Image Recognition for Fruit Stores using Deep Learning

Jose Luis Rojas Aranda José Ignacio Núñez Varela

UASLP, Facultad de Ingeniería
Fundamentos de Inteligencia Artificial

October 26, 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 accessible

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 features.

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

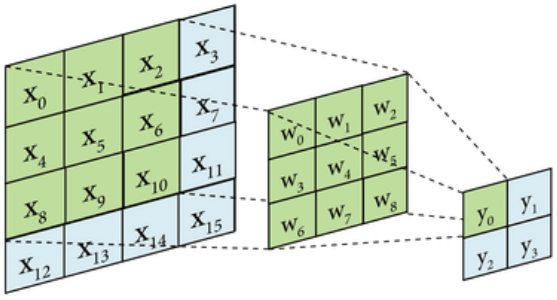


Figure 1: Example of convolution operation in a matrix.

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[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 achieve 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 achieve better performance. Also in the recently MobileNetV3[5] 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

Currently much of the progress in the machine learning world of today is due to big data, this is because this algorithms works better with more data. The problem with data is that it isn't cheap.

3.1 Fruits 360

One 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. The problem with this dataset its that there is not enough variety in order for the model to generalize for the problem we are trying to solve, for this reason we also perform tests on seeing the impact of the dataset size [9].

3.2 Creating A Dataset

The Fruits 360 dataset isn't 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 much data is really needed to solve the problem and how difficult is to gather this data.

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

For the model to perform well we need that the training data to be as close as possible as those that the model will see in production, the advantage here is that we are not trying to make a general purpose fruit detector, rather one for fruit stores. So we need to replicate the images, to do this the fruits are over a stainless steel sheet and the photos are taken from the top. We selected 3 different fruits per categories for training and other 3 for testing, and we did combinations of different position, rotation and amount of fruits in the photo. We also need to consider that in the checkout process the fruits are inside a transparent plastic bag, so we also took photos of the fruits inside a bag.

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.
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 4.

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

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).

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 5.

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

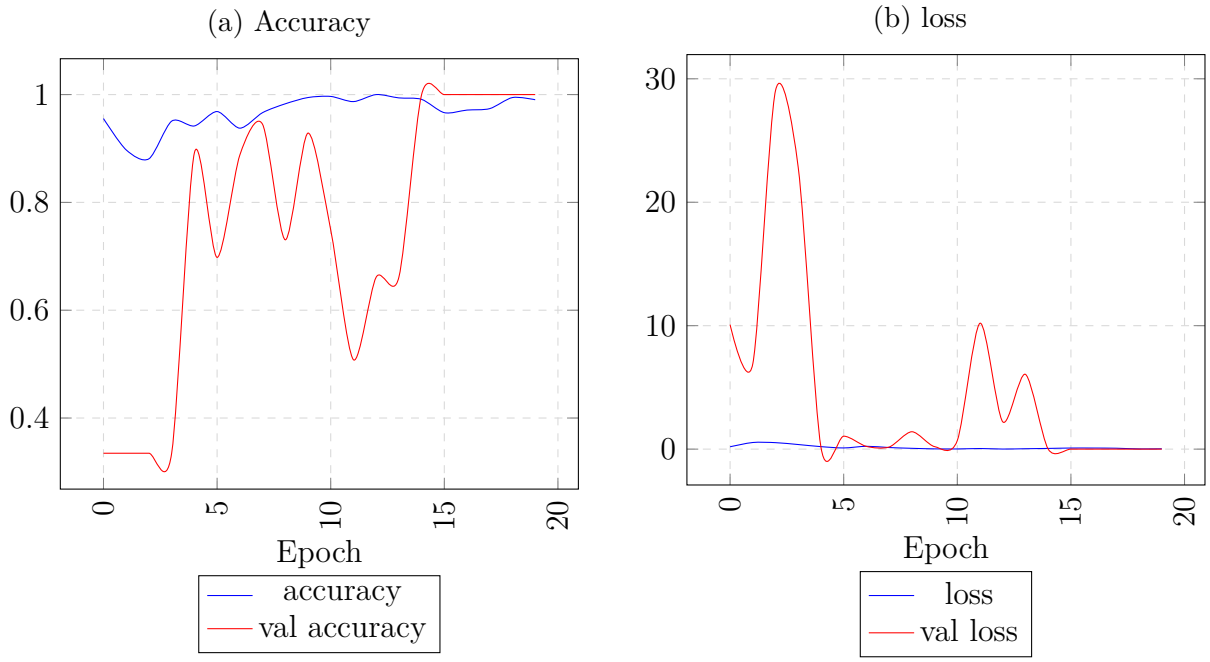


Figure 4: Training logs for SmallerVGGNet

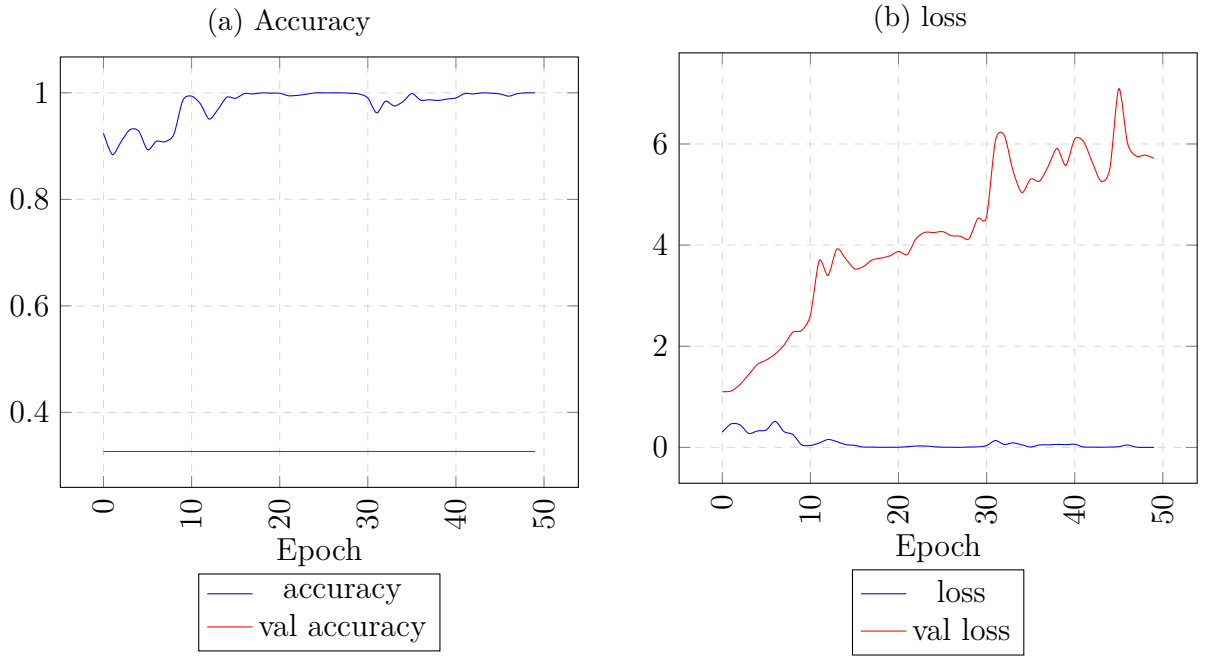


Figure 5: Training logs for MobileNetV2 with weight intialized randomly

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 6.

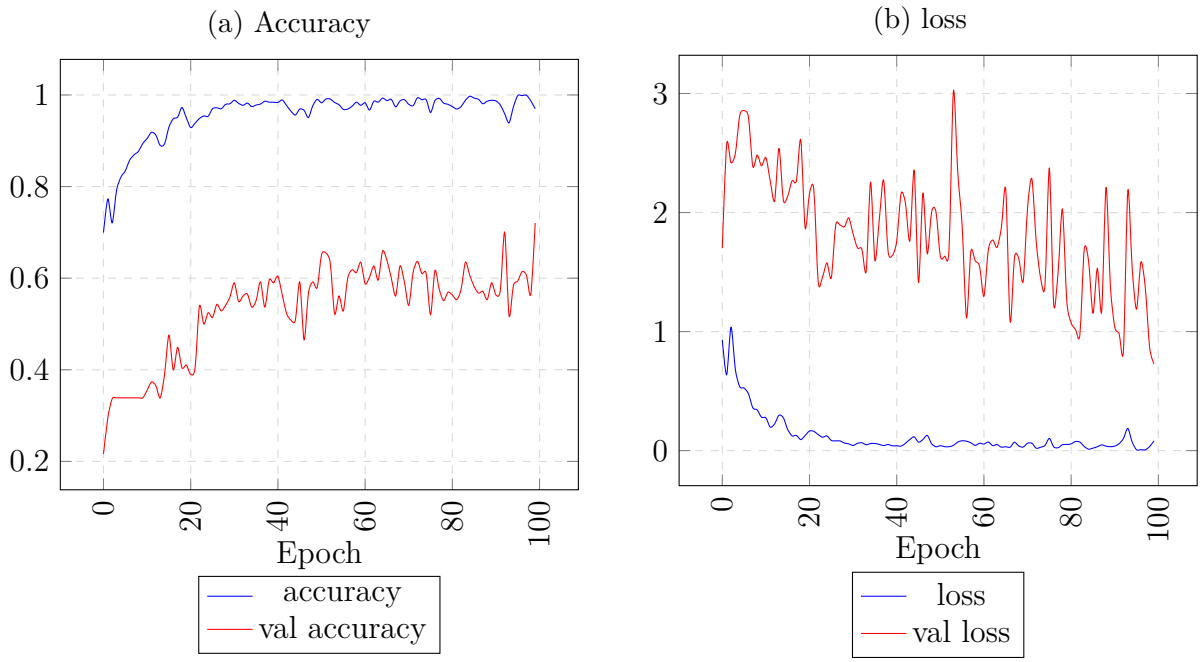


Figure 6: Training logs for MobileNetV2 with ImageNet weights and only training the last fully connected layer.

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 7.

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

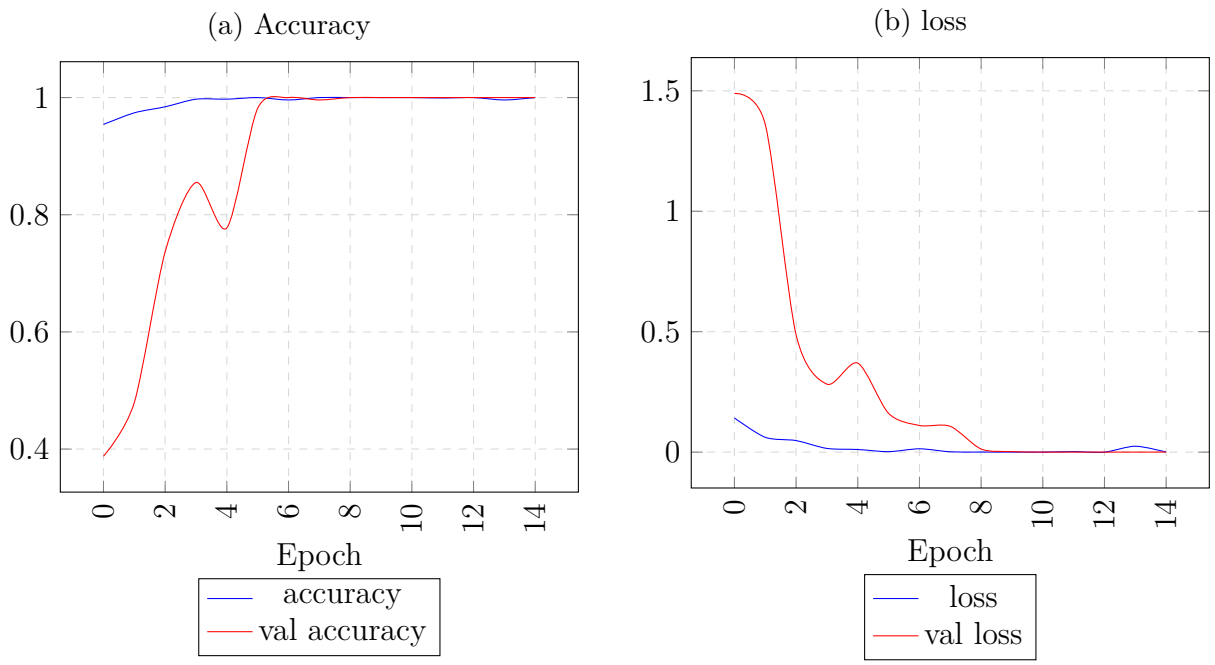


Figure 7: Training logs for MobileNetV2 with ImageNet weights and retraining all layers.

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.

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

4.4 Impact of dataset size

Datasets are very important for machine learning and specially deep learning, this kind of algorithms works better with more data. But it also has been shown that the quality of the data is important [9], it has to be representative and as similar as possible as those the model will get as input when deployed. In the Dataset of Fruits 360, most of the train and test images are very similar thus imposing the question whether how much of the dataset is really needed?, and whether training with a subsection of the dataset is enough for the model to achieve good performance in the test set.

In order to test this, we trained the model with different size of subsections of the dataset. In order to do this we need to consider that because we are reducing the dataset size, we need to increase the number of epochs to achieve roughly the same amount of training iterations as the one with the full dataset size, we can calculate this:

$$iterations = epochs * (size(dataset)/size(batch))$$

We are using as base model MobileNetV2 with ImageNet weights, trained with 15 epochs thus getting $1,369$ iterations. So we change the number of epochs depending on the sub-dataset size to get the same number of iterations.

4.4.1 Reducing dataset size

The whole Fruits 360 dataset of the categoires Apple Golden 1, Banana, Orange; consists of 1461 images for training and 490 for testing, what we want to reduce is the amount of training images and see whether they are all necessary to achieve good performance. To do this what we did is that when pre-processing the dataset to use it with the TensorFlow Dataset API, we add a step at each image where we ignore some n number of images, with this approach we can reduce the dataset size systematically choosing different step sizes to reduce it.

Dataset Size	Categorie Size	Test Accuracy
1461	490	0.99
731	245	0.99
488	164	0.95
244	82	1.0
98	32	1.0
74	24	1.0
68	22	1.0
60 (offset 25)	20	1.0
50	16	1.0
43 (custom)	14	0.99
33	11	1.0
24	8	1.0
18	6	0.99

Table 2: Model accuracy comparing different sub-dataset sizes

With this experiment we can say that, even tho this algorithms works better with more data, this data need to be representative with respect of the data that the model will be evaluate on. Size matters, but quality is important.

4.4.2 Hyperparameters for small datasets

During the project we really haven't done experiments with different hyperparameters, mainly because of the complexity behind and time consuming that it can be, what we do is that we take the default values and try to not move them. But reducing the dataset causes the model to be more sensitive to hyperparameters, especially the batch size. For example we trained a model with a subdataset that has 43 total training images hand picked from the original Fruits360 dataset, with this dataset we trained the model with 3 different batch size Figure 8. When we have a batch size of 1 or also known as stochastic gradient descent Figure 8(a), training isn't as stable but is prone to getting stuck in a local minimum only when dealing with small datasets because the loss goes to 0 rapidly. When we have a batch size of 43 which is the size of the whole data set is known as batch gradient descent 8(b), the training is very stable but once the model reaches a local minimum there is no way for the optimizer to get out, in this case the local minimum is reached from the beginning. And when we have a batch size of 12 also know as mini-batch gradient descent Figure 8(c), we see that the training is is not as stable, thus the optimizer can get out from local minimums, but also the loss doesn't reach 0 as fast as with stochastic gradient descent.

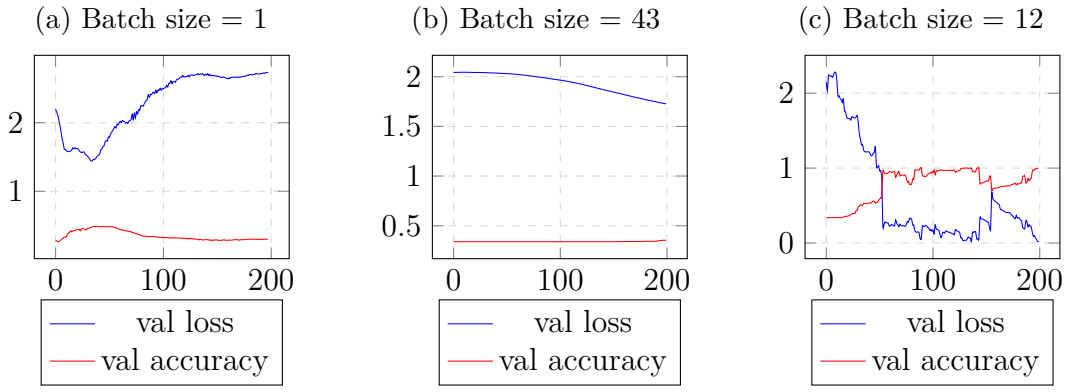
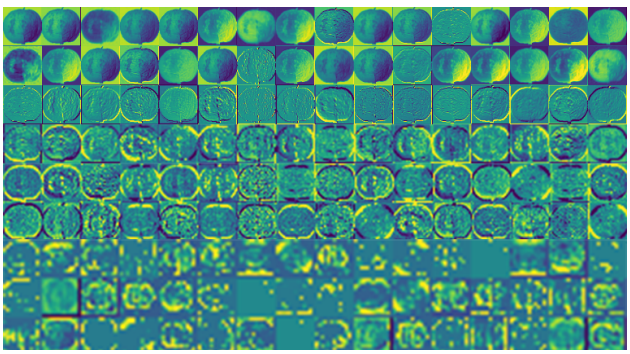


Figure 8: Comparisons of the effect of batch size, on the same sub-dataset.

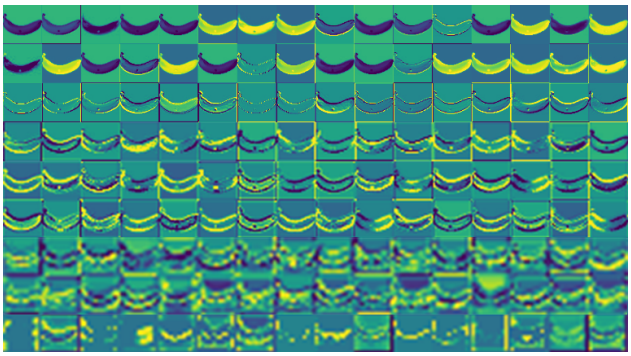
4.5 Visualizing activations

One disadvantage of using deep neural networks for machine learning models is that we don't have a good way of telling what is the model really learning, this imposes a lot of problems in the deep learning world. One technique to visualize what CNN's are learning is to graph the activations of the convolution layers, by doing this we can see what information is being retained by the layers

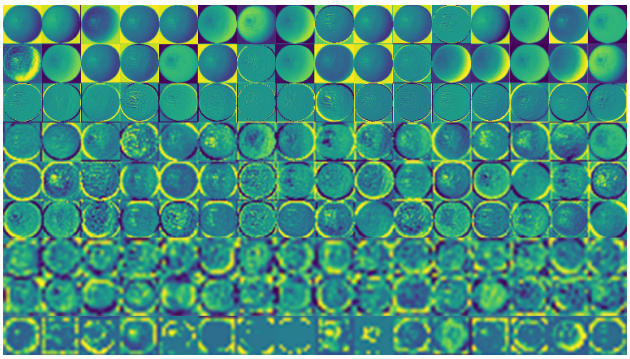
When we see the activations of our model with images of apple Figure 9(a), banana Figure 9(b) and orange Figure 9(c). We can see that our model is retaining mainly the form of the fruit and the texture. This is a good thing and we can also see that for the apple and orange, even tho the two have a similar shape the layers are recognizing different textures of the fruit. However, what we can not see with this technique is the importance of the color for the model, we only see how it is able to extract different patterns.



(a) Activations of a apple image.



(b) Activations of a banana image.



(c) Activations of a orange image.

Figure 9: Visualizing activation of MobileNetV2, layers 1, 7, 16, 27, 42, 57..

With this in mind, we can say that if we want to improve the accuracy of our model in the real world. We need the following:

1. More photos of the fruit at different positions and different size, so the model can learn different shapes of the same fruit.

2. Variety in the texture of the same fruit.

5 Experiment Results

If our training data isn't sufficient to satisfy the necessities of our problem, we cannot expect it to perform well in images of our problem. And this is what we see if we test the model trained in the Fruits 360 dataset Image **??**, that it miss-classifies the images. We also see a that the model is overfiting the data and making erroneous predictions.

References

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

Figure 10: Example of images miss-clasifed by the model, trained with the Fruits 360 dataset

[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.

[9] Xiangxin Zhu, Carl Vondrick, Charless C. Fowlkes, Deva Ramanan. *Do We Need More Training Data?.* 2015