Image Recognition for Fruit Stores using Deep Learning

Jose Luis Rojas Aranda José Ignacio Núñez Varela

UASLP, Facultad de Ingeniería
Fundamentos de Inteligencia Artificial

September 14, 2019

1 Introduction

The aim of this project is to create an intelligent system for fruit stores, that is able to classify different fruits, thus making the check out process more efficient and with out the need of human intervention. In order to achieve this the system needs a machine learning model that is able to achive good accuracy and good performance.

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. Since AlexNet[1] vastly outperformed every state of the art algorithm on image recognition in the ImageNet competition back in 2012, this kind of technology has become more accesible

2.1 Deep Convolutional Neural Networks

Convolutional Neural Networks are the way to go when it comes to image recognition, this is because they are very good at pattern recognition and processing bigger 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(tensor more generally) 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 and producing a feature map. Figure 1 shows how a convolution operation is perform.
The 2 purposes of a convulutional layers are:

1. To extract high level features, such as edges or more complex featurse.

2. Reduce the input size. This is not always the case, it depends on the padding and the stride size.

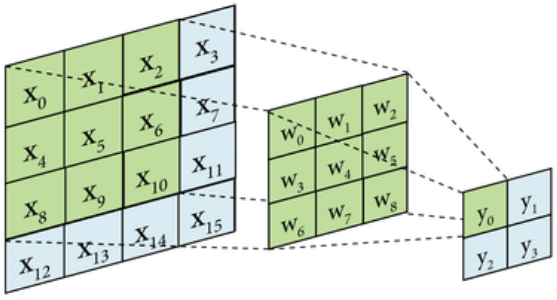

Figure 1: Example of convolution operation in a matrix.

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 max 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[1], there has been a lot of investigation when it comes to designing CNN architectures. Until today the top accuracy on ImageNet uses a ResNet architecture, with a top 5 accuracy of 98%. 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, this reached a limit, because it is shown that the deeper the net becomes it makes the optimizer struggle to find an optimal solution due to exponentially increasing weights [2]. One solution to this problem are Residual Blocks that allow to propagate deeper the input signal. For example most of ResNet's models have more than 100 layers!

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 production, we need to consider this limitations.

2.3 MobileNets

MobileNets are CNN models that addresses the computational complexity problem, by making an architecture that is more efficient, that can run on mobile and embedded devices and still achieve top level performance.
They achive this dividing a normal convolution into two steps[3], first it performs a point wise convolution then a depth wise convolution, 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[4] uses inverted residual blocks and a bottle neck layers to achive better performance. Also in the past week MobileNetV3[5] paper was realeased 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

The dataset in which the model will be train is the Fruits 360 dataset [8] which contains images of 118 fruits and vegetables. Figure 2 is an example of an image from the training set.

4 Training

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, currently no data augmentation is applied.

Figure 2: Example of training dataset image of an apple.

Training setup the models are trained using Keras backed by Tensorflow. We use the RMSProp optimizer with a learning rate of 0.0001, batch size of 16 and categorical crossentropy loss. The number of epochs depend on the model.

4.1 Smaller VGGNet

The first model we train is a smaller version of VGGNet called SmallerVGGNet[6], 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. The model is trained with 20 epochs. As shown by the graph below, good accuracy is achived in the trainig dataset since the beginning and validation accuraccy starts getting good results from epoch 5. Training graph shown in image 3.

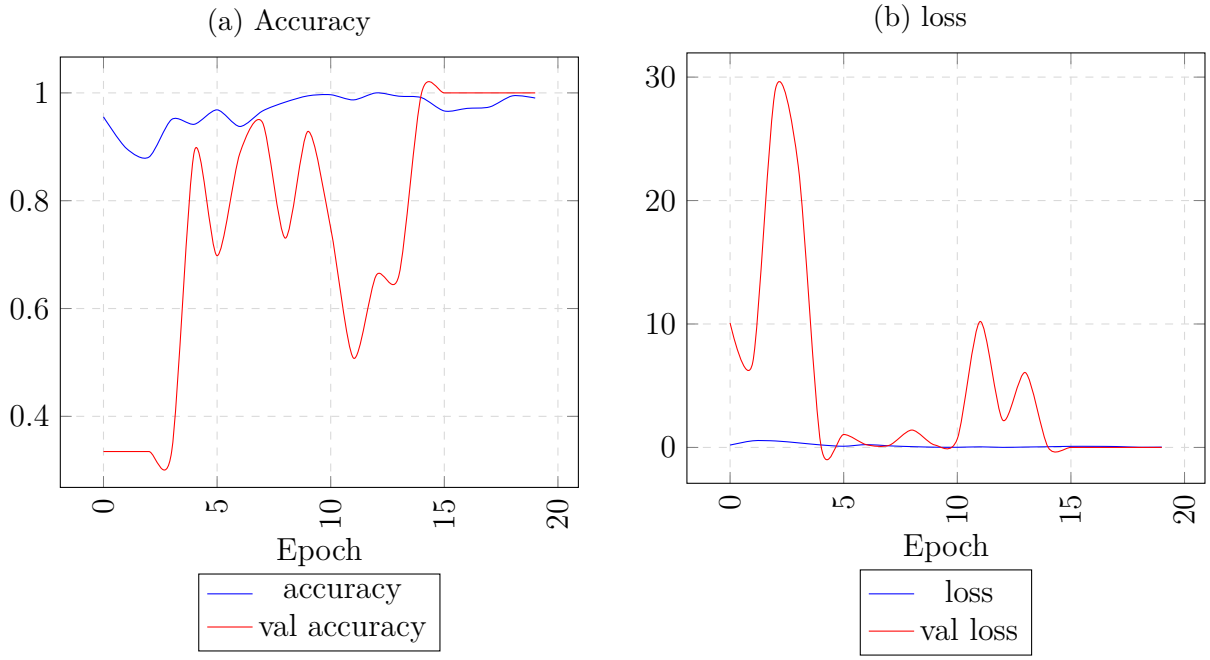

Figure 3: Training logs for SmallerVGGNet

4.2 MobileNetV2

For the MobileNetV2 model architecture[4], we are using the Keras implementation, this has several advantages. First it enable us to load trained weights trained in common datasets like ImageNet. And second has flexibility for the input size of the model, by default the input size of MobileNetV2 is (224, 224, 3) but the images in our dataset are (100, 100, 3).

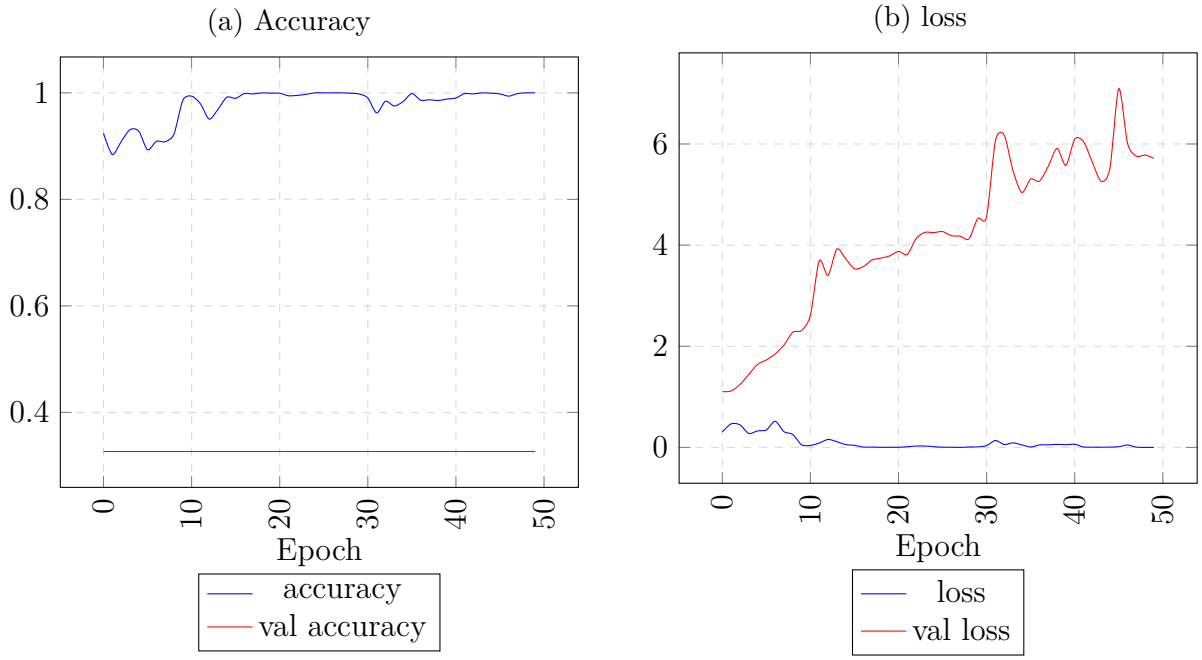

Figure 4: Training logs for MobileNetV2 with weight intialized randomly

For the training of the full model, we see that it is able to have a really good accuracy from the beginning, but is unable to generalize and ends up overfiting the train dataset. Training graph shown in image 4.

4.2.1 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[7]. This kind of techniques works really well for when our dataset isn't 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 ImageNet. There are several way you can train a model with transfer learning, in this case we trained it two ways:

First we load the trained model and cutoff the last layer, that is a dense layer with 1000 neurons, this 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 repurpose it to the fruit classification

problem. With this approach we get better results on the test dataset and the model is able to generalize better. Another big advantage is the training times, due that we only train the last layer of the model each epoch is trained way faster. Training graph shown in image 5.

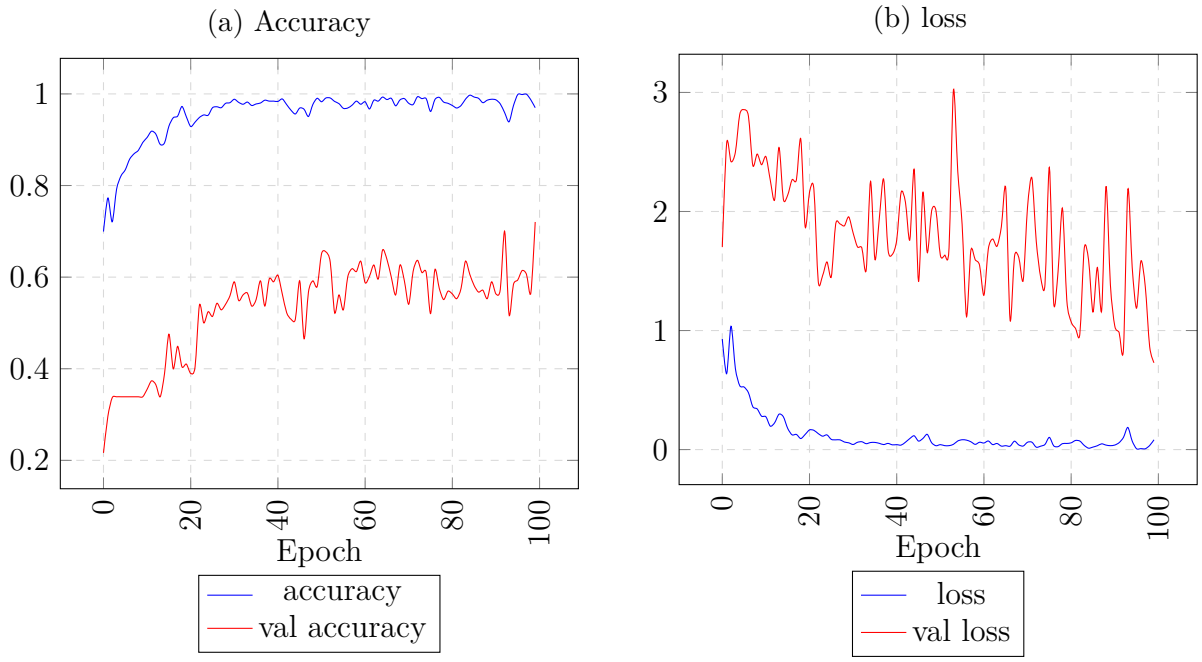

Figure 5: Training logs for MobileNetV2 with ImageNet weights and only training the last fully connected layer.

The second approach is to also cut 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, 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. Training graph shown in image 6.

4.3 Comparisons

In total there were 3 models trained, the best overall model was MobileNetV2 using transfer learning from ImageNet weights and training all layers, this model achieved perfect classification in the training dataset and in the test dataset. One of the main reason the be more focused in a more complex model like MobileNetV2 instead of SmallerVGGNet even though the current problem is quite simple, is that MobileNetV2 will always outperform because is designed to achive very good performance in the most complex classification task out there.

One of the problems of training more complex models is that for the training process you need a computer with powerful GPU's if you don't want to wait days to the model finish training. When using transfer learning training is easier because it doesn't requiere many epochs to arrive to a good solution, for example when we train MobileNetV2 with ImageNet weights and retraining all layers it only required 10 epochs to achive acurracy of 1.0.

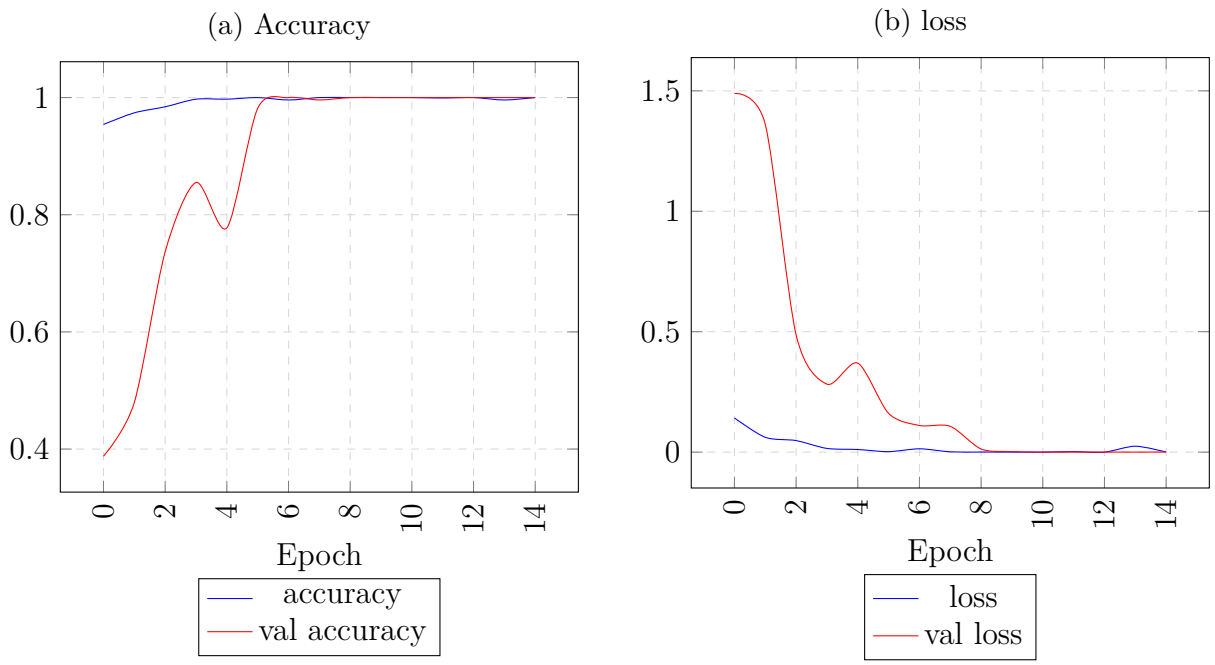

Figure 6: Training logs for MobileNetV2 with ImageNet weights and retraining all layers.

A good way to tell if a model architecture isn't as efficient is to look at its number of parameters that it has, we can see a comparisons of the model in the table 1.

Model	Num params	Train accuracy	Test accuracy
SmallerVGGNet	$8,676,227$	0.96	1.0
MobileNetV2	$2,261,827$	1.0	0.326
MobileNetV2 + ImageNet weights (only head classifier)	$2,261,827$	0.98	0.65
MobileNetV2 + ImageNet weights	$2,261,827$	1.0	1.0

Table 1: Trained model comparisons

5 Experiment Results

Also some tests were done on images different to the test dataset, for purposes of checking how well it performs with more general images. As show in the figure 7. There is still more tests to be done for the next report.

References

[1] Krizhevsky. A, Sutskever. I, and E. Hinton. G, *ImageNet Classification with Deep Convolutional Neural Networks.* 2012

Figure 7: Example of image classification using MobileNetV2 model.

[2] He. Kaiming, Zhang. Xiangyu, Ren. Shaoqing and Sun, Jian. *Deep Residual Learning for Image Recognition.* 2015

[3] 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

[4] 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

[5] 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

[6] Keras and Convolutional Neural Networks (CNNs)
`https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/`

[7] A Gentle Introduction to Transfer Learning for Deep Learning
`https://machinelearningmastery.com/transfer-learning-for-deep-learning/`

[8] Horea Muresan, Mihai Oltean, *Fruit recognition from images using deep learning* , Acta Univ. Sapientiae, Informatica Vol. 10, Issue 1, pp. 26-42, 2018.