QMX-BdSL49: An Efficient Recognition Approach for Bengali Sign Language with Quantize Modified Xception

Nasima Begum*, Saqib Sizan Khan, Rashik Rahman, Ashraful Haque, Nipa Khatun, Nusrat Jahan, Tanjina Helaly
Department of Computer Science and Engineering, University of Asia Pacific,
Dhaka, Bangladesh

***Abstract*—Sign language is developed to bridge the communication gap between individuals with and without hearing impairment or speech difficulties. Individuals with hearing and speech impairment typically rely on hand signs as a means of expressing themselves. However, people, in general, may not have sufficient knowledge of sign language, thus a sign language recognition system on an embedded device is most needed. Literature related to such systems on embedded devices is scarce as these recognition tasks are very complex and computationally expensive. The limited resources of embedded devices cannot execute complex algorithms like Convolutional Neural Network (CNN) properly. Therefore, in this paper, we propose a novel deep learning architecture based on default Xception architecture, named Quantized Modified Xception (QMX) to reduce the model's size and enhance the computational speed without compromising model accuracy. Moreover, the proposed QMX model is highly optimized due to the weight compression of model quantization. As a result, the footprint of the proposed QMX model is 11 times smaller than the Modified Xception (MX) model. To train the model, BDSL 49 dataset is utilized which includes approximately 14,700 images divided into 49 classes. The proposed QMX model achieves an overall F1 accuracy of 98%. In addition, a comprehensive analysis among QMX, Modified Xception Tiny (MXT), MX, and the default Xception model is provided in this research. Finally, the model has been implemented on Raspberry Pi 4 and a detailed evaluation of its performance has been conducted, including a comparison with existing state-of-the-art approaches in this domain. The results demonstrate that the proposed QMX model outperforms the prior work in terms of performance.**

Keywords*—*Bengali sign language; CNN; computer vision; model quantization; Raspberry Pi 4; transfer learning; Tiny ML

I. Introduction

Disability is a crucial issue in terms of human rights because a person with an impairment is usually deprived of ordinary public welfare. Almost a billion of the world's population has some form of physical disability[1]. Individuals with disabilities experience more negative socioeconomic consequences, resulting in a poorer standard of life. While over 430 million people worldwide suffer from hearing impairment[2], there are more than 1.7 million hearing and speaking impaired people in Bangladesh alone[3]. These impaired people belong to the Bangladesh Deaf and Mute Community (BDMC). Due to their communication impediment, the BDMC, faces numerous obstacles while attempting to participate in education, work, social activities, and other aspects of everyday life.

Sign language employs the visual-manual paradigm to communicate meaning. Sign language is conveyed via hand and finger movements to create gestures. The only way to communicate with people with hearing or speaking disabilities is through sign language. Similar to every other language, the Bengali language has its own sign language, which is known as Bengali Sign Language (BdSL). The BDMC uses only BdSL to communicate with everyone, which restricts their ability to converse with society, as the majority of the society does not know sign language due to a lack of social awareness.

In the aforementioned scenario, communication between the BDMC and society requires a sign language interpreter. However, a skilled interpreter may not always be readily available, and in such circumstances, paying fair fees may be a serious worry. An automated recognition system for sign language can play a vital role in reducing the basic and social differences between society and BDMC. Therefore, sign language recognition is a popular area of study. Current research in this area focuses mostly on either sensor-based [1] or vision-based [2] systems.

Numerous studies have been conducted on BdSL recognition, and there are numerous benchmarking datasets [3], [4], [5], [6], [7] for BdSL recognition. However, these datasets are insufficient for training and evaluating deep learning models, and the majority are not open-source. CNN [8], [9] is a popular choice along with the transfer learning [10], [11], [12] model to recognize BdSL.

Several research implements the CNN model for recognizing BdSL. Hossain et al. [13] proposed a CNN-based sign language recognition model and achieved 98.75% accuracy. Islalm et al. [14] also proposed a CNN-based model, and they evaluated their model using 10-fold cross-validation. They achieved 99.80% accuracy. Some research utilized CNN-based transfer learning models for recognition. Rafi et al. [4] utilized the VGG19 transfer learning model with 89.6% accuracy. To our knowledge, no prior work exists on constructing an efficient deep learning model that can be implemented in embedded or IoT devices via model quantization. The majority of recent work employed mainstream or pre-trained models. Therefore, these models cannot be implemented on devices with a low configuration.

[1]https://www.who.int/news-room/fact-sheets/detail/disability-and-health
[2]https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss
[3]https://en.wikipedia.org/wiki/Deafness_in_Bangladesh

Therefore, this research proposes a BdSL recognition deep learning system in which the available default Xception [15] model has been extensively altered to obtain a new model with greater accuracy. In addition, the novel Modified Xception (MX) model has been quantized in order to be applied in embedded systems. Thus in this research, a novel Quantized Modified Xception (QMX) model is proposed. This paper also offers a comprehensive investigation of QMX, Modified Xception Tiny (MXT), Modified Xception (MX), and the default Xception model. Furthermore, the developed QMX model has been successfully implemented on an embedded device, namely the Raspberry Pi 4. The QMX recognition model is only 3.3MB in size and contains just 3,317,201 trainable parameters, making it an extremely lightweight model for embedded system implementation. To train and evaluate the model, an open-source benchmark dataset named BDSL 49 [16] that contains 14,700 images divided into 49 classes is utilized. The QMX model achieved an overall F1 accuracy of 98%.

The main contributions of this research are as follows:

- A quantization algorithm for the Modified Xception (MX) model.
- A Quantized Modified Xception (QMX) model has been proposed to recognize hand signs and predict the characters. The QMX model is 72 and 11 times lighter than the default Xception and Modified Xception (MX) models respectively. Moreover, it achieved an average F1-score of 98%.
- For IoT implementation, the proposed QMX model is deployed on an embedded system, which is Raspberry Pi 4 for evaluating the model efficiency and inference time.

The rest of the paper is structured as follows: Section II represents the literature review. The dataset description is provided in Section III. The proposed methodology is described in Section IV. Section V analyzes the results of the conducted experiments. Lastly, Section VI concludes the paper with some future works.

II. Literature Review

Recognition of sign language is a very intriguing area of study. Although research on BdSL recognition is abundant, few have managed to make it implementable and attainable in a practical situation; therefore, it remains unexplored. This section outlines the evolution of research regarding BdSL recognition.

Islam et al. [17] provided a Bengali sign language digit recognition system using deep learning, which delivers the output in text form. Using the Ishara-Lipi dataset, they developed the proposed model, which gained about 95% accuracy. However, they did not use RGB images during model training. Khan et al. [18] proposed a CNN and Region of Interest (ROI) segmentation-based BdSL translator device that can translate only five sign gestures. It has a 94% accuracy rate for recognizing signs in real-time. Due to the lack of available signs for different words, they built the device using five words only. The authors of [19], reviewed the research approaches of BdSL from 2002 to 2021 and discussed each work's contributions and weaknesses. The Scale Invariant Feature Transform (SIFT) technique and CNN were used in the proposed system of [3] to detect one-handed gestures of 38 Bengali signs. However, they used grayscale images for training their proposed model. Using CNN, the authors of [13] achieved 98.75% accuracy in recognizing Bengali signs. However, they only used one-handed sign gestures. Shurid et al. [9], proposed a Bengali sign language recognition and sentence building CNN-based model and achieved 90% accuracy using their augmented dataset. However, their proposed model could not work properly to recognize critical sign gestures.

Ishara-Lipi [6] is a commonly used dataset for BdSL recognition, though it consists of only 36 characters out of 49. “BenSignNet” a CNN and concatenated segmentation-based model, was proposed in [8] that only detects Bengali Sign Language alphabets using three different datasets. However, the model is computationally expensive as it used several image processing techniques. Ilias et al. [20] proposed a Sign Language Recognition Generative Adversarial Network (SLRGAN) using a Context-Aware Generative Adversarial Network architecture. The proposed model achieved 23.4%, 2.1%, and 2.26% word error rates for three primarily used datasets such as RWTH-Phoenix-Weather-2014, Chinese Sign Language (CSL), and Greek Sign Language (GSL). However, they considered only the contextual information of the sign language. A CNN-LSTM model was proposed to recognize both hands' lexical signs in Bangla [7]. However, the BdSL dataset has only 36 classes with 13,400 images and produced 90% training accuracy and 88.5% testing accuracy. Rafi et al. [4], proposed a VGG-19-based model to recognize 38 different classes of Bengali sign gestures and obtained 89.6% test accuracy. However, their dataset contains a low amount of sample images. In order to enhance inter-dataset performance, the research work [10] uses a variety of deep learning models and angular loss functions to highlight the significance of generalization in finger-spelled BdSL recognition. Due to a lack of diversity in the dataset, they achieved 55.93% and 47.81% test accuracy using the SphereFace loss function in the VGG-19 architecture. A pre-trained model called “MobileNet” was proposed in [11]. The authors proposed an approach for converting signs made in BdSL into their appropriate Bengali letters. They evaluated the model using the Ishara-Lipi dataset and achieved 95.71% accuracy. However, this model could not detect the hand signs in different backgrounds.

The authors of [21] presented a quantization method to estimate the floating-point calculations in a neural network using just integer arithmetic. The network quantization techniques are discussed by Garifulla et al. [22] which are used for disease diagnosis on portable medical devices to reduce the CNN models' size and inference time. The authors of [23] examined the mathematical properties of quantization parameters and assessed the choices made for a large variety of neural network models for various domains of application, such as voice, language, and vision. Koutayni [24] introduced a low-energy solution for depth camera-based hand posture estimation methods and compressed the deep neural network model using dynamic quantization approaches at various levels to gain maximum compression without sacrificing accuracy.

Most of the existing BdSL recognition models are trained with datasets, which are insufficient due to the lack of data

TABLE I. Classes Label and Name

label	0	1	2	3	4	5	6
Class Name	অ	আ	ই	উ	এ	ও	ক
label	7	8	9	10	11	12	13
Class Name	খ	গ	ঘ	চ	ছ	জ	ঝ
label	14	15	16	17	18	19	20
Class Name	ট	ঠ	ড	ঢ	ত	থ	দ
label	21	22	23	24	25	26	27
Class Name	ধ	প	ফ	ব	ভ	ম	য়
label	28	29	30	31	32	33	34
Class Name	র	ল	ন	স	হ	ড়	◌ং
label	35	36	37	38	39	40	41
Class Name	◌ঃ	০	১	২	৩	৪	৫
label	42	43	44	45	46	47	48
Class Name	৬	৭	৮	৯	_◌্	space	ঞ

TABLE II. Classwise Data Distribution

Label	Train Sample	Test Sample	Pixel	Format
0	240	60	128 x 128	RGB
1	240	60	128 x 128	RGB
2	240	60	128 x 128	RGB
...	...	...	....	...
48	240	60	128 x 128	RGB

samples in them. Furthermore, hardly any research focuses on the real-life implementation of BdSL recognition systems on low-end devices.

III. Dataset Description

The use of sign language is essential to communicate with persons with hearing disabilities or persons with speaking disabilities. A dataset is highly useful for an automated system to recognize the hand signs of Bengali Sign Language. For this research purpose, a dataset named BdSL 49 [16] is utilized with 49 classes. Each class represents a Bengali alphabet, numeric character, or special character (space, Hasantha). The dataset consists of 14,700 images organized into 49 categories. In Table I, 49 labels of the dataset referring to the naming of Bengali letters are listed. Fig. 1 illustrates sample images of each class.

Each sample image is labeled with the appropriate Bengali characters. Each class has approximately 300 images and is divided into two sections: one for training and the other for testing the model. 80% of the images for each class is considered for training, while the remaining 20% is considered for testing. Table II illustrates the data distribution between the train and the test set of the BdSL 49 dataset. As shown in Table II, there are 240 samples in the train set and 60 samples in the test set of each class. All images are in RGB format with a 128X128 pixel size.

IV. Methodology

This section discusses the proposed methodology. As the embedded low-end devices are unable to run computationally expensive models, thus a novel Modified Xception (MX) model is proposed and described in subsection IV-A. However, when the MX model is implemented in an embedded device to perform in real-time, its footprint needs to be compressed further. Hence, in subsection IV-B, a quantization method is proposed that quantizes the MX model, and thus, the QMX model is achieved by converting the float 32-bit MX model into an int 8-bit QMX model. The QMX model is proposed to run the model specifically in a relatively low-configuration devices in real time environment.

Initially, some benchmark transfer learning models, namely Xception, InceptionV3, InceptionResNetV2, MobileNet, MobileNetV2, ResNet50V2, ResNet101V2, and ResNet152V2 are trained using the BdSL 49 dataset. Among these eight models, the Xception model performed comparatively well. However, it provides only 93% accuracy. Based on the performance analysis, Xception has the best level of accuracy. The Xception model architecture is chosen as a framework for developing the proposed QMX architecture. However, the Xception model size is around 240MB, which is very large. Therefore, to improve the model's performance and decrease the model's size, the Xception model's architecture is altered, and a novel MX model is proposed. Afterward, the MX model is quantized to implement it in embedded devices. Thus, the Quantized Modified Xception (QMX) model is attained. Finally, the QMX model is implemented on an embedded device, namely the Raspberry Pi 4, for inference.

A. Proposed Modified Xception Architecture

The proposed MX model is a highly comprehensive architectural model featuring 31 layers instead of the 71 layers of the original Xception model. The architecture of Xception is divided into three components. Initially, the data passed via the entry flow, then eight times repeated in the middle flow, and finally, passed through the exit flow. However, the precision of hand sign recognition is inadequate. Additionally, since Xception is a generalized architecture, it contains an extensive number of trainable parameters that are not optimal for all types of recognition. Hence, the purpose of our research is to design an effective architecture with the essence of the Xception model and achieve adequate precision for BdSL recognition.

The proposed MX model utilizes the Depthwise Separable Convolutional (DSC) layer which is a variant of the Separable Convolutional layer. The DSC layer partitions the process into two or more sub-processes. Upon the conclusion of each sub-process, the outcomes are integrated with the overarching result. Therefore, it reduces the multiplication costs resulting for a similar type of process. The utilization of Separable Convolution reduces the computational cost as well as the number of trainable parameters. The standard convolution operation incurs a substantial computational expense due to the simultaneous processing of all color channels, as indicated by the multiplication cost outlined in Eq. (1). In contrast, separable convolution is a technique that separates the color channels. Consequently, the multiplication of the number of kernels presented in Eq. (2) is performed on a single channel.

$$Conv2D = K^2 * d^2 * C * N \tag{1}$$

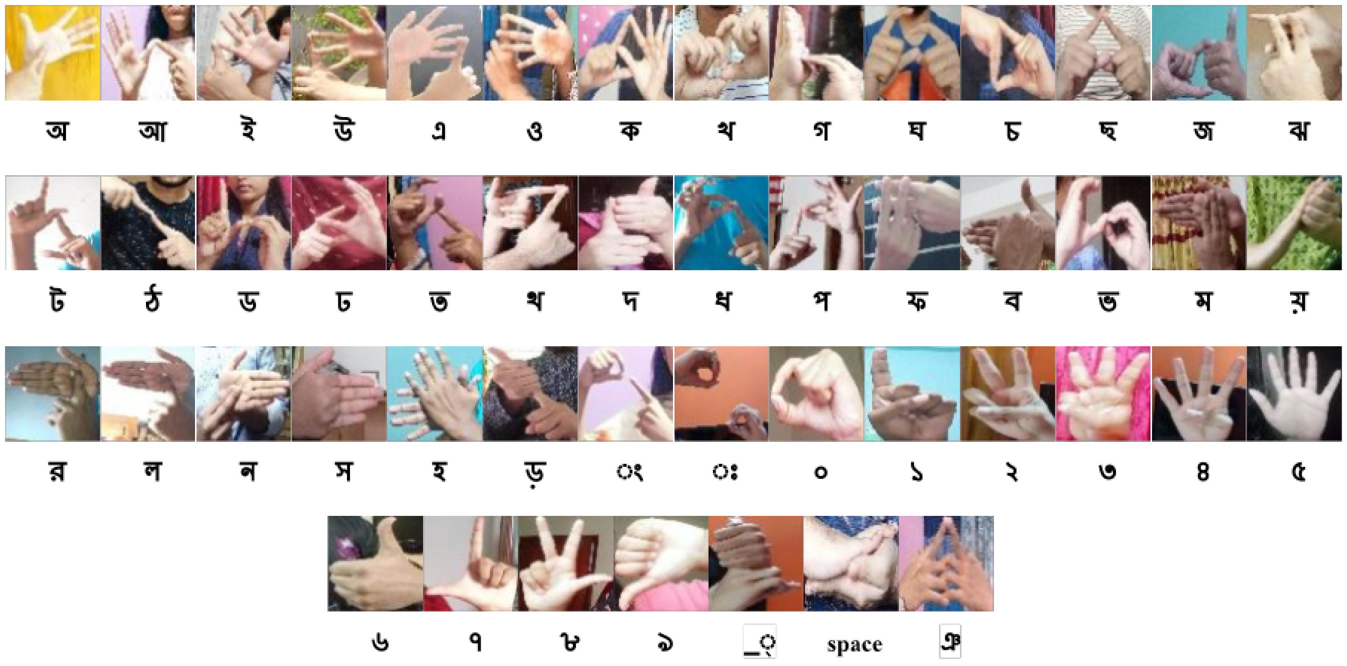

অ আ ই উ এ ও ক খ গ ঘ চ ছ জ ঝ
ট ঠ ড ঢ ত থ দ ধ প ফ ব ভ ম য়
র ল ন স হ ড় ং ঃ ০ ১ ২ ৩ ৪ ৫
৬ ৭ ৮ ৯ space

Fig. 1. Sample images from the dataset.

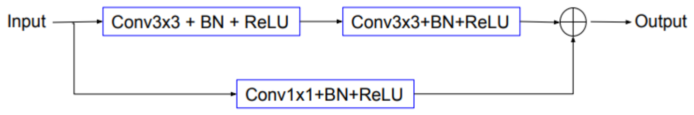


Fig. 2. Sample of the residual network.

$$SepConv2D = d^2 * 1 + 1^2 * N \tag{2}$$

Eq. (1) and Eq. (2) represents the complete multiplication costs for ordinary and separable convolution, respectively. The dimension K, as derived from the convolution process in Eq. (2), undergoes a transformation where it is multiplied by N after switching to a value of 1. Similarly, the variable d, which represents the size of the filter, is multiplied by 1 after undergoing a transformation and the variable C, representing the color channels in Eq. (2), is reduced to a value of 1. Finally, the variable N denotes the number of kernels. On the other hand, the neural network's architecture which comprises an excessive number of layers, may suffer from data loss resulting from a vanishing gradient. The MX model employs residual connections (a type of skip connection) which is illustrated in Fig. 2. The implementation of a residual layer mitigates this phenomenon and effectively incorporates various forms of data.

Fig. 3 illustrates the changes made to the default Xception architecture. Many layers are removed and a few layers are added in the Xception architecture in order to achieve an efficient architecture with sufficient precision. Changes in the architecture are classified in colors where "Green" signifies the addition of a new layer or connections to the Xception architecture. "Blue" signifies a modification of the parameter values of any existing layer in the Xception which is known as fine-tuning. "Red" signifies the elimination of the layers from the default Xception model.

The starting layer of the MX architecture is a standard convolutional layer with the kernel size modified to 5x5 to avoid overfitting. It was only performed in the top two convolution layers. After the first Separable Convolution layer, several layers are further added to the structure. The new architectural layers include Depthwise Convolution, Batch normalization, and LeakyReLU activation function. The RGB images that are provided to the Depthwise Convolution undergoes a process of channel separation, convolution, and subsequent re-stacking, as demonstrated in Eq. 3. The Separable Convolution process corresponds to the Depthwise Convolution, with the exception of an additional step. The process of pointwise convolution involves an additional step. Following the stacking process, the features depicted in Eq. 2 are extracted through the utilization of a 1x1 filter.

$$DepthwiseConv2D = d^2 * 1(C1) + d^2 * 1(C2) + d^2 * 1(C3) \tag{3}$$

In the given context, the variable d denotes the dimensions of the filter, while the C variables correspond to the color channels. Each color channel is assigned a value of 1 and subsequently multiplied by d. Hence, the aforementioned methodology exhibits swifter and more effective outcomes in comparison to traditional convolution. Batch Normalization is employed subsequent to the Depthwise Convolution layer to expedite the training process and streamline the learning

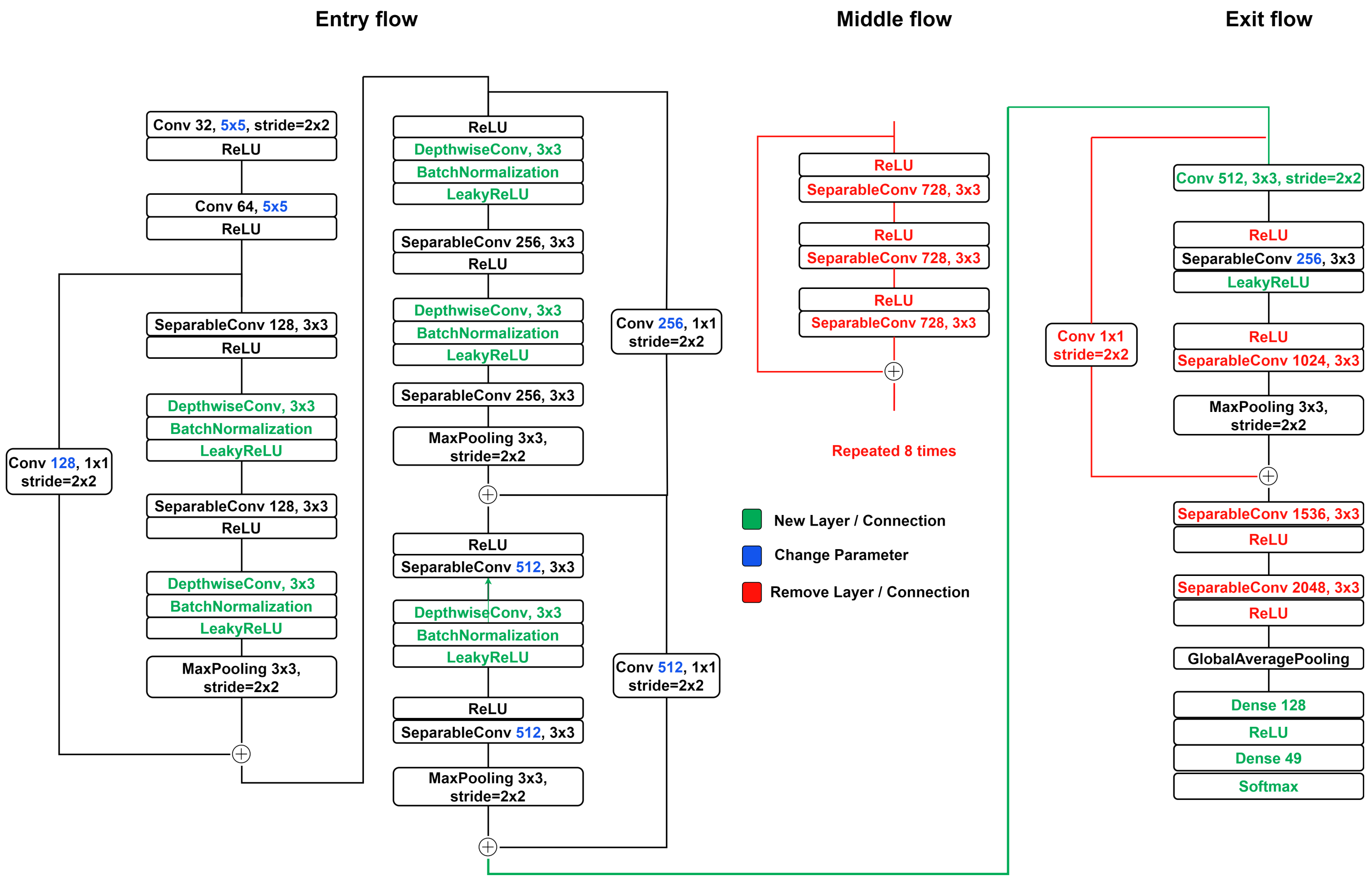


Fig. 3. Architectural modification of the Xception model.

procedure. Subsequently, the Leaky Rectified Linear Unit (LeakyReLU) activation function is employed. The Rectified Linear Unit (ReLU) activation function is triggered when the neurons within a neural network produce a positive value. In the absence of a positive value, the output of the function is zero. The LeakyReLU activation function is designed to mitigate the issue of "dead ReLU" that arises when the output of a ReLU is consistently negative. The system is capable of accepting certain numerical values that fall within the negative range and are in close proximity to zero. The LeakyReLU is mathematically represented by Eq. (4), where α is a constant parameter which is assigned a value of -0.01.

$$f(x) = max(\alpha x, x) \tag{4}$$

The objective of utilizing Depthwise Convolution and LeakyReLU is to derive supplementary features from images. It improves the precision of the model. The "Entry Flow" architecture employs the Depthwise Convolution and Batch Normalization techniques, along with the LeakyReLU activation function, at multiple locations, as illustrated in Fig. 3. Additionally, the filter value of the final two separable convolution layers in the "Entry Flow" segment is reduced, and the parameters of the residual connection layers are modified. This makes the model more lightweight. To make the model lighter and more streamlined, the entirety of the "Middle Flow" design is eliminated. This modification extensively reduces the computational cost of the model. An immediate connection between "Entry Flow" and "Exit Flow" is established. "Exit Flow" is also customized. A new convolution layer and LeakyReLU activation are added to this segment, and most of the separable convolution layers are omitted. The residual layer is also removed from this portion. After the GlobalAveragePooling layer, two fully connected layers are added with ReLU and Softmax activation functions respectively. An overview of the proposed MX model architecture is illustrated in Fig. 4.

B. *Quantized Modified Xception (QMX)*

Model Quantization is a well-known model compression approach that reduces the computational load and memory usage of the neural network models. The proposed quantization technique for the MX model is presented in this section. In this method, r represents the real number, q denotes the bit representation of the values, or quantized values. Most of the time, quantized networks are trained using floating point numbers, and then the weights are quantized. The before and after quantization of each of the convolutional layers are illustrated in Fig. 5, where (a) introduces 8-bit integers and 32-bit integer accumulator and (b) illustrates that the convolution layers are trained using simulated quantization. All variables and calculations utilize 32-bit floating-point arithmetic. To imitate the effects of variable quantization, the computation graph

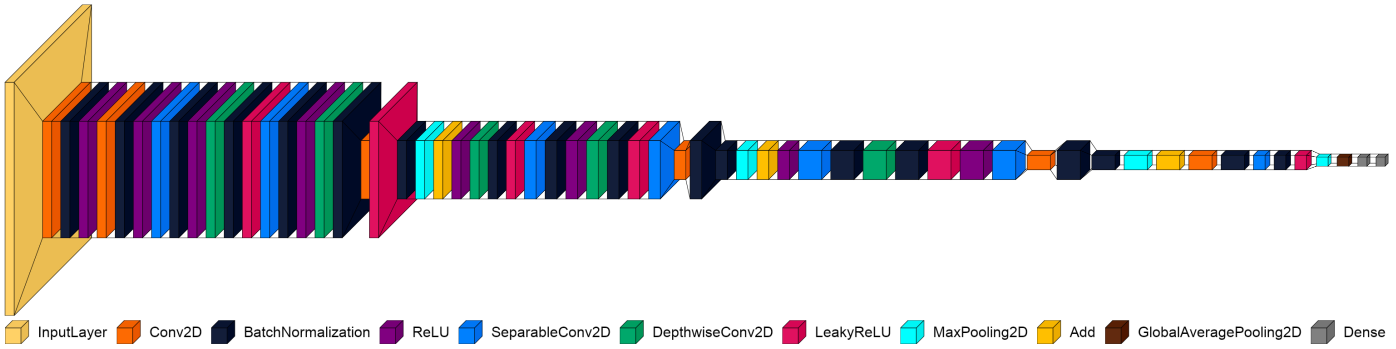


Fig. 4. Proposed MX architecture.

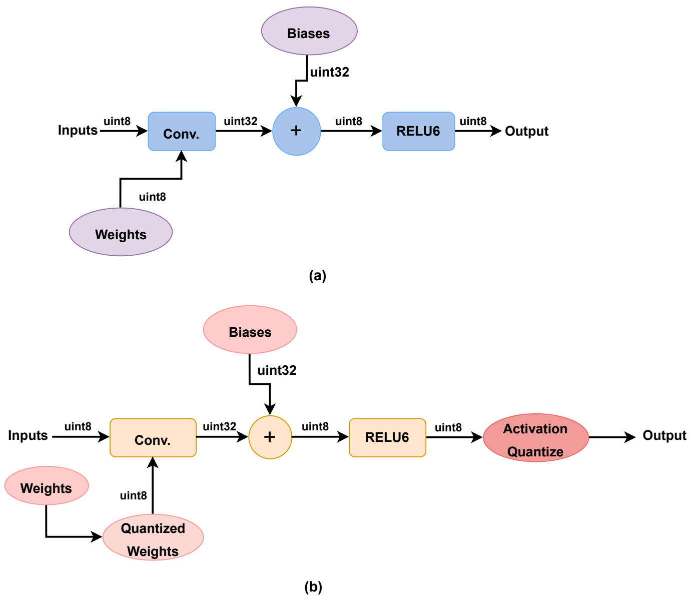


Fig. 5. Integer quantization in the convolution layer.

is infused with weight quantization and activation quantization endpoints. The resulting graph estimates the integer-only computation graph during training with standard optimization techniques for floating-point models.

A technique is provided during the forward pass of training for simulating quantization effects. The Backpropagation proceeds as usual, and floating point values are utilized for all weights and biases, allowing straightforward correction of minor values. Conversely, the forward propagation pass simulates quantized inference as implemented by the inference engine. The quantization technique's rounding behavior is integrated into floating-point arithmetic, such as: 1) The weights are quantized before convolving. The layer is normalized with batch normalization prior to quantization. The weights, $\hat{w}$ are incorporated with the batch normalization parameters using Eq. (5), where γ is the batch normalization scale parameter. The symbol φ is a very small constant, and η is a moving average estimation of the variance of convolution results throughout the batch. 2) Activation functions are quantified where they arise during inference.

$$\hat{w} = \frac{\gamma \times w}{(\eta + \varphi)^{\frac{1}{2}}} \tag{5}$$

The point-wise quantization function q in Eq. (6) is used to perform quantization, which is controlled for each layer by the quantization level numbers and the clipping range.

$$clip(R; p, q) = min(max(x, p), q)$$
$$s(p, q, \kappa) = \frac{q - p}{\kappa - 1}$$
$$q(R; p, q, \kappa) = \lfloor \frac{clip(R, p, q) - p}{s(p, q, \kappa)} \rceil * s(p, q, \kappa) + p \tag{6}$$

Here, R represents the real value that must be quantized and the notation for the quantization spectrum is p, q, κ indicates the number of quantization levels, while "$\lfloor\rceil$" indicates rounding to the nearest integer. The value of κ is set to 8 because, for 8-bit quantization, $\kappa = 2^8 = 256$ is utilized.

The quantized model's workflow is described by Algorithm 1. At first, the model receives the sign image and its corresponding labels as initial input. Following this, the MX floating point deep learning architecture shown in Fig. 4 is constructed (step 1). Following the development of the model, it is quantized to enable quantize-aware training (step 2). The quantized model is finally trained until it reaches convergence (step 3). The model is thus ready for inference, at which point, it can predict the desired output based on an input image (steps 4 and 5). Here, Quantization Aware Training (QAT) is used for compressing the proposed model. QAT is a model quantization technique where quantization operations are inserted before training the model. It enables the quantized weights and activation functions of the model to be adjusted.

C. Raspberry Pi Integration

The QMX model is implemented into Raspberry Pi 4 embedded system in order to evaluate the performance of the model in a real-world configuration. Python is utilized as the programming language for model implementation and TensorFlow 2.6 for inference. Initially, the input images or video frames are streamed by the device's camera module. Then, it predicts the corresponding signs. Finally, the output is displayed on a display screen connected to the device through

Algorithm 1 Training Steps of Quantization

```
Input: Sign images and corresponding labels.
Output: Recognize class name of sign images.
1: Create a training graph for a floating point model.
2: During training and inference, tensors will be reduced into fewer bits while inserting the quantization operations.
3: Train the quantized model using training data until convergence.
4: Perform the necessary improvements and build the inference graph for usage in a low-bit inference engine.
5: Use the quantized inference graph to draw conclusions from the data.
```

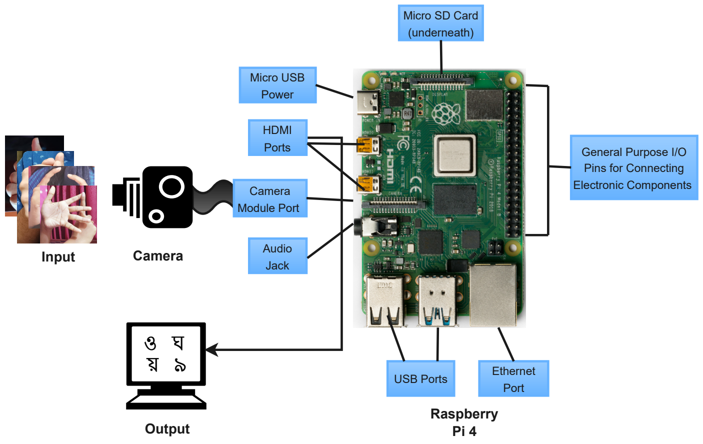


Fig. 6. Overview of raspberry pi 4 implementation.

its HDMI connector. Fig. 6 represents an overview of the Raspberry Pi 4 implementation.

D. Experimental Setup

The Google Colab Pro edition is utilized to execute the training procedure. A variety of Python packages are utilized for implementation. Pickle is used for data loading. Tensorflow serves as the backend to the Keras framework for model development. “Categorical Cross-Entropy” is chosen as a loss function for model compilation. Adaptive Moment Optimization (Adam) is utilized as an optimizer with a learning rate of 0.01. The Adam optimizer offers a quicker computation time and requires fewer training parameters. The gradient descent algorithm uses this approach by taking the gradients’ exponentially weighted average into account. Using averages accelerates the algorithm’s convergence towards minima. The batch size is set to 58 for 60 epochs and evaluates the model using the test dataset. The batch size of 58 is determined since the entire train sample is a factor of 58.

After the QMX model has been established, it is implemented on a Raspberry Pi 4 for inference. The original Xception, MX, and MXT models are also incorporated into the embedded device, enabling the examination of real-world performance. As a result of implementing all four models on Raspberry Pi 4, a quantitative analysis of the performance of these models in terms of model flash occupancy, inference time, and energy consumption is obtained. Energy consumption and inference time are important considerations for evaluating real-world performance. The embedded device is a stand-alone device connected to a power bank, thus if the model is very efficient, then it consumes less power. Additionally, shorter estimation time indicates greater real-time accuracy.

V. Result Analysis

In this section, experimental results and model assessments are quantitatively analyzed. The QMX model has been subjected to comparative analysis with several existing models. Besides, an estimation is conducted on the memory consumption, average operational duration, and average energy consumption of the MX, MXT, and QMX variants. The precision, recall, and F1 accuracy of the quantized model are also measured.

A. Evaluation Matrices

To evaluate the QMX model, five evaluation metrics are selected, namely precision, recall, F1-score, ROC-AUC curve, and confusion matrix. For an ideal classification model, both precision and recall tend to be one (1). The F1 score is determined by the weighted average of precision and recall. Therefore, this score takes into account both false positives and false negatives.

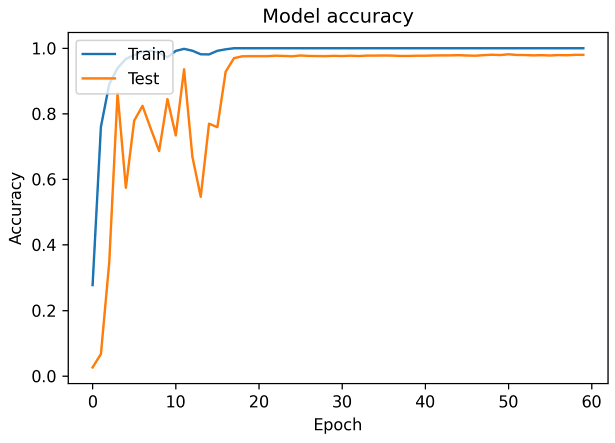
The confusion matrix is a prominent metric used in classification problems. It is applicable to both binary classification and multiclass classification problems. The comparison between the predicted outcome and the actual result can be derived using the confusion matrix. The AUC (Area under the ROC Curve) - ROC curve (Receiver Operating Characteristic curve) is a performance metric for classification problems with different threshold values.

B. Evaluation of the QMX Model

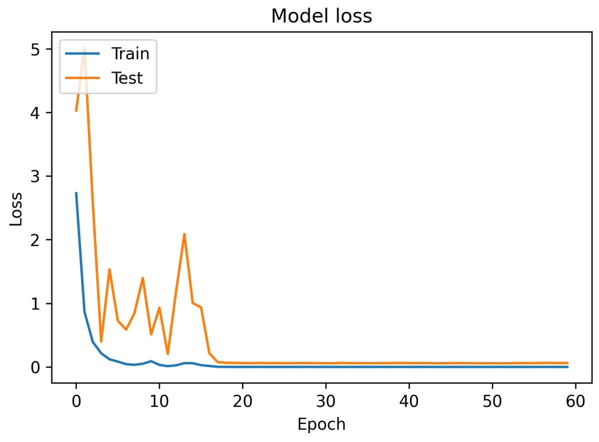
Fig. 7 illustrates the accuracy and loss graph of the QMX model while training. Here, the training score is represented by the blue line and the test score is represented by the yellow line during the training phase. From this figure, it is observed that the training score converges rapidly just within 10 epochs and remains constant for the rest of the epochs. On the contrary, the test score converges within 20 epochs after significant fluctuation. In addition, after convergence, the train and test scores become almost the same, the test score is slightly lower than the training score. Therefore, it can be concluded that the model exhibits a high degree of generalization.

Fig. 8 depicts the confusion matrix of the QMX model, revealing that the model is quite accurate at recognizing 49 distinct hand signs. Sixty images per class are used to evaluate the trained model. Approximately 59 to 60 images, and in some instances 57 or 58, are accurately predicted for each class.

Table III presents the classification report which contains evaluation metrics, namely precision, recall, and F1-score for each class of the MX, MXT, and QMX models. Although the average, macro average and mean average are the same for all three models, the QMX model is lighter, faster, and more efficient among these three. Moreover, embedded devices can utilize the QMX model better. Thus, the QMX model is



(a)



(b)

Fig. 7. QMX model accuracy and loss graph.

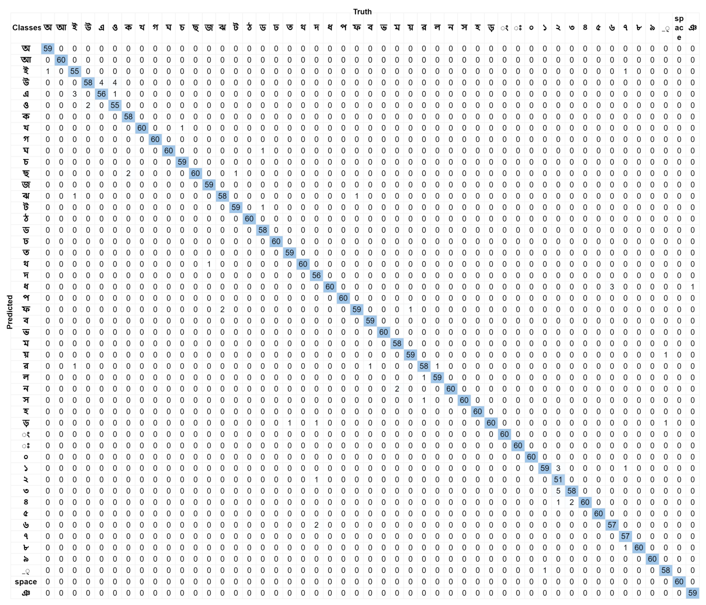


Fig. 8. Confusion matrix of QMX model.

TABLE III. PERFORMANCE EVALUATION OF DIFFERENT MODELS

	MX			QMX		
label	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
0	0.97	0.98	0.98	1.00	0.98	0.99
1	0.98	1.00	0.99	1.00	1.00	1.00
2	0.98	0.97	0.97	0.96	0.88	0.92
3	0.89	0.93	0.91	0.87	0.97	0.91
4	0.96	0.92	0.94	0.93	0.93	0.93
5	0.95	0.97	0.96	0.97	0.93	0.95
6	0.97	1.00	0.98	1.00	0.97	0.98
7	0.97	1.00	0.98	0.98	1.00	0.99
8	1.00	1.00	1.00	1.00	1.00	1.00
9	0.97	1.00	0.98	0.97	1.00	0.98
10	1.00	0.97	0.98	1.00	0.98	0.99
11	0.98	0.97	0.97	0.95	1.00	0.98
12	1.00	0.98	0.99	1.00	0.98	0.99
13	0.98	0.92	0.95	0.97	0.97	0.97
14	1.00	0.98	0.99	1.00	0.98	0.99
15	1.00	1.00	1.00	1.00	0.98	0.99
16	1.00	1.00	1.00	1.00	0.97	0.98
17	0.98	1.00	0.99	0.98	1.00	0.99
18	1.00	0.98	0.99	1.00	1.00	1.00
19	0.98	1.00	0.99	0.98	1.00	0.99
20	1.00	0.95	0.97	0.98	0.93	0.96
21	0.95	1.00	0.98	0.92	1.00	0.96
22	0.97	0.97	0.97	1.00	1.00	1.00
23	0.92	0.98	0.95	0.95	0.98	0.97
24	1.00	1.00	1.00	0.98	1.00	0.99
25	1.00	1.00	1.00	1.00	1.00	1.00
26	0.98	0.97	0.97	0.98	0.97	0.97
27	1.00	1.00	1.00	0.98	0.98	0.98
28	0.98	0.95	0.97	0.98	0.98	0.98
29	0.98	0.98	0.98	0.97	0.98	0.98
30	0.95	1.00	0.98	0.97	0.98	0.98
31	1.00	0.98	0.99	1.00	1.00	1.00
32	1.00	1.00	1.00	1.00	1.00	1.00
33	0.94	1.00	0.97	0.97	1.00	0.98
34	1.00	0.98	0.99	0.98	1.00	0.99
35	1.00	1.00	1.00	1.00	1.00	1.00
36	1.00	1.00	1.00	1.00	1.00	1.00
37	1.00	1.00	1.00	0.92	0.98	0.95
38	1.00	0.97	0.98	0.98	0.85	0.91
39	0.95	0.98	0.97	0.92	0.95	0.93
40	1.00	0.98	0.99	0.97	1.00	0.98
41	1.00	1.00	1.00	1.00	1.00	1.00
42	0.98	0.95	0.97	0.97	0.93	0.95
43	1.00	0.95	0.97	1.00	0.92	0.96
44	0.97	1.00	0.98	0.97	1.00	0.98
45	1.00	0.98	0.99	1.00	1.00	1.00
46	1.00	1.00	1.00	1.00	0.97	0.98
47	1.00	1.00	1.00	1.00	1.00	1.00
48	1.00	0.98	0.99	1.00	0.98	0.99
accuracy	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
macro avg	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
weighted avg	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**

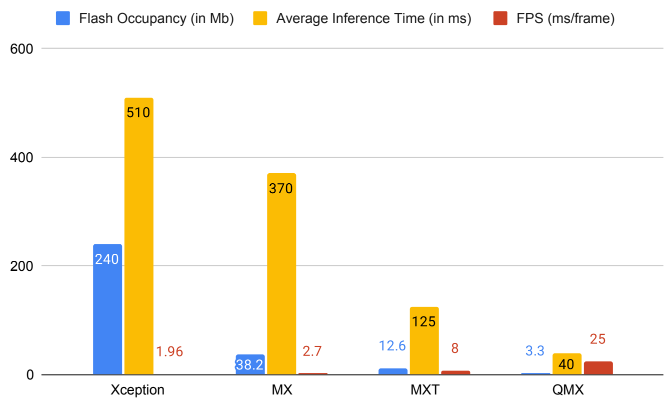


Fig. 9. Comparison of Xception, MX, MXT, and QMX model regarding flash Occupancy, inference time and Frames Per Second (FPS).

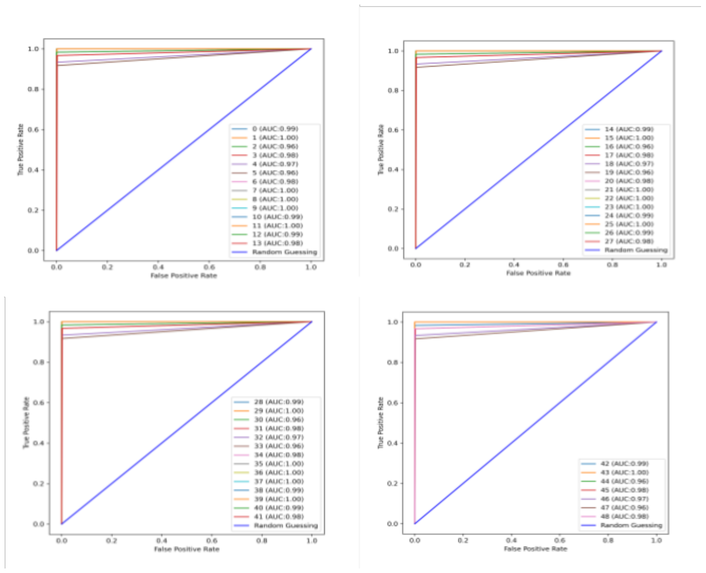


Fig. 10. ROC and AUC curve of QMX model.

better in terms of space and time complexity while maintaining accuracy.

Fig. 9 establishes that, compared to the original Xception, MX, and MXT models, the int8 QMX model consumes less flash memory and average inference time. The average inference time of these four models is calculated after 100 iterations using the Raspberry Pi 4. The QMX model's flash occupancy is 72, 11, and 3 times less than the original Xception, MX, and MXT models, respectively. Besides, the recognition performance of the QMX model is 12, 9, and 3 times faster than the original Xception, MX, and MXT models, respectively. In a nutshell, Fig. 9 displays the lightweight features of the QMX model, and these percentages imply a significant increase in the model's lightweight characteristics. In addition, the QMX model can process 25 frames per second, as demonstrated in Fig. 9; hence, the model can predict hand gestures in real-time.

The average F1-score is 0.98, which means the model's F1 accuracy is 98%. Fig. 10 represents the ROC and AUC graph. Here, the ROC threshold value is plotted for each class and then the AUC curve for each class is determined. For all classes, AUC lies between 0.96 and 1.00, making it a very generalized model that can perform well in real-life configurations.

C. Saliency Visualization

A saliency map is a visual representation of the area where viewers' attention tends to first focus [25]. It can be used to focus on the most prominent areas of a picture that are most likely to influence the model's prediction. By using Eq. (7) and (8), a saliency map can be visualized.

$$S_x(i) = {w_x}^T * i + b_x \tag{7}$$

$$w = \frac{\partial S_x}{\partial i} \tag{8}$$

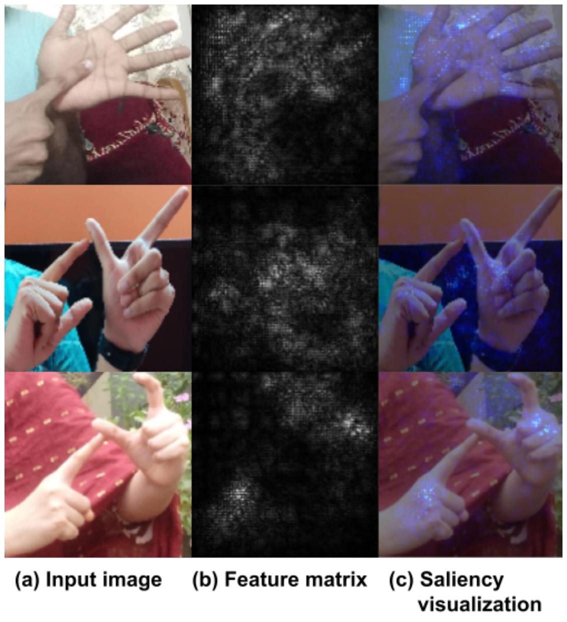


Fig. 11. Saliency map visualization.

Here, $S_x(i)$ represents the score of the predicted class x, the one-dimensional image vector is referred to by i, w indicates the weight and b represents the bias for the predicted class x. The gradient specifies how strongly each pixel of the image (i) can influence the outcome of the prediction (S). Knowing the weights (w; using Eq. (8)) for each pixel allows us to display the information as a saliency map, where each pixel represents the strength with which it influences the outcome of the prediction. Fig. 11 reveals which pixels are crucial for predicting the signs. And it can be seen from this figure that the saliency map highlights the pixels of the hand signs, indicating that the QMX model takes into account the hand elements from the image to produce precise predictions.

To generate a saliency map, the first input images shown in Fig. 11(a) are fed to the model. Afterward, the corresponding class x is predicted by the model. A 2D matrix is constructed utilizing a Gaussian pyramid from an image vector, which is responsible for activating class x, by employing the established weights (w). This 2D matrix is shown in Fig. 11(b) and it depicts the important pixel areas of the images responsible for the prediction of class x. Furthermore, upon projecting the aforementioned 2D matrix onto the input image, it becomes apparent that the salient features of the image are distinctly emphasized, as illustrated in Fig. 11(c). Depending on the prediction score S_x, the important areas are strongly or weakly highlighted. Fig. 11(c) reveals that the areas of the hand signs are prominently highlighted and concentrated, indicating that the model considers the hand signs to be the most essential attribute for predicting the related signs.

D. Comparison with State of the Arts

The present investigation employs BDSL 49 for the purpose of training pre-existing architectures. Table IV presents a visual representation of the results obtained by various architectures on the BDSL 49 dataset. By recreating, training and testing the model of [14],[4],[3],[6] and [8] on the BDSL

TABLE IV. Performance Comparison Between Proposed QMX Model with the State-of-the-Art Architectures when Trained on BDSL 49 DataSet

Research	Dataset	Classes	F1 Accuracy
Islalm et al. [14]	BDSL 49	49	92%
Rafi et al. [4]	BDSL 49	49	93%
Shanta et al. [3]	BDSL 49	49	93%
Islam et al. [6]	BDSL 49	49	94%
Miah et al. [8]	BDSL 49	49	77%
Proposed QMX Architecture	BDSL 49	49	98%

TABLE V. Performance Analysis of Proposed QMX Model when Trained on other Available Datasets

Dataset	Train Data	Test Data	Classes	F1 Accuracy
38 BdSL [14]	11061	1520	38	89%
KU-BdSL [26]	1200	300	30	99%
Ishara-Lipi [6]	3333	792	36	93%
BdSL 49 [16]	11774	2940	49	98%

49 dataset, the QMX model achieved an F1 accuracy of 92%, 93%, 93%, 94%, and 77%, respectively. The QMX model got the highest accuracy of 98% for the proposed QMX architecture. On the other hand, Table V shows the performance of the proposed QMX architecture using some benchmark datasets. For the proposed QMX architecture, it achieved 89%, 99%, and 93% respectively, using the datasets [14], [26] and [6]. Besides, utilizing BDSL 49 dataset the QMX model achieved an F1 accuracy of 98%. This comprehensive comparison indicates that the proposed QMX architecture is quite standard for BdSL recognition.

VI. Conclusion

Persons with hearing impairment may face challenges in interacting with those who primarily use verbal language for communication. On the other hand, the majority of people in society can not understand sign language, which creates a communication gap between them. Therefore, in order to address this concern, this research has devised a QMX framework for the identification of BdSL that can be implemented on an embedded system. The model under consideration has been trained on a large dataset, referred to as BdSL 49, which has been designed to closely resemble real-world scenarios. This has resulted in the development of a model that exhibits a high degree of accuracy. The dataset maintains the standard sign representation of Bengali Sign Language, containing 49 different classes. Furthermore, the proposed QMX model is efficient as it requires low resources and can run in real time on a low-end device or embedded system. Besides, it is 11 times smaller than the proposed MX model and achieves an overall accuracy of 98%. In order to alleviate the challenges faced by persons with hearing disabilities or persons with speaking disabilities, our proposed future research endeavors involve the development and implementation of a language model capable of generating text from video streaming.

Acknowledgment

We give special thanks to the Institute of Energy, Environment, Research, and Development (IEERD) and the University of Asia Pacific (UAP), for supporting this research project.

References

[1] K. Kudrinko, E. Flavin, X. Zhu, and Q. Li, “Wearable sensor-based sign language recognition: A comprehensive review,” *IEEE Reviews in Biomedical Engineering*, vol. 14, pp. 82–97, 2020.

[2] S. Sharma and S. Singh, “Vision-based sign language recognition system: A comprehensive review,” in *2020 International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 2020, pp. 140–144.

[3] S. S. Shanta, S. T. Anwar, and M. R. Kabir, “Bangla sign language detection using sift and cnn,” in *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2018, pp. 1–6.

[4] A. M. Rafi, N. Nawal, N. S. N. Bayev, L. Nima, C. Shahnaz, and S. A. Fattah, “Image-based bengali sign language alphabet recognition for deaf and dumb community,” in *2019 IEEE global humanitarian technology conference (GHTC)*. IEEE, 2019, pp. 1–7.

[5] P. P. Urmee, M. A. Al Mashud, J. Akter, A. S. M. M. Jameel, and S. Islam, “Real-time bangla sign language detection using xception model with augmented dataset,” in *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, 2019, pp. 1–5.

[6] M. S. Islam, S. S. S. Mousumi, N. A. Jessan, A. S. A. Rabby, and S. A. Hossain, “Ishara-lipi: The first complete multipurposeopen access dataset of isolated characters for bangla sign language,” in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. IEEE, 2018, pp. 1–4.

[7] N. Basnin, L. Nahar, and M. S. Hossain, “An integrated cnn-lstm model for bangla lexical sign language recognition,” in *Proceedings of International Conference on Trends in Computational and Cognitive Engineering*. Springer, 2021, pp. 695–707.

[8] A. S. M. Miah, J. Shin, M. A. M. Hasan, and M. A. Rahim, “Bensignnet: Bengali sign language alphabet recognition using concatenated segmentation and convolutional neural network,” *Applied Sciences*, vol. 12, no. 8, p. 3933, 2022.

[9] S. A. Shurid, K. H. Amin, M. S. Mirbahar, D. Karmaker, M. T. Mahtab, F. T. Khan, M. G. R. Alam, and M. A. Alam, “Bangla sign language recognition and sentence building using deep learning,” in *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE, 2020, pp. 1–9.

[10] S. K. Youme, T. A. Chowdhury, H. Ahamed, M. S. Abid, L. Chowdhury, and N. Mohammed, “Generalization of bangla sign language recognition using angular loss functions,” *IEEE Access*, vol. 9, pp. 165 351–165 365, 2021.

[11] T. M. Angona, A. S. Shaon, K. T. R. Niloy, T. Karim, Z. Tasnim, S. S. Reza, and T. N. Mahbub, “Automated bangla sign language translation system for alphabets by means of mobilenet,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, pp. 1292–1301, 2020.

[12] K. K. Podder, M. E. Chowdhury, A. M. Tahir, Z. B. Mahbub, A. Khandakar, M. S. Hossain, and M. A. Kadir, “Bangla sign language (bdsl) alphabets and numerals classification using a deep learning model,” *Sensors*, vol. 22, no. 2, p. 574, 2022.

[13] S. Hossain, D. Sarma, T. Mittra, M. N. Alam, I. Saha, and F. T. Johora, “Bengali hand sign gestures recognition using convolutional neural network,” in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, 2020, pp. 636–641.

[14] M. S. Islalm, M. M. Rahman, M. H. Rahman, M. Arifuzzaman, R. Sassi, and M. Aktaruzzaman, “Recognition bangla sign language using convolutional neural network,” in *2019 international conference on innovation and intelligence for informatics, computing, and technologies (3ICT)*. IEEE, 2019, pp. 1–6.

[15] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[16] A. Hasib, S. S. Khan, J. F. Eva, M. Khatun, A. Haque, N. Shahrin, R. Rahman, H. Murad, M. Islam, M. R. Hussein *et al.*, “Bdsl 49: A comprehensive dataset of bangla sign language,” *arXiv preprint arXiv:2208.06827*, 2022.

[17] S. Islam, S. S. S. Mousumi, A. S. A. Rabby, S. A. Hossain, and S. Abujar, "A potent model to recognize bangla sign language digits using convolutional neural network," *Procedia computer science*, vol. 143, pp. 611–618, 2018.

[18] S. A. Khan, A. D. Joy, S. Asaduzzaman, and M. Hossain, "An efficient sign language translator device using convolutional neural network and customized roi segmentation," in *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*. IEEE, 2019, pp. 152–156.

[19] A. Khatun, M. S. Shahriar, M. H. Hasan, K. Das, S. Ahmed, and M. S. Islam, "A systematic review on the chronological development of bangla sign language recognition systems," in *2021 Joint 10th International Conference on Informatics, Electronics & Vision (ICIEV) and 2021 5th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, 2021, pp. 1–9.

[20] I. Papastratis, K. Dimitropoulos, and P. Daras, "Continuous sign language recognition through a context-aware generative adversarial network," *Sensors*, vol. 21, no. 7, p. 2437, 2021.

[21] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[22] M. Garifulla, J. Shin, C. Kim, W. H. Kim, H. J. Kim, J. Kim, and S. Hong, "A case study of quantizing convolutional neural networks for fast disease diagnosis on portable medical devices," *Sensors*, vol. 22, no. 1, p. 219, 2021.

[23] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.

[24] M. R. Al Koutayni, V. Rybalkin, J. Malik, A. Elhayek, C. Weis, G. Reis, N. Wehn, and D. Stricker, "Real-time energy efficient hand pose estimation: A case study," *Sensors*, vol. 20, no. 10, p. 2828, 2020.

[25] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[26] I. R. Abdullah Al Jaid Jim, M. Z. Akon, and A.-A. Nahid, "Ku-bdsl: Khulna university bengali sign language dataset," 2021. [Online]. Available: https://data.mendeley.com/datasets/scpvm2nbkm/1