sizes to extract multi-scale data as well as remember low-level features so you can add spatial information to the layers behind. Thanks to that, the model achieves an accuracy of up to 99.5% with 975K parameters. Because it must run many kernel sizes of different sizes in parallel to extract model data, this model is not optimized for RAM-limited hardware. For the OUHANDS data set, there is a model like ExtriDeNet [13] that also uses a multi-scale feature extractor to extract data, but this model's evaluation on the OUHANDS set is only 65.1% even though it has up to 1.3 million parameters. The method shows obvious ineffectiveness when compared to our method when the proposed model achieves an accuracy up to 92% but only needs 42.1K parameters. A more optimized approach produces particularly good results up to 98.75% using two architectures: Multi-scale structure and lightweight attention to enhance the power of the model[26]. This model is also lighter than ExtriDeNet with only 666.7K parameters but still 15× larger than the proposed model. When comparing with us, our proposed model offers an acceptable trade-off

Table 3. Summarize the result of the proposed model and previous model in HGR System

Method	Dataset	Accuracy	Benchmark
[25]	ASL	99.99%	-
[24]	ASL	98.72%	1018 KiB Flash 271.97 KiB RAM $\overline{t_{inference}}$ = 2025 ms
[23]	ASL	99.5%	956.38 KiB Flash 510.69 KiB RAM $\overline{t_{inference}}$ = 3282 ms
[13]	OUHANDS	65.1%	1.3 MiB KiB Flash 269 KiB RAM $\overline{t_{inference}}$ = 500.4 ms
[26]	OUHANDS	98.75%	-
Our method	ASL/ OUHANDS	99.7%/ 92%	140 KiB Flash 290.13 KiB RAM $\overline{t_{inference}}$ = 269 ms

between accuracy and model size, making it suitable for deployment on microcontrollers with various hardware constraints. The summary of comparison is described in Table 3.

4. Conclusion

Hand Gesture Recognition (HGR) has emerged as a promising avenue poised to revolutionize human-computer interaction, particularly with the proliferation of microcontroller devices playing an increasingly pivotal role in our digital landscape. Motivated by these trends, We proposed a lightweight micro-bottleneck model tailored for HGR systems on microcontrollers. Our model demonstrates competitiveness with state-of-the-art methods in terms of accuracy, achieving 99.6% on the American Sign Language (ASL) dataset and 92% on the OUHANDS dataset. Notably, despite its high accuracy, the model maintains a significantly lighter size, comprising just over 42,000 parameters. Specifically, it is 8.6× smaller than ExtriDeNet [13] and 10× faster than CrossFeat [23] in inference time. This opens up many possibilities for integrating newer and more natural control systems into existing systems in the easiest and most cost-effective way.

Acknowledgement

The calculations in this work are done at Phenikaa University's HPC Systems.

References

[1] Debajit Sarma and Manas Kamal Bhuyan. "Methods, databases and recent advancement of vision-based hand gesture recognition for hci systems: A review". In: *SN Computer Science* 2.6 (2021), p. 436.

[2] Mohammad Alsaffar et al. "Human-computer interaction using manual hand gestures in real time". In: *Computational Intelligence and Neuroscience* 2021 (2021).

[3] Hermann Baumgartl et al. "Vision-based hand gesture recognition for human-computer interaction using MobileNetV2". In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE. 2021, pp. 1667–1674.

[4] Jayashree R. Pansare, S. H. Gawande, and Maya Ingle. "Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background". In: *Journal of Signal and Information Processing* 2012 (2012), pp. 364–367. URL: https://api.semanticscholar.org/CorpusID:28173070.

[5] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[6] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[7] Andrew G. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *CoRR* abs/1704.04861 (2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.

[8] Xiangyu Zhang et al. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *CoRR* abs/1707.01083 (2017). arXiv: 1707.01083. URL: http://arxiv.org/abs/1707.01083.

[9] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[10] Xiao Yan Wu. "A hand gesture recognition algorithm based on DC-CNN". In: *Multimedia Tools and Applications* 79.13 (2020), pp. 9193–9205.

[11] Adnan Hussain et al. "An Efficient and Robust Hand Gesture Recognition System of Sign Language Employing Finetuned Inception-V3 and Efficientnet-B0 Network." In: *Computer Systems Science & Engineering* 46.3 (2023).

[12] Manju Khari et al. "Gesture Recognition of RGB and RGB-D Static Images Using Convolutional Neural Networks". In: *Int. J. Interact. Multim. Artif. Intell.* 5 (2019), pp. 22–27. URL: https://api.semanticscholar.org/CorpusID:208016465.

[13] Gopa Bhaumik et al. "ExtriDeNet: an intensive feature extrication deep network for hand gesture recognition". In: *The Visual Computer* 38.11 (2022), pp. 3853–3866.

[14] Shiniu Sun et al. "ShuffleNetv2-YOLOv3: a real-time recognition method of static sign language based on a lightweight network". In: *Signal, Image and Video Processing* 17.6 (2023), pp. 2721–2729.

[15] Weina Zhou and Xile Li. "PEA-YOLO: a lightweight network for static gesture recognition combining multiscale and attention mechanisms". In: *Signal, Image and Video Processing* 18.1 (2024), pp. 597–605. DOI: 10.1007/s11760-023-02755-0.

[16] Ji Lin et al. "MCUNet: Tiny Deep Learning on IoT Devices". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 11711–11722. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/86c51678350f656dcc7f490a43946ee5-Paper.pdf.

[17] Filip Svoboda et al. "Deep learning on microcontrollers: a study on deployment costs and challenges". In: *Proceedings of the 2nd European Workshop on Machine Learning and Systems*. EuroMLSys '22. Rennes, France: Association for Computing Machinery, 2022, pp. 54–63. ISBN: 9781450392549. DOI: 10.1145/3517207.3526978. URL: https://doi.org/10.1145/3517207.3526978.

[18] Mark Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". In: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[19] Andrew Howard et al. "Searching for MobileNetV3". In: *CoRR* abs/1905.02244 (2019). arXiv: 1905.02244. URL: http://arxiv.org/abs/1905.02244.

[20] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: *CoRR* abs/1709.01507 (2017). arXiv: 1709.01507. URL: http://arxiv.org/abs/1709.01507.

[21] Akash Nagaraj. *ASL Alphabet*. 2018. DOI: 10.34740/KAGGLE/DSV/29550. URL: https://www.kaggle.com/dsv/29550.

[22] Matti Matilainen et al. "OUHANDS database for hand detection and pose recognition". In: *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. 2016, pp. 1–5. DOI: 10.1109/IPTA.2016.7821025.

[23] Gopa Bhaumik et al. "CrossFeat: Multi-scale Cross Feature Aggregation Network for Hand Gesture Recognition". In: *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*. 2020, pp. 274–279. DOI: 10.1109/ICIIS51140.2020.9342652.

[24] Khushi Gupta et al. "Hand gestures recognition using edge computing system based on vision transformer and lightweight CNN". In: *Journal of Ambient Intelligence and Humanized Computing* 14.3 (2023), pp. 2601–2615. DOI: 10.1007/s12652-022-04506-4.

[25] Yong Soon Tan et al. "Convolutional neural network with spatial pyramid pooling for hand gesture recognition". In: *Neural Computing and Applications* 33 (2021), pp. 5339–5351. DOI: 10.1007/s00521-020-05337-0.

[26] Mingyue Zhang et al. "A Lightweight Network Deployed on ARM Devices for Hand Gesture Recognition". In: *IEEE Access* 11 (2023), pp. 45493–45503. DOI: 10.1109/ACCESS.2023.3273713.