Garage Door Opener for Motorcycles Using Image Classification

Minsung Cho
University of Nevada, Las Vegas
Electrical and Computer Department
Multidimensional DSP
Spring 2021

***Abstract*—This paper proposes a machine learning image classification application to the automatic garage door opener for motorcycles. It solves the problem of motorcycle riders to open the garage door hands-free with the use of custom data set collection from a several videos taken, light CNN model, and background subtraction and more. This project uses Python 3.8, Tensorflow 2.4, and CUDA 11.0 on Windows 10 OS. The hardware used for the project is AMD Ryzen 3600X and NVidia 1050Ti. The code is available at https://github.com/cho-minsung/multiDSP/tree/main/final.**

***Index Terms*—Image classification, machine learning, background subtraction, object detection**

I. Introduction

Most of the car owners do not experience any difficulty opening the garage door opener in their cars when they park their car in the garage. This is because driving a car to the driveway while reaching for the garage door opener remote can be done with a free hand before reaching the garage. However, this is not the case for many motorcycle riders. Motorcycle requires both hands to be occupied while operating the motorcycle. The majority of the motorcycles are parked in the garage. Especially when the most driveway is slanted, it is hard to temporarily stop to press the garage door opener remote. On top of that, many motorcycles are prone to tip over at a lower speed or a stopped position. It is ideal to keep both hands on the motorcycle for safety and convenience.

This project aims to tackle such a problem by utilizing the machine learning computer vision method of binary image classification. It will have a main device that captures the wide view of the driveway in the direction of the motorcycle coming into the garage. This device will be equipped with a power supply, a camera, a storage module, and a communication module, like Wi-Fi or bluetooth. The computation will be done through the mobile device or the computer connected to the main device to collect the video captured and to perform the model training and adaptive background subtraction. This is a challenge of its own because the training has to be done with a limited computing power. Therefore, a very light and efficient CNN architecture and MobileNetV2 were used.

For the background subtraction, the adaptive background mixture model for real-time tracking can be used. And for the object detection, single shot multi-box detector (SSD) can be used. These are the features that will be added in the future.

The sample dataset collected for the experiment is from videos capturing various motorcycles, cars, and bicycles. From the videos, 221 images were selected to be trained and be validated.

The result of the experiment was 96.36% confidence level for the daylight image classification and 99.99% confidence level for the sunset image. Also, it successfully classified 9 random images from all the objects using MobileNetV2.

Identify applicable funding agency here. If none, delete this.

II. Project structure

The project structure is attached as the fig. 1. The camera is set up on the top of one end of the garage door pillar facing at the way the vehicle comes into the garage.

III. Design and Performance Constraint

In order for the detector to work without making the motorcyclists stop at any point, the system must have no delay. It would defeat the sole purpose of the project if the recognition is slow or inaccurate.

Here are the calculations for the time constraint on the detection and all the operations in between. First, the video capturing should be done every 1/12 or 1/6 seconds for the balance between the performance and efficiency. The photos taken are background subtracted by adaptive background subtraction to recognize any significant change in the background as an object. Then the immediate series of photos will be collected and will be fed into the prediction function that uses the pre-trained model. The garage door opens between 12-15 seconds (9-11 seconds just enough to get in). The average driveway is 18-20ft. Including a few feet of the road, this would make the total window time of less than 25 feet. The average motorcycle speed during parking and during pulling up to the driveway is 3-5mph. This leaves 3-8 seconds for the device to detect an object, classify an object, and send the signal to the door opener remote.

There are few obstacles to tackle during this small window of detection. There might be a car parked like in figure 2. Also other parallel parked cars and bushes can be a visual obstacle. Therefore, the model should be able to classify the motorcycle even when the visibility is partial. The decision algorithm to open the door then should be the classification

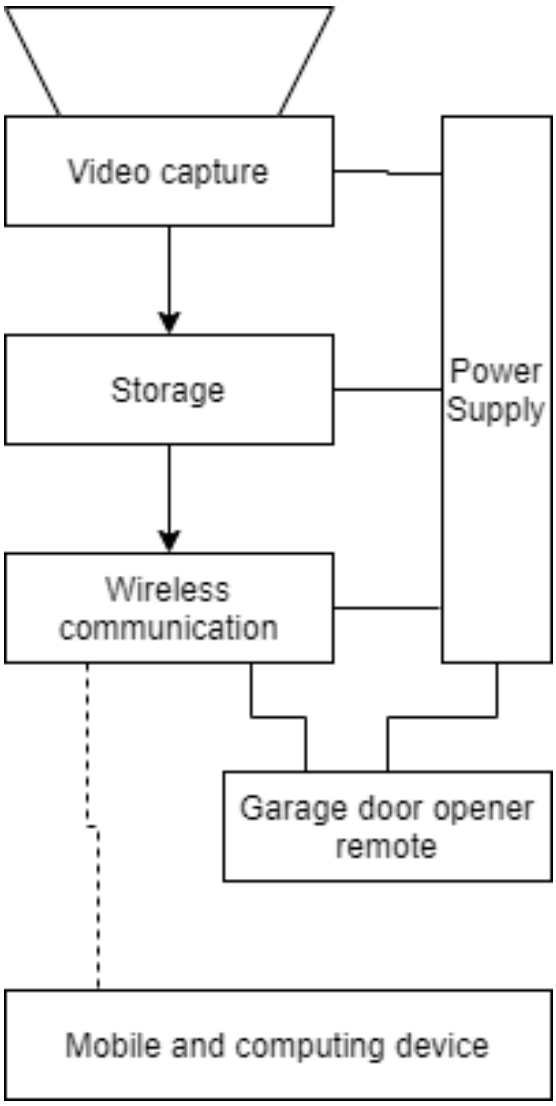

Fig. 1. General structure of the device.

Fig. 2. Image of a car parked.

of the collection of images gathered as the motorcycle pulls up to the garage.

IV. Design considerations

The project consists 3 main parts of the body. The first is the background subtraction that will be the basis of the object detection. The second is the image classification that will be the one of the primary mechanism for the garage door opening decision factor. The last is the object detection that will provide another evidence variable for the decision factor.

The current version of the project concentrates mainly on the image classification. This is because there should be a problem of bounding box labeling for object detection. Because the application is going to be for a specifically registered vehicle from the user, in order to detect a certain vehicle, it would have to have a custom labels that the user must input. Drawing bounding boxes for hundreds of images to get the bounding box label is a rather tedious labor to ask for the end-user. So this part is left for the future revision as another critical feature to be added.

The background subtraction should be used to detect when to take the image and to classify the image. Due to the computational limit, ”mixture of Gaussian(MoG)” method is used. The adaptive background subtraction is not an adequate solution to this application because the moving objects are slow and the change of lighting requires multiple thresholding. The MoG will be invariant to lighting changes and will be computationally efficient. This will also be the feature to be added when there will be a physical device that will need an optimization to detect the object on a certain time.

A. Image classification

In order for the computational efficiency, the image class was a binary decision of true or false. The dataset is divided between the specific registered vehicle by the user and the rest that are either other objects or a background. This dataset will be collected first manually by the user when registering for the device. Then it will be updated and retrained as the use-time increases. This will further improve the performance and accuracy. Another performance improvement will be covered in the later section.

Due to the restriction of the storage, the dataset should be the minimum size. For the experiment, the dataset size of 221 images were used. Another computational limit asks for a relatively lighter convolutional neural net(CNN). Therefore, two options of model are tested for the application.

B. Custom light CNN

A custom CNN tested has an architecture as shown in figure 6. It has roughly 4M parameters with 3 convolutional layers followed by max pooling layer to reduce the size. Then this output is flatten and is fed into a dense fully connected layer to classify the image. For the training time, with the specification of the PC written in the introduction, it took 300 seconds with 20 epochs. For overfitting optimization to overcome the small dataset size, dropout layers and rotation, intensity variation,

dilation and etc. were applied before and in between the training layers. Two images out of the training and validation dataset were used for predicting the image classification.

Layer (type)	Output Shape	Param #
rescaling_5 (Rescaling)	(None, 180, 180, 3)	0
conv2d_6 (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d_6 (MaxPooling2	(None, 90, 90, 16)	0
conv2d_7 (Conv2D)	(None, 90, 90, 32)	4640
max_pooling2d_7 (MaxPooling2	(None, 45, 45, 32)	0
conv2d_8 (Conv2D)	(None, 45, 45, 64)	18496
max_pooling2d_8 (MaxPooling2	(None, 22, 22, 64)	0
flatten_2 (Flatten)	(None, 30976)	0
dense_4 (Dense)	(None, 128)	3965056
dense_5 (Dense)	(None, 2)	258

Total params: 3,988,898
Trainable params: 3,988,898
Non-trainable params: 0

Fig. 3. Custom CNN model.

Fig. 4. daytime and nighttime prediction image.

C. MobileNetV2

MobileNet is a ”depthwise separable convolution” neural network created by Google which dramatically reduces the complexity cost and model size of the network, which is suitable to Mobile devices, or any devices with low computational power. In MobileNetV2, a better module is introduced with inverted residual structure, Non-linearities in narrow layers are removed. As in figure 3, the model has a residual block with stride of 1 and a block with stride of 2 for downsizing. Noticeable characteristics of the layers are that the first layer is 1x1 convolution with ReLU6, the second layer is depthwise convolution like MobileNetV2, and the third layer is another 1x1 convolution but without any non-linearity. If ReLU is used again, the deep network only have the power of a linear classifier on the non-zero volume part of the output domain. Lastly, there is an expansion factor t of 6. If the input has 64 channels, then the internal output would be 64x6=384. See figure 4 for the summary of the input/output relationship.

The architecture of the MobileNetV2 is shown as in figure 5.

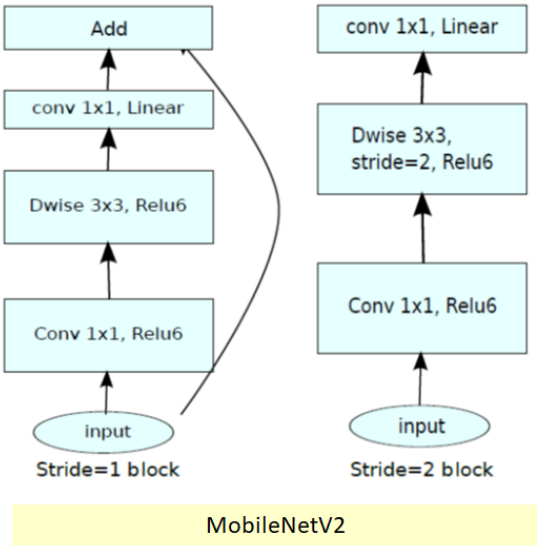

Fig. 5. MobileNetV2 sturcture.

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Fig. 6. Input, output, and operators

For the image classification performance comparison, MobileNetV2 outperformed MobileNetV2 and ShuffleNet with comparable model size and computational cost. The table of comparision is shown in figure 6.

V. The Results

The result using the custom CNN shows that the daytime prediction had a 96.36% confidence in guessing the specific motorcycle right and the nighttime prediction had a 99.99% confidence in guessing the specific motorcycle right. The images used are shown in the figure 4.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fig. 7. Overall architecture

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Fig. 8. Classification performance comparison.

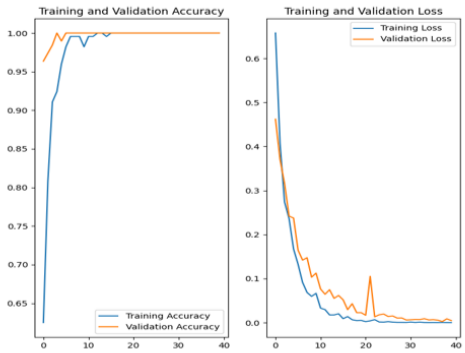

Fig. 9. The result of MobileNetV2

The experiment with the MobileNetV2 was done with the same epoch of 20. Despite that the training time is 10 times slower, the result identified all 9 of the random images correctly. Figure 7 shows the performance of the MobileNetV2. Figure 8 shows the classification results of the 9 random images.

Fig. 10. The classification of MobileNetV2

VI. The Conclusion

The results show that the classification with the custom CNN passed all the hardware constraints with less training time than the MobileNetV2. However, the next experiment should be done on a mobile phone with a mobile specifications. Then further development in hardware for the physical implementation as well as in software to include object detection and background subtraction. Regarding the object detection that will be added in the future, it would be highly beneficial to try both the pre-labeled open-source dataset that labels motorcycles and cars to generally detect and label the motorcycle for additional security variable in terms of opening the garage, or to use the fixed-size bounding box around a recognized object from the MoG.