**Aim:**

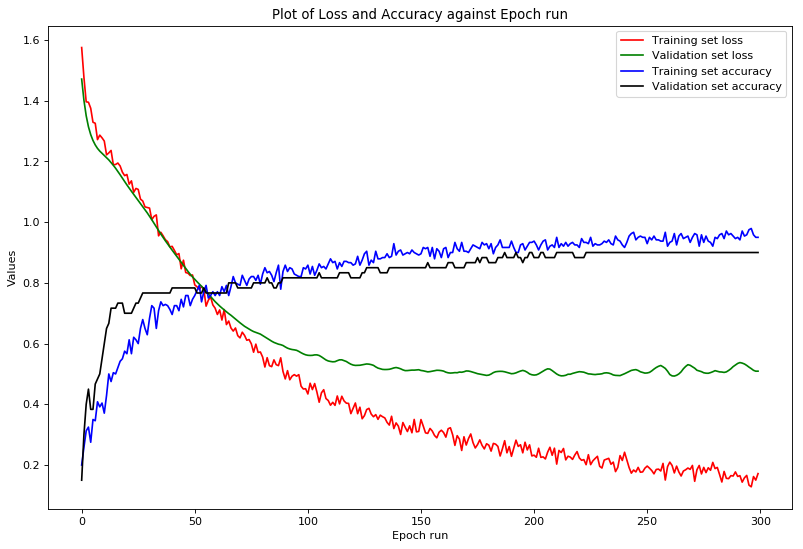
To classify a Fruits image dataset using Convolutional Neural Network into the following classes: (0) Apples, (1) Bananas, (2) Mixed, (3) Oranges.

**What did you do to increase the accuracy of the image classifier?**

1. Performed stacking of Convolutional (Conv2D) layers with different filters to ensure improved feature extraction from the image source
2. Performed Max Pooling with varying window sizes to reduce the size of hidden layers and reduce overfitting
3. Added Dropout Layers with different dropout rates to prevent over-fitting by ensuring random fraction of neurons in each layer is used for learning of weights during the forward-pass and backpropagation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Configurations to improve accuracy | CNN Model 1 | CNN Model 2 | CNN Model 3 | CNN Model 4 |
| Conv2D Layers | 1 | 1 | 2 | 4 |
| Number of Filters | 32 filters | 32 filters | Conv2D layer 1: 16 filters  Conv2D layer 2: 32 filters  Total: 48 filters | Conv2D layer 1: 32 filters  Conv2D layer 2: 64 filters  Conv2D layer 3: 64 filters  Conv2D layer 4: 128 filters  Total: 288 filters |
| Size of Filters | 3pixels by 3pixels | 3pixels by 3pixels | Conv2D layer 1: 5pixels by 5 pixels  Conv2D layer 2: 3 pixels by 3 pixels | Conv2D layer 1: 32 filters  Conv2D layer 2: 64 filters  Conv2D layer 3: 64 filters  Conv2D layer 4: 128 filters |
| MaxPooling | 3x3 window | 6x6 window | 2x2 window | 2x2 window |
| Dense Layers | 64 units with ReLu activation function | 0 layers | 256 units with ReLu activation function | 32 units with ReLu activation function |
| Dropout Layers | Layer1: Dropout rate of 0.5  Layer2: Dropout rate of 0.25 | Dropout rate of 0.5 | Dropout rate of 0.5 | Layer1: Dropout rate of 0.25  Layer2: Dropout rate of 0.5 |
| Epochs | 100 | 300 | 100 | 100 |
| Model Results | | | | |
| Training Loss | 0.0627 | 0.1767 | 0.0241 | 0.0623 |
| Training Accuracy | 0.9917 | 0.9417 | 1.0000 | 0.9667 |
| Validation Loss | 0.6363 | 0.4992 | 0.7232 | 0.8913 |
| Validation Accuracy | 0.8500 | 0.9000 | 0.8833 | 0.9000 |
| Training Time | 8.3695s | 20.2121s | 9.8789s | 20.8939s |
| Prediction Time | 0.0375s | 0.0409s | 0.0403s | 0.0459s |

**Plot the loss and accuracy of your training**



**Specify the best/final accuracy of your image classifier on the test data**

The best models are CNN Model 2 and Model 4 as they both have a prediction accuracy of 90%. However, CNN Model 2 is faster to train than Model 4.

**A reflection of lessons learnt in building your network**

***Data Balance:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Volume of input data/ images | | | | |
| Data Sets | Apples | Banana | Mixed Fruits | Oranges |
| Train | 76 | 76 | 20 | 76 |
| Test | 19 | 18 | 5 | 18 |

We noted mixed fruits were consistently inaccurately predicted in all 4 CNN models whilst apples, oranges and bananas had more accurate predictions across all the models. This is due to the imbalance in the volume of mixed fruit data in both the training and test dataset. Since the model was trained only on 20 images of mixed fruits, it learnt fewer features and was unable to consistently predict images of mixed fruits.

***Model Architecture vs Accuracy:***

We noted CNN Model 2 could consistently produce a higher accuracy than the other CNN models. CNN Model 2 does not utilise any Conv2D layer stacking or any hidden dense layers in the architecture. It has a simple structure which utilises 1 Conv2D layer of 32 filters of size 3 pixels by 3 pixels, a pooling window of size 6x6 and 1 dropout layer.

As our training dataset is small, our goal is to avoid overfitting our model. Hence, by using less hidden units together with a shallow network structure, we are better able to reduce bias.

We learnt that a complex model architecture does not automatically imply a greater accuracy. CNN Model 2 and 4 have similar accuracies despite Model 4 having a more complex architecture.

***Redesigning data: Input image size***

We trained our models on 2 different input image sizes to observe the impact on CNN model performance

First, we used an input image of dimension 128px by 128px. As the model is trained on a larger input image, there are more features to learn from and this leads to model improvement. However, this results in a substantially longer training time of 15 minutes without providing any significant improvements in accuracy.

Hence, we found a better alternative for the input image size was 34px by 34px. The model training time was shorter, and the accuracy was relatively high at 90%.

**Appendix A: Feature map**

