

Image Recognition for Fruit Stores Using Deep Learning

José Luis Rojas Aranda José Ignacio Núñez Varela

UASLP, Facultad de Ingeniería
Fundamentos de Inteligencia Artificial

November 29, 2019

Abstract

The purpose of this investigation is to create a intelligent system for a Fruit Store that makes the checkout process easier. For this we propose using a Convolutional Neural Network, and during training we use techniques like transfer learning, fine tuning and data augmentation in order to make the system robust with a small dataset. To test this we introduce a new dataset of fruits images, and explore what will it take to make the system affordable and scalable.

1 Introduction

Whether in a supermarket or in your local store, we all have been there; when the cashier cannot remember what is the code for the fruit or vegetable we are trying to buy, thus making the checkout process inefficient, but what if Deep Learning can help us solve this? When we hear about Deep Learning we generally think of self driving cars or some high tech industry, but this technologies can help a lot of sectors. So we propose a deep learning model that is capable through image recognition, classify the fruits during the checkout process. The complexity of this task is big due to all the variations in a single type of fruit that there exists.

To solve this we use Convolutional Neural Networks that are powerful classification models and use an architecture called MobileNet, that is both light weight and efficient. On the other hand we also consider that for this algorithms to work we need a lot of data, the catch here is that quality really matters, thus making the process of data gathering very expensive. To alleviate this problem, we use a technique called transfer learning to improve the accuracy and robustness of the model on small datasets.

2 Deep Learning

Deep Learning is sub-field of machine learning, that uses stacks of artificial neural networks to extract features from an input, Deep Neural Networks are able to outperform other machine

learning algorithms. AlexNet[1] was one of the first Deep Convolutional Neural Network architecture, very similar as that of LeNet[2] but deeper, and used GPUs to reduce training times. AlexNet vastly outperformed every state-of-the-art algorithm on image recognition in the ImageNet competition back in 2012. Since then, this kind of technology has become more accessible.

2.1 Convolutional Neural Networks

Convolutional Neural Networks(CNN's) are the way to go when it comes to image recognition, this is because they are very good at pattern recognition and processing higher input sizes. These kind of neural networks are composed mainly by 3 types of layers: convolution layers, pooling layers and fully connected layers.

2.1.1 Convolution Layer

Given an input image of $n \times n \times n_c$ and a kernel of smaller shape $f \times f \times n_c \times n'_c$ where n'_c is the number of total filters of the layer, the layer is parameterized by the kernel. A convolutional layer convolves the kernel and the input, producing a feature map. Figure 1 shows how a convolution operation is performed.

The two purposes of a convolutional layers are:

1. To extract high level features, such as edges or more complex features.
2. Reduce the input size. This is not always the case, it depends on the padding and the stride size.

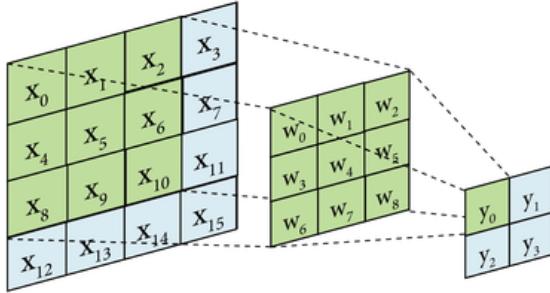


Figure 1: Example of convolution operation in a matrix.

2.1.2 Pooling Layer

A pooling layer has no parameters, its purpose is to reduce the input size. Given a receptive field of size $f \times f$, a pooling layer can perform one of the following operations:

1. Max Pool: Takes the max element of the receptive field.
2. Min Pool: Takes the min element of the receptive field.
3. Average Pool: Takes the average of the receptive field.

2.1.3 Fully Connected Layer

Fully connected layers are the classic neural networks, where each layer l has $n^{[l]}$ neurons, where each neuron is connected to all $n^{[l-1]}$ neurons of the last layer through weights and a bias, with this learnable parameters the neuron computes some activation function.

2.2 CNN Architectures

Since AlexNet, there has been a lot of research when it comes to designing CNN architectures. Which architecture to use depends on the problem you are trying to solve, there isn't one best for all cases, and is still an area with a lot of research.

2.2.1 Going Deeper

One thing that at first seemed to be improving the CNN's was to add more layers, but it reached a limit. It is shown that the deeper the net becomes, it makes the optimizer struggle to find an optimal solution due to exponentially increasing weights[3].

One solution to this problem are Residual Blocks, that allow to propagate deeper the input signal. They do this by adding the input of the convolution block at the end of the same block, allowing that at least the identity function is retained at the end of the block. The first architecture that used this Residual Blocks was ResNet, some versions of this architecture have more than 100 of these blocks. Currently the top accuracy on ImageNet uses a ResNet architecture, and achieve an accuracy of 98% (better than humans).

2.2.2 Computational Complexity

One disadvantage of the big CNN architectures is the computational complexity. For example, a normal convolution has a computational cost of:

$$n \cdot n \cdot n_c \cdot n'_c \cdot f \cdot f$$

This is not a big problem when doing research thanks to GPUs, but when it comes to using this technology in real world applications, we need to consider these limitations.

2.3 MobileNets

MobileNets are CNN models that solve the computational complexity problem, by making an architecture that is more efficient, and that can run on mobile or embedded devices and still achieve top level performance.

They achieve this by dividing a normal convolution into two steps. First, it performs a point-wise convolution, then a depth-wise convolution[4]. With this change the computational complexity of the operation reduces to:

$$n \cdot n \cdot n_c \cdot f \cdot f + n_c \cdot n'_c \cdot f \cdot f$$

MobileNetV2[5] uses inverted residual blocks and a bottleneck layers to achieve better performance. Recently MobileNetV3[6] paper was released, in which they use novel techniques like Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP) to achieve even better performance with mobile CPUs.

3 Dataset

Machine learning has created a new paradigm in the computer science world, because in order to create models, we need to also think in terms of data. As Andrej Karapathy in his great blog post *A Recipe for Training Neural Networks*[12] says, "you need to become one with the data". So dedicating the time to analyze the data, gives us great answers and can save us time.

3.1 Fruits 360

One dataset in which the model was trained is the Fruits 360 dataset [13] which contains images of 118 fruits and vegetables, we only used images of Apples, Oranges and Bananas. Figure 2 is an example of different images of the training set, and the problem here is obvious, and its that there is not enough variety in the data in order for the model to generalize for the problem we are trying to solve.



Figure 2: Example of training images of the Fruits360 dataset.

3.1.1 Impact of Dataset Size

We also perform tests on seeing the impact of the dataset size [14], for this we found that quality really matters. In the following Table 1 we see how the model performs in subsets of the Fruits360 dataset, and see that it can get high accuracy ($> 95\%$) even with a subset of size eighteen. Meaning that the general approach of gathering more data to improve the system matters most, if we introduce more variety on the data

Dataset Size	Category Size	Test Accuracy
1461	490	0.99
488	164	0.95
43 (custom)	14	0.99
18	6	0.99

Table 1: Model accuracy trained in Fruits360, comparing different subset sizes of the dataset

3.2 Creating a Dataset

The Fruits 360 dataset is not enough to solve the problem, this is why we embark into the task of creating our own dataset. Doing this can give us an idea of how much data is really needed to solve the problem, and how difficult is to gather this data. Generally it is said that you need 1,000 images per categories to make a good classifier, this idea comes from the ImageNet dataset which contains 1,000 images per category, but this is not a good answer to the data problem. Because as demonstrated earlier, dataset size is relative and more does not generally mean better, so we need to ask the following questions: What kind of images will the model see when deployed? and, how can we introduce variety for the model to generalize well?

The advantage here, and is that we are not trying to make a general purpose fruit detector, rather one for fruit stores, so we need to replicate the store environment. To do this the fruits are over a stainless steel sheet and the photos are taken from the top, we selected three different fruits per categories for training and other three for testing. We introduce variation in the dataset, by taking images of the fruits at different positions and rotations. We also consider that in the checkout process the fruits are generally inside a transparent plastic bag, so we also took photos of the fruits inside a bag.



Figure 3: Example of training images of the dataset created.

4 Training

The models are trained using Keras backed by TensorFlow. We use the Adam Optimizer[7] and categorical cross-entropy loss, for the hyperparameters choice we use a learning rate of $3e - 4$ and default parameters of the Keras implementation of the optimizer.

Learning rate is a very important parameter and recent research has found great progress. For example, techniques like Super-Convergence found that under certain parameters, neural networks can be trained an order of magnitude faster using high learning rates[8]. We chose not to try different configurations of hyper-parameters because is hard to find an explanation of the results. And instead use default parameters that are known to work well, so we can focus on making the model reliable.

For the training process, the dataset is processed and converted to a TFRecord using the TensorFlow Dataset API, which enable us to create an efficient input pipeline for training the model. For example, we can use this pipeline to re-scale, crop or apply data augmentation to all the dataset.

4.1 Choosing a CNN Architecture

As mentioned earlier in the Deep Learning section, there exists a lot of CNN's architectures, and knowing what to choose depends on the problem at hand. In this section we try to answer the question of what architecture to use, that is able to solve our problem and at the same time be reliable for real world applications. So we compare two architectures.

The first architecture we use to train a model, is a smaller version of VGGNet[9] called SmallerVGGNet[10]. Which is characterized using only 3×3 convolutional layers, reducing volume size by max pooling and fully-connected layers at the end of the network prior to a softmax classifier.

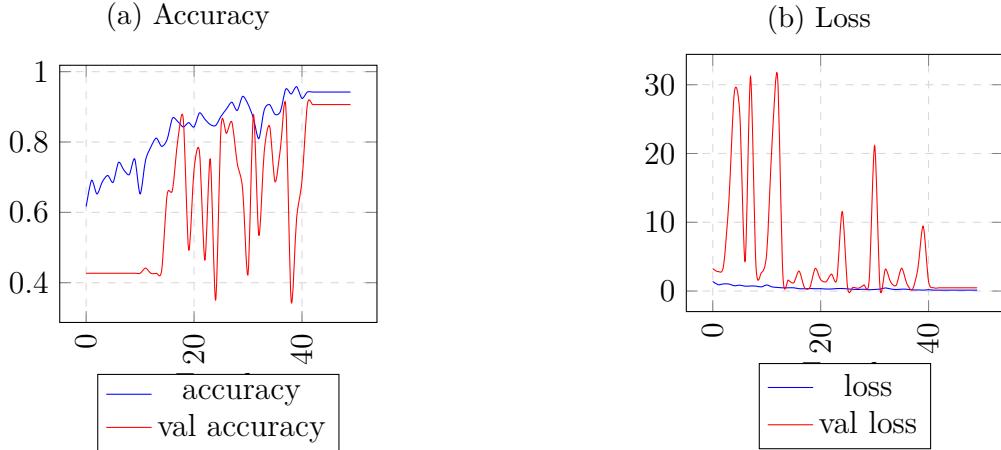


Figure 4: Training logs for SmallerVGGNet

For the MobileNetV2 model architecture[5], we are using the Keras implementation. For the width multiplier hyper-parameter, we are using a value of 0.5.

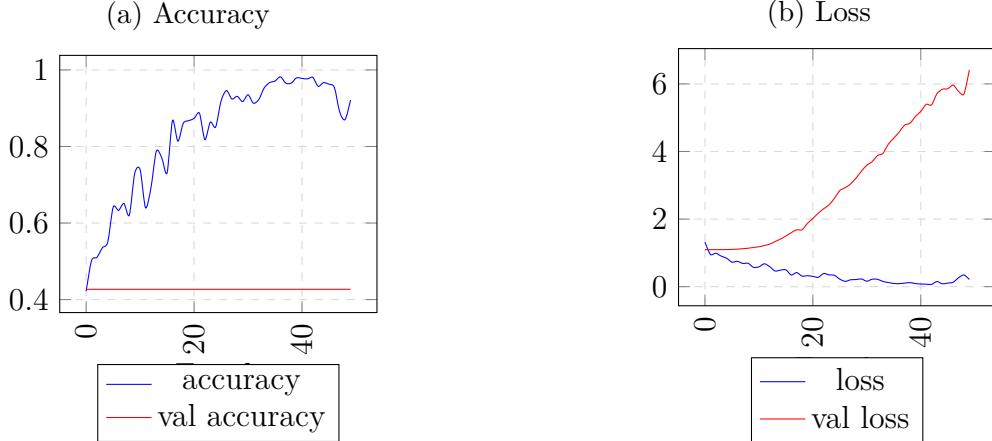


Figure 5: Training logs for MobileNetV2 with weight initialized randomly.

As we can see in Figures 4 and 5, both architectures are able to learn the training set pretty well, and we clearly see that SmallerVGGNet is a better model because is not overfitting the training set. The disadvantage here becomes obvious if we see the number of parameters in the models:

Model	Num params
SmallerVGGNet	8,676,227
MobileNetV2	2,261,827

Table 2: Model size comparisons

SmallerVGGNet has more than double the amount of parameters, this means that the model needs more computing power to make a prediction. And because we want the system to work in fruit stores, we need to consider that they may not have very good computers, thus making the system useless. One way to solve this is to gather more data but this is expensive, so in the next section we use transfer learning to improve the accuracy of the model.

4.2 Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task[11]. This kind of technique works really well when our dataset is not big enough, and it always makes the model converge faster to a solution. We trained MobileNetV2 with transfer learning using weights from a model trained in the ImageNet dataset. There are several ways you can train a model with transfer learning, in this case we trained it two different ways:

First, we load the trained model and cutoff the last layer, that is a dense layer with 1000 neurons that serves as a classifier of the previous feature map. Once cut, we set the rest of the layers to not trainable, and to the end of the network a dense layer but in this case with the number of classes of fruits we are going to predict. This enable us to keep all the feature extracted with the ImageNet model, and re-purpose it to the fruit classification problem.

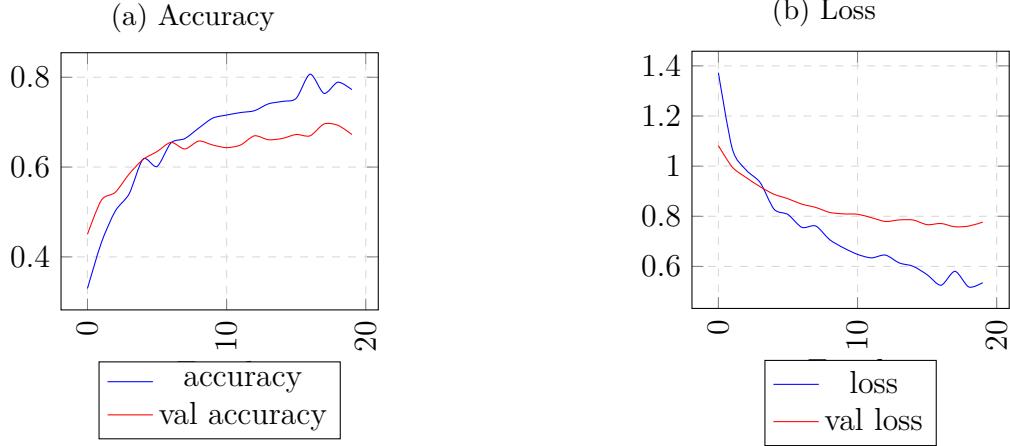


Figure 6: Training logs for MobileNetV2 with ImageNet weights and only training the last fully connected layer.

The second approach, is to also remove the last layer and add a classifier with the number of classes we are going to classify. But instead of setting the layers to not trainable (meaning that the weights are not updated), we let the optimizer retrain the full network. This makes the model start training at some point and not with random values, thus making the convergence to a solution way faster and to not overfit the training dataset.

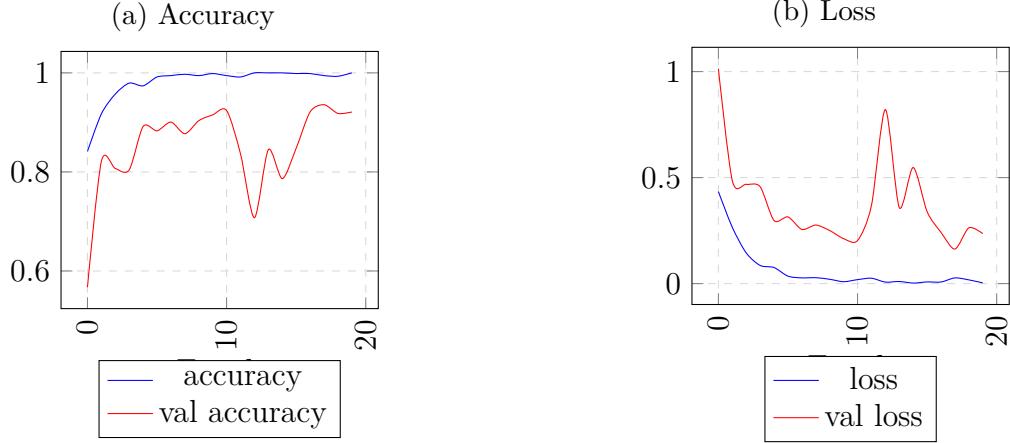


Figure 7: Training logs for MobileNetV2 with ImageNet weights and training all network.

For the first approach we get better results on the test dataset and the model is able to generalize better, as shown in Figure 6. Another big advantage is the training times, due that we only train the last layer of the model, each epoch is trained way faster.

In the second approach training is not as stable and it is doing a little overfitting, but we get +0.9% accuracy, as shown in Figure 7.

4.3 Comparisons

4.3.1 Training with Fruits 360

At the start of the project, the model was trained using the Fruits360 dataset, and the results were not good in images of our problem, but in the test and train set of the Fruits360 it performed well. This is because our problem images are not of the same distribution as those of the Fruits360, and is naive of us to expect that it will perform well. The accuracy was of 0.33% that means that is clearly not generalizing.

4.3.2 Data Augmentation

Also we trained the model using data augmentation, with random changes in hue, saturation and illumination, but did not obtain better results. We believe this is because the model is giving more weight to the patterns of the image, so the data augmentation we are applying is not improving the accuracy.

4.3.3 Final Results

In Table 3, we compare the trained models. The results show us that transfer learning makes the model perform better due to the small dataset we are dealing, SmallerVGGNet also isn't bad but, as mentioned earlier the disadvantage is the model size. Also we compare how the model performs when we remove the images of the fruits inside the plastic bag, and see that it gets better results, but the difference is not as big as we expected.

Model	Plastic Bag	Train accuracy	Test accuracy
SmallerVGGNet	Yes	0.91	0.82
MobileNetV2	Yes	0.97	0.436
MobileNetV2 + ImageNet weights (only head classifier)	Yes	0.77	0.70
MobileNetV2 + ImageNet weights	Yes	0.99	0.92
MobileNetV2 + ImageNet weights	No	0.99	0.98

Table 3: Trained model comparisons

5 Visualizing Activations

One disadvantage of using deep neural networks for machine learning models is that we don not have a good way of telling what is the model really learning, this imposes a lot of problems in the deep learning field. One technique to visualize what CNN's are learning is to

visualize the activations of the convolution layers, by doing this we can see what information is being retained by the layers.

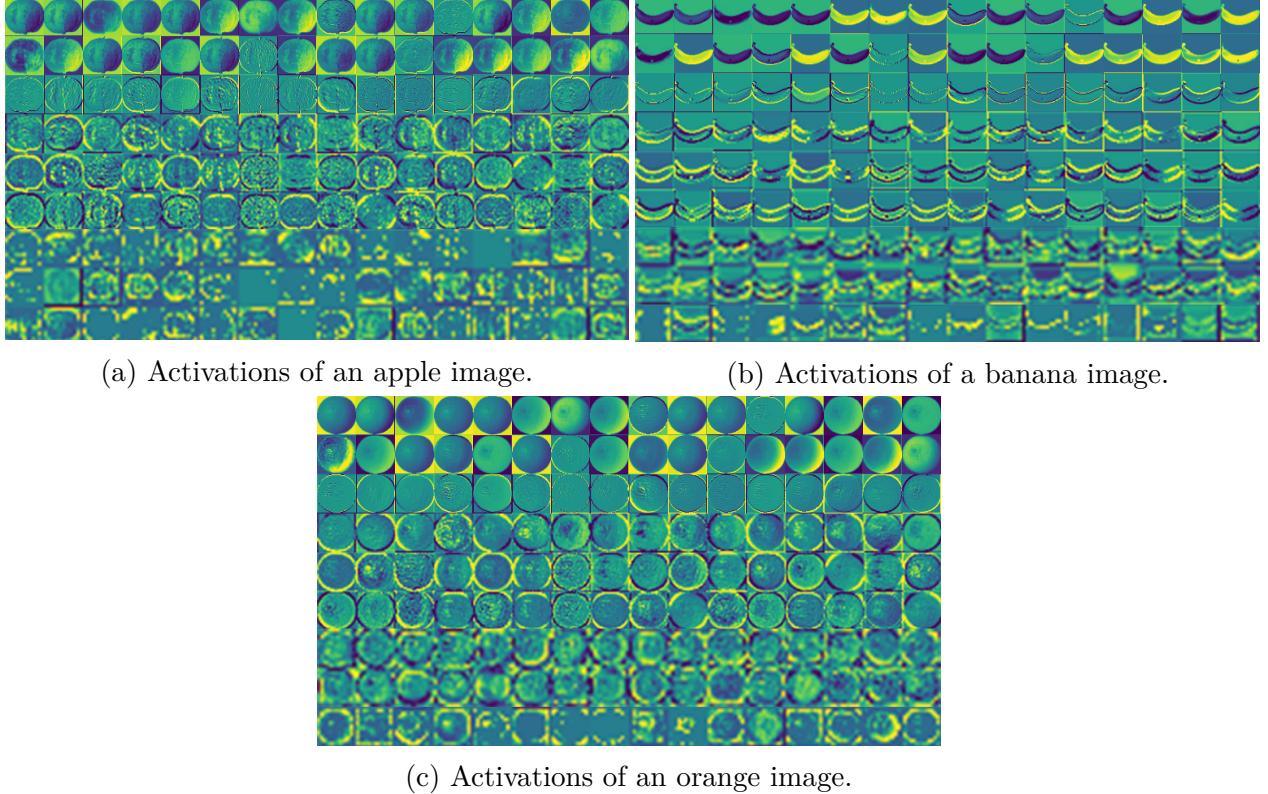


Figure 8: Visualizing activation of MobileNetV2, layers 1, 7, 16, 27, 42, 57. The model is trained in the Fruits360 dataset

When we see the activations of our model with an image of an apple, banana and an orange (Figure 8 respectively), we see that our model is retaining mainly the shape of the fruit and the texture.

With this in mind, we can say that if we want to improve the accuracy of our model for real world applications, the following can be done:

1. Gather more images of the fruit at different positions and size, so the model can learn different shapes of the same fruit.
2. Variety in the texture of the same fruit.

6 Future Work

There is deficiency in the model, CNN's are very good at pattern recognition and extracting features, but we believe that the problem here is that is not giving as much weight as it should to the colors of the image. Causing the model to miss-label images of apples and oranges, because they have very similar shapes, but different colors.

On possible way to address this problem is to create a multi-input model, this means that the models do not only takes as input only the whole image, rather the image and some numerical or categorical data. In this case data about the colors of the image, so that in the last fully connected layer of the neural network, the feature map and the color data is concatenated to produce the final prediction. To obtain the color data we can do it by hand, using a K-Means machine learning algorithm or the information of the histogram.

We saw that Convolutional Neural Networks are powerful image classification models, but for them to work well we need a lot of data, and we also saw that more is not always better. But if we use techniques like transfer learning and fine tuning, we can achieve good performance as long as we have a representative dataset.

References

- [1] Krizhevsky. A, Sutskever. I, and E. Hinton. G, *ImageNet Classification with Deep Convolutional Neural Networks*. 2012
- [2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner *Gradient-Based Learning Applied to Document Recognition*. 1998
- [3] He. Kaiming, Zhang. Xiangyu, Ren. Shaoqing and Sun, Jian. *Deep Residual Learning for Image Recognition*. 2015
- [4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* 2017
- [5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: inverted residuals and linear bottlenecks,” in 2018 IEEE Conference on Computer Vision and Pattern Recognition
- [6] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le and Hartwig Adam. *Searching for MobileNetV3*. 2019
- [7] Diederik P. Kingma, Jimmy Lei Ba. *Adam: A Method for Stochastic Optimization*. 2017
- [8] Leslie N. Smith, Nicholay Topin. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates*. 2018
- [9] Karen Simonyan, Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015
- [10] Keras and Convolutional Neural Networks (CNNs)
<https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/>
- [11] A Gentle Introduction to Transfer Learning for Deep Learning
<https://machinelearningmastery.com/transfer-learning-for-deep-learning/>

- [12] A Recipe for Training Neural Networks
<http://karpathy.github.io/2019/04/25/recipe/>
- [13] Horea Muresan, Mihai Oltean, *Fruit recognition from images using deep learning* , Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.
- [14] Xiangxin Zhu, Carl Vondrick, Charless C. Fowlkes, Deva Ramanan. *Do We Need More Training Data?*. 2015