Development of M-Health Application with Seven-Segment Display Reading Feature from Glucose meter

Daffa Romyz Aufa, Rinaldi Munir, Nur Ahmadi, *Member, IEEE*

***Abstract*—M-Health application is useful for monitoring blood glucose conditions for diabetics. The entry of blood glucose values from a glucose meter into the application is generally done manually. This process is time-consuming and prone to error. Therefore, developing a prototype m-health application that uses a model to read blood glucose measurement results can be a solution. This paper proposes the development of an m-health (mobile health) application that has 4 features: taking a picture of the glucose meter with a phone camera, reading the glucose value with a model, storing the reading results in a database, and data visualization. The model was trained using 3764 images of blood glucose measurement results from glucose meters. The model ultilized is YOLO11 with small, medium, and large variations. In order to be used in the application, the trained model is converted into TensorFlow Lite format (.tflite). The TensorFlow Lite model is then quantized to 16-bit float precision (FP16) and 8-bit integer (INT8) to reduce inference time and model size. Based on the test results, the model chosen to be implemented in the application is the small model with INT8 precision. The model was chosen because it has a small inference time and file size while having an accuracy that are not too far from other model variants. The model has an accuracy of 94.14%, an f1-score of 97.03%, an inference time of 330.7 ms, and a file size of 11.4 MB. Model is tested on the m-health application with 156 images from the test data set resulted in an accuracy of 97.77%, an f1-score of 98.34%, and an average inference time of 1918.15 ms.**

***Index Terms*—Seven-segment display, mobile health, glucose meter, YOLO11, blood glucose, diabetes.**

I. Introduction

THE treatment given to people with diabetes is to control their blood sugar so that it has normal levels [1]. To keep blood sugar at a normal level, diabetics are asked to monitor their blood sugar with frequent blood sugar tests. Depending on the treatment provided, patients may be asked to test their blood sugar up to four times a day. The results of these blood glucose measurements should be recorded to ensure that blood glucose levels are within the target range and to determine the patient's blood glucose trend.

The m-health application is able to help diabetics by storing blood sugar measurement results and visualizing the data in the form of graphs. For this reason, the results of blood sugar measurements must be entered into the m-health application. Currently, there are glucose meter that use Bluetooth technology to send data to m-health application. However, only 1.4% of glucose meter recommended for personal use have Bluetooth features to send data to the application [2].

In general, the results of measuring blood sugar levels are still entered into the m-health application manually. This process is time-consuming and there is a possibility of entering the wrong blood glucose measurement results. Glucose meter with Bluetooth features generally have higher prices. Therefore, replacing a glucose meter by buying a new device that has a Bluetooth feature is inefficient. Therefore, solution is needed to automatically enter the blood glucose measurement results into the m-health application so that there is no need to replace the glucose meter.

Mobile phone camera can be used to capture the image of the blood glucose measurement results from the seven-segment display of the glucose meter. The image of the blood glucose measurement results can be read with an image recognition model. Thus, a model is needed that can read the blood glucose level measurement results from the image captured with a mobile phone camera. The model will be integrated with an m-health application that is able to capture images as input for the model and record the reading results.

The solution is suitable because diabetics who already use the m-health application also have a smartphone and glucose meter. Diabetics who have not used the m-health application can also use it immediately. This is because around 60% of the population in Indonesia has a smartphone [3]. This application also allows diabetics who already have a smartphone to buy a glucose meter that does not have Bluetooth technology because the price tends to be more expensive.

The paper is organized as follows. Section II presents the related works, research gap, and summary of our contribution in the present work. Section III describes alternative solutions that can be applied to solve the problem, the selection of alternative solutions and its justifications. Section IV details the implementation process of the proposed solution including model and application development. The result of model and application testing are presented and discussed in Section V. Finally, conclusions are drawn in Section VI.

II. Related Works

There has been much research exploring various methods to read the seven-segment display. Shenoy and Aalami developed

Daffa Romyz Aufa is with the School of Electrical Engineering and Informatics, Bandung Institute of Technology, 40132, Indonesia (e-mail: 13520162@std.stei.itb.ac.id).

Rinaldi Munir is with the School of Electrical Engineering and Informatics, Bandung Institute of Technology, 40132, Indonesia (e-mail: rinaldi@staff.itb.ac.id).

Nur Ahmadi is with the Center for Artificial Intelligence (U-CoE AI-VLB), School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, 40132, Indonesia (e-mail: nahmadi@itb.ac.id).

a smartphone-based application to read seven-segment display images from health monitors using machine learning [4]. The machine learning used in this study is a random forest classifier.

This application reads the seven-segment display by capturing the health monitor image and determining the box where the seven-segment display is located. Next, the application performs Otsu thresholding on the input image to convert the color image into a black and white binary image. Then, the image is segmented to divide the image into individual digits. Then, each digit will be classified using a random forest classifier.

The random forest classifier is chosen because it is fast and accurate. The random forest classifier is a combination of 100 decision tree classifiers that perform voting to determine the class of the digits so that it is not easy to overfitting. The model was able to read the digits from the health monitor with 98.2% accuracy and 97.8% f1-score. This random forest classifier has a lightweight file size of only 847 kB. The drawback of this application is that the user must specify the box where the seven-segment display is located. Blurry images cannot be read by the application and will ask the user to take a clearer photo. The image processing time is long because the model does not reside on the smartphone but on the server so it takes up to 2 to 3 seconds.

Finnegan *et al.* developed a model for detecting and reading seven-segment displays from glucose meter and blood pressure monitors [2]. The model is divided into two parts, namely digit location detection and digit classification. Digit location detection uses the Maximally Stable Extremal Regions (MSER) algorithm and looks for connected components in binary images. Digit classification uses a neural network model with features in the form of Histogram of Orientated Gradients (HOG).

Digit location detection is performed by preprocessing the input image with filtering and histogram equalization. Then, each digit segment is detected and stored as a blob using MSER and by searching for connected components in the binary image. Then, any blobs that are not segments are discarded by rule-based filtering and logistic classifier. A clustering algorithm is then used to combine the segments into one seven-segment digit. Next, each digit is passed to the digit classification model.

Digit classification is performed using a neural network using HOG features. The model is trained using binary images of digits. The model has three layers, namely input layer, hidden layer and output layer.

This model has an accuracy of 93%, an f1-score of 87% for the glucose meter, and an f1-score of 80% for the blood pressure monitor image. This model has the advantage of being able to detect digits so there is no need to manually determine the digit bounding box. The models for digit detection and classification are separate so that each model can be optimized. The drawback of this model is the lack of accuracy compared to the random forest classifier and end-to-end CNN. The digit classification model depends on the digit detection model. If the digit detection model cannot identify digits properly, the classification model will not be able to classify them.

Moreira developed an end-to-end model to detect and read seven-segment displays from medical devices using deep learning models that can detect objects [5]. The deep learning models used in this research are EfficientDet and EfficientDet-lite which have been previously trained with the MS-COCO dataset. Both models is fine-tunined with images of medical devices that have seven-segment displays. The images have bounding boxes and labels for each digit.

The model rescales the input image into a square shape. The image is not stretched, but pixels with zero value are added to the sides of the image to make it square. Then, the pixel values of the image are normalized using the mean and standard deviation. The normalized image is then used to train the model.

The model output is the bounding boxes of digit along with coordinates, classification, confidence score. There are several digit bounding boxes that could point to the same digit. Therefore, only the one with the highest confidence score is kept.

This model has the advantage of having light preprocessing and postprocessing. The EfficientDet-lite1 model has 100% precision and 99.7% f1-score in digit detection and has 100% accuracy in digit classification. The image processing time is very fast at only 49 milliseconds. The drawback of this model is its size of 5.8 MB. The size of this model is almost 7 times larger than the random forest classifier.

All of the above studies did not have a method to determine the blood glucose unit used by the glucose meter. glucose meter displays the measurement results in two units: mg/dL and mmol/L [6]. These two units have different value ranges so they need to be differentiated in their readings. Storage of measurement results must also be able to handle unit differences. Data visualization also needs to have a function to be able to display graphs in both units.

In this paper, a prototype m-health application is developed with the capability of reading the results of the blood glucose level measurement represented in seven-segment display from the image captured by the cell phone camera. The reading is done using YOLO11, a convolutional neural network deep learning model. The model was designed so as to be able to distinguish between the blood glucose units used. The difference between the two units is the presence of a comma in the mmol/L unit. Therefore, the reading model is able to detect the comma along with other measurement numbers. The model will be implemented directly on the mobile device. Thus, model quantization techniques are performed to reduce the computational and memory burden on such devices with limited resources.

III. Solution Analysis

There are several approaches to reading numbers on seven-segment displays: machine learning and deep learning. The machine learning approach consists of two steps: feature extraction and classification. The feature extraction step is a step to segment each number and produce the features of each number. Each number feature will then be classified using a machine learning model to produce the value for each number.

The deep learning approach is able to read numbers end-to-end because the model can learn image features and perform classification on the same model.

Based on research conducted by Moreira, the deep learning approach produces a model with higher accuracy than the machine learning approach [5]. The deep learning model used is convolutional neural network (CNN). The CNN model is suitable for reading images because it could receives input in the form of a matrix. Image is a matrix of pixels.

The CNN model architecture that will be used is YOLO because it can predict quickly and accurately. YOLO (You Only Look Once) is a trained CNN model for object detection developed by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in 2015. YOLO detects objects through a single regression approach to generate bounding boxes and class probabilities [7]. YOLO's ability to detect and recognize objects from whole images in one evaluation is the reason this model is called You Only Look Once.

YOLO has several advantages over other object detection models. First, YOLO has a very high speed because it detects using regression approache so that it can make predictions on real-time video input. Secondly, YOLO analyzes the full image during training and testing so that it is able to implicitly recognize contextual and appearance information about the class. This prevents YOLO from identifying the background as an object because YOLO is able to see the image as a whole. Thirdly, YOLO learns general representations of objects, making it easy to apply to new domains. However, YOLO has a disadvantage in accuracy compared to other object detection models. In addition, YOLO has difficulty in detecting objects that are small in size.

YOLO11 is a YOLO model introduced by Ultralytics on September 30, 2024. YOLO11 has accuracy and speed performance built on the latest advances in deep learning and computer vision [8]. The advancements of YOLO11 from previous versions are the improved backbone and neck architecture for better feature extraction, more efficient refined architectural designs and optimized training pipelines, higher accuracy with fewer parameters, ability to be implemented in various devices, and model support in performing various computer vision tasks. The YOLO11 architecture has better accuracy and inference time than the previous YOLO model.

Mobile phones have several operating systems that can be used for the development of m-health applications such as Android and iOS. Android was chosen as the operating system in this development of m-health applications. This is because Android has a market share in Indonesia of 93.47% in December 2024 [9]. Thus, the number reader model on the seven-segment display developed must pay attention to the computational load that can be handled by Android phones. The developed m-health application has four main features: image capture from camera, number reading using model, saving result to database, and data visualization.

IV. Implementation

The implementation of the m-health application is divided into two parts, namely model development and application development. Model development includes dataset collection, model training, model conversion and model quantization. The implementation process is carried out on two working environments. The first environment is utilized for model training and testing. The second environment is carried out for the development and testing of m-health applications. The working environments is as follows.

1) Model Development
   - Google Colab Pro
     - GPU : NVIDIA® T4
     - System RAM : 12.7GB
     - GPU RAM : 15.0GB
     - Disk : 201.2GB
2) Application Development
   - Laptop (Asus A412F)
     - CPU : Intel® Core™ i7-10510U CPU @ 1.80GHz 2.30 GHz
     - RAM : 8GB
     - OS : Windows 11 23H2
   - Smartphone (Oppo A54)
     - Processor : MediaTek MT6765V/CB
     - RAM : 6GB
     - OS: ColorOS V11.1

A. Dataset Collection

The data is collected from public image datasets. The dataset used is an image of blood glucose measurement results from glucose meters that has a seven-segment display screen. The image dataset has variations in lighting, distance, and viewing angle so that it reflects the real detection environment. Details about the data source can be seen in Table I.

The acquired images are first cleaned to remove images that are not good for model training. The images that is not good includes images that use numbers that are not seven-segment displays, too tilted, too far away, or too difficult to read even by humans. The glucose meter shows the measurement value in one of two types of units, namely the mg/dL device and the mmol/L device. The distribution of unit types in the image dataset is shown in Table II.

TABLE I
Number of images from the data sources

Source	Amount	Percentage
Datacluster Labs	223	14.22%
Workspace	1345	85.78%
Total	1568	100.00%

TABLE II
Distribution of unit types

Unit	Amount	Percentage
mg/dL	1103	70%
mmol/L	465	30%
Total	1568	100%

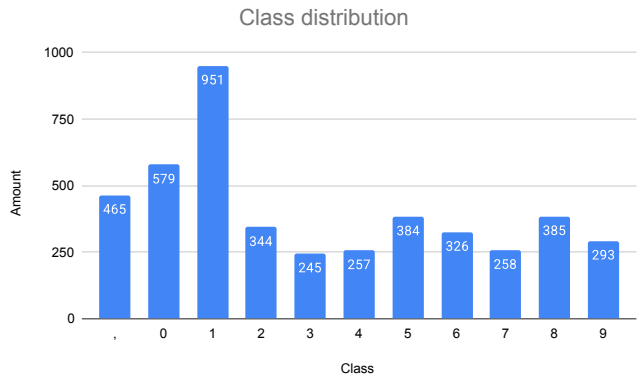

Fig. 1. Class distribution

TABLE III
DATASET SPLIT

Split	Amount	
	Before Augmentation	After Augmentation
Train	1098 (70 %)	3294 (88%)
Validation	314 (20%)	314 (8%)
Test	156 (10%)	156 (4%)
Total	1568 (100%)	3764 (100%)

The image is then annotated using the Roboflow application. Annotation is in the form of giving bounding boxes and classes to each object in the image. The image can contain several objects in the form of numbers and commas. The objects are grouped into 11 classes: ",", "0", "1", "2", "3", "4", "5", "6", "7", "8", and "9". Images that have comma object are images that use mmol/L units. Meanwhile, images that do not have comma object are images that use mg/dL units. The Class distribution is as shown in Figure 1.

The image is then divided into three sets: train, validation, and test. The training set is used in model training as a data source. The validation set is used for validation during training. The testing set is used in testing the model. The training data is augmented to increase the number and variety of images. Data augmentation was performed in the form of rotation by ±15°, shear by ±15° horizontally and ±15° vertically, brightness by ±15%, and blur up to 2.5 pixels. Augmentation resulted in three augmented images from one training set image. The dataset splits before and after the augmentation process can be seen in Table III.

B. Model Training

The YOLO11 model has several model variants: nano (YOLO11n), small (YOLO11s), medium (YOLO11m), large (YOLO11l), extra large (YOLO11x). The comparison between the five models is shown in Table IV. The training process is carried out on small, medium, and large variants. The nano model is not trained because it has a very low mAP value compare to other models even though it has a small size. The extra-large model is not trained because it has a mAP that is not much larger than the large model despite being twice the size.

TABLE IV
COMPARISON OF YOLO11 VARIANTS

Model	Size (pixels)	mAP Val 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	Params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x large	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

The three model variants are trained with the following hyperparameters.

- Epoch : 1000
- Patience : 100
- Image size : 640
- Optimizer : Auto (SGD)
- Learning rate : 0.01
- Momentum : 0.9

The hyperparameter optimized in this training process is the epoch value. Other hyperparameters use the default values provided by the Ultralytics library. This is done to simplify the training process and reduce hyperparameter tuning. The epoch and patience work together in the early stopping mechanism. The epoch value is the maximum number of iterations the model performs training using the entire dataset. The patience value is the limit on the number of epochs that must be passed without any increase in the validation metric. If the number of epochs exceeds the limit, training will be stopped (early stopping). The patience value is useful for preventing overfitting when training process does not experience an increase in the validation metric.

The epoch value is chosen with a value that is much higher than the patience value. The default patience value from the library is 100 and the selected epoch value is 10 times the patience value, which is 1000. This is done so that training process only stops when the validation metric stops increasing for a number of epochs that is determined by the patience value. The expectation of this approach is that the resulting model has the most optimal validation metric value because continuing training after that epoch does not result in an increase in the validation metric.

The hyperparameter value for the input image size is 640. This value was chosen because the models are pre-trained models that have been trained using an input image size of 640. Using the same input image size value is done to minimize the difference between the data that the model already knows and the data used in this training. The expectation of this approach is to minimize the weight changes that must be made in training to achieve optimal metrics.

The optimizer selection is done automatically by the library. In addition to determining the optimizer used, the library also automatically determines the best learning rate and momentum values. The optimizer selected by the library is Stochastic

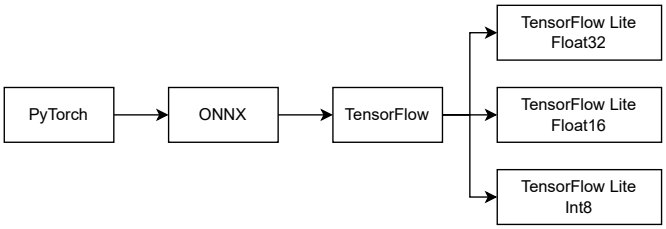

Fig. 2. Conversion and quantization process

Gradient Descent (SGD) with a learning rate of 0.01 and a momentum of 0.9.

The training process produces two models, namely the last epoch model and the best epoch model. The last epoch model is the model at the time the last epoch was run before the training process stopped. The best epoch model is the model at which the best validation metric was obtained.

C. Model Conversion and Quantization

The trained model has a different format from the format that can be used by the m-health application. Therefore, the model will be converted to that format. The converted model is then quantized into a model with a smaller bit precision. Quantization is done to reduce the number of bits needed to store the model weights so that the file size becomes smaller. This can reduce the model computational load so they are suitable for mobile phones with limited resources. However, quantization can cause a reduction in model accuracy.

The training process produces a model file that has the PyTorch (.py) format. Meanwhile, the model used in the m-health application must be in TensorFlow Lite (.tflite) format. Therefore, the model is converted to ONNX format first. Then, the ONNX format model is converted to TensorFlow format. Then, the Tensorflow format model is converted to TensorFlow Lite format. In this conversion, model quantization can be carried out from 32-bit float (FP32) precision to 16-bit float (FP16) and 8-bit integer (INT8). The conversion and quantization process is as shown in Figure 2.

This conversion and quantization process produce three files for each model: models with FP32, FP16, and INT8 precision. The FP32 precision model is the default model that stores weights in 32-bits. Meanwhile, the FP16 precision model is a quantization model that cuts the number of bits from 32-bits to 16-bits. The INT8 precision model cuts the number of bits from 32-bits to 8-bits and converts the weights and activation outputs into integers.

D. Application Development

The m-health application will use the previously converted model to perform reading of seven-segment display. The m-health application developed has four main features: taking images with a camera, reading images, storing reading results to a database, and data visualization. In addition, there are three additional features: manual value entry, taking images from internal storage, and displaying list of all saved data.

This m-health application will have 7 use cases as shown in Figure 3. The first three use cases are three ways that users

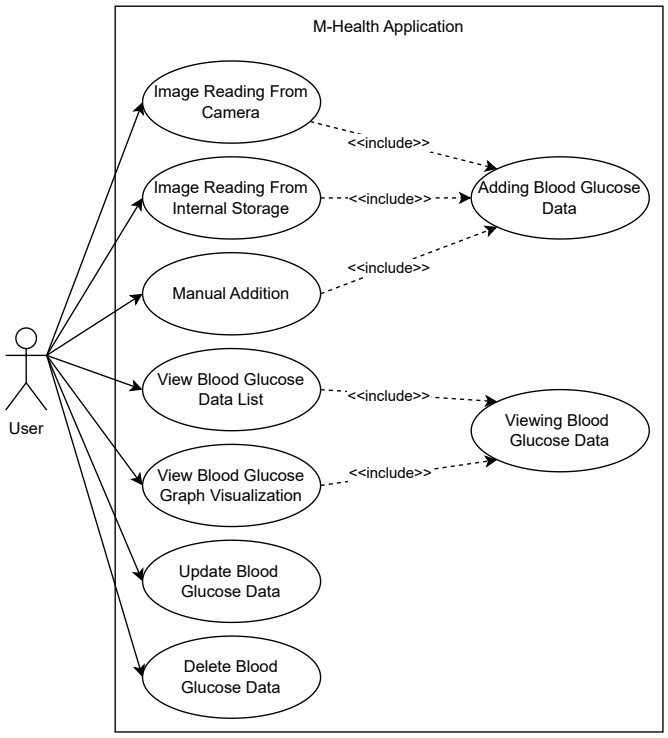

Fig. 3. Use case diagram of the m-health application

can add blood sugar data to the database. In the use case of reading images from the camera, users must use the mobile phone camera to get an image of a glucose meter. In the use case of reading images from storage, users must search for an image of a glucode meter from the mobile phone's internal storage. The image from both methods will be read by the model to get the blood sugar values. The blood sugar values are then inserted into the database. In addition to adding data through image reading, users can also add blood sugar data manually as explained in the manual addition use case.

The data that has been saved to the database can be displayed to the user through two use cases: the use case of viewing a list of blood sugar data and the use case of viewing a visualization of a blood sugar graph. The use case of viewing a list of blood sugar data displays all blood sugar data in the form of a list. The use case of viewing a visualization of a blood sugar graph displays all blood sugar data in the form of a bar graph, line graph, or pie graph. In the use case of updating blood sugar data, the user can make changes to the time of collection and blood sugar values. In the use case of deleting blood sugar data, the user is able to delete certain data from the database.

The workflow of reading the blood glucose measurement is as shown in Figure 4. The process of reading seven-segment display numbers is divided into two steps: the number identification process and the unit classification process. The number identification process aims to detect all individual numbers and commas in the image. The unit classification process will identify the units used by the glucose meter based on the presence of a comma symbol.

The presence of a comma indicates that the unit used is

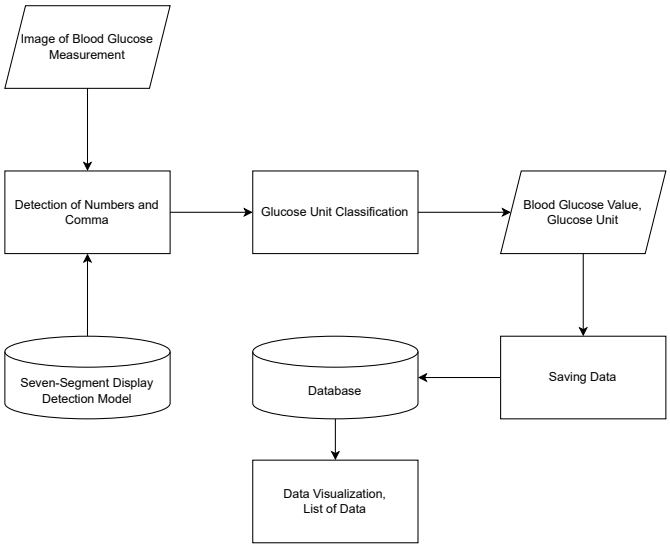

Fig. 4. Workflow of reading blood glucose measurement

mmol/L. If there is no comma, then the unit used is mg/dL. The reading process results are blood glucose values and glucose units. Then, the reading results are inserted into the database. Blood glucose values are stored in the database in mg/dL units. Therefore, glucose values in mmol/L units must first be converted to mg/dL units. If glusose unit used is mmol/L, the blood glucose value will be multiplied by 18.0156 before being entered into the database.

Blood glucose data that has been stored in the database will be used to create data visualizations for users. When the user opens the dashboard page. Then, the application will take blood glucose data to create bar charts, line charts, or pie charts.

The m-health application has five pages, namely the dashboard page, the camera page, the manual data addition page, the data list page, and the edit page. The user interface of the five pages is displayed in Figure 5. The dashboard page contains data visualization in the form of graphs. Data can be visualized in the form of bar graphs, line graphs, and pie charts by selecting the graph option. There is a button to change the units used by the graph. This display the graph in either the mg/dL or mmol/L unit.

The camera page serves to read numbers using images from the camera. In addition to images from the camera, this page provides the option to use images from the phone's internal storage. The camera page consists of a PreviewView to show images from the camera, an OverlayView to show the bounding box above the PreviewView, two TextViews to show measurement values and inference time, and a button with the words "Save Value" to save the measurement value to the database. There is a button above the PreviewView with the words "Open Gallery" to retrieve image from internal storage.

The manual data addition page contains an EditText to enter blood glucose measurement values and a RadioGroup to select the glucose units used. The "Save Value" button will save the data to the database. This page is useful for entering data if the image reading on the camera page is unsuccessful.

TABLE V
EVALUATION METRICS OF MODEL TRAINING

Metric	YOLO11s	YOLO11m	YOLO11l
Accuracy (%)	96.67	96.67	**97.77**
Precision (%)	97.90	98.31	**98.82**
Recall (%)	98.32	98.32	**98.77**
F1-score (%)	98.07	98.29	**98.77**
Inference time (ms)	**11.00**	20.20	25.80
File size (MB)	**18.30**	38.70	48.90

The data list page shows all stored blood glucose data. The data list is implemented into a RecyclerView. Each element has a glucose value in mg/dL units, a glucose value in mmol/L units, the date and time of measurement, the class of the glucose value, a button to delete data, and a button to edit data. The edit data button functions to open the edit page.

The edit page contains two EditTexts, one RadioGroup, and a Button. The first EditText is a place to enter the time and date of the blood glucose measurement. The second EditText contains the blood glucose value in mg/dL units. The RadioGroup contains two elements that are the units of the blood glucose value. The "Save Value" button will change the glucose measurement on the data that has been selected on the edit page.

V. RESULTS AND DISCUSSION

A. Evaluation of Model Training

Evaluation on the trained model is done to obtain the control metric values that will be compared with the metrics of the converted and quantized model. The results of the model testing produce metrics of accuracy, macro-averaged precision, macro-averaged recall, macro-averaged f1-score, inference time, and model file size. The evaluation metrics of the YOLO11 trained model are as shown in Table V.

In general, more complex and larger models will have higher performance. However, larger models also have larger inference times and file sizes. Among the three trained models, the YOLO11l model has the highest accuracy, precision, recall, and f1-score. Meanwhile, the YOLO11s model has the smallest inference time and file size. There is no significant difference in the accuracy, precision, recall, and f1-score metrics of the three models. The significant differences are in the inference time and file size metrics.

B. Evaluation of Model Conversion and Quantization

Evaluation on the conversion and quantization result models is done to select the model to be implemented in the m-health application. The resulting metrics are compared with the base model to identify and measure the differences in metric values due to the conversion and quantization process. The evaluation metrics of the YOLO11 FP32 model resulting from conversion and quantization is displayed in Table VI. Based on the test results of the YOLO11 FP32 model, there is no difference in the accuracy, precision, recall, and f1-score metrics between the base and FP32 models. However, there is a significant difference in the inference time. The increase in inference

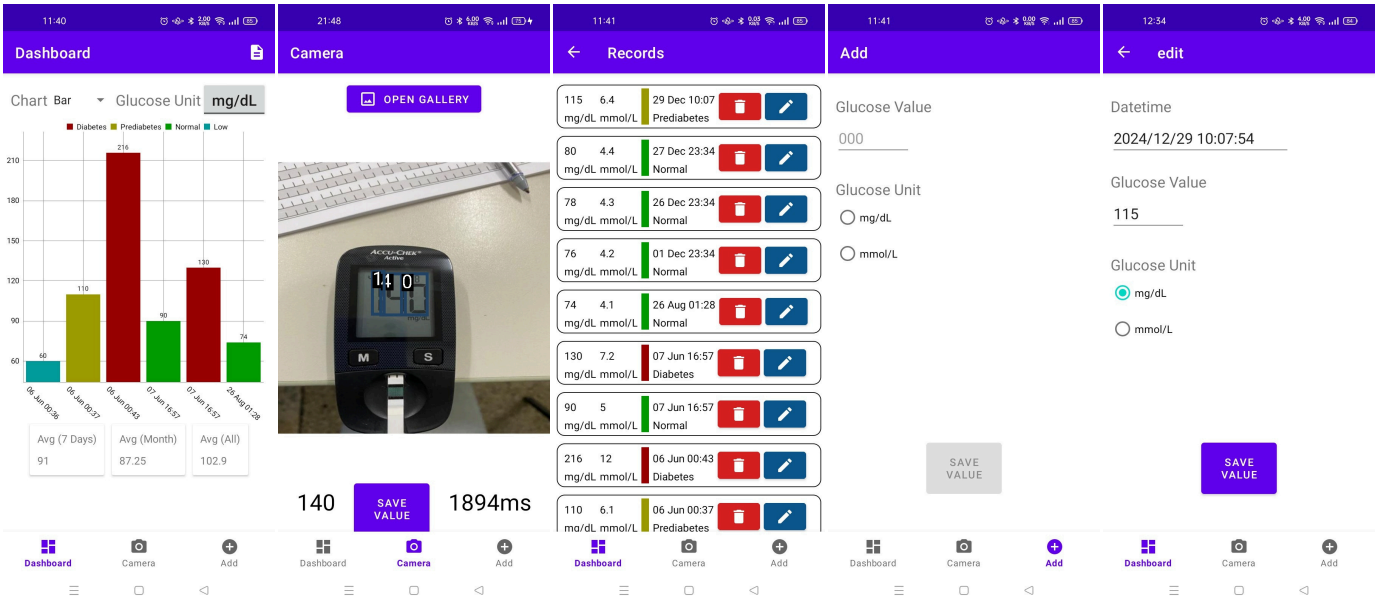

Fig. 5. User interface of the m-health application

time in the FP32 model is due to the file format conversion from PyTorch (.py) to TensorFlow Lite (.tflite). The file size in the FP32 model has increased by about 100% from the base model. This is because the PyTorch model stores weights and activation outputs in 16-bit precision. Therefore, the file size doubled when converting from 16-bit precision to 32-bit precision. Among the three FP32 models, the YOLO11l FP32 model has the highest accuracy, precision, recall, and f1-score. Meanwhile, the YOLO11s FP32 model has the smallest inference time and file size.

The evaluation metrics of the YOLO11 FP16 model resulting from conversion and quantization is displayed in Table VII. The test results of the FP16 precision YOLO11 model are similar to the FP32 model. There is no difference in the accuracy, precision, recall, and f1-score metrics between the base and FP16 models. There is a significant difference in the inference time due to the change in file format from PyTorch (.py) to TensorFlow Lite (.tflite). The file size of the FP16 model is almost the same as the baseline model because the base model stores weights and activation outputs in 16-bit precision. Among the three FP16 models, the YOLO11l FP16 model has the highest accuracy, precision, recall, and f1-score. Meanwhile, the YOLO11s FP16 model has the smallest inference time and file size.

The evaluation metrics of the YOLO11 INT8 model resulting from conversion and quantization is displayed in Table VIII. Based on the test results of the YOLO11 INT8 precision model, there is a significant increase in the file size metrics for the medium and large models. The increase in file size in the YOLO11 INT8 medium and large models is due to changes in file format. The file size in the YOLO11s INT8 model has decreased by about 50% from the base model. This is because the PyTorch model stores weights and activation outputs in 16-bit precision. Therefore, the file size halved when converting from 16-bit precision to 8-bit precision. There has been a slight decrease in the accuracy, precision, recall, and f1-score metrics between the base and INT8 models. This is due to some information loss when reducing the model precision from 16-bit to 8-bit. Among the three INT8 models, the YOLO11l INT8 model has the highest accuracy, precision, recall, and f1-score. Meanwhile, the YOLO11s INT8 model has the smallest inference time and file size.

The YOLO11 model converted and quantized to FP32 and FP16 precision has no difference in accuracy, precision, recall, f1-score, and inference time metrics. Therefore, the FP16 precision model is preferred over FP32 because it has the smaller inference time and file size. The three FP16 models have differences in accuracy, precision, recall, and f1-score metrics that are not too far apart.

The significant difference is in the inference time and file size metrics. Inference time determines the computational load on the application and file size increases the size of the application. These two metrics are important considerations for mobile devices that have limited resources. Therefore, the model chosen in the FP16 model is the small model.

There is no very significant difference in accuracy, precision, recall, and f1-score metrics between the FP16 and INT8 model. Thus, the INT8 model is preferred over the FP16 model because of the smaller inference time and file size. The differences in accuracy, precision, recall, and f1-score metrics between INT8 models are also not significant. Therefore, the YOLO11s INT8 model was chosen because it has the smallest inference time and file size.

Comparison of the YOLO11s INT8 model with related studies is displayed in Table IX. In general, the YOLO11s INT8 model has been able to detect seven-segment display numbers accurately. The model has higher accuracy metrics and f1-scores than the research of Finnegan *et al.* However, the accuracy and f1-score of the model are smaller than the research of Shenoy and Aalami and the research of Moreira.

TABLE VI
EVALUATION METRICS OF YOLO11 FP32

Metric	YOLO11s			YOLO11m			YOLO11l		
	Base	FP32	Delta (%)	Base	FP32	Delta (%)	Base	FP32	Delta (%)
Accuracy (%)	96.67	96.67	0.00	96.67	96.67	0.00	97.77	**97.77**	0.00
Precision (%)	97.90	97.90	0.00	98.31	98.31	0.00	98.82	**98.82**	0.00
Recall (%)	98.32	98.32	0.00	98.32	98.32	0.00	98.77	**98.77**	0
F1-score (%)	98.07	98.07	0.00	98.29	98.29	0.00	98.77	**98.77**	0.00
Inference time (ms)	11.00	**431.50**	+3822.73	20.20	1260.40	+6139.60	25.80	1649.50	+6293.41
File size (MB)	18.30	**36.30**	+98.36	38.70	77.40	+100.00	48.90	97.60	+99.59

TABLE VII
EVALUATION METRICS OF YOLO11 FP16

Metric	YOLO11s			YOLO11m			YOLO11l		
	Base	FP16	Delta (%)	Base	FP16	Delta (%)	Base	FP16	Delta (%)
Accuracy (%)	96.67	96.67	0.00	96.67	96.67	0.00	97.77	**97.77**	0.00
Precision (%)	97.90	97.90	0.00	98.31	98.31	0.00	98.82	**98.82**	0.00
Recall (%)	98.32	98.32	0.00	98.32	98.32	0.00	98.77	**98.77**	0.00
F1-score (%)	98.07	98.07	0.00	98.29	98.29	0.00	98.77	**98.77**	0.00
Inference time (ms)	11.00	**410.90**	+3635.45	20.20	1279.90	+6236.14	25.80	1627.90	+6209.69
File size (MB)	18.30	**18.30**	0.00	38.70	39.20	+1.29	48.90	49.40	+1.02

TABLE VIII
EVALUATION METRICS OF YOLO11 INT8

Metric	YOLO11s			YOLO11m			YOLO11l		
	Base	INT8	Delta (%)	Base	INT8	Delta (%)	Base	INT8	Delta (%)
Accuracy (%)	96,67	94,14	-2,62	96,67	93,10	-3,69	97,77	95,00	-2,83
Precision (%)	97,90	96,14	-1,80	98,31	96,09	-2,26	98,82	96,83	-2,01
Recall (%)	98,32	98,01	-0,32	98,32	97,59	-0,74	98,77	98,41	-0,36
F1-score (%)	98,07	97,03	-1,06	98,29	96,82	-1,50	98,77	97,59	-1,19
Inference time (ms)	11	330,7	+2906,36	20,2	945,3	+4579,70	25,8	1226,2	+4652,71
File size (MB)	18,3	11,4	-37,70	38,7	93,5	+141,60	48,9	125,7	+157,06

TABLE IX
COMPARISON OF YOLO11S INT8 WITH RELATED STUDIES

Research	Model/Algorithm	Accuracy (%)	F1-score (%)
Shenoy and Aalami (2016)	Random Forest	98,20	97,80
Moreira (2022)	CNN	100,00	99,70
Finnegan et al. (2019)	ANN	93,00	87,00
Proposed model	YOLO11s INT8	94,14	97,03

This is due to the difference in objects that can be detected by the model of this research with the model of related studies. The model in the related studies is only able to detect numbers, while the model in this research is able to detect commas. The ability to detect commas causes a decrease in accuracy metrics and f1-scores because the size of the comma is quite small so that it is difficult for the model to detect. This is indicated by the difference in f1-score for commas with the average f1-score, which are 84.09% and 97.03%. In addition, there are differences in the data sets used in this research with the related studies so that the differences in accuracy metrics and f1-score cannot be compared directly.

C. Evaluation of M-Health Application

The YOLO11s INT8 model was selected for use in the m-health application. The m-health application testing was carried out by reading 156 images from the test dataset. The images have various variations, such as mg/dL images, mmol/L images, tilted images, distant images, and blurry images. The sample results of the m-health application testing are is displayed in Figure 6.

From the test results, an accuracy metric of 97.77% and an f1-score of 98.34% were obtained. The average inference time for one image is 1918.15 ms. The inference time on the application increased by 5 times from the model testing results. This is due to the difference in the model testing environment and application testing. In addition, the smartphone used in the application testing has a GPU that is not supported by the library used. The inference computational load is given to the cellphone CPU which causes slower inference time.

VI. CONCLUSION

In this study, a m-health application that is able to read the blood glucose value and determine the glucose unit from a blood glucose measurement of a glucose meter has been successfully designed, implemented and tested with a good accuracy. The application use YOLO11, a convolutional neural network model. The model has been trained, converted to TensorFlow Lite format, and quantized to 16-bit and 8-bit precision. The model used in the application is YOLO11s

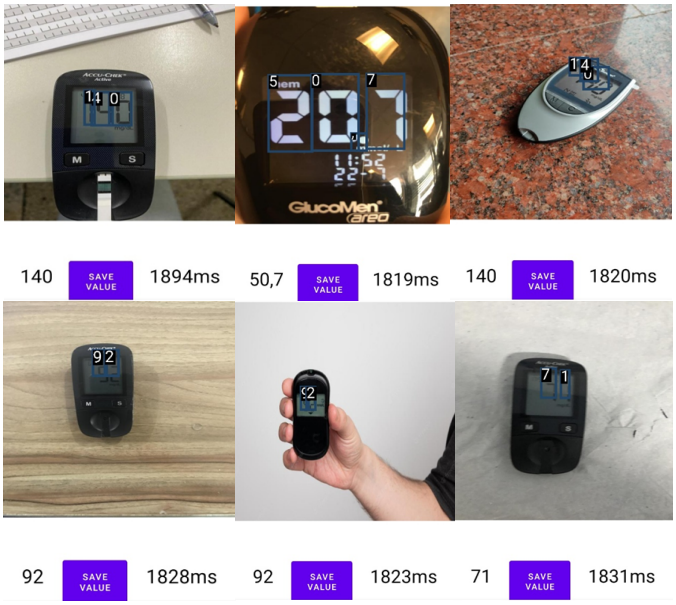

Fig. 6. Sample results of the m-health application testing

INT8. The model has an accuracy of 94.14% and an f1-score of 97.03%. The model has a small file size of 11.4 MB, making it suitable for mobile devices. The model has an average inference time of 330.7 ms on Google Colab and 1918.15 ms on smartphone. Through testing with 156 images, the application produced an accuracy of 97.77% and an f1-score of 98.34%.

REFERENCES

[1] Mayo Clinic, "Diabetes : Diagnosis treatment," 2024, accessed: 2025-01-22. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/diabetes/diagnosis-treatment/drc-20371451

[2] E. Finnegan, M. Villarroel, C. Velardo, and L. Tarassenko, "Automated method for detecting and reading seven-segment digits from images of blood glucose metres and blood pressure monitors," *Journal of medical engineering & technology*, vol. 43, no. 6, pp. 341–355, 2019.

[3] Statista, "Number of smartphone users in indonesia from 2019 to 2029," 2024, accessed: 2025-01-22. [Online]. Available: https://www.statista.com/forecasts/266729/smartphone-users-in-indonesia

[4] V. N. Shenoy and O. O. Aalami, "Utilizing smartphone-based machine learning in medical monitor data collection: Seven segment digit recognition," in *AMIA Annual Symposium Proceedings*, vol. 2017, 2018, p. 1564.

[5] L. P. Moreira, "Automated medical device display reading using deep learning object detection," *arXiv preprint arXiv:2210.01325*, 2022.

[6] J. Sanchez, *Glucose Meters and Strips*. New York, NY: Springer New York, 2013, pp. 870–871. [Online]. Available: https://doi.org/10.1007/978-1-4419-1005-9_1191

[7] J. Redmon, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[8] Ultralytics, "Ultralytics yolo docs : Home," 2024, accessed: 2025-01-22. [Online]. Available: https://docs.ultralytics.com/

[9] Statcounter, "Mobile operating system market share indonesia," 2024, accessed: 2025-01-22. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/indonesia

Daffa Romyz Aufa currently pursuing B.Eng. degree in informatics engineering from Bandung Institute of Technology (ITB), Indonesia. His research interests include web development, data engineering, machine learning, computer vision, and software engineering.

Rinaldi Munir received the B.Eng. degree in informatics engineering and the M.Sc. degree in digital image compression from the Bandung Institute of Technology (ITB), Indonesia, in 1992 and 1999, respectively. He received his Ph.D. degree in image watermarking from the School of Electrical Engineering and Informatics, ITB, in 2010. In 1993, he started his academic career as a Lecturer with the Department of Informatics, ITB. He is currently an Associate Professor with the School of Electrical Engineering and Informatics, ITB, and the Informatics Research Group. His research interests include cryptography and steganography-related topics, digital image processing, fuzzy logic, and numerical computation.

Nur Ahmadi (Member, IEEE) received the B.Eng. degree in electrical engineering from Bandung Institute of Technology (ITB), Indonesia, in 2011 and M.Eng. degree in communication and integrated systems from Tokyo Institute of Technology, Japan, in 2013. He received his Ph.D. degree in Electrical and Electronic Engineering from Imperial College London, UK, in 2020. His Ph.D. research focused on signal processing and deep learning for intracortical brain-machine interfaces. He is now with the Center for Artificial Intelligence and School of Electrical Engineering and Informatics, Institut Teknologi Bandung. His current research interests include biomedical signal processing, artificial intelligence, machine learning, digital and embedded systems.