Predicting Ripeness and Freshness   
of Fruits

Ho Ming Jun, Kyi Nuu Khin Khin, Ng Ngai Teng Colin, Rani Karthigeyan Rajendrakumar,   
Toh Hai Jie Joey, Toh Hong Xian (Ray)

National University of Singapore  
e0406253@u.nus.edu, e0425394@u.nus.edu, e0389113@u.nus.edu, e0395480@u.nus.edu,   
e0425064@u.nus.edu, e0406975@u.nus.edu

Abstract

Using Convolutional Neural Networks (CNNs) to predict the freshness and ripeness of fruits.

Introduction*[[1]](#footnote-1)*

Grocery shopping is an important aspect of our lives and has been ingrained in us since young. Many of us have creative ways of choosing fresh products, including a visual inspection, by looking at signs of oxidation or damage.

However, it is a problem for those who are unable to do so, such as visually impaired people or children and domestic helpers choosing fruits for the first time. Our group proposes to devise an application with two functions utilising the phone’s camera function. The first tells apart pictures of fresh fruits and rotten fruits, while the second tells the ripeness level of fruits.

This will help visually impaired people or inexperienced shoppers in choosing their fruits, and even aid experienced shoppers in confirming their choices. This application can also help users identify if fruits they have already purchased at home have gone bad, so that users can decide if the fruits are still edible or should be thrown away.

Convolutional Neural Networks

The Machine Learning (ML) algorithm that we have decided to implement in our application is Convolutional Neural Networks (CNNs). A CNN is a Deep Learning algorithm which can take in an input image, allocate significance (through weights and biases) to various aspects in the image and be able to distinguish one from the other[[2]](#footnote-2).

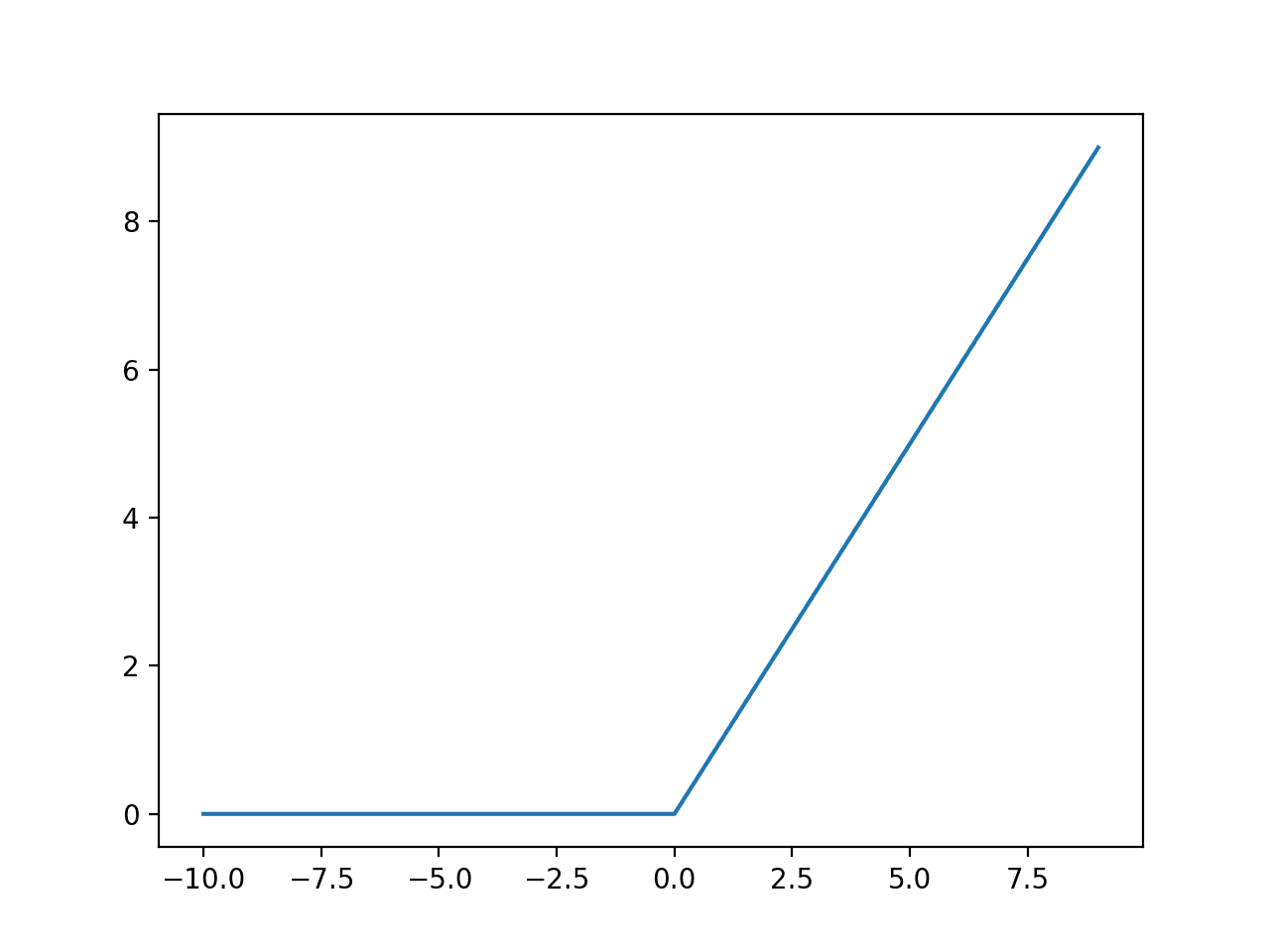
CNNs consist of several layers, each containing one or more kernels, which will perceive small sections of the input image. The results from all the collections in a layer partially overlap to create the entire image representation. This new image is then passed onto the next layer and the process repeats, allowing the system to learn about the image composition[[3]](#footnote-3).

In our application, image recognition CNNs will aid users in the fruit choosing process. They do so by passing the input image of a fruit through each layer of the network, which we have coded to detect different features in the fruit (e.g. colour, signs of bruising), and then ultimately deciding whether the fruit is fresh or rotten, and how ripe the fruit is.

Performance of CNN

CNNs have been known to yield state-of-the-art results on challenging datasets, which often consists of many diverse and nuanced classes. Since there can be varying numbers of kernels at each layer of the network, CNNs can combine the benefits of kernel convolution, meaning that they learn sparse representations of features, such as edges, corners, lines, and even textures (Zachary 2019).

This was a crucial point of consideration for our group when choosing a ML algorithm/technique to implement. This is because the same fruit classes come in a wide variety of colours and textures at different stages of ripeness, making it visually challenging to distinguish fresh from rotten, especially when there are certain parts of the fruit that indicate a different level of freshness from the rest.

Hence, our group has chosen to use CNNs in order to exploit its accuracy in image classification, as CNNs will be able to detect various features of the fruit over many deep layers, allowing accurate classification of the freshness of the fruit.

VGG16

We chose VGG16 as our first proposed model to train our data. VGG16 is a pre-trained model that allows us to save time spent on building our own neural network from scratch. Additionally, the model reduces the number of parameters, by incorporating three non-linear rectification layers instead of a single one, which makes the decision function more discriminative.

Benefits of Proposed Model

VGG has several properties that make it distinct from standard CNNs (Simonyan, Zisserman 2015). They are:

* Use of 3\*3 convolutional kernel size instead of larger 5\*5 or 7\*7 kernel with stride of 1
* Use of the RELU function
* Use of max pooling instead of average pooling.

One factor that distinguishes the VGG16 model from the other CNNs is the use of smaller 3\*3 convolutional kernel size. Convolution layers in a model help to identify certain features in an image being processed. The interesting feature about these convolutional layers is that the larger 5\*5 and 7\*7 kernels can actually be broken down into smaller pieces using the 3\*3 convolutional kernels. Using smaller kernel sizes also allows for the model to have more convolution layers without using too much computer resources.

Pooling layers help to reduce the size of the feature maps in a convoluted neural network. When a pooling layer is applied, the features of an image that has been detected by the convolutional layers before it gets condensed down and summarised.

There are two pooling algorithms that can be used: average pooling and max pooling. Each pooling filter usually consists of a matrix of size 2\*2 that moves along the previous layer image by a stride of 1.

In average pooling, the values recorded by each of the 4 squares in the 2\*2 filter is summed up and the average value is passed on. In max pooling, the highest value among the 4 squares in the 2\*2 filter is recorded.

Using the max pooling method is preferred today as it allows the most prominent feature in the image to be made more visible[[4]](#footnote-4).

Fig. 1 *Graphical representation of the RELU activation function*

The above figure shows the graphical representation of the RELU activation function. Any positive value will retain its positive value while any negative values get reduced to 0. Unlike nonlinear functions such as sigmoid or tanh, this function is used to avoid the vanishing gradient problem where the information gained within each additional layer would be decreasing, causing the effective learning of complex networks to be reduced[[5]](#footnote-5).

Training a VGG16 Based Model to   
Predict Freshness of Fruits

For an application of such a scale, where we need to classify whether various fruits are fresh, we will need a large amount of data. We have found a large dataset[[6]](#footnote-6) which comprises an image dataset of fresh fruits and rotten fruits, including apples, bananas and oranges, that have been pre-allocated as training examples and test sets. There are a total of over 10,000 training examples, and each image will be pre-processed into a numpy array with dimensions:

Horizontal\_pixel\_value \* vertical\_pixel\_value \* 3  
(3 because of RGB values)

This means that training time and space consumption increases quadratically with resolution, resulting in a possible space+time/accuracy tradeoff. These important requirements can be satisfied using CNN, where we control the resolution of processed images (and to choose the optimal resolution by trial and error) to reduce training time.

After the model has been trained, users can upload a picture and use the model to predict whether the fruit is fresh or rotten. If the model predicts that the image contains fresh fruit, it will print “fresh” and “rotten” otherwise, which will help our users in their decision-making process.

However, like all other machine learning applications, the output is merely a prediction. If the picture is supposedly a fresh fruit but the model predicts a rotten, the grocer will miss out on fresh produce. In contrast, when the picture is supposedly a rotten fruit but the model predicts a fresh, it might cause food poisoning or other health problems when an unknowing person consumes it. Therefore, the application should be used as an aide in choosing fruits, but not a replacement.

Currently our model only predicts the quality of the fruits for only a few fruits namely apples, bananas and oranges. As an experiment, our team decided to use our trained model and see if it could still function well in detecting the quality of other kinds of produce through transfer learning. We proceeded to take another sample of another fruit which the model has not seen before and see its accuracy in determining the quality of the fruit.

MobileNETV2

The MobileNetV2 is a convoluted Neural Network that can also be used for our ripeness recognition application. The unique property of this network is its ability to achieve good image recognition performance while at the same time reduce the number of calculations needed to execute the model (Sandler et. al. 2013).

Properties of Proposed Model

The technique used in the convolution is known as DepthWise Separable Convolution[[7]](#footnote-7). Suppose that we have an image that is 10 pixels by 10 pixels and it has three colour channels. We would like to create a convolution layer that outputs 256 channels. Assume that we have a convolutional kernel that is 3 pixels by 3 pixels.

In the standard convolution algorithm, we can set the step size to 1. In our example, the 3\*3 convolutional kernel needs to move through the image 8\*8 times. At each stage, it needs to make 27 multiplications. With a target output channel of 256, that will produce the total number of calculations to be 27 \* 8 \* 8 \* 256 = 442,368 computations required.

If we however separate the calculations depthwise, we split the calculation into two stages. The first stage is known as Depthwise Convolution. We will first apply the convolution calculation to each of the three individual layers rather than as an entire image. In this case, each layer will be convoluted with a kernel that is 3\*3\*1 pixels in size. The total number of computations required for now is 9 \* 8 \* 8 \* 3.

Next, now that we have our three convoluted layers, we will pass it through the next stage known as pointwise convolution. This is where the three layers obtained just now are combined, and they will be passed through a convolutional kernel of 1\*1\*3 pixels. Since we have a target output of 256 channels, the total number of calculations is 256 \* 3\*8\*8. If we total the two stages, the number of computations required is 50880, which is significantly lesser than the previous process.

Benefits of Proposed Model

The process of using the MobileNetV2 may be more beneficial for our application as the reduction in the number of computations will help the application run better in mobile applications where the amount of computational power is limited.

Furthermore, VGG is a model that is optimised for image classification between different objects as opposed to classifications on the same object. As a result, the trained models using VGG16 were less accurate than those using   
MobileNETV2.

Training a MobileNETV2 Based Model to   
Predict Ripeness of Fruits

We found a small dataset[[8]](#footnote-8) which comprises an image dataset of ripe, underripe and overripe bananas. There are a total of 250 training examples, and similar to the previous dataset, each image will be pre-processed into a numpy array with dimensions horizontal\_pixels \* vertical\_pixels \* 3 (Image RGB Values).

Once the model has been trained, users can upload a picture and the model will give a prediction of its ripeness level. Depending on the ripeness, the model will print “ripe”, “underripe” or “overripe” respectively.

To overcome the limitation on dataset, we could explore using images from user’s camera devices to improve and/or validate our model. For example, once a user used the application to predict the freshness of the fruit, we could prompt the user to validate whether our prediction was correct.

Final Trained Model

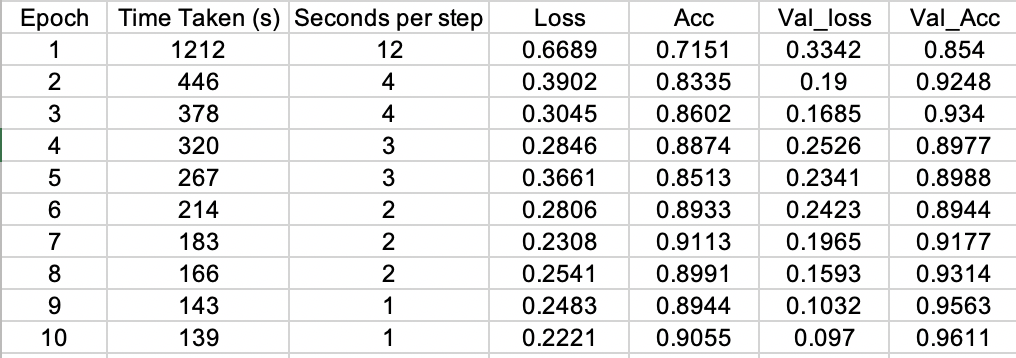


Table 1 *Results of the Freshness Model*

As seen from Table 1, using VGG16, we were able to obtain a final accuracy of 0.9611 and loss of 0.097 on validation dataset. 10 epochs was used to train the model, with 100 steps per epoch.

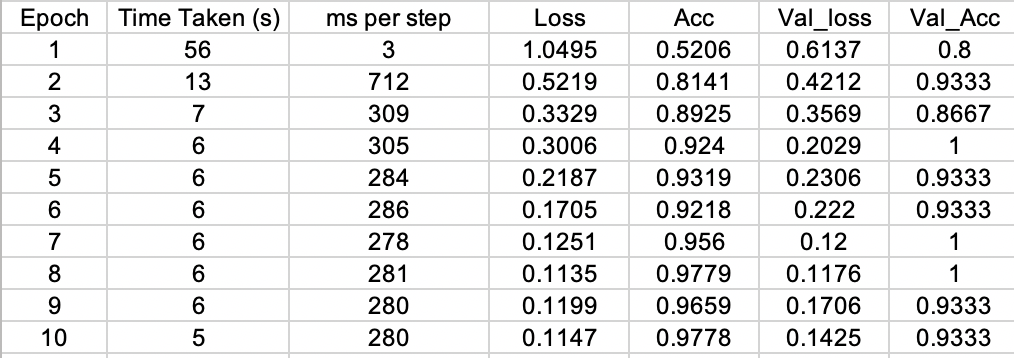


Table 2 *Results of the Ripeness Model*

Similarly from Table 2, using MobileNetV2, we were able to obtain final accuracy of 0.9333 and loss of 0.1425 on the 15 test examples, with standard deviation 0.06289. 10 epochs was used to train the model, with 20 steps per epoch.

Although our current ripeness model is trained using banana data, to further test our model, we selected a few test inputs from Google images. To make sure our model could be applied to other types of fruits, we tried to select fruits that would have looked similar during different levels of freshness and ripeness.

In Figures 2, 3 and 4 below, we observe that the underripe apples and oranges share a similar green undertone to an underripe banana. Whereas in Figures 5, 6 and 7, we observe that the overripe apples and oranges have several black spots that are also apparent on overripe bananas.



Fig. 2 *Underripe Apples*



Fig. 3 *Underripe Oranges*

  
Fig. 4 *Underripe Bananas*

Fig. 5 *Overripe Apples*

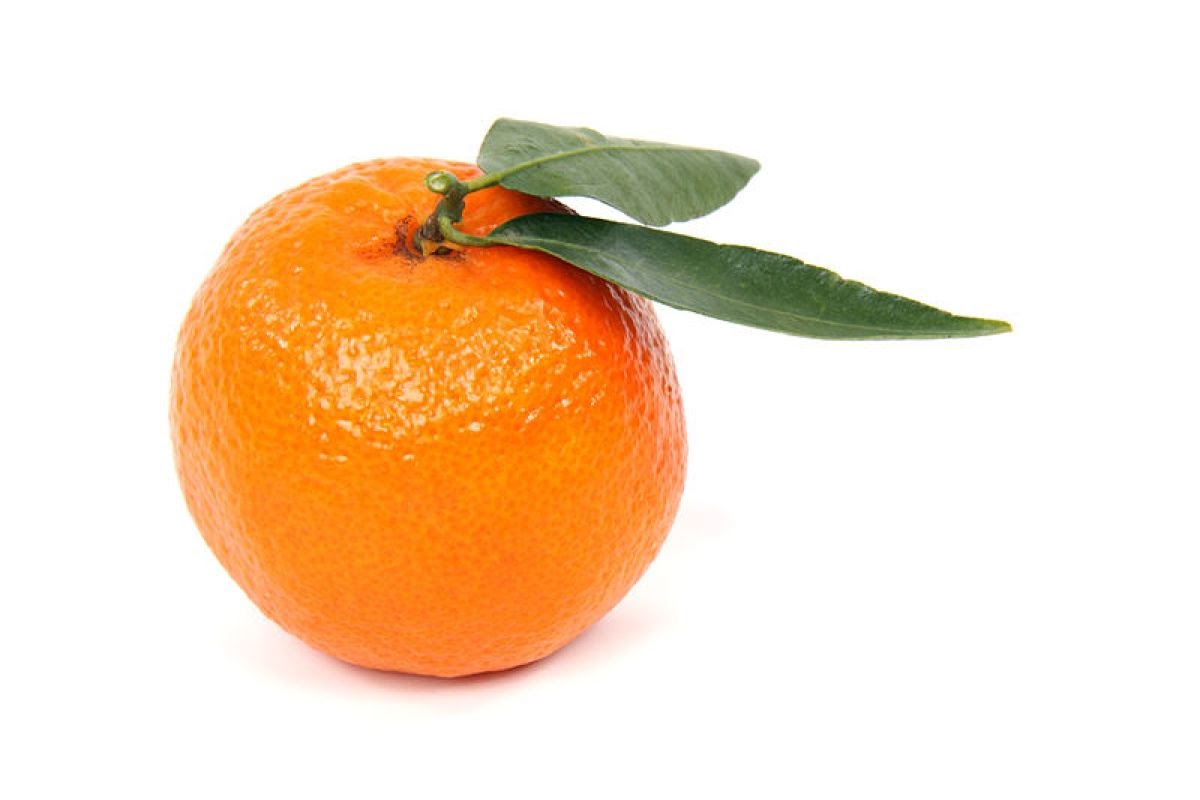


Fig. 6 *Overripe Oranges*



Fig. 7 *Overripe Bananas*

As such, with the similarities in the fruits, we were able to obtain the following results in Table 3.

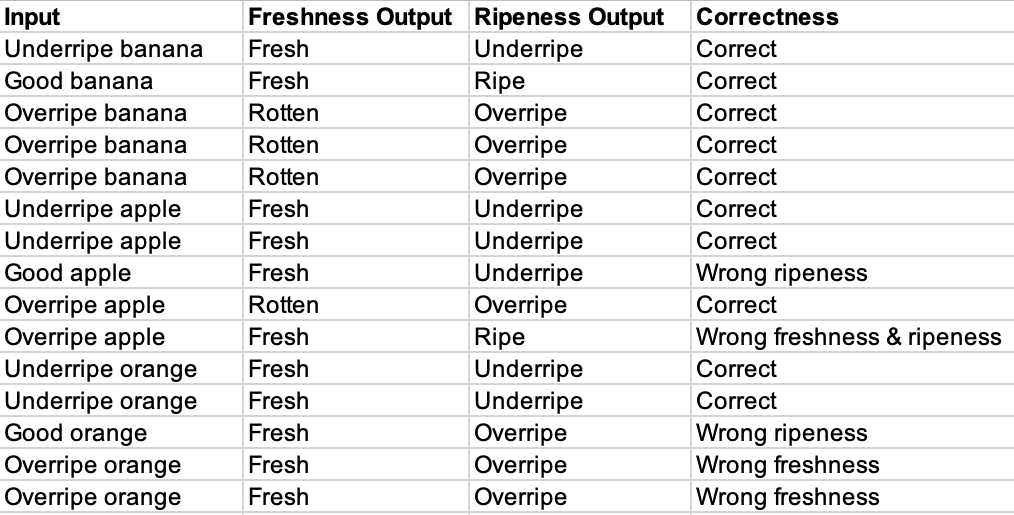


Table 3 *Prediction on unseen data using our final models*

The accuracy on both models was 80%. All the banana inputs were correctly classified by both models, whereas for apples and oranges, there were some misclassifications. Taking a closer look at the inputs that were misclassified by the ripeness model, we attempt to provide reason for the inaccuracy.

Fig. 8 *Ripe Orange Misclassified as Overripe*

Fig. 9 *Overripe Apple Misclassified as Ripe*

Fig. 10 *Ripe Apple Misclassified as Underripe*

In Fig. 8, Fig. 9 and Fig.10 are images that were wrongly classified. They suggest that our ripeness model is less accurate in correctly classifying fruits that do not look similar to the banana at the respective stage of ripeness. For instance, ripe apples and oranges are red and orange in colour respectively, but ripe bananas are yellow in colour.

This further implies that if we test our model on fruits such as blackberries, it is likely that they will all be classified as overripe. Therefore, to increase the accuracy of our model, we should train it with a variety of fruits.

Conclusion

From our experiment, we can conclude that there are limits to the use of transfer learning. While transfer learning using predetermined weights in the CNN can help to speed up the training process, we must be careful in selecting fruits that have similar visual properties with one another to avoid misclassification by the model.

In addition, the small dataset used to train the ripeness model may not have provided the model with enough information to distinguish between different fruits with similar visual properties.

Hence, if a larger dataset with more images of similar fruits is used to train the model, the model might be able to correctly classify the fruits. image recognition performance while at the same time reduce the number of calculations needed to execute the model (Sandler et. al. 2013).

Team Members’ Contributions

Ho Ming Jun

* VGG16 Model
* Writing of project report

Kyi Nuu Khin Khin

* MobileNETV2 Model
* Writing of project report

Ng Ngai Teng Colin

* VGG16 Model
* Writing of project report

Rani Karthigeyan Rajendrakumar

* VGG16 Model
* Writing of project report

Toh Hai Jie Joey

* MobileNETV2 Model
* Writing of project report

Toh Hong Xian (Ray)

* MobileNETV2 Model
* Writing of project report

References

Espinoza, Alejandro Zachary. (2019). An Application of Deep Learning Models to Automate Food Waste Classification. *Dissertations and Theses*. *Paper 5365.* doi:10.15760/etd.7238

Sandler, Mark & Howard, Andrew & Zhu, Menglong & Zhmoginov, Andrey & Chen, Liang-Chieh. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 201 pp. 4510-4520*. doi:10.1109/CVPR.2018.00474.

Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Vision and Pattern Recognition. Xiv 1409.1556.*

1. Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved. [↑](#footnote-ref-1)
2. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [↑](#footnote-ref-2)
3. https://www.altexsoft.com/blog/image-recognition-neural-networks-use-cases/ [↑](#footnote-ref-3)
4. Jason Brownlee, A Gentle Introduction to Pooling Layers for Convolutional Neural Networks, April 2019.

   https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/ [↑](#footnote-ref-4)
5. Jason Brownlee, A Gentle Introduction to the Rectified Linear Unit (ReLU), January 2019.

   https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ [↑](#footnote-ref-5)
6. Kaggle, Fruits Fresh and Rotten for Classification, August 2018. https://www.kaggle.com/sriramr/fruits-fresh-and-rotten-for-classification [↑](#footnote-ref-6)
7. Chi-Feng Wang, A Basic Introduction to Separable Convolutions, Aug 2018. https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728 [↑](#footnote-ref-7)
8. Giovanni Carvalho, Banana Ripeness Classification, March 2019. https://github.com/giovannipcarvalho/banana-ripeness-classification [↑](#footnote-ref-8)