Investigation into Object Classification for Images

Convolutional Neural Network to Classify Fruit

Ming Shane Tong

Student Number: 300522795

Course Code: COMP309
Lecturer: Dr Qi Chen, Dr Marcus Frean
Date: 31 October 2022

1 Abstract

A Convolutional Neural Network was created to classify if an image contained a cherry, strawberry or tomato. This used a training set of 3000 images, a validation set of 600 images and a testing set of 900 images. The final model achieved and accuracy of 80.77%, a better result than the Multi Layer Perceptron model which achieved an accuracy of 37.89%. This paper will explore Image Classification, Convolutional Neural Networks and techniques for tuning these model.

2 Introduction

The purpose of this project is to explore how a computer can be trained to identify objects within images. The finding from this project will be used to develop a model to help us classify these images into one of the 3 categories of fruit.

3 Problem Investigation

3.1 Dataset

A collection of 4500 image of each category of fruit were collected from various sources including the internet to help train this model. To avoid data leakage during testing, 300 images from each category were removed to become the testing set and 200 images were removed to become the validation set.

There is a lot of variations within the set of images. The quantity of the fruit present in the image varies from image to image, some contains only 1 fruit whereas some contains many. While commonly cherries, strawberries, and tomatoes are red, there are also have different colours which indicates its type or ripeness. We also notice that each fruit is a different type of red with cherries having a darker red, strawberries having a vibrant red and tomatoes having a lighter red. Images of the fruits sliced are also included and it contains distinctive patterns on the inside.

Figure 1: Variations of fruit

Amongst the images there are also images that are not the fruit being classified. We also see that paintings and clip art are also included which are not good representation of the fruit. As we want to classify images of real fruit, these images should not be included in our dataset.

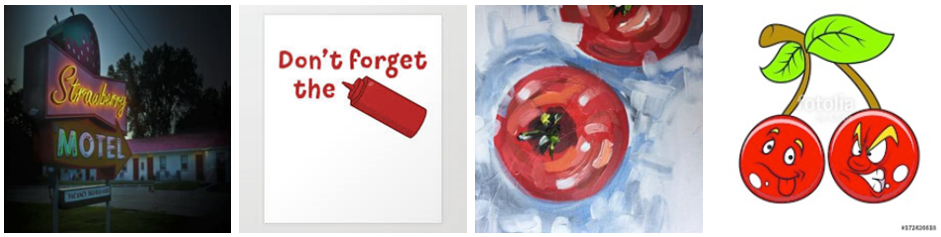


Figure 2: Invalid Images

There are also various quality differences within our data. Some fruit are too small to be seen in our images. We also have images that have been edited to look differently. Some of these completely lose certain features of the fruit due to lack of contrast.

Figure 3: Some boundary cases

3.2 Pre-Processing

3.2.1 Data Selection

From exploring the dataset of images, several images were discarded as they did not represent the category of fruit it was meant to. Images that did not contain the fruit or contained a non physical version of the fruit (such as a cartoon) were removed. Images that were considered for removal were images that contain the fruit in low resolution, messy images with many objects or images with many edits where the fruit cannot be recognised.

3.2.2 Data Transform

Images were transformed into tensors in order to be in a usable state for the computational model. To normalise pixels values, each pixel were converted from a integer between 0 - 255 to a floating point number within a distribution with the mean and the standard deviation is 0.5 (Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) in PyTorch). The images are also resized to 300 pixels in width and height to ensure that all images can be processed equally.

3.3 Baseline Multi Layer Perceptron Model

The Multi Layer Perceptron Model (MLP) is a machine learning model that takes an array of values as an input and passes that it through several layers of processing to get an output. To evaluate its effectiveness, a MLP model was created with a single hidden layer of 150 perceptrons.

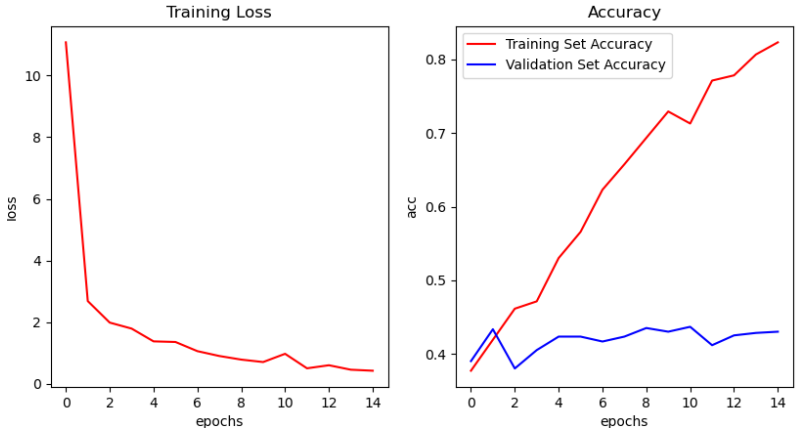


Figure 4: Performance of the MLP

In the graphs presented the red line represents accuracy of the model on the training set and the blue line represents the accuracy on the validation set. We see that the MLP barely improve on its initial accuracy within 4 epoch. When the train model was used on the test set it achieved an accuracy of 37.89%.

3.4 Convolutional Neural Network Model

The Convolutional Neural Network Model (CNN) iterates on the MLP model by using a 3 dimensional array as its input and using convolution layers. Convolution layers extract features within data by running a kernels across the image to discover features like edges and shapes. Pools were used to reduce the dimension of the images by grouping multiple pixels into a single pixel.

To iterate on the baseline model, one convolutional layer that extracts 6 layers is added with an additional 2x2 MaxPool which keeps the highest value pixel in a 2x2 pixel area.

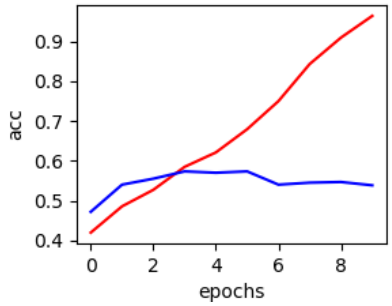


Figure 5: Performance of the CNN

In Figure 5, when the accuracy of the model reached close to 100% in 9 epochs, the accuracy of the validation remained close to 50% which indicates model overfitting. We see that the accuracy of the 2 models start to diverge around 4 epochs where overfitting began to occur.

3.5 Investigating of AlexNet

The CNN Model have been a model in use for a long time for training models to identify images. One such instance to classify images in the ImageNet database. The ImageNet database has 1000 classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images. Using this database, ImageNet hosted the competition, ImageNet Large Scale Visual Recognition Challenge (ILSVRC), that asked its contestants to submit a model that can classify the object within the images of this database. [RDS+15]

One of the contestants was AlexNet in 2012. The AlexNet CNN model achieved an error rate of 15.3% on the ImageNet database with the second best achieving 26.2%. AlexNet has several properties used in its model. [AKH12]

3.5.1 ReLU

As oppose to the tanh function or the sigmoid function which were common at the time, AlexNet used the ReLU function

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x \geq 0 \end{cases}$$

From their findings, using the ReLU function reduced the error in fewer iterations than the other functions. It was tested that the ReLU function achieved a 25% error rate on the CIFAR-10 dataset in fewer epochs than the other 2 alternatives. This is beneficial as faster learning prevents overfitting of the model. [AKH12]

3.5.2 Overlapping Pools

Usually during the pooling process, adjacent pools do not overlap however, in AlexNet, overlapping pools were used. This means the outputted array will be larger as extra pixels between adjacent pools will be added. The use of the overlapping pools prevented overfitting from the AlexNet model slightly so adding this into the network could be beneficial. [AKH12]

3.5.3 Data Augmentation

AlexNet augmented its data by converting it to size 256x256 images then cropping the center 224x224 pixels to be used in the model. This means that a the edges of the images were discarded which makes sense to do as the focus is going to be in the center and removing part of the border also removes unnecessary noise from the image. [AKH12]

3.5.4 Dropout Layers

A regulation technique used by the AlexNet model was the dropout technique. What happens at this layer is that every node is given a 50% chance of outputting 0 instead of its usual output. This means that the node will not participate in back propagation. This essentially samples a range of

network architectures and the model must be trained to use all nodes to inform its output. This prevents overfitting as the model will teach the model to find alternative patterns in the data. This is at the expense of the model needing to complete double the number of iterations to converge. AlexNet used dropout in the first 2 convolutional layers. [AKH12]

3.5.5 Architecture

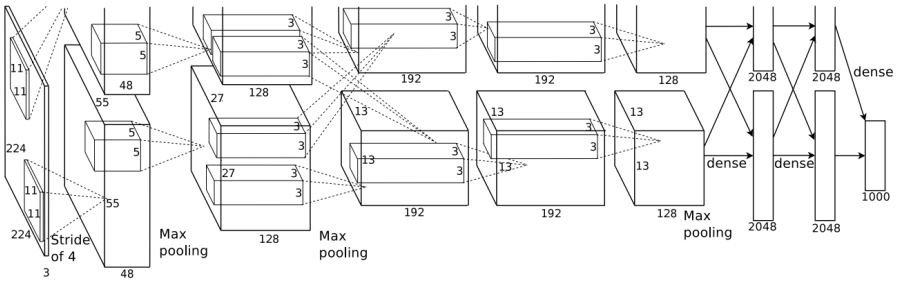


Figure 6: AlexNet model architecture

In the AlexNet paper, an image of the CNN architecture was provided where its inner workings can be examined. It can be seen that the kernel size at the convolutional layer varied with the largest being 11 for the first layer then reducing to 5 then 3 in the later layers. We also see the use of strides which is a variable that determines how much to move the kernel each time. It can also been see that the number of layers extracted is quite large with the most layers at one point being 192 layers.

4 Tuning

4.1 Pre-Processing

Adapting what was used in the AlexNet Model, after images were sized, the images were cropped to include only the pixels in the center of the image.

In addition to this change, RandAugment in PyTorch can apply random changes to the images as they are loaded. With random augmentations, images could be oriented differently by flipping them vertically and horizontally, and rotated. The images could also be colored different such as increasing the contrast or saturation.

Figure 7: Cherry with increased contrast and sharpness

Figure 8: Strawberry Images with Posterising

These changes trains the network to see a range of patterns that may not be obvious if all images were all the same. By learning these patterns from edited images, the network should improve its predictions for the unedited testing set.

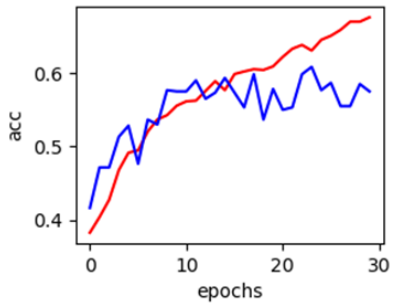


Figure 9: Performance with Random Augmentation

We see from Figure 9 that our accuracy is between 55% and 60% which improves upon the previous model by about 5%.

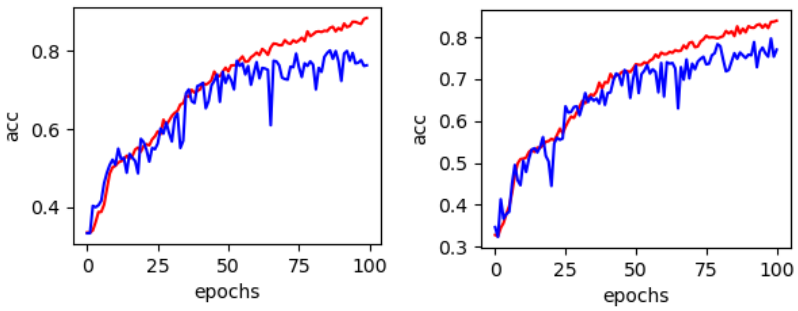
4.2 Increasing Network Complexity

4.2.1 Increasing Layer

The model of AlexNet was adopted and changed to create the base of the model. The network was increased to a network with 4 layers while using similar techniques such as decreasing convolutional kernel sizes and strides.

4.2.2 Dropout Layer

The paper for the AlexNet model discussed the use of the dropout layer technique which was used to prevent overfitting.



(a) Without Dropout (b) With Dropout

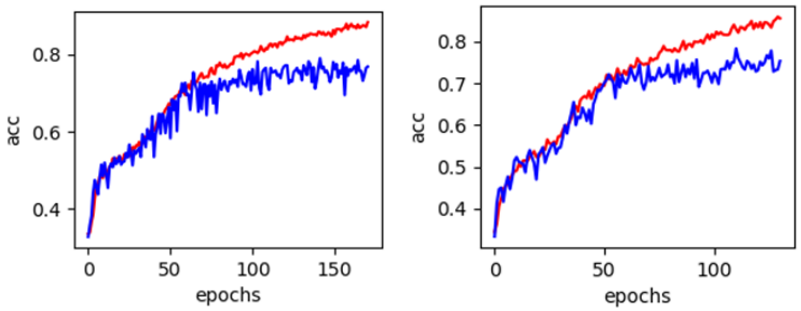
Figure 10: Dropout Performance

We see that for both graphs, the model starts to become overfit as the rate the training set accuracy is improving become greater than the validation set. However, we see that with dropout, this 'branching out effect' is minimised whereas the first graph converges to an accuracy of around 75%. It is predicted that with more epochs, the model with dropout will achieve a greater accuracy than without dropout.

4.2.3 Pooling Layer

Similar to what was used in AlexNet, the use of the overlapped pooling technique was used in the model. A pooling layer using a size 3 kernel and the stride of 2 was tested.

We see on Figure 11, with the regular pooling, the model achieved an accuracy of 70% whereas with overlapping pooling, the accuracy was close to 75% which is a 5% improvement. Hence this technique is effective in improving the accuracy of the model.



(a) With Normal Pooling (b) With Overlapping Pooling

Figure 11: Overlapping Pooling Performance

4.3 Optimisation Function

The previous optimisation function used in the model used the stochastic gradient descent (SGD). What this means is that the gradient descent performed is the sum of the proposed changes for every instance in the batch.

However, the steps taken by this method can be large as it is the sum of its changes. Another weakness is that the learning rate is fixed and so it reaches the optima in more epochs than other algorithms. Other optimisation functions can be tested to see if there is any improvement.

4.3.1 Adam

Adam is an optimisation algorithm that can optimize the model faster than SGD by increasing momentum towards the optima. This usually results in fewer steps being needed for our model to converge to the optima.

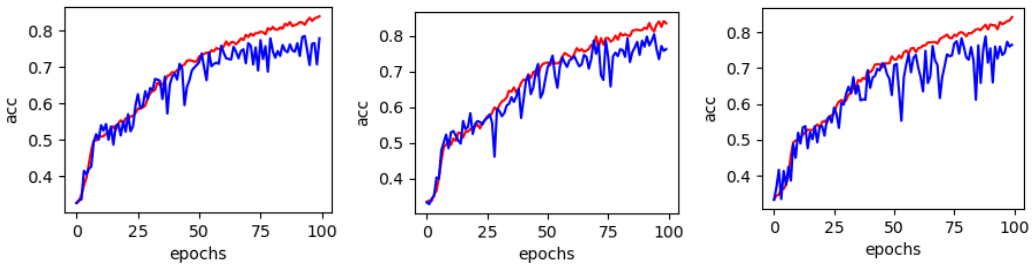
During testing of this function, Adam not only didn't improve the performance of the model, it failed to change the accuracy of both sets of data within 20 epochs. This means that for this model, Adam is not suitable.

4.4 Batch Sizes and Learning Rates

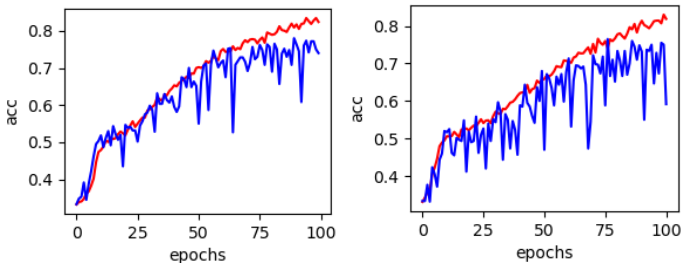
The batch size being used to train the model impacts the amount that is updated on each step. Larger batch sizes means fewer updates. To compensate for the decrease in updates, the learning rates need to be adjusted. Currently, there is only 1 image in every batch. To test the impact of batch size on our data, several quantities of batch sizes should be tested.

4.4.1 Related Work

Dominic Masters and Carlo Luschi [ML18] concluded from their paper that "the best results have been obtained with batch sizes m = 32 or smaller, often as small as m = 2 or m = 4". In this paper, different batch sizes were trialled and they found that there was a general performance increase with reduced batch sizes. This result was obtained with the use of the linear scaling rule which states that an increase in the batch size requires a proportional increase to the learning rate to keep the mean weight update constant during gradient descent. Using the suggestions from the paper, several batch sizes up to 32 are tested.



(a) Batch Size 2 (b) Batch Size 4 (c) Batch Size 8



(a) Batch Size 16 (b) Batch Size 32

From these results, it can be seen that the models had about the same highest accuracy of around 75% or 80%. This result does not align with the study because the training set of 3000 images is small compared to the training set of CIFAR10 containing 50000 images used in the paper.

However, when the batch size and learning rate are larger, the accuracy fluctuates much more. This is because the learning rate is too high so too large of updates are occurring. Because of this fluctuation of the accuracy, the confidence we can have in the model is much less so lower batch sizes should be used.

4.5 Loss Functions

The loss function is used to inform the model on how to do gradient descent. While there are many types of loss functions available, the outputs of this model are categorized and so loss functions like Mean Squared Error (MSE) cannot be used as it outputs either numerical or ranked data.

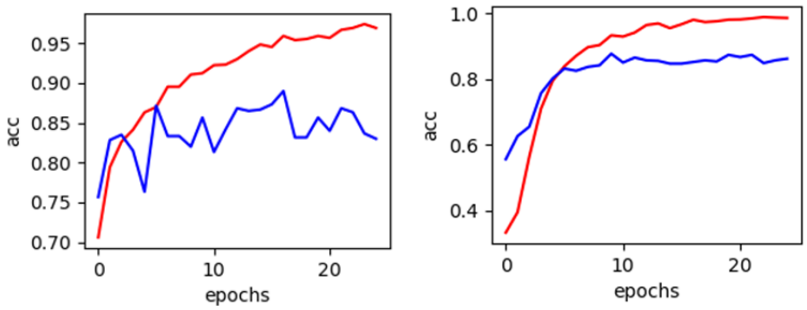
In the first model, the CrossEntropyLoss was used as the loss function to optimise the model. Another loss function that is appropriate for categorical data is the NLLLoss. This method was tested against the training data and the validation data.

After testing, the use of NLLLoss achieved an accuracy of 60%, improving the accuracy of the validation set by 10% increasing from 50% using CrossEntropyLoss.

4.6 Transfer Learning

The transfer learning technique uses a pre-trained model that classifies the images in one dataset and applies the outputs of that model to inform a different model of a different purpose. This is beneficial as the classes of the previous model can reveal patterns in the images that share properties with the 3 fruit. This means at least 1 additional layer needs to be added to map the outputs to the 3 classes of fruit.

To test this method, 2 pre-trained models will be used which are AlexNet and ResNet18. Both models were used on the ImageNet dataset.



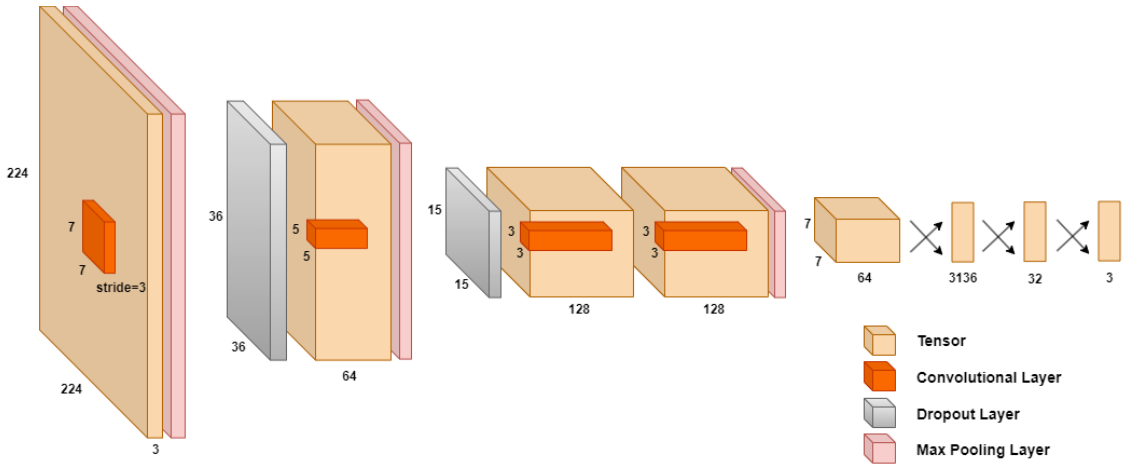
(a) Performance for AlexNet Transfer Learning (b) Performance for ResNet18 Transfer Learning

It can be observed that both models reached a final accuracy of above 80% within the first 10 epochs. This accuracy is better than the accuracy that was given by the model created. This suggest that the models have learnt patterns from the ImageNet dataset that can also be applied to this dataset of fruit. We see however that the accuracy of the AlexNet model is fluctuates unlike the ResNet18 which remained smooth. It is predicted if AlexNet could be trained for more epochs then it would converge to an accuracy around 85% for the validation set.

It can also be observed that the training set accuracy for both models are close to 100%. This suggest that the model will become overfit if left continued to train which will decrease the accuracy of the validation set. This means that more layers after the initial pre-trained model is needed to improve the model.

5 Methodology

5.1 Network architecture



(a) Diagram of the Network

The model contains 5 convolution layers and 3 fully connected classification layers.

The kernels of the first layer is 7 pixels in height and width with a stride of 3, extracting 64 layers. The overlapping max pooling layer is then added with kernel size 3 and stride 2.

A dropout layer with drop probability of 50% is added before the second convolutional layer with size 5 kernel extracting 128 feature into a max pooling layer.

Another dropout layer is added for the 3rd set of convolution layers. The inputs then go through 2 sets of convolution layers with kernel size 3. The first of the set extracts 128 layers and the second extracts 64 layers. The output is put through the final max pooling layer.

The output will be flatten and inputted into 3 classification layers with 3136 input nodes, 32 hidden nodes and 3 output nodes representing each fruit class.

This model was adapted from the AlexNet model and utilised several of its key features. The use of overlapping pooling layers which was found to reduce overfitting slightly. Another way overfitting was avoided was through the use of the dropout layer that randomly dropped nodes from the output with a 50% probability, forcing the network to find alternative features which reduces the weights given to any singular feature.

5.2 Pre-Processing

During training, random augments of the images were performed onto the images to diversify our dataset. By diversifying the dataset, the model will be trained to identify patterns in the data and from that learn to identify fruit in unmodified images.

Following the changes, the images are then resized to 256 pixels in height and width before cropping a 224 pixel square in the middle. This allows the model to focus on the middle of the image without noticing any noise that may exist in the background.

Finally, the image is converted to a tensor and every value is normalised by mapping each value to a value in a normal distribution with a mean of 0.5 and standard deviation of 0.5.

The pre processing steps are performed on the validation and testing images as well except the random augments are removed.

5.3 Optimisation Function

The chosen optimisation function was the Stochastic Gradient Descent (SGD) function, this function was chosen as the Adam function failed to improve from the model's initialisation.

5.4 Loss Function

The loss function selected for this model was the NLLLoss because through testing found that it improved the accuracy of the model by 10%.

5.5 Activation Function

The use of the ReLU function was used at each layer except for the output layer. The ReLU function was used as it reduced the error of the network in fewer iterations than other activation functions. The output layer used the SoftMax function as required by the NLLLoss function.

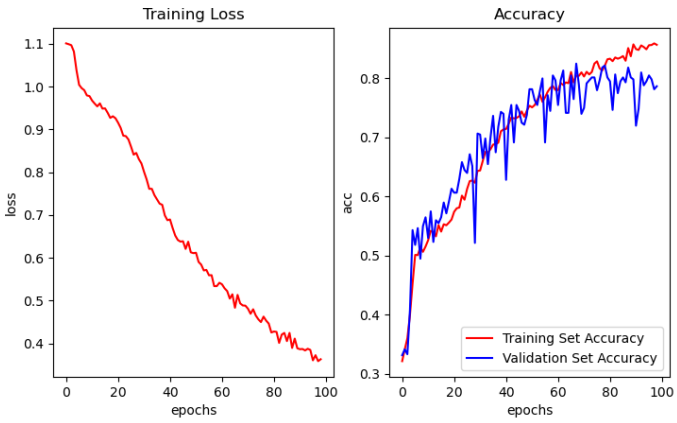
5.6 Batch Size and Learning Rate

It was determined that lower batch sizes were beneficial due to less of a need to compensate for the lack of updates by increasing the learning rate. The increase in learning rate caused greater changes to the accuracy which can cause the model not to converge.

5.7 Epochs

With the given parameters, the model converged to an accuracy of 80% on the validation set within 68 epochs.

6 Results



(a) Results of CNN Model

The model achieved an accuracy of 80.78% on the test set of 900 images. This is an improvement on the baseline MLP model which achieved an accuracy of 37.89%. The training of the final model took 71 minutes whereas the MLP model took 3 minutes to converge.

7 Discussions

The difference in accuracy is due to the type of model that are being used. The MLP model considers the image as an array which is not a representative way of analysing the image. However, using convolutional layers, the kernel is able to select parts of the image that helps identify the fruit such as shapes and other features. It is because of the addition of kernels in the CNN that helps it achieve a higher accuracy on the unknown test set.

This came at the cost of time as the CNN model took significantly longer to train. The addition of the convolution layers could add extra computation time as it is more expensive of an operation to apply a kernel to the image as compared to multiplying and adding 2 arrays which is the most common operation in a MLP network.

The addition of the dropping layer also increases the amount of time it takes for the model to converge. The dropping layer stops certain nodes from being trained in each batch which means that each node will take longer to train. It was predicted that the dropout layer required twice the

number of iterations for the model to converge compared to when a dropout layer was not used. [AKH12]

A technique that was used to reduce the training time of the model was to use the pooling layers that reduced the dimensions of the image by grouping adjecent pixels together. However, the ones used in this model were overlapping which added extra pixels to the image. With non overlapping pools, the model would train faster as there are fewer pixels in the outputted image.

8 Conclusion

In this paper, a Convolutional Neural Network was trained to identify whether an image contained a cherry, strawberry or tomato. The model trained utilised many techniques used within the AlexNet model for the ImageNet database. The final model achieved an accuracy of 80.78% on the test set compared to the MLP model which achieved an accuracy of 37.89%.

Common techniques used for Convolutional Neural Networks were tested.

Pooling can be overlapping or non-overlapping. It was discovered that overlapping pools has the potential to reduce overfitting within the model. Dropout layers also help reduce overfitting by getting the model to learn from multiple features rather than focus on one as nodes are randomly removed from the network during training.

It was also reported that the lower batch sizes tends to perform better than larger batch sizes due to the need to increase the learning rate. It was also found that random augmenting was beneficial to helping the model learn the patterns of the fruit and cropping the center image helped the model by removing noise around the border of the image.

9 Future Work

9.1 Network Architecture

Part of what made this network successful was that it was built similar to the architecture of AlexNet, an existing working model. However, this model was developed in 2012 and many other models have been developed since then.

Another model used in this paper was the ResNet18 which was used in transfer learning to classify the fruits. In the ResNet18 model, the concept of the residual learning was developed which means that the original inputs are given to other hidden layers and not just the input layer. [HZRS15] This technique was not used in this model so utilising this technique could create some success.

ResNet is not the only model that can be studied but GoogleNet and VGG are also CNN models that could be studied furthur.

9.2 Batch Size and Learning Rate

Part of the investigation into batch size was following the linearity rule which states when there is an increase in batch size then a proportional increase in learning rate. However, the batch size to learning rate ratio was constant through out the test. It was concluded that lower batch sizes are more suitable which aligned with the study referenced. However, no optimal ratio was found that maximised testing accuracy. Investigation into the optimal ratio would be valuable in optimising the neural network and its performance. Additional research into whether the same ratio can be used in another model would also be beneficial.

9.3 Overlapping Pools

The use of pools were used as an attempt to reduce the time to train the model by reducing the number of pixels for the next layer. In this model, the overlapping pools were used to reduce overfitting by adding extra pixels to the reduced image, which contradicts its original purpose. Research into the overlapping pools' effectiveness and computational cost would inform developers of this technique's effectiveness in CNN models and whether they should be used in certain contexts.

References

[AKH12] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. 2012. Online; accessed 21-October-2022.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[ML18] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.