EVOLUTION IN ELECTRICAL AND ELECTRONIC ENGINEERING
e-ISSN: 2756-8458

EEEE

Vol. 5 No. 1 (2024) 292-297
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Beef and Pork Meat Classification using MobileNet V2 Algorithm via Android Smartphone Application

Alif Fazhan Abdol Rahman [1], Siti Zarina Mohd Muji [1]*, Chessda Uttraphan Eh Kan[1]

[1] *Department of Electronic EngineeringFaculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, MALAYSIA*

*Corresponding Author: szarina@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2024.05.01.037

Article Info

Received: 11 January 2024
Accepted: 01 March 2024
Available online: 30 April 2024

Keywords

MobileNet V2, Android Studio, Beef, Pork, Epochs

Abstract

Each type of animal meat exhibits distinct color and texture characteristics. For instance, beef typically presents a dark red hue with a chewy texture, while pork displays a paler red color with a smoother fiber. Previous research has utilized methods such as the gray level co-occurrence matrix (GLCM), hue saturation value (HSV), and color intensity for meat classification. In this study, we employed a MobileNet-V2 implemented in a Jupyter notebook, utilizing the MobileNetV2 model, to classify beef and pork meat. The dataset comprised 488 of each meat image after the augmentation process, partitioned into training (70%), testing (20%), and validation (10%) sets. Before partitioning, images were resized to 128×128 pixels. The model was trained using the training dataset with 100 epochs and the Adam optimizer, resulting in an accuracy of 96.93%.

1. Introduction

Due to the high price of beef, some vendors have begun to cheat consumers by selling beef with cheaper pork. In 2021, there are cases in Jakarta, Indonesia where the seller deceives consumers by selling pork, which they claim is beef [1]. In Muslim-majority countries, it's not allowed to eat food with pork. Pork and beef look different—pork is paler and smoother, while beef is brighter and rougher [2]. The beef and pork issue underscores the importance of ensuring the integrity of meat products in the market. Instances of deceptive practices, such as selling pork as beef due to cost considerations, not only impact consumer trust but also raise concerns about adherence to dietary restrictions, particularly in Muslim-majority countries where pork consumption is prohibited. Addressing this issue requires innovative solutions that leverage technology, such as the implementation of advanced algorithms like Convolutional Neural Networks (CNN) in mobile applications. By tackling the beef and pork issue, we aim to provide consumers with a reliable tool to distinguish between these meats accurately and uphold transparency in the food industry. We have technology that can tell them apart using pictures. One way is using deep learning, specifically Convolutional Neural Network (CNN), which works better than other methods. This research suggests a solution: using a CNN in an Android app to identify pork and beef through image classification [3].

This study's goals are to develop an algorithm for meat classification and design a graphical user interface (GUI) for the system that has been created [4]. By creating an Android app using MobileNet V2 for beef and pork meat classification. The objectives include training the dataset in Google Colab, saving it as a TFLite file, and integrating it into an Android Studio application. The ultimate aim is to provide users with a simple tool for distinguishing between beef and pork meats through their smartphones.

Evolution in Electrical and Electronic Engineering Vol. 5 No. 1 (2024) p. 292-297

1.1 MobileNet-V2

This experiment on beef and pork classification uses MobileNetV2 via an Android smartphone application to train the model on a broad dataset of labeled photos of both beef and pig. Taking use of MobileNetV2's lightweight and efficient architecture, I used transfer learning to fine-tune the pre-trained model for my specific dataset. This method allowed the model to learn the distinct properties of beef and pork, resulting in reliable categorization findings.

Following model training, the MobileNetV2 model was effortlessly integrated into the Android application via TensorFlow Lite for efficient deployment. The application's intuitive interface allows users to effortlessly take or select meat photographs from their device's gallery. Following image selection, the application sends the image to the MobileNetV2 model for inference, which swiftly analyses the image and returns a classification result indicating whether the meat is beef or pork. This user-friendly interface allows for easy and precise meat classification using a smartphone.

2. Materials and Methods

In this section, 2 stages lead to the success of the work by acknowledging each system development and application development.

2.1 System Overview

In this system overview, illustrated in Fig. 1, the workflow is divided into system development and application development. The initial component involves utilizing an Android smartphone for data upload and collection. Visual data is pre-processed, encompassing scaling, augmentation, data separation, and normalization, to prepare it for deep learning. This includes shrinking images to an appropriate pixel size and employing techniques like rotation, flipping, and skewing to augment the training dataset for robust beef and pork classification systems [5]. The preprocessed images are then utilized in training the MobileNetV2 model and formulating the feature extraction approach. To implement the model on a smartphone, Android Studio, and TensorFlow Lite are employed, contributing to the development of the application system.

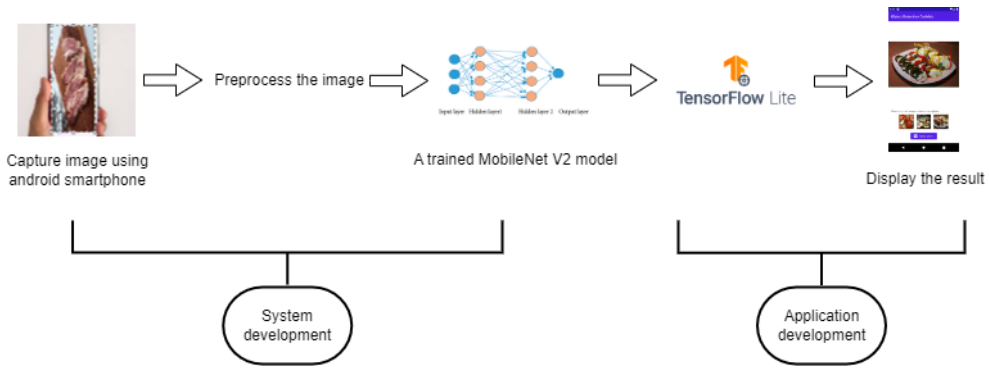

Fig. 1 *System development and Application development process*

2.2 System development

2.2.1 Training dataset

Fig. 2 shows the process of training, validation, and testing the dataset. The training data is divided into two sections. We use 10% of the data (57 images) to carefully check how well the final model matches the training data, while the remaining 70% (488 images) is employed to teach the dataset and improve accuracy. The dataset used for this work was sourced from Kaggle and subjected to an augmentation process to enhance its diversity and improve the robustness of the model's training

2.2.2 Data Augmentation

Fig. 3 shows the process of the augmentation process to increase the value of the dataset because it needs a large dataset to be trained. It is to increase the performance throughout the training phase [6]. During the augmentation process, the images underwent rotations (20 degrees), flips (horizontal), shears (20%), and zoom (20%) until normalization was applied to all of them.

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)

Found 40 images belonging to 2 classes.

validation_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)
```

Fig. 2 *Process to train and test dataset*

```
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
```

Fig. 3 *The pre-processing of images*

2.2.3 Epochs training and save to Tf.lite file

The Mobile Net V2 model was imported because this work employs a pre-trained model for transfer learning. The findings show that the model obtained 96.93% with 100 epochs after training as shown in Fig 4. The model weight and architecture are saved once the confusion matrix has reviewed the performance test data. TensorFlow Lite converts the model into a (Tf.lite) file for deployment in smartphone hardware as shown at Fig 5. Tf.lite file is very important to upload in the Android studio which is needed to complete the application development process.

```
Epoch 98/100
8/8 [==============================] - 9s 1s/step - loss: 0.0836 - accuracy: 0.9649 - val_loss: 1.4517e-04 - val_accuracy: 1.0000
Epoch 99/100
8/8 [==============================] - 9s 1s/step - loss: 0.0833 - accuracy: 0.9605 - val_loss: 5.4620e-04 - val_accuracy: 1.0000
Epoch 100/100
8/8 [==============================] - 10s 1s/step - loss: 0.1023 - accuracy: 0.9693 - val_loss: 6.4898e-04 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x7d36d780e2c0>
```

Fig. 4 *Epochs 100 training*

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

WARNING:absl:Found untraced functions such as _jit_compiled_con

[28] with open('beef_pork_classifier2.tflite', 'wb') as f:
         f.write(tflite_model)
```

Fig. 5 *Model saved into (tflite_model)*

2.3 System development

2.3.1 Import TensorFlow Lite interpreter

The model that is saved needs to be uploaded into an Android Studio. A classifier is developed to load the model and read the labeled file. The coding was run, and the application was launched on the smartphone. Fig 6 shows part of the Java script coding of a classifier to complete this work.

```
public void classifyImage(Bitmap image) {
    try {
        Model model = Model.newInstance(getApplicationContext());

        // Creates inputs for reference.
        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 128, 128, 3}, DataType.FLOAT32);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect( capacity: 4 * imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());
```

Fig. 6 *Create a classifier*

2.3.2 Layout of the application

Before the Java script is run, layout is very important (Fig. 7) and it needs to be set up according to the work. Based on this work, the button of camera and upload is needed to classify either by taking photos or uploading.

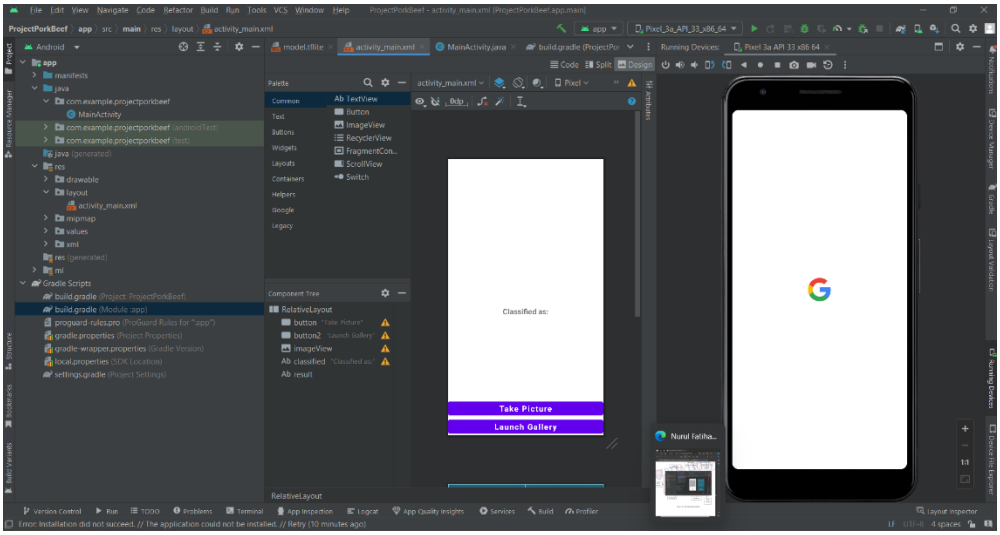

Fig. 7 *Process to build a layout of the application*

3. Result and Discussion

3.1 Application functionality

The Android application's functionalities include a main activity that displays an image, a confidence result, and several buttons such as a button to launch the diagnosis, a select photo button, and a start camera button. There was also a camera activity that required the user to shoot an image as shown at Fig. 8.

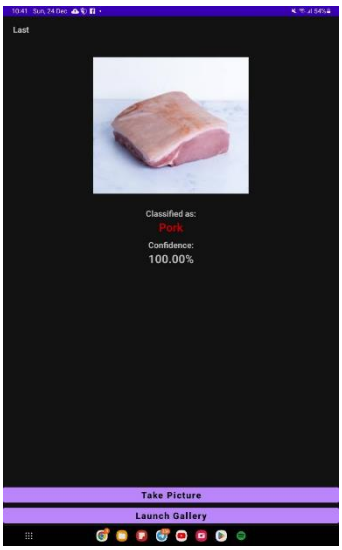

Fig. 8 *App Interface*

3.2 Application Info

The Android app, created with Android Studio, uses a compact 14.64MB TensorFlow Lite model to differentiate between pork and beef. It ensures accurate results without taking up much space on your phone. The development fine-tuned the model in Android Studio for optimal performance, prioritizing a user-friendly experience. Fig. 9 displays the app info on Galaxy Tab A8 for beef and pork classification.

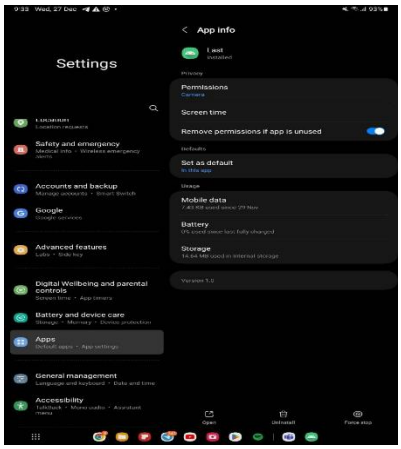

Fig. 9 *Application Info*

3.3 Testing result

Upon capturing or selecting an image from the gallery, the system resizes it to the precise input dimensions (128x128 pixels). The MobileNet V2 model proposed in this study is then applied for image classification. Fig. 10 presents a preview of a beef image, while Fig. 11 displays a preview of a pork image, both achieving a confidence and accuracy level of 100%. Additionally, the application, with a size of 14.64MB, ensures an inference time of under 317ms for image display on a smartphone. The testing involved 20 images each for beef and pork. Table 1 presents the test results conducted on the Galaxy Tab A8, indicating a flawless performance without any errors. This suggests that the application is running smoothly, and the dataset has been effectively trained without encountering any issues.

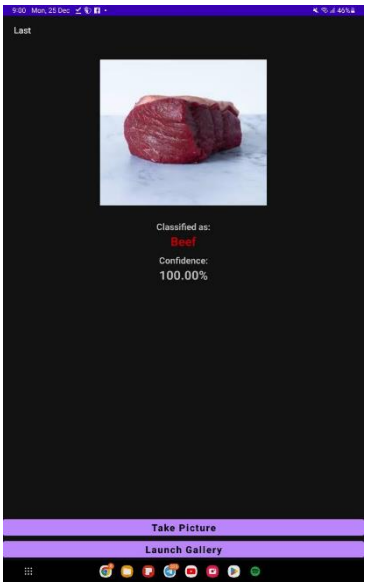

Fig. 10 *Beef testing*

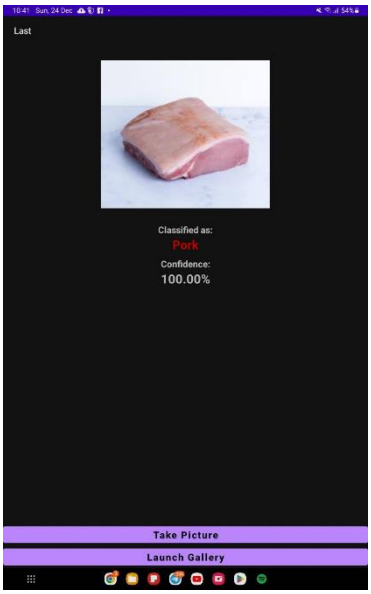

Fig.11 *Pork testing*

Table 1 *Results of testing*

Testing images	Comment
Beef (20 images)	All beef was classified correctly without error
Pork (20 images)	All pork was classified correctly without error

3.4 Epoch

In our tests, the model did well. At 20 epochs, it got 96.05% accuracy and a perfect 100% validation. When we pushed it to 100 epochs, it improved a bit to 96.93% accuracy, still holding strong at 100% validation. This shows the model is not just good from the start but also gets even better with more training. Fig. 12 and 13 show the result of the accuracy and validation accuracy.

```
Epoch 17/20
8/8 [==============================] - 10s 1s/step - loss: 0.1044 - accuracy: 0.9693 - val_loss: 0.0629 - val_accuracy: 1.0000
Epoch 18/20
8/8 [==============================] - 10s 1s/step - loss: 0.0837 - accuracy: 0.9825 - val_loss: 0.0503 - val_accuracy: 1.0000
Epoch 19/20
8/8 [==============================] - 11s 1s/step - loss: 0.0949 - accuracy: 0.9605 - val_loss: 0.0399 - val_accuracy: 1.0000
Epoch 20/20
8/8 [==============================] - 11s 1s/step - loss: 0.1080 - accuracy: 0.9605 - val_loss: 0.1093 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x7f3bc67d3400>
```

Fig. 12 *Epochs 20*

```
Epoch 98/100
8/8 [==============================] - 9s 1s/step - loss: 0.0836 - accuracy: 0.9649 - val_loss: 1.4517e-04 - val_accuracy: 1.0000
Epoch 99/100
8/8 [==============================] - 9s 1s/step - loss: 0.0833 - accuracy: 0.9605 - val_loss: 5.4620e-04 - val_accuracy: 1.0000
Epoch 100/100
8/8 [==============================] - 10s 1s/step - loss: 0.1023 - accuracy: 0.9693 - val_loss: 6.4898e-04 - val_accuracy: 1.0000
<keras.src.callbacks.History at 0x7d36d780e2c0>
```

Fig. 13 *Epochs 100*

4. Conclusion

In conclusion, this work successfully addresses the challenge of accurately classifying beef and pork through the development of an Android application. Leveraging advanced technologies, including the TensorFlow Lite model and MobileNet V2, the app provides users with a user-friendly tool to distinguish between these meats conveniently. The compact size of 14.64MB ensures minimal impact on device storage, emphasizing efficiency. Through rigorous testing and fine-tuning in Google Colab and Android Studio, the application achieves an impressive accuracy rate of 96.93%. This work not only contributes to combating deceptive meat practices but also aligns with dietary preferences, particularly in Muslim-majority regions. Overall, the developed Android application stands as a practical and effective solution, merging technology and user convenience for reliable beef and pork classification.

Acknowledgment

The author would also like to thank the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation

References

[1] Agensi. (2021, April 24). Daging babi dikata daging lembu. Harian Metro. https://www.hmetro.com.my/global/asia/2021/04/698433/daging-babi-dikata-daging-lembu Committee, B. C. A. (2015, September 8). *Meat Colour*. Pressbooks.

[2] S. Rahmati, N.M. Julkapli, W.A Yehye, and W.J Basirun. "Identification of meat origin in food products–a review." Food Control, vol. 68, 2016, pp. 379-390.

[3] Metwalli, S. (2023, March 1). *What is a Graphical User interface (GUI)?* Built In. https://builtin.com/software-engineering-perspectives/graphical-user-interface

[4] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

[5] Awan, A. A. (2022, November 23). *A complete guide to data augmentation*. https://www.datacamp.com/tutorial/complete-guide-data-augmentation

[6] Salsabila, Anwar Fitriano, & Bagus Sartono. (n.d.). Image Classification Modelling of Beef and Pork Using Convolutional Neural Network.