Banana Ripeness Classification using a Convolutional Neural Network

Vincenzo Alessandro Macri[1]
Dept. of Electrical Engineering and Computer Science[1]
Florida Atlantic University
Boca Raton, Florida
vmacri2017@fau.edu
May 1, 2022

I. Introduction

Bananas are one of the most popular fruits in the world, eaten by millions world wide on a daily basis in ranging forms from a milkshake, or a pastry to just the plain fruit. Due to their popularity bananas hold the title of the most traded fruit in the world, with their global export value estimated at 8 billion dollars as of 2016 and a retail value between 20 and 25 billion. [1] Although their popularity increases yearly, around the world we face the growing issue of food waste. According to Dana Gunders, ”Americans throw away 5 billion bananas every year — and a lot more food. Across the U.S. food supply in 2019, 35 percent of it — some 80 million tons of food — went unsold or uneaten.” [2] To make make matters worse, there are millions around the world living in developing countries with food not regularly available, struggling to put meals on the table daily. The ability to limit this food waste, especially regarding bananas and fruit in general is an important field of interest we will target in this paper. Using a convolutional neural network, based on the popular AlexNet architecture [3], we created a classifier trained on varying images of bananas that classifies pictures of the fruit into three separate categories: unripe, ripe and over-ripe. Using these classifications, we plan to create an intelligent robotic system that can be deployed in supermarkets to activity scan through an classify the bananas being offered for purchase in the store. After classifying the given bananas on the supermarkets shelf, the system could then sort the bananas, putting the unripe bananas in the back, followed by the most ripe bananas and then finally by slightly over-ripe bananas in the front of the display. Any notably over-ripe bananas could be flagged or sorted into an area to be taken to a bakery to make pastries or other items that require the banana to be over-ripe. The intended goal of this sorting would be a reduce in store banana waste by selling off bananas that will be going bad the quickest first while saving the unripe bananas to be sold later. In this paper we will discuss the algorithms, methods and techniques that are used in our convolutional neural network image classifier.

II. Methodology

The convolutional neural network architecture we will be using is a slightly altered version of the popular AlexNet architecture, famous for its remarkable performance in the ImageNet Large Scale Visual Recognition Challenge, an image classification challenge consisting of 1000 image categories. The AlexNet architecture consists of multiple blocks of convolutional layers, ReLU layers and max pooling layers, all of which we will discuss in more detail. The architecture ends with a fully connected deep neural network that outputs 1000 values for each of the 1000 image categories. The basic AlexNet architecture is visualized by the block diagram in [Fig. 1]. The only change we will be making to the AlexNet structure is in the output layer of the deep neural network where we will have 3 output neurons instead of 1000 since we will only be classifying our images into 3 ripeness categories. We will discuss the principles behind convolutional layers and why they are used, the ReLU activation function, max pooling and finally the inner workings of each neuron in the fully connected network.

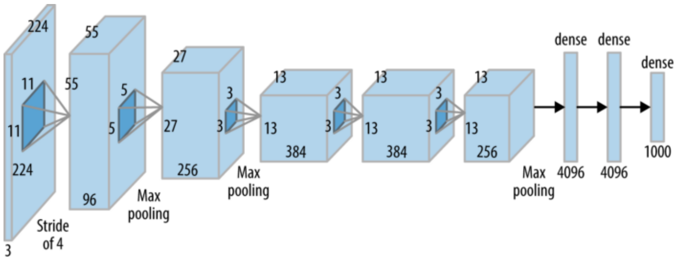

Fig. 1. AlexNet block diagram

A. Convolutional Layers

The main exclusive feature of AlexNet which makes it so powerful is its use of convolutional layers. Convolutional layers are used in convolutional neural networks to extract different features from a given image. These features are then used to logically classify the images into desired categories. In a general form, a convolutional layer is made up of a set number of filters, all of the same size. Each one of these filters will be a matrix of numbers that will be selected randomly at the beginning of training. For a given filter, governed by its size and stride, its matrix will traverse over the entire image and at each group of pixels on the image the filter stops over, the dot product between the filter matrix and the corresponding images matrix will be performed and saved to a new matrix called the activation map. This activation map will essentially be a filtered variation of the original image according to the values of the filter matrix. This process of a filter running over an image is visualized in [Fig. 2].

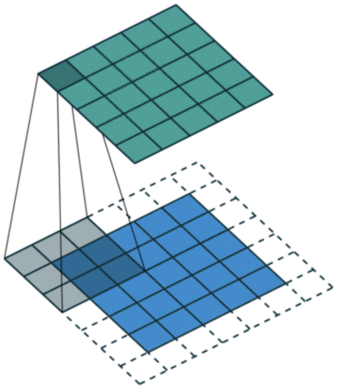

Fig. 2. Convolutional Layer filter visualization with a zero padding value of 1 and a stride of 1.

Overtime, through back propagation, the random numbers in the filter will converge to specific values that when run across an image will produce an activation map that will highlight certain image attributes that might be useful in classifying the image. For example, one filter might converge to have values that specifically look for vertical lines while another filter in the same convolutional layer might converge to have values that look for diagonal lines.

There are different aspects of a convolutional layer that can be fine tuned and adjusted to produce a desired activation map: the filter size, stride and zero padding. The filter size, which are the dimensions of the filter matrix, control the amount of pixel values in an image that will be used to generate a single value in the activation map. The stride of a convolutional layer's filter controls the amount of filter shifts that will be made while the filter traverses an image. Since the nature of filters inherently reduce the dimensionality of an image, a technique known as zero padding can be used to control the size of the activation map generated. Zero padding in a convolutional layer is the process of putting zeros around the border of an image thus changing the original image size and influencing the final size of the resulting activation map. In the AlexNet architecture we used, there are 5 convolutional layers throughout the network each with varying filter sizes, strides and zero padding values customized to optimize the networks performance. Convolutional layer filters towards the beginning of the network will converge to detect lower level features such as lines while convolutional layer filters towards the end of the network will converge to detect more complex task specific features such as geometrical patterns or in our case bananas. An interesting visualization of what types of feature extractions can be found at different complexity levels in the network are shown in [Fig. 3].

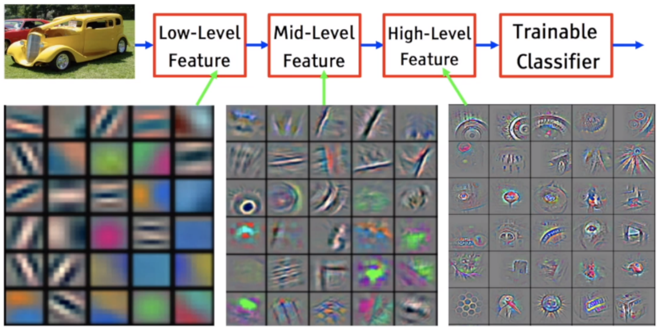

Fig. 3. Different feature complexity levels from a Convolutional Layer

B. ReLU

After each of the 5 convolutional layers in our convolutional neural network architecture, there is a ReLU activation function layer. The purpose of this layer is normalize the values in the produced activation maps from each of the filters in the layer. This process is done by applying the function shown in the graph of [Fig. 4] which takes all values less than zero, and makes them equal to zero leaving all other values as they were.

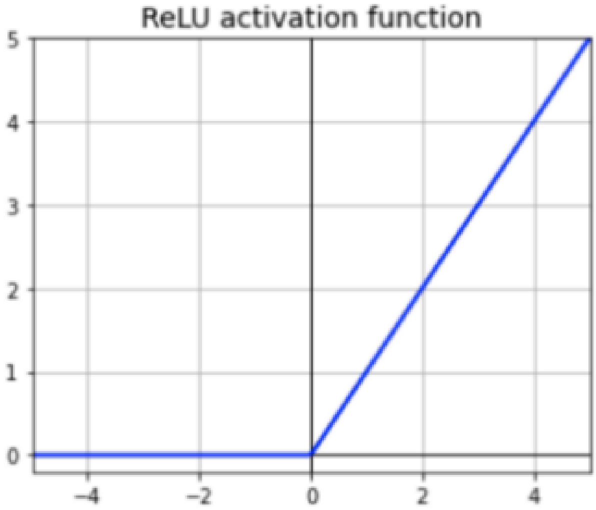

Fig. 4. Graph of the ReLU activation function

C. Max Pooling

In a convolutional neural network, an image as input eventually gets its dimensionality diminished into a relatively small set of output values according to the desired task. This process is often not done using just convolutional layers and requires the assistance from pooling layers. In the particular variation of the AlexNet architecture we used, a max pooling layer is used 3 times, after the first, second and last convolutional layer's ReLU. The max pooling layer effectively downsizes the image by traversing a 3x3 pooling filter across the image with a stride of 2. For each of the filter shifts, the largest number in the filter space is written to a new matrix that will then be passed to the next layer of the network.

D. Fully Connected Network

Following the the 5 convolutional layer blocks in the variation of the AlexNet architecture we used, there is a dense fully connected neural network that takes all 9216 pixel values from the last max pooling layer and lines them up end to end as a one dimensional vector. These 9216 values are then passed through 3 dense layers, the first layer consisting of 4096 neurons, followed by the second layer also consisting 4096 neurons and then finally the output layer producing 3 values from its 3 neurons. These final 3 numbers that are outputted from the network are the corresponding probabilities that the input image was of an over-ripe, ripe or unripe banana. The way this neural network functions is through the passing of information between neurons. Each neuron will accept inputs from its previous layer and from those inputs will compute its local field value. For each neuron in the network, the local field is computed using the formula (1).

$$\mathrm{v} = \sum_{j=0}^{m}(x_j w_j) \tag{1}$$

In this formula x_j is an input value to the neuron and w_j is its corresponding weight. The value m represents the total number of inputs and their corresponding weights that are being fed into the neuron and the variable v is the neurons local field, or output value. This output value from the neuron is then be passed through the ReLU activation function discussed previously concluding the neurons function. Depending on the layer of the network, the output of the ReLU function will either be an input into the next layers neurons or one of the final output values.

III. Parameters

To implement, train and test our model we used Python 3 in the Google Colab environment using the PyTorch library. For data to train and test our convolutional neural network, we found 80 images each of over-ripe bananas, ripe bananas and unripe bananas through a google image search. We then randomly selected 20 percent of the images from each class to be reserved for testing and kept the remaining 80 percent of the images from each class to train the network placing both groups of images into their respective folders. We trained the network for 30 epochs with a learning rate of 0.003 and a batch size of 16. The epoch size controls the number of times the entire training data set will be passed through the convolutional neural network during training, the learning rate controls the magnitude of updates that will be made to the weights and filters in the network and the batch size controls the number of training images the network will be given at a time before it is given the opportunity to update its weights. These parameters were all fine tuned to maximise the performance of our network and ensure steady but efficient training. To help generalize the functionality of our network and make it robust in detecting correct classifications regardless of banana size, shape or orientation, before each training image is sent into the network for training, it is randomly cropped, flipped horizontally and rotated. In doing these image transforms, we ensure that no two training images will be exactly the same minimizing the risk of the network becoming overfit to the training data. Additionally, rather than training the entire model from scratch which would take an extended period of time, we used a pretrained AlexNet model that we could begin our training from. This prevented our network from having to relearn low level features such as lines and curves that can be found in practically any image and focused the training process on learning higher level features that are relevant to the specific task of classifying the ripeness of the bananas.

IV. Results and Discussion

After training the model for 30 epochs, the convolutional neural network reached a training accuracy of 99 percent by only the 15th epoch, maintaining the near perfect accuracy for the remainder of the training process. Additionally, after the model was fully trained, we ran inference on the model for each of the images we saved for testing that the model had never seen before and it correctly classified the validation data with an accuracy of 95 percent. An example of 4 testing results can be seen in [Fig. 5].

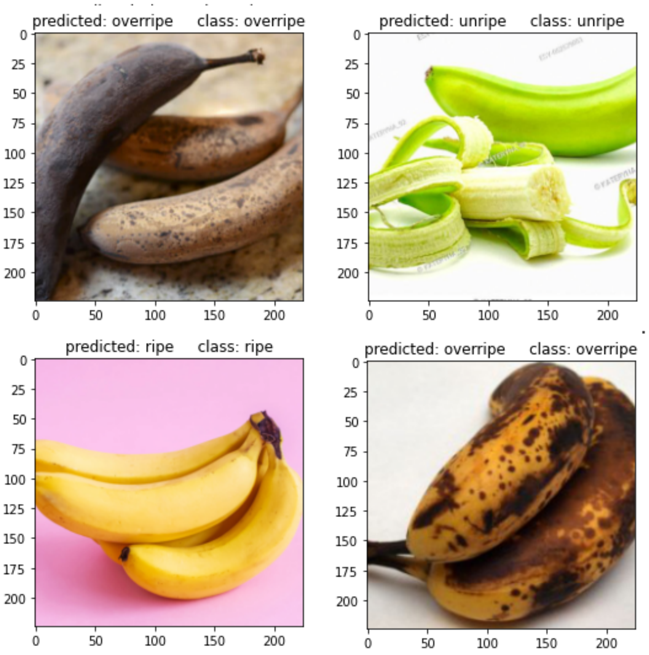
predicted: overripe class: overripe
predicted: unripe class: unripe
predicted: ripe class: ripe
predicted: overripe class: overripe
0
25
50
75
100
125
150
175
200
0
50
100
150
200

Fig. 5. Testing results from fully trained banana ripeness classifier.

After analyzing the results of our trained convolutional neural network, it was clear that the variation of the AlexNet architecture we used was proficient in accurately classifying images of bananas by their level of ripeness. In reviewing the few images in the testing data that the classifier did not sort correctly, the majority of errors seem to have resulted in variations in the training data according to personal opinion. While some might consider a banana with a few brown spots to be over-ripe, others might claim the banana is still ripe and okay to eat thus drawing a fine line between what an over-ripe banana should look like and what a ripe banana should look like. In picking training data for this model, we selected some bananas with minimal amounts of brown spots to be classified as ripe, however in doing so some of the training images in our over-ripe and ripe classes were mildly similar resulting in inaccurate classifications for some testing images. We plan to retrain this network again with a higher level of focus on our training data, making sure to have clear distinctions between what a ripe banana should look like in comparison to an over-ripe or under-ripe banana. Additionally we plan to explore different types of architectures and models in the future that might allow us to classify banana ripeness in an analog fashion, providing a placement on a spectrum of ripeness rather than a digital classification.

V. Conclusion

In light of the results we observed from training and testing our classification model, we are pleased with the models ability to accurately classify images of bananas according to their ripeness stage: over-ripe, ripe and unripe. We discussed the key components that make up the AlexNet convolutional neural network architecture that we used including the convolutional layers, ReLU activation function layers, max pooling layers and fully connected dense layers at the end of the network. Through this discussion we delved into the way convolutional neural networks utilize feature extraction to maximise performance and how low level features in the first convolutional layers lead to more complex high level features in the final convolutional layers to aid in generalizable classification. With the unacceptable rates of fruit waste, especially in the banana market, we hope this technology will grow into a product that can help minimize the amount of bananas that go over-ripe before being seen or purchased by consumers. Additionally we plan to continue working to solve this problem using machine learning tools, looking into different techniques that may be able to advance ripeness detection accuracy leading to a less wasteful world and a more efficient banana market.

References

[1] A. Caublot. “Bananalink.” (), [Online]. Available: https://www.bananalink.org.uk/all-about-bananas/.

[2] D. Gunders. “5 billion bananas get thrown away each year — how reducing food waste can help solve the climate crisis.” (), [Online]. Available: https://www.chicagotribune.com/opinion/commentary/ct-opinion-food-waste-20210409-3k3llled4fbmlp3nwhiej3o354-story.html.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.