Clean Bottle: A reverse vending machine that detects clean PET bottles with machine learning

Jeongmin Yoo, Yeeun Sohn, Yooha Bae

Abstract

Recycled transparent PET bottles have the highest added value among plastics, but in South Korea, only 10% of PET bottles are recycled with high quality. Therefore, the South Korea Ministry of Environment planned to impose a new policy that makes it mandatory to separate clear PET bottles. Clean Bottle is a reverse vending machine helping the new policy that takes one more step by adding machine learning so that only clean PET bottles without seals and content are accepted. The vending machine uses a machine learning technique called a convolutional neural network to determine pass or fail. A prototype of the machine was designed by using a Raspberry Pi 4B and PiCamera. The validation accuracy of the model resulted as approximately 99.3% and was able to correctly classify new data of PET bottles as well. Also, to optimize the model, this study reduced the model size by changing the number of filters.

Keywords

Machine Learning; Recycling; PET bottles; Convolutional Neural Network; Deep Learning

---

1. Introduction

1.1 Problem statement

Plastic causes severe pollution on Earth. About 300 million tons of plastic waste are produced every year in the world ("Our Planet", n.d.). Research by Ritchie and Roser (2018) stated that only 6 percent of plastics that are produced from 1950 to 2015 are recycled and others are still in use, went to landfill, or were incinerated. For plastic bottles, one million plastic bottles are sold every minute around the world, and in 2016, less than half of purchased plastic bottles were gathered for recycling but only 7% of these bottles were recycled ("The World's Population", 2020). As is the case with the world, pollution from plastic bottles is serious in South Korea. The annual use of plastic bottles in Korea reached 4.9 billion in 2017 (Kang, 2020). To reduce pollution from plastic bottles, not only the amount of use should be decreased, but also the amount of recycling should be increased.

Transparent PET bottles have the highest added value among plastics, making fiberfill, fabric, automotive parts, industrial strapping, sheet and film, new containers for both food and non-food products, and containers for baked goods by being recycled

(Yoo, 2020; Oromiehie & Mamizadeh, 2004). Therefore, in the case of PET (Poly-Ethylene Terephthalate) bottles, transparent PET bottles are required for high-quality recycling, but in Korea, only 10% of PET bottles are recycled with high quality because they are discharged regardless of color, and it is difficult to separate only transparent PET bottles (Yoo, 2020; Seoul Mediahub, 2020). Also, another reason is impurities like seals on the bottles are mixed during the process of discharge and collection (Yoo, 2020). During the process of recycling, if the wind separation and gravity separation are repeated three times, the impurities are removed. However, the more the separation process is repeated, the more money is needed, and the more pieces of PET are washed away with impurities (Kang, 2019). Therefore, to reduce the separation process, transparent PET bottles should be separately collected in the right way.

1.2 Policy

The right way to throw away transparent PET bottles is, first, transparent PET bottles should be separately collected not together with other plastics. Empty the contents of the clear PET bottle then remove the seal and crush it (Lee, 2020). When recycling the clear PET bottles, the right way is to remove the plastic cap and ring but according to the South Korea Ministry of Environment, it is okay to not remove the plastic cap and ring because it can be separated through the separation process (Yoo, 2020). To recycle transparent PET bottles with high quality, the South Korea Ministry of Environment not only emphasized the right way to throw away PET bottles but also planned to impose a new policy that makes it mandatory to separate clear PET bottles. They are planning to apply this policy to all apartments throughout the nation from 2020 December and expand to detached houses from 2021 December (Kim, 2020). This paper is about the project, Clean Bottle, that is aiming at helping this new policy using Machine Learning.

The rest of the paper is structured as follows: Section 2 presents the proposed solution in the hardware and software parts. Section 3 contains the explanation of Clean Bottle's prototype. Section 4 centers on possible future improvements of Clean Bottle, while the final section gives the conclusion.

2. Proposed solution

In this paper a reverse vending machine with capability to identify bottle status real-time is proposed. This is the contribution to collecting clean PET bottles. The purpose of the solution is to encourage people to recycle appropriate conditions of PET bottles which are without seals and content.

2.1 Hardware conception

Clean Bottle is a reverse vending machine that takes one more step by adding machine learning so that only clean PET bottles without seals and content are accepted. The configuration involves several devices. There are requirements in order to configure all the devices to work well. Pi camera is the core requirement which is used for detecting bottles. Four LEDs will be attached to the top of the reverse vending machine. By using two conveyor belts, the device is able to make bottles stand for easily compressing them with an equipped compressor. As mentioned previously, the compressor will be installed at the top of the second conveyor belt.

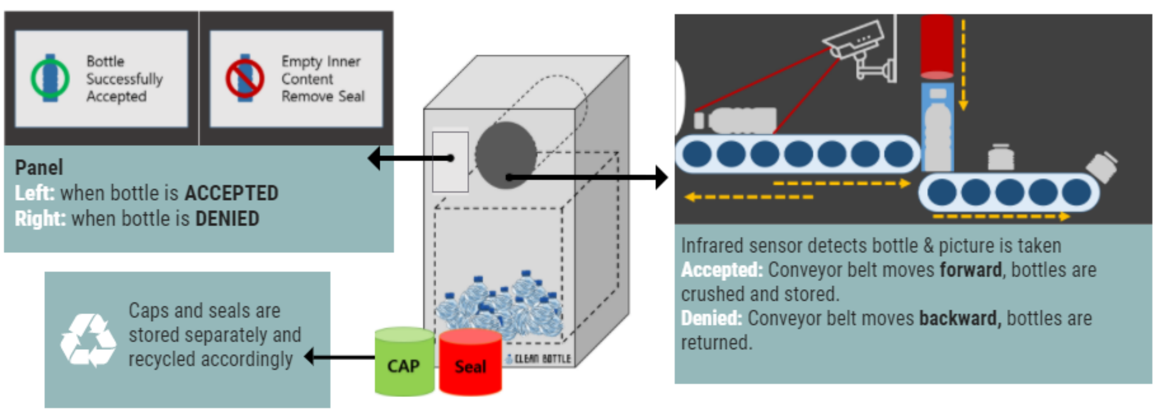

Fig. 1. Hardware structure of Clean Bottle.

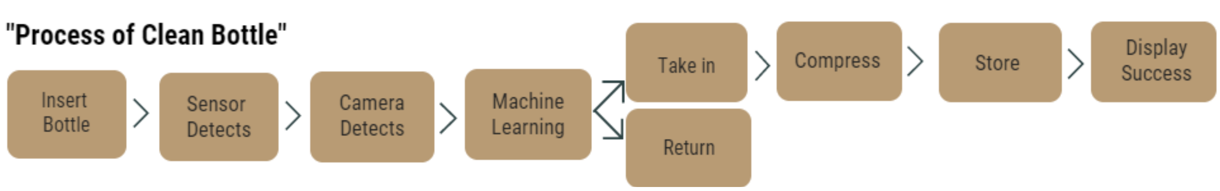

Fig. 2. The process of Clean Bottle.

Following the Fig. 2 which shows the process of Clean Bottle, when the PET bottle is inserted, the first conveyor belt takes the bottle in as shown on the figure above. When the infrared sensor detects the bottle, the conveyor belt stops and takes a picture of the bottle with a PiCamera. By machine learning, pass or fail is determined. The core conception related to machine learning will be explained below. The bottle is passed when there is no content or seals. If the bottle does pass, then the first conveyor belt moves forward, pass the bottle to the second conveyor belt by standing it up, and the bottle will be crushed by the compressor and stored. When failed, the first conveyor belt returns the bottle to the user. At the top of the machine, the panel is attached. This panel is for letting people know the status of the process. The panel shows “ACCEPTED” with green LED board or “DENIED” with red LED board accordingly. Next to the machine, there are buckets to collect caps and seals.

2.2 Software conception

Using deep learning technology to detect the PET bottle's status in real-time in a reverse vending machine and decide to pass or fail. Clean bottle uses a machine learning technique called a convolutional neural network, CNN. To train the AI, our team has divided our dataset into training data and validation data. Each image has a classification labeled to them, which is pass or fail. The labeled image goes through data augmentation, where images are modified to increase the amount of data.

Each image goes through the convolutional neural network. Convolutional neural networks called CNNs have been applied to visual tasks since the late 1980s (Wang, 2017). CNNs are hierarchical neural networks whose convolutional layers alternate with subsampling layers, reminiscent of simple and complex cells in the primary visual cortex (Wiesel and Hubel, 1959). CNNs vary in how convolutional and subsampling layers are realized and how the nets are trained (Ciresan, 2011). The convolution is a kind of product operation of a filter, which is also called a kernel. It is with a matrix of the image to extract some predetermined characteristics (Kone, 2019). When the image goes through multiple layers of the network, the model can recognize objects such as dogs, cats, and in our case, the passing or failure of PET bottles.

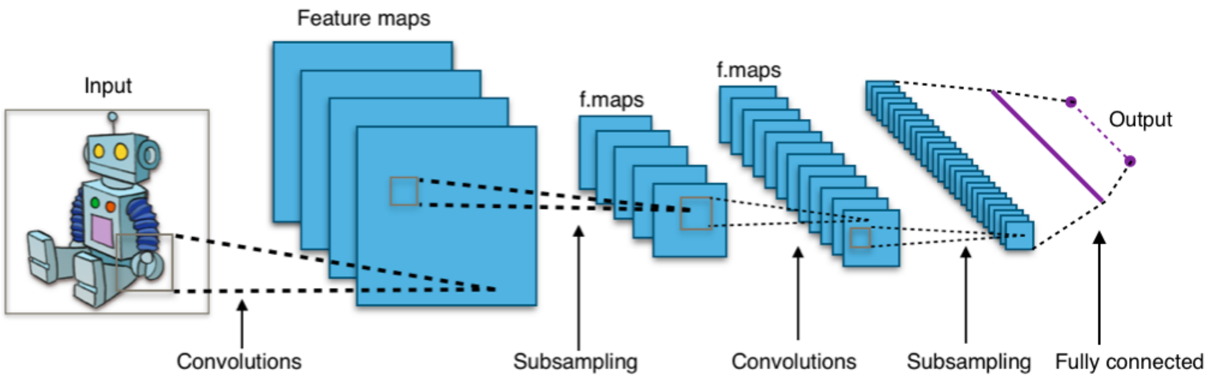

Fig. 3. Full convolutional neural network.
Source: Kone, 2019.

This project imports Keras library from tensorflow which is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow (Rieuf, 2017). For scientific computing with Python, the project uses numpy.

3. Prototyping

3.1 Creating the hardware

As no image dataset included images of PET bottles with content, without content, with a seal, and without seal, a controlled environment to collect the dataset was required. A prototype was constructed to create a stationary situation assuming that the conveyor belt is stopped and the picture of the bottle is being taken. A Raspberry Pi 4B, PiCamera, and a LED was used in this prototype. The Raspberry Pi 4B was used as it can easily be connected with sensor modules, and because it is cheap and small (Raspberry Pi). The cost of the vending machine can be reduced by using Raspberry Pi 4B and space can be saved as well. The following variables were controlled for a consistent dataset: light, background, bottle placing. To control the light, a cardboard box was used. On the top side of the box, a small hole was made to attach the Raspberry PiCamera. The PiCamera had been attached to the Raspberry Pi 4B. The Raspberry Pi 4B and the PiCamera was fixed on the top of the box so that the images would be taken without movement. To give consistent lighting, a LED light that goes by battery was attached inside the box. The background of the image dataset was kept constant by using black cardboard paper. The exact location to place the bottle was marked on the cardboard paper to ensure that the bottle placing is consistent. One side of the cardboard box was fixed to the table to safely lift the box and close it repeatedly while taking the images.

A breadboard was attached to the Raspberry Pi and an RGB LED module was connected with the breadboard (Mwangi, 2019). The LED turns red when the bottle is a fail, turns blue when the bottle is a pass.

Fig. 4. Raspberry Pi and sensors attached to the cardboard box.
Fig. 5. The inside of the prototype.

3.2 Creating the dataset

Creating the dataset was one of the largest challenges of this research. Previously, the dataset consisted of images collected from the internet. However, although the images went through preprocessing, the accuracy of the model was low for new untrained data. Thus, to increase the accuracy of the model in recognizing new untrained data, the hardware of the prototype was designed, and the image dataset to train the model was created by using this hardware. Two classifications for the dataset were set; pass and failure. Various types and sizes of PET bottles were used to create the image dataset. For passing PET bottles, the seal and inner content were removed, and the PET bottles were cleaned. The images for failure were created by differentiating the degree of crushing and the number of inner contents of the PET bottle. The PiCamera's image resolution was changed to 256x256. The dataset was ensured that an equal amount of images were taken for both classifications. As a result, 215 images of failures and 213 images of passes were taken.

3.3 Creating the software

To create the model for "Clean Bottle," CNN(Convolutional Neural Network) was used in Tensorflow and Keras. TensorFlow is an end-to-end open-source platform for machine learning (*TensorFlow*). First, the image dataset was called and the classification name was given by 'pass' and 'fail.' The dataset was split into the training dataset and the validation dataset. The training dataset is the actual data used to train the model (Shah, 2017). The validation dataset is used to evaluate the performance of the model trained by using the training dataset. The dataset is split into the training dataset and the validation dataset as there is a limited amount of data that can be provided. In this study, 30% of the dataset was used as the validation dataset and the rest was used for the training. The values of the batch of the image were rescaled and offset so that the inputs were re-ranged from [0, 255] to [0, 1] range (Chollet & Omernick, 2020). Then the images of the dataset went through the process of data augmentation. Data augmentation is a strategy to increase the diversity of the training data for models without the need for collecting new data (Seita, 2019). By cropping, padding, and horizontally flipping the images, large neural networks can be trained. Each image in the dataset was augmented to 9 images by rotation.

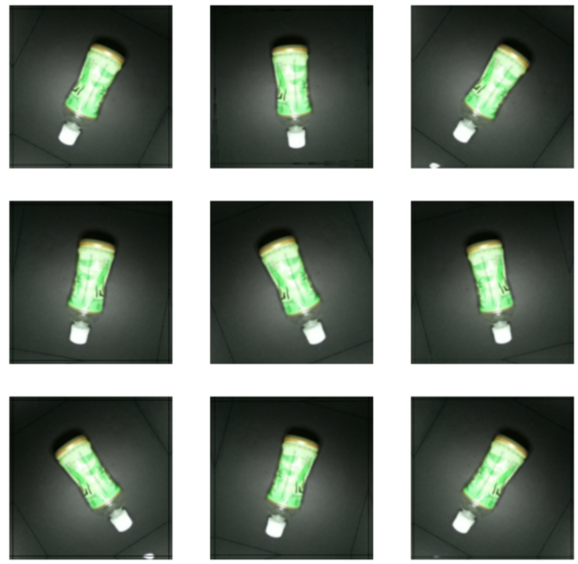

Fig. 6. Pass and fail classification on image dataset.
Fig. 7. Data augmentation of one image in dataset.

The augmented dataset was passed to CNN. The model was designed to have a kernel size of 3. The images were first processed through a Conv2D layer that has 16 kernel layers. 448 parameters were created by this layer. Then the output from the first Conv2D layer was passed through the second conv2D layer with 32 kernel layers and a consistent kernel size of 3. As a result, 4640 numbers of parameters were created. The output from the second Conv2D layer was passed through the last Conv2D layer that has the kernel size of 64. The resulting parameter number was 18496. The output data from the last Conv2D layer was then processed by dropout. Dropout allows neural networks to prevent overfitting (Srivastava, 2014). Overfitting occurs when a model is overly sensitive to the details and noise of the training data (Brownlee, 2019). When new data is inserted into the overfitted model, the performance greatly drops as the model is unable to generalize. Dropout allows models to randomly drop units during training. After the dropout, the dense layer was created by 128 dimensionalities of the output space (Keras documentation: Dense layer). The resulting trainable parameters were 8,412,578.

Table 1. Summary of the model.

Layer (type)	Output Shape	Param #

sequential (Sequential)	(None, 256, 256, 3)	0
rescaling_1 (Rescaling)	(None, 256, 256, 3)	0
conv2d (Conv2D)	(None, 256, 256, 16)	448
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_1 (Conv2D)	(None, 128, 128, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 64)	0
dropout (Dropout)	(None, 32, 32, 64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 128)	8388736
dense_1 (Dense)	(None, 2)	258
Total params: 8,412,578 Trainable params: 8,412,578 Non-trainable params: 0		

Finally, the training dataset and the validation dataset was passed to the convolutional neural network with the epoch of 25. One epoch is when the entire dataset is passed forward and backward through the neural network only once (Sharma, 2017). Only using one epoch leads to underfitting but by increasing the number of epochs the model becomes more optimal. However, if the epoch is too large, it may result in overfitting. The model for this research's optimal epoch was determined as 25 because larger epochs did not increase the training and validation accuracy. Multiple models were created and the model with the highest training and validation accuracy and the lowest training and validation loss was selected. The resulting validation accuracy was 99.3%. The model was then saved so that when new data is inserted into the model, the same model would classify the image.

Classification of new data was conducted by the following process. A PET bottle was inserted into the prototype and the PiCamera took the image. The image was saved inside the Raspberry Pi and it was sent to a remote server by using WinSCP. WinSCP allows file transfer between a local and remote computer (*Introducing WinSCP* 2017). The Raspberry Pi and a laptop were connected to the same IP address. The image was transferred from the Raspberry Pi to the laptop, and in the laptop the image was classified as pass or fail. The classification was saved as a .txt file and sent back to the

Raspberry Pi again by WinSCP. According to the classification, the LED color was changed.

Fig. 8. The training and validation accuracy graph of the model.

3.4 Results

When this study tested the prototype, the result came out with high accuracy.

Fig. 9. Test cases which is "Pass" with accuracy 99.99%, 99.94%, and 99.96%

These are “Pass” cases because they have no contents and seals. The Raspberry Pi camera took pictures of these test cases, and “Pass” and the accuracy are shown on the screen. Then, the LED turned on with blue light because these are “Pass” cases.

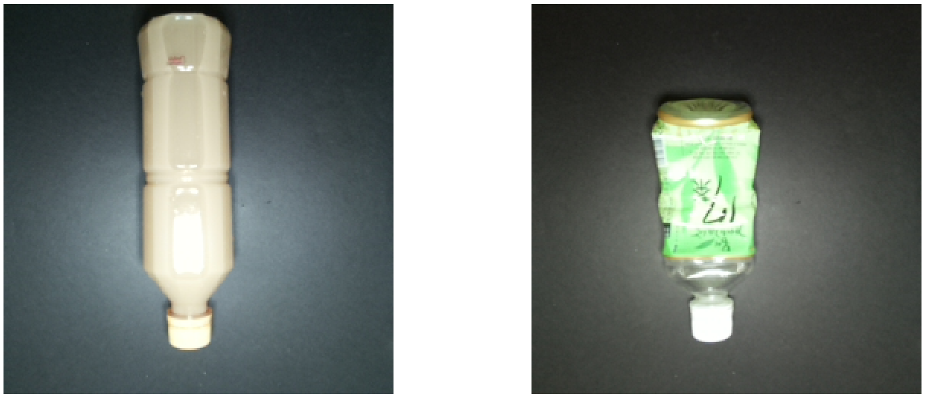

Fig. 10. Test cases which is “Fail” with accuracy 99.99% and 99.99%

These are “Fail” cases because the left one has content and the right one has a seal. The Raspberry Pi camera took pictures of these test cases, and “Fail” and the accuracy are shown on the screen. Then, the LED turned on with red light because these are “Fail” cases.

3.5 Improvement of software

To make the model suitable to run on a single-board computer, the project tried model optimization to reduce the size of the model. When the model size becomes smaller, the model will need smaller storage size and download size and use less memory (“Model Optimization”, n.d.). It’s crucial to note that there is no one-size-fits-all approach: different use cases require different techniques. Various stages of the model building lifecycle determine possible and preferred optimization strategies. This study chose two main strategies to optimize the model: change kernel size and change number of filters.

3x3

0.91	0.32	0.07
0.73	0.26	0.81
0.53	0.68	0.14

5x5

0.27	0.64	0.44	0.84	0.29
0.28	0.06	0.89	0.99	0.33
0.64	0.67	0.08	0.38	0.03
0.04	0.31	0.16	0.57	0.08
0.87	0.85	0.97	0.71	0.96

Fig. 11. Kernel size (3x3 and 5x5).
Source: Rosebrock, 2020.

First, this study tried to change kernel size to reduce the model size. The kernel size determines the width and height of the 2D convolution window (see Fig. 11). The kernel size usually includes 1x1, 3x3, 5x5, 7x7, and it's rare that kernel size is larger than 7x7 (Rosebrock, 2020). This is because it needs extremely longer training time such as two to three weeks and expensiveness (Pandey, 2020). Also, kernel size is odd-sized because odd-sized filters symmetrically split the previous layer pixels around the output pixel and if this symmetry does not exist, there will be malformation across the layers (Pandey, 2020). As a result, the kernel size has a relationship between model size, this study expected to optimize the model size by decreasing kernel size.

```
num_classes = 2

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Fig. 12. The code and graph for Kernel-size: 3x3, Model-size: 96.6MB

```python
num_classes = 2

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 1, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 1, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 1, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

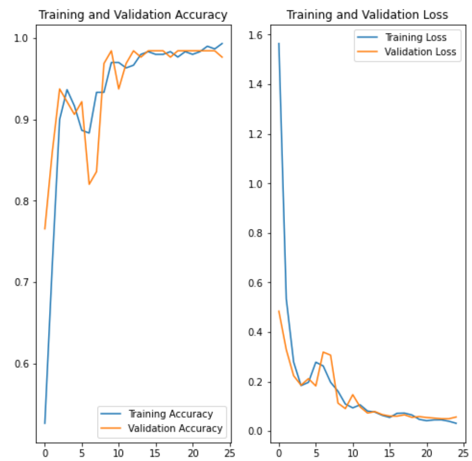

Fig. 13. The code and graph for Kernel-size: 1x1, Model-size: 96.4MB

When this project first created the model, the kernel size was set to 3x3. Thus, there was not much choice for the size of the kernel. A 1x1 kernel size is not used frequently as it only captures the interaction of input channels in just one pixel of the feature map. Thus, features extracted from the 1x1 kernel will not include information from neighboring pixels (Pandey, 2020). Despite these limitations of the 1x1 kernel, this study tried reducing the kernel size to 1. However, the model size was not significantly affected (See Fig. 12 and Fig. 13). Thus, for the model, changing kernel size was not an effective strategy.

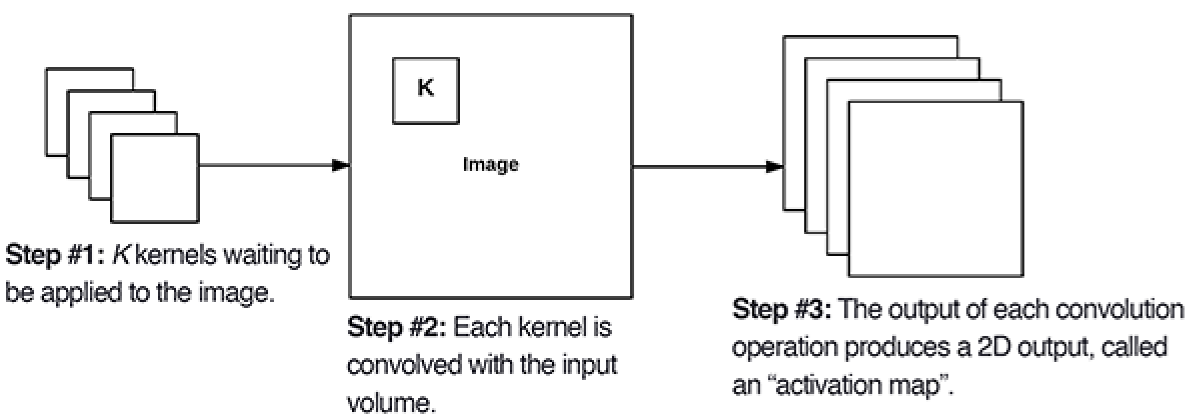

Fig. 14. Number of filters.
Source: Rosebrock, 2020.

The other strategy to reduce the size of the model is changing the number of filters. The number of filters convolves with the input volume, and each of these operations makes a 2D activation map (Rosebrock, 2020) (See Fig. 14).

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$

Fig. 15. Formula that produces a volume of size in a convolutional layer.
Source: “CS231n”, n.d.

In the Convolutional layer, the volume of size is W2×H2×D2 and D2 is K, which is the number of filters (“CS231n”, n.d.). According to the formula, this study forecasted that decreasing the number of filters will decrease the model’s volume of size, then the size of the model will be reduced. Therefore, this study reduced the number of filters for the test while maintaining the epoch to 25 and the filter size to 3 to control the test environment.

```
num_classes = 2

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Fig. 16. The number of filters: 16, 32, 64, Model-size: 96.6MB, Training Accuracy: 1.0

```
num_classes = 2

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(8, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(64, activation='relu'),
  layers.Dense(num_classes)
])
```

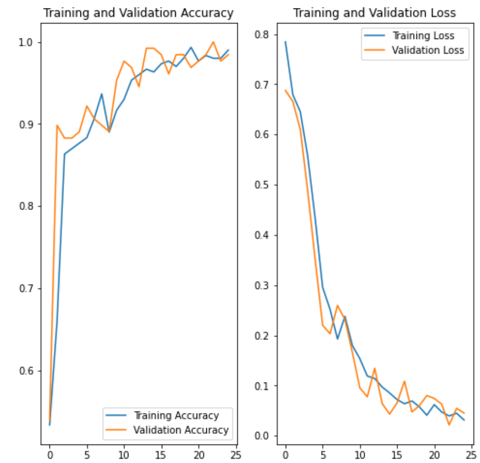

Fig. 17. The number of filters: 8, 16, 32, Model-size: 24.4MB, Training Accuracy: 0.99

```
num_classes = 2

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(4, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(8, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(32, activation='relu'),
  layers.Dense(num_classes)
])
```

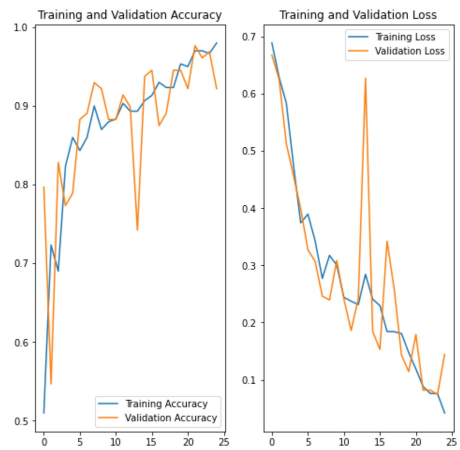

Fig. 18. The number of filters: 4, 8, 16, Model-size: 6.43MB, Training Accuracy: 0.97

```
num_classes = 2

model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

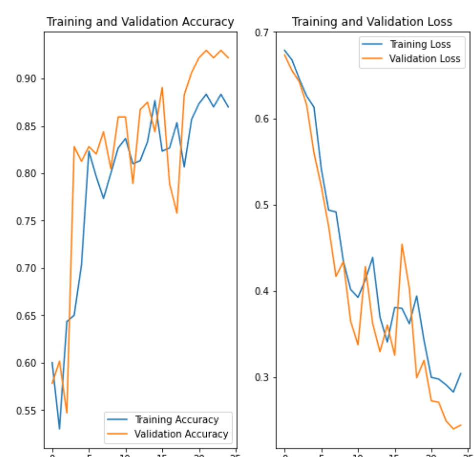

Fig. 19. The number of filters: 2, 4, 8, Model-size: 1. 92MB, Training Accuracy: 0.87

When the number of filters decreased, the size of the model was significantly reduced and the training accuracy decreased a little. Finally, the size of the model was reduced from 96.6MB to 1.92MB. However, Fig. 19's performance decreased considerably compared to Fig. 18. Thus, this study judged that the cost of reducing the size of the model was not worth the loss of performance in the case of Fig. 19. Finally, this study concluded that the best way to optimize the model is to decrease the number of filters to 4 on the first Conv2D layer as the model's performance was maintained and the model size was significantly reduced from 96.6MB to 6.43MB, 6% of the base case model size.

4. Conclusion

If Clean Bottle becomes available nationwide in Korea, the percentage of high quality PET bottles that are recycled will increase. Also, additional labor that will be required in the recycling process can be greatly reduced by individual efforts if the Clean Bottle is used. Clean Bottle can be additionally improved by further training it to check whether there are substances on papers, or styrofoam. Cans and glass bottles can also be added to check whether there are seals on them. Clean Bottle can also be improved by having its own application. The application will give users each an unique QR code. The Clean Bottle can detect the QR codes and users can gain points according to the amount of PET bottles inserted into the machine. Users can also check the nearest locating Clean Bottle and the previous Clean Bottle machines they have used. Also, the application can keep track of how many bottles the user has recycled and how much the user has contributed to saving the planet.

References

Brownlee, J. (2019, August 12). Overfitting and Underfitting With Machine Learning Algorithms. Machine Learning Mastery. machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

Chollet, F., & Omernick, M. (2020, July 25). *Keras documentation: Working with preprocessing layers*. Keras. keras.io/guides/preprocessing_layers/.

CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). cs231n.github.io/convolutional-networks/.

Ciresan, Dan. (2011, July 16). Flexible, High Performance Convolutional Neural Networks for Image Classification. www.researchgate.net/publication/220812758_Flexible_High_Performance_Convolutional_Neural_Networks_for_Image_Classification.

Convolutional Neural Networks Tutorial in TensorFlow. Adventures in Machine Learning. (2017, April 24). https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/.

HUBEL, D. H., & WIESEL, T. N. (1959, October). Receptive fields of single neurones in the cat's striate cortex. The Journal of physiology. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130/.

Introducing WinSCP. WinSCP. (2017, December 21). winscp.net/eng/docs/introduction.

Kang, E. (2020, January 7). *1년간 쓴 플라스틱컵 쌓으면 하늘까지 닿겠네 [If We Stack Plastic Cups That We Used for a Year, It Will Reach the Sky]*. DongA.com. www.donga.com/news/Society/article/all/20200107/99110106/1.

Kang, E. (2019, February 19). 분리수거한 페트병, 재활용율 절반도 안되는 이유 *[The reason why the recycling rate of PET bottles separately collected is less than 50%]*. DongA.com. www.donga.com/news/Society/article/all/20190219/94194986/1.

Keras documentation: Dense layer. Keras. keras.io/api/layers/core_layers/dense/.

Kim, J. (2020, June 23). 투명 페트병 분리배출 시범사업 성과... 페트병 더 모으기 위해 제도 개선 *[Achievement of a Pilot Project for Separation and Discharge System* *of Transparent PET Bottles... Policy Improvement to Collect More PET Bottles].* KBS News. news.kbs.co.kr/news/view.do?ncd=4477291&ref=A.

Kone, C. (2019, November 6). Introducing Convolutional Neural Networks in Deep Learning. Medium. https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9.

Lee, J. (2020, May 4). 서울시 '분리배출제' 이모저모... "아파트에서 비닐과 페트병은 어떻게 버려야 하나요*?" [Seoul's 'Waste Sorting System'... "How Do I Throw out Vinyl and PET Bottles in the Apartment?"].* Etoday. www.etoday.co.kr/news/view/1890554.

Mwangi, H. W., & Mokoena, M. (2019). Using Deep Learning to Detect Polyethylene Terephthalate (PET) Bottle Status for Recycling. *Global Journal of Computer Science and Technology*.

Model optimization : TensorFlow Lite. TensorFlow. www.tensorflow.org/lite/performance/model_optimization.

NumPy. https://numpy.org/.

Oromiehie, A., & Mamizadeh, A. (2004). Recycling PET beverage bottles and improving properties. *Polymer International*, *53*(6), 728–732. doi.org/10.1002/pi.1389

Our planet is drowning in plastic pollution. This World Environment Day, it's time for a change. (n.d.) www.unenvironment.org/interactive/beat-plastic-pollution/.

Pandey, S. (2020, July 1). How to choose the size of the convolution filter or Kernel size for CNN? Medium. medium.com/analytics-vidhya/how-to-choose-the-size-of-the-convolution-filter-or-kernel-size-for-cnn-86a55a1e2d15.

Raspberry Pi. *Buy a Raspberry Pi 4 Model B*. Raspberry Pi. www.raspberrypi.org/products/raspberry-pi-4-model-b/?resellerType=home.

Rawat, W., & Wang, Z. (2017, June). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. ResearchGate. www.researchgate.net/publication/317496930_Deep_Convolutional_Neural_Networks_for_Image_Classification_A_Comprehensive_Review.

Rieuf, E. (2017, May 20). Keras: Deep Learning library for Theano and TensorFlow. https://www.datasciencecentral.com/profiles/blogs/keras-deep-learning-library-for-theano-and-tensorflow.

Ritchie, H., & Roser, M. (2018, September 1). *Plastic Pollution*. Our World in Data. ourworldindata.org/plastic-pollution?utm_source=newsletter.

Rosebrock, A. (2020, June 4). Keras Conv2D and Convolutional Layers. PyImageSearch. www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/.

Seita, D. (2019). *1000x Faster Data Augmentation*. The Berkeley Artificial Intelligence Research Blog. bair.berkeley.edu/blog/2019/06/07/data_aug/.

Seoul Mediahub. (2020, May 4). *Q&A로 알아보는 비닐, 투명페트병 분리배출제 [Understanding with Q&A about Waste Sorting of Vinyl and Transparent PET bottles]*. mediahub.seoul.go.kr/archives/1279330.

Shah, T. (2017, December 7). *About Train, Validation and Test Sets in Machine Learning*. Medium. towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7.

Sharma, S. (2017, September 23). *Epoch vs Batch Size vs Iterations*. Medium. towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

TensorFlow. www.tensorflow.org/.

The world's population consumes 1 million plastic bottles every minute. Plastic Soup Foundation. (2020, March 2). www.plasticsoupfoundation.org/en/2017/07/the-worlds-population-consumes-1-million-plastic-bottles-every-minute/.

Yoo, J. (2020, July 19). *라벨은 떼고, 뚜껑은? 투명페트병 분리수거 이렇게 하세요 [Remove the Seal, and Then How about the Cap? The Way of Recycling Transparent PET Bottles]*. Joongang Daily. n.news.naver.com/article/025/0003018686.