Image Classification of Disposable Bag using Computer Vision

- ✧ Author: Yuyang Xiao
- ✧ Github:https://github.com/22049236-Yuyang-Xiao/DL4SN-Household-Disposable-Bag-Classification-using-Arduino-Nano-33-
- ✧ Edge Impulse: https://studio.edgeimpulse.com/public/184553/latest

1. Introduction

Overview

Computer vision, a field of artificial intelligence, has gained significant attention in recent years due to its potential to revolutionize various industries. The main application for this project is training appropriate model and subsequently embed it into tiny machine learning device like Arduino Nano 33 to automatically classify household disposable bags: garbage bag, plastic bag and paper bag. With accurate identifying and sorting those bags could contribute to improve waste management processes and reduce environmental pollution.

Inspiration

Disposable materials exist everywhere in global world, citizens mostly did not recognize the complexity and importance of dispose process. For example, the plastic bags, when improperly discarded or left to decompose in landfills, can have detrimental effects on the environment, wildlife, and human health. Traditional methods of manually sorting waste can be time-consuming, labor-intensive and error prone. Therefore, there is a need for automated solutions that can streamline the waste sorting process and ensure accurate classification.

Example

An expected industrial professional equipment in Figure 1 has the function while camera capturing object with a special image to be processed, the program in device like robot arm or mini household cleaning vehicle will be worked to classify relevant object through pre-trained neural network model.

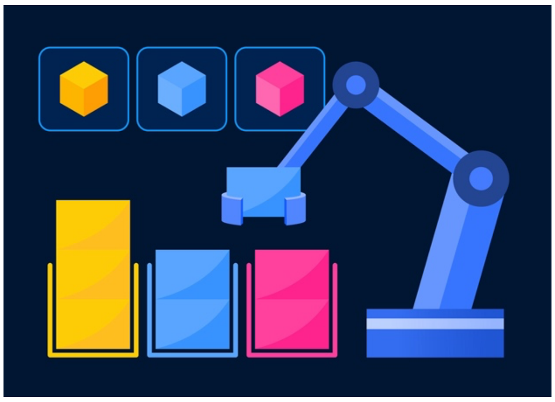
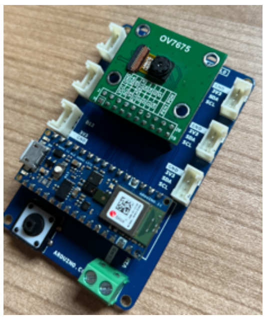
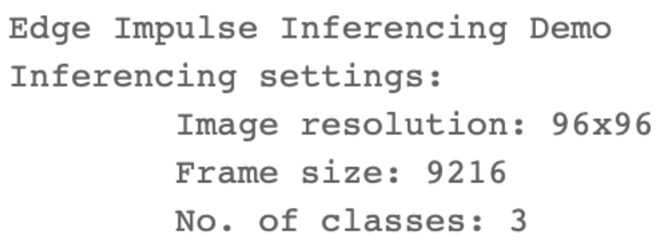


Figure 1 Example of Expected Prototype





```
Edge Impulse Inferencing Demo
Inferencing settings:
        Image resolution: 96x96
        Frame size: 9216
        No. of classes: 3
```

Figure 2 Arduino Nano Board and Camera Module

Figure 2 is the information about experiment equipment. By using module of OV 7675 Camera [1], the image will be captured as 96 * 96 resolution using special inference demo which was loaded in Arduino Nano board. images are then processed using computer vision algorithms to identify and classify the bags based on their level of decomposition. For instance, the algorithm can analyze the feature, color, and other information which can then be used to sort the bags into appropriate categories for further processing, recycling, or disposal.

2. Research Question

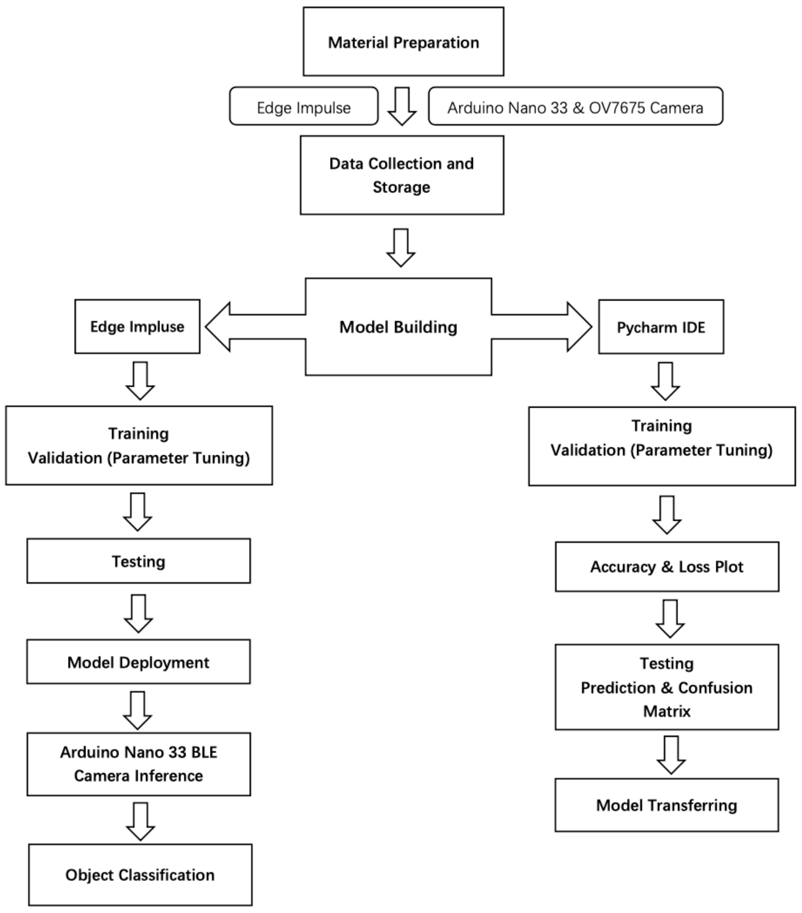
a) Can Arduino Nano 33 classify disposable bags using OV 7675 camera accurately?
b) How to deal with the resolution issue of image since it is captured by OV 7675 camera?
c) What approaches do experiment applies to improve model performance?

3. Application Overview

Application diagram in Figure 3 is composed of all components block within project. Firstly, the physical material should be prepared before the experiment starts, Arduino Nano 33 BLE board and OV 7675 Camera is used to collect image datasets and could be downloaded from Edge Impulse. After collecting adequate datasets, experiment will begin with building model.

This project firstly build model by using local IDE Pycharm and Google Colab to construct network around with MobileNet V1 [3]. Training validation process will determine which sets of parameters will be used in the last testing session and provide reference for the work of Edge Impulse. In the meantime, the accuracy and loss plot could evaluate model performance after completing training, user could be told whether it is overfitting or underfitting. By combing next block, testing report provides other performance metrics such as F1 score, which further prove the reliability of model performance. In addition, the model should be transferred to the specific source file in order to embed into TinyML device, in this case Arduino Nano 33 BLE.

Edge Impulse is a powerful platform that could permit user building expected model and deploying into device through compiled source file. In this step, training, validation, testing blocks rely on previous tuned parameters so that the model will get comparative good performance. Model deployment could be made by chosen any required device, this makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Camera module block is an evitable item because this project requires using camera to make image classification in live. Eventually, device will execute program and make corresponding classification depending on each category's probability.



Material Preparation
Edge Impulse
Arduino Nano 33 & OV7675 Camera
Data Collection and Storage
Edge Impluse
Model Building
Pycharm IDE
Training Validation (Parameter Tuning)
Testing
Model Deployment
Arduino Nano 33 BLE Camera Inference
Object Classification
Training Validation (Parameter Tuning)
Accuracy & Loss Plot
Testing Prediction & Confusion Matrix
Model Transferring

Figure 3 Diagram Block of Application

4. Data

Data source of this project is collected by myself since the quality and resolution of image from OV 7675 was more special than normal image. Figure 4 is three images with labels captured by OV 7675 camera. An important point for image is the distance between the camera and the object location is increased than the previous demo test, that is because device mostly is not adjacent to target object in real world. In general, the dataset is composed of three categories, which are already introduced before: garbage bag, plastic bag and paper bag. Edge Impluse was utilized to connect with OV 7675 camera installed on Arduino Nano 33 BLE, the squashing image will be transmitted as long as camera module begins to work.

Eventually the dataset of this project contains raw images which 60% are treated as training set, 20% are validation test and rest of them are testing set.

Figure 4 Experiment Dataset Image - Label

5. Model

With the progress of scientific research, many high-quality neural network models have appeared, but some of them cannot run on mobile devices or embedded devices due to its large memory and computation requirements. For example, the weight of VGG network is 490M and that of ResNet152 is about 640M. Image classification task could be achieved through lots of complex but delicate model, but it is evitable to choose a simple but capable one because Arduino Nano 33 BLE restricts its ram size only has 256KB of RAM. Therefore, MobileNet V1[3] is chosen due to its computation amount are greatly reduced and the parameters are only 1/32 of VGG.

As the framework and working diagram in Figure 5 and Figure 6 presents, MobileNet v1 consists of a series of depthwise separable convolution blocks followed by global pooling and a softmax output layer. Each depthwise separable convolution block consists of a depthwise convolution layer followed by a pointwise convolution layer. The depthwise convolution layer applies a separate convolutional filter to each input channel, while the pointwise convolution layer combines the output of the depthwise convolution layer using a 1x1 convolutional filter. Eventually, softmax output layer produces a probability distribution over the output classes based on the feature vector. In addition, dropout with 0.2 to ignore partial neurons could make network less sensitive to the weight change of a neuron, also increasing the generalization ability and reducing overfitting.

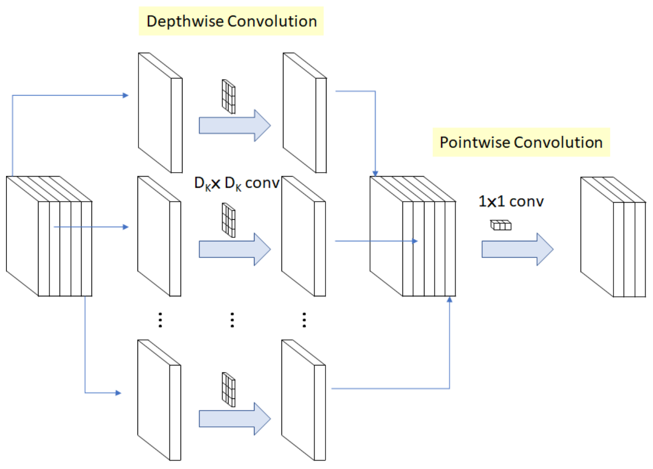


Figure 5 Architecture of MobileNet V1

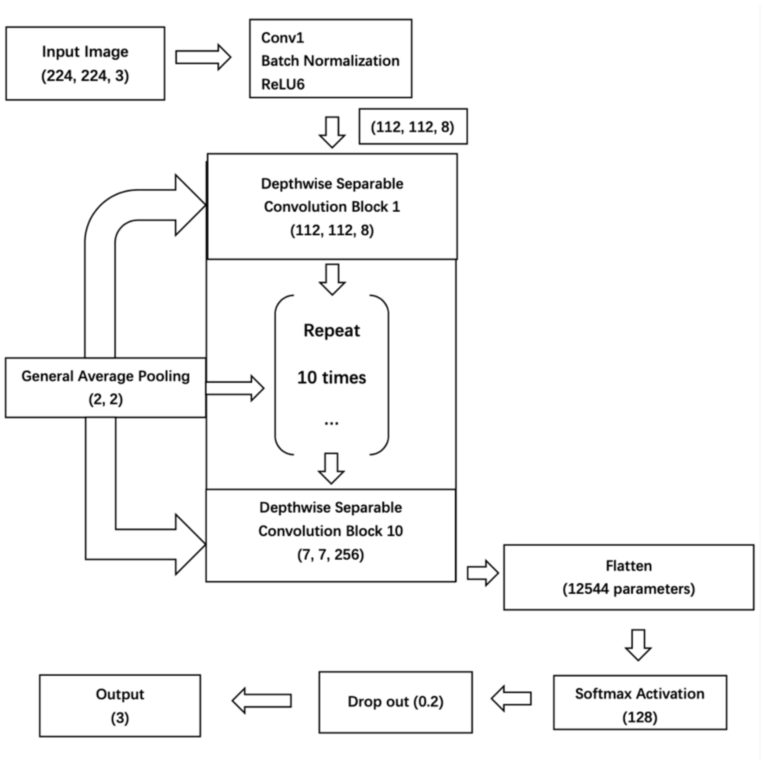


Figure 6 Working Diagram Flow of MobileNet V1

6. Experiments

In general, this project is attempting to train a comparative good MobileNet V1 model and deploy it into Arduino Nano 33 BLE to make image classification through OV 7675 camera capturing. Experiment was set to focus on constantly improving the performance of model by tuning hyper parameters and determining best group of parameters.

Initially, two sets of data were set as the experimental group and the control group. One of them is the original image data which will not be enhanced and adjusted, is treated as control group. Another one is experimental image data, will be augmented and transferred to array type through a series processing such as rescaling to [0,1], rotation, horizontal flip etc. Furthermore, each image was enhanced through so that the generalization ability of the model could be improved as well. At the same time, this group of experimental results will be used to prove the necessity of image data preprocessing.

When datasets were processed, the training process will begin by loading previous divided training sets into network with default hyper parameters. Eventually, program will return the preliminary results of this session, but normally the score is not good as expectation, statistics information in Table 1 provides the evidence. Therefore, following validation process should be prepared.

In order to find best performance model among those ranges of value, this part set grid search to tune parameter according to the corresponding score. Expected parameter candidates are: ‘alpha’, 'learning rate’, ‘batch size’ and ‘number of epoch’.

```python
learning_rates = [0.001, 0.01, 0.1]
batch_sizes = [5, 10, 16, 32, 64]
num_epochs = [10, 20, 30]

def mbv1(learning_rate, batch_size, num_epochs,train_generator,test_generator):
  model = Model(inputs=base_model.input, outputs=x)

  optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

  model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer=optimizer)

  history = model.fit(train_generator, batch_size = batch_size,epochs = num_epochs, validation_data=test_generator, shuffle=True, validation_steps=len(test_generator))
  scores = model.evaluate(test_generator)

  return scores[1]

best_accuracy = 0.0
for lr in learning_rates:
    for bs in batch_sizes:
        for ne in num_epochs:
            accuracy = m(lr,bs,ne,train_generator,test_generator)
            if accuracy > best_accuracy:
                best_accuracy = accuracy
                best_lr = lr
                best_bs = bs
                best_ne = ne
# Print the optimal hyperparameters
print('Optimal learning rate: {:.4f}'.format(best_lr))
print('Optimal batch size: {}'.format(best_bs))
print('Optimal number of epochs: {}'.format(best_ne))
```

Each parameter will be run in program according to its range. In the meantime, it requires refining the search space based on the results of the previous experiments, repeating steps 4 to 6 until the performance of the model is satisfied.

According to validation results, program will give the best group of hyper parameters among all combinations. 20 for Epoch, 0.001 for learning rate and 10 for batch size, which are convinced as final parameters to draw accuracy and loss plot in order to evaluate performance. Figure 7 concludes the overall training performance of experiment group is good although there exits distinct fluctuation at the first two epoch. Generally, the trend of line shows convergence while epochs gradually increased to 20, accuracy result seems around 0.96 and loss value decreased to around 0.1.

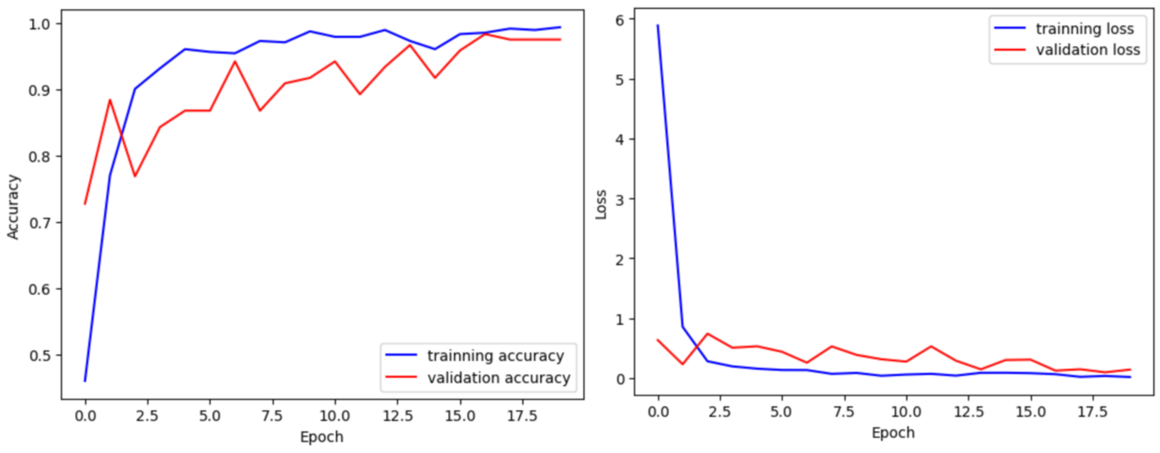


Figure 7 Augmented Image Data - Accuracy and Loss Plot for 20 Epoch

By combing results of experiment group and control group, it has the compelling differences no matter how delicate tuning was made. As Table 1 exhibits, two groups used same parameter and training process, but augmented datasets performed much better than original datasets for both metrics. In sum, image classification task in deep learning requires amount of delicate preprocessing which should not be ignored. Thus, the subsequent process will continue using augmented image data so that the Arduino Nano 33 BLE will get a good classification result.

Table 1 Testing Evaluation - Experiment Group vs Control Group

	EXPERIENCE GROUP (AUGMENT)	*CONTROL GROUP (NO AUGMENT)*
TEST DATA LOSS	0.20018	6.24
TEST DATA ACCURACY	96.05%	45.87%

Except for value of accuracy and loss, Precision (P), Recall (R) and F1 Score (F1) could be treated as metric to evaluate model performance during testing process. As the confusion matrix and Table 2 presents, the overall performance of pre-tuned model is good, all metrics have comparative high score value, which means the testing result for each image basically equals to label itself.

Table 2 Confusion Report for Test Dataset

	Precision	*Recall*	*F1-score*	*support*
Garbage Bag	0.98	0.96	0.97	50
Paper Bag	0.96	0.95	0.96	58
Plastic Bag	0.93	0.98	0.96	44
Accuracy			0.96	152
Macro avg	0.96	0.96	0.96	152
Weighted avg	0.96	0.96	0.96	152

Arduino Nano 33 BLE device could deploy classification program only if the pre-tuned model is transferred to C source file with int 8 data type and then to be embed to device corresponding pointer location. Model deployment could be achieved on Edge Impulse and the performance will not be affected as long as the relevant dataset, model and parameters are equals to above experiment's setting. By loading the source file to Arduino Nano 33 BLE through IDE after compiling process on Edge Impulse completed, two mandatory modules contain pre-trained model and camera inference were included in the compiler board as below figure. Eventually, the classification on device could begin once Nano board and camera are linked to IDE compiler.

7. Results and Observation

Main Experiment Results

After experiencing training, validating and testing, MobileNet V1 model gained a good classification performance. Figure 8 is the confusion matrix of overall 152 testing images, each row and column are the specific category, the number of each square represents the value that matches a condition for a row and column. For instance, it is seen that there are 50 items in first rows (garbage bag), 48 of them belongs to first column (garbage bag) and only 2 items are not belonged to this category. Basically, there is no anomaly data appearing and it could conclude that this pre-trained model is good for classifying disposable bag.

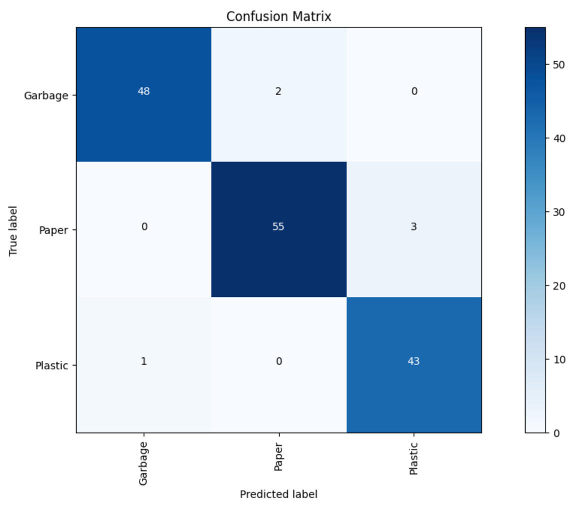
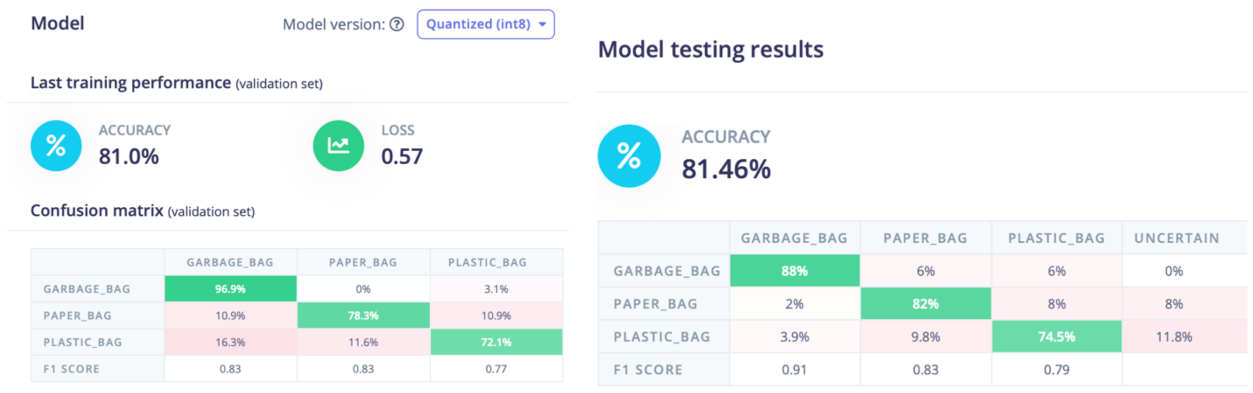


Figure 8 Confusion Matrix of Tuned Model Using Testing Results

Figure 9 is the visualization of training and testing performance on edge impulse with model type equals to quantized (int 8). For training process, the overall accuracy and loss are based on F1 score through confusion matrix among all labels, 81% and 0.57 respectively. Results from testing part also provided confidence that pre-trained model could be evaluated well because of the similarity of accuracy between former training and testing process, each item presents comparative good score although it did exist some defects that appeared uncertain condition of plastic bag and paper bag. Generally speaking, the subsequent deployment on device will cause well classification results, at least for most scenarios.



Last training performance (validation set)

ACCURACY 81.0%

LOSS 0.57

Confusion matrix (validation set)

	GARBAGE_BAG	PAPER_BAG	PLASTIC_BAG
GARBAGE_BAG	96.9%	0%	3.1%
PAPER_BAG	10.9%	78.3%	10.9%
PLASTIC_BAG	16.3%	11.6%	72.1%
F1 SCORE	0.83	0.83	0.77

Model testing results

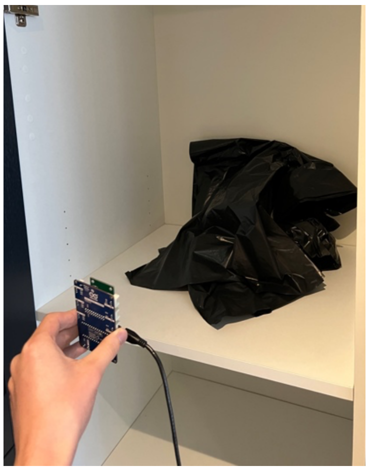
ACCURACY 81.46%

	GARBAGE_BAG	PAPER_BAG	PLASTIC_BAG	UNCERTAIN
GARBAGE_BAG	88%	6%	6%	0%
PAPER_BAG	2%	82%	8%	8%
PLASTIC_BAG	3.9%	9.8%	74.5%	11.8%
F1 SCORE	0.91	0.83	0.79	

Figure 9 Visualization Result of Simulated Quantized Model on Edge Impluse

According to three group of live capture through the camera on nano board in Figure 10, 11 and 12, console platform exhibits relevant instruction message ‘taking photo’ every 2 seconds. If the image size is correct then the embed model will begin analyzing image information such as feature, eigen value. Finally, it will make prediction and demonstrate to user via printing probability score among three categories. Obviously, the model is capable to clearly recognize and classify each object based on probability value which are all higher than 85%, it is convincing enough to justify the results for regular image classification task.

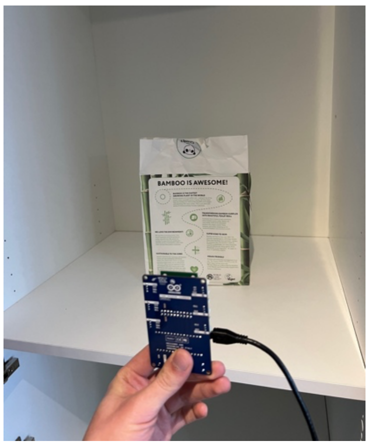
Garbage Bag



```
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 13 ms., Classification: 758 ms., Anomaly: 0 ms.):
    garbage_bag: 0.99219
    paper_bag: 0.00391
    plastic_bag: 0.00781
```

Figure 10 Garbage Bag Classification Result From Embedded Model in Arduino Nano Board

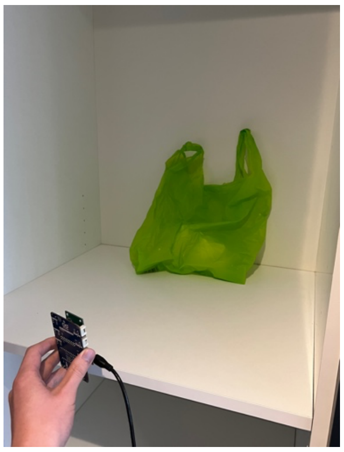
Paper Bag



```
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 13 ms., Classification: 758 ms., Anomaly: 0 ms.):
    garbage_bag: 0.03125
    paper_bag: 0.95312
    plastic_bag: 0.01562
```

Figure 11 Paper Bag Classification Result From Embedded Model in Arduino Nano Board

Plastic Bag



```
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 13 ms., Classification: 758 ms., Anomaly: 0 ms.):
    garbage_bag: 0.08594
    paper_bag: 0.03906
    plastic_bag: 0.87109
```

Figure 12 Plastic Bag Classification Result From Embedded Model in Arduino Nano Board

Conclusion

- Training, Validation and Testing observations: Adjusted model by adding last dense layer has 128 neurons and drop out with rate of 0.2. Accuracy and loss of training are better than validation, which represents it might not lead overfitting and underfitting problem. Validation process provides tuned parameter, which is suitable among all combination, also proves it is not necessary to use higher epoch and batch size in order to influence training performance. Testing results comes from pre-trained model that make a valid classification and model was embedded into Arduino Nano 33 BLE got a distinguished performance.

- Experiment Improvements: What have former experiment done were not sufficient to convince people to believe the pre-trained model is capable to make classification due to the final score of probability was less than 0.5. Not only the images were not enhanced, but also the parameters were not tuned through validation, which detrimentally influenced performance. Thus, several vital improvements were made including increase distance between camera and object while collecting, add more dataset, enhance quality of image, adjust layers of neural network, hyper parameter tuning.

- Limitations: Experiment didn't consider the case which normally could happen in the real world: more than two categories will appear such as garbage bag and plastic bag when image is collecting by camera. Or sometimes the image only has half or even less pixel information of object. If those condition exists, the normal image classification could not be worked perfectly because pre-trained model was not covered in those cases.

- Challenges: According to the existence of limitation stated before, one of challenges is how to train a model that incorporates comprehensive scenarios without increasing more ram usage and latency on TinyML device like Arduino Nano 33 board. Second, although it was successful to make classification for each image with a valid probability score, in most scenarios it was happened in normal light source condition. Considering one expected application of this project is making classification followed with robot movement, the background of image shooting is depended on working time and weather condition, thus it could be serious luminous or darkness, none of them will be benefit with the accuracy of classification.

- Future work: If excluding special case, the overall performance of embed pre-trained model in nano board is good for now. What will be attempted to achieve in future is possibly incorporating complex cases and working out with latent defects of model. Considering user-friendly view, camera module could be replaced to web camera or other advanced functional substitute, with high quality raw image plus subsequent augment the performance of model could be improved to some extent.

After all, Arduino Nano 33 BLE board is not a main tool, it has many restrictions of ram usage and other elements. If possible, the next experiment will be deployed on useful tools such as Raspberry Pi 4 or phone client in order to achieve a real classification. Thus, the object detection and instance segmentation should be considered as well to make each block connected in expert deep learning filed and that will lead a consistent result of object detection, localization and classification.

8. Biblography

1. Arduino nano 33 ble sense, Arduino Nano 33 BLE Sense - Edge Impulse Documentation. Available at: https://docs.edgeimpulse.com/docs/development-platforms/officially-supported-mcu-targets/arduino-nano-33-ble-sense (Accessed: April 22, 2023).

2. Marionette (2022) Plastic - paper - garbage bag synthetic images, Kaggle. Available at: https://www.kaggle.com/datasets/vencerlanz09/plastic-paper-garbage-bag-synthetic-images (Accessed: April 22, 2023).

3. Howard, A.G. et al. (2017) MobileNets: Efficient convolutional neural networks for Mobile Vision Applications, arXiv.org. Available at: https://arxiv.org/abs/1704.04861 (Accessed: April 22, 2023).

9. Declaration of Authorship

I, Yuyang Xiao, confirm that the work presented in this assessment is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Yuyang Xiao

April 22, 2023