Report on Binary Classification Model for Grapes and Raisins using Transfer Learning

Group 12: 1910576101, 1910476108, 1911176114, 1910676148

March 16, 2024

1 Introduction

The objective of this experiment is to construct, train, and evaluate a convolutional neural network (CNN) model using transfer learning. The model aims to efficiently classify images belonging to two distinct classes of grape and raisin within a given image dataset. The transfer learning technique will help our model learn better and faster. Transfer learning means our model can borrow knowledge from other models that have already learned a lot about pictures. We'll teach our model using a set of pictures, and then test how well it can identify the two different things in new pictures. By doing this experiment, we want to compare between our CNN model and the model using transfer learning.

2 Dataset

The dataset used in this study has 1730 colourful pictures of grapes and raisins. These pictures were captured with different smartphones very carefully to help our computer learn better. Each subject has two or three photos taken from different angles. We also took pictures of single grapes and raisins and some group pictures of grapes and raisins. We split the dataset into three parts: the training set, the validation set, and the test set. There is no overlapping object in these sets. The resolution of the images are 32x32.

Training Set: This set has 1070 pictures, with an equal number of raisins and grapes. We made sure all these pictures have the same black and white background. We also labeled each picture, telling the computer if it's a grape (0) or a raisin (1). This labeling helps the computer learn which picture belongs to which group.

Validation Set: For fine-tuning model hyperparameters and monitoring performance during training, a separate validation set containing 240 images has been used. Similar to the training set, this subset maintains a balanced representation of raisins and grapes, with images captured against a black and white background.

Test Set: This set is the final test for our models. It has 420 pictures with different backgrounds, like black, white, red, green and blue. We made sure to have an equal number of raisins and grapes in this set too.

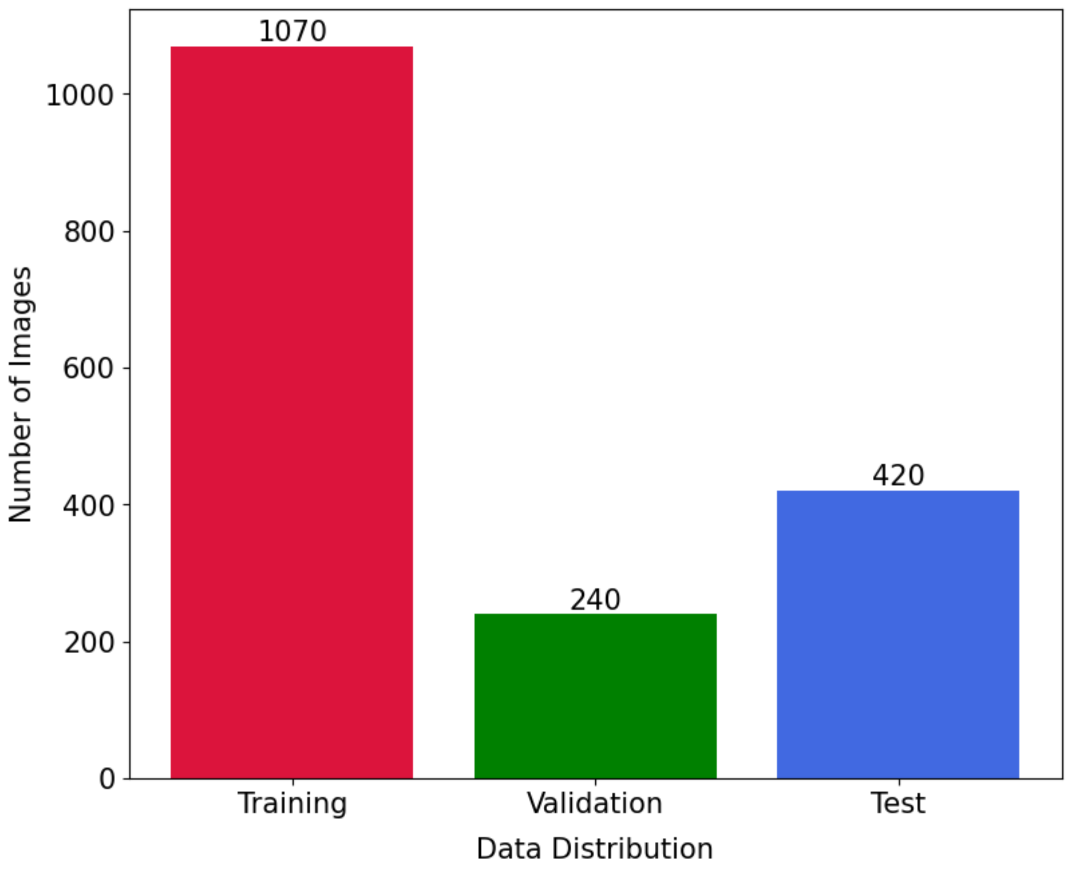


Figure 1: Number of images in the Training, Validation and Test Set.

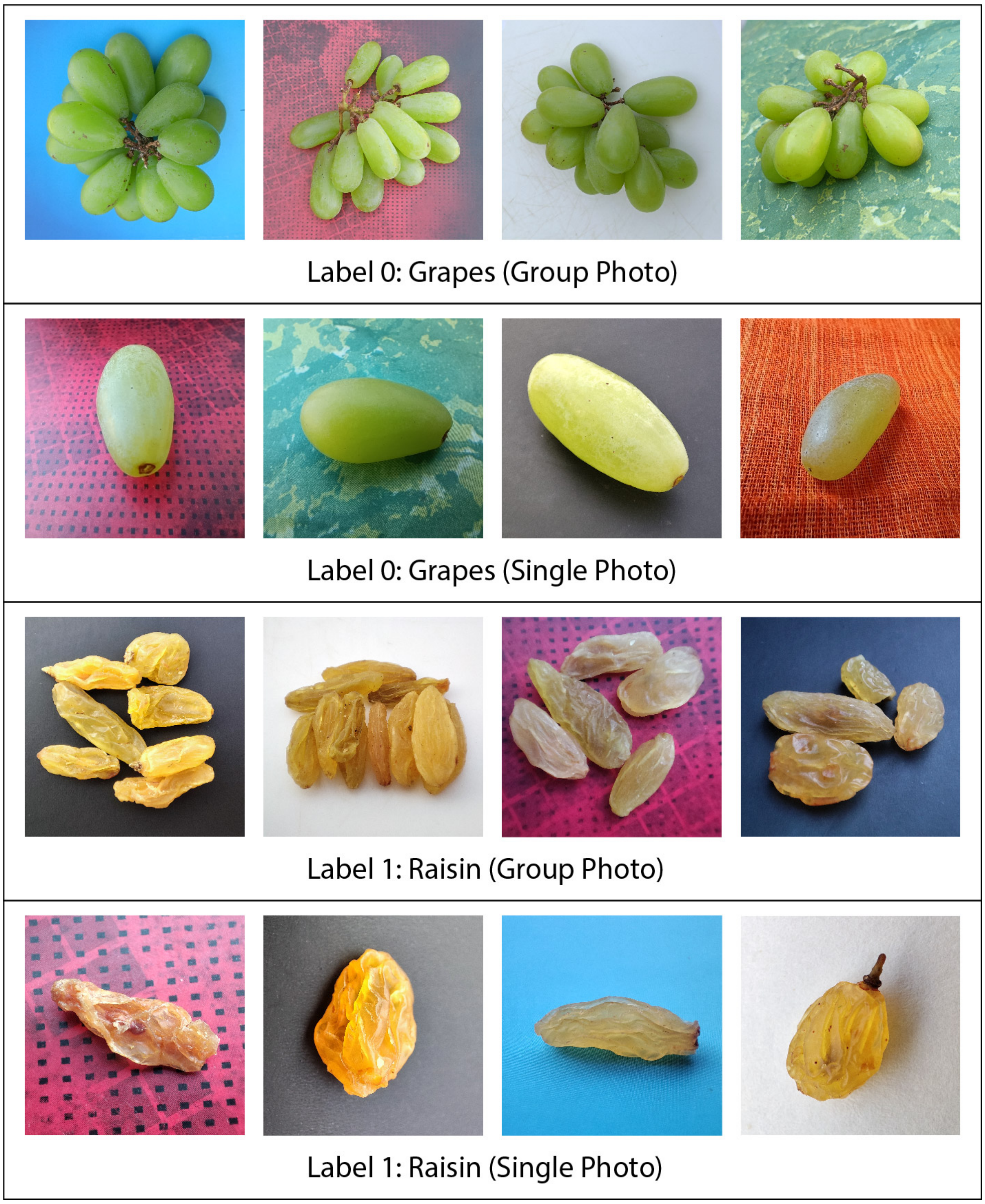

Label 0: Grapes (Group Photo)
Label 0: Grapes (Single Photo)
Label 1: Raisin (Group Photo)
Label 1: Raisin (Single Photo)

Figure 2: Some examples of images from our dataset. Train and Validation Set consists of white and black background and Test Set consists of red, green and blue background.

3 Statistical Analysis of Standard Deviation for Grapes and Raisins

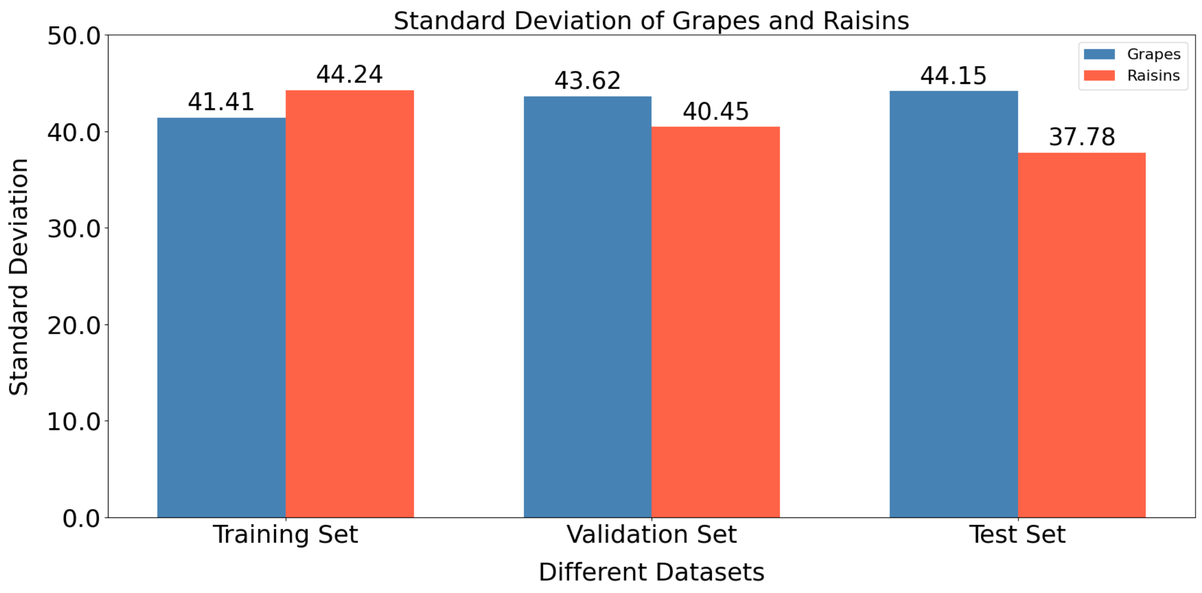


Figure 3: Statistical Analysis in terms of Standard Deviation of our Training, Validation and Test Dataset.

The graph shows pixel intensity variability within the images of grapes and raisins across different datasets. For grapes, the standard deviation progressively increases from the training set (41.41) to the validation set (43.62) and further to the test set (44.15). This indicates a gradual rise in image variability as we transition from training to validation and test datasets. Conversely, for raisins, the pattern is different. The training set exhibits the highest standard deviation (44.24), followed by a decrease in variability in the validation set (40.45), and a further reduction in the test set (37.78). This suggests that raisin images show a decreasing trend in variability across the datasets.

The variability in standard deviation across different datasets (training, validation, test) indicates potential differences in the image characteristics between these datasets. It's essential to consider these variations during model training and evaluation to ensure robust performance across different scenarios.

4 Model Architecture

Both the model architectures are designed using the Keras functional API.

4.1 Convolutional Neural Network

Three sets of Convolution layers with Max Pooling layers and lastly three set of Dense layers consisting of 512, 256, 128 neurons were used as the model architecture. The model has 349217 trainable parameters. Adam optimizer is used for parameter updates during training. Binary crossentropy is employed as the loss function, suitable for binary classification tasks. Model performance is evaluated using accuracy.

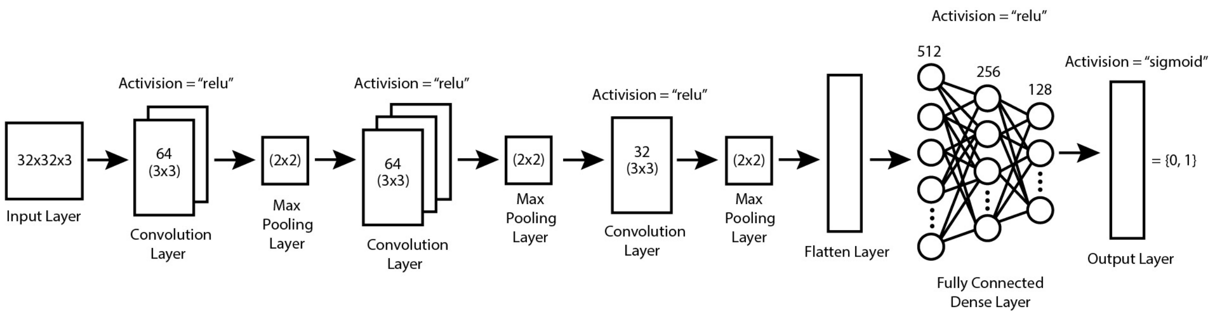


Figure 4: Proposed Convolutional Neural Network Architecture.

4.2 Convolutional Neural Network With Transfer Learning

We built a convolutional neural network (CNN) architecture for image classification tasks, using transfer learning. Specifically, we employed the feature extraction layers from the VGG16 model, utilizing an input shape of 32x32x3. Following this, we added three dense layer consisting of 512, 256 and 128 neurons. The weights were transferred from the pre-trained VGG16 model, which was initially trained on the ImageNet dataset containing 14 million images. The model has 427009 trainable parameters.

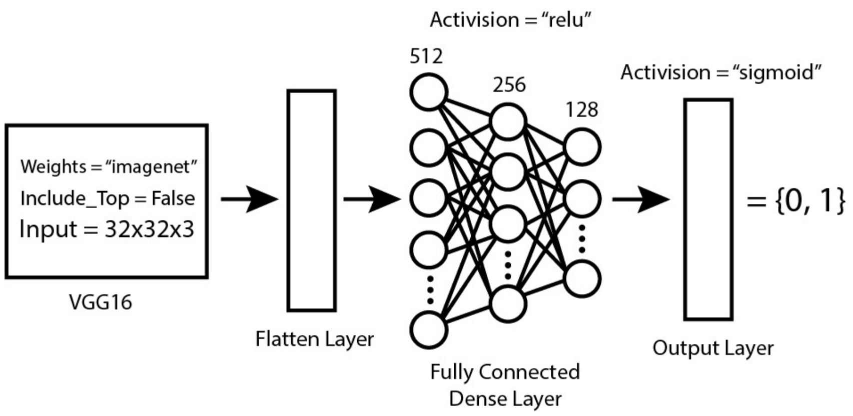
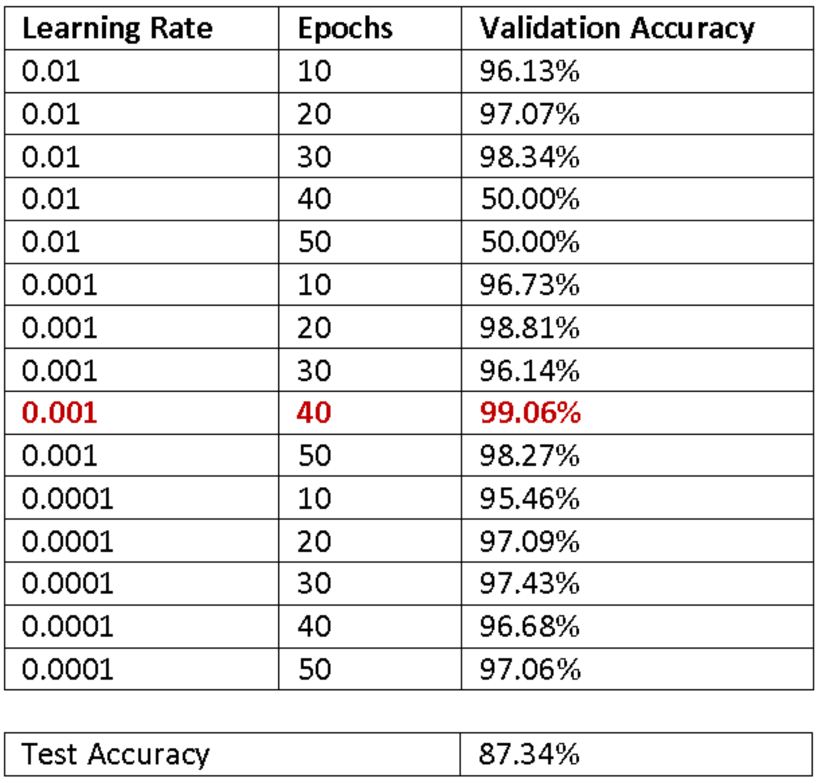


Figure 5: Proposed CNN with Transfer Learning Architecture.

5 Results

5.1 Results without using Callback Functions

After training the model, we achieved an accuracy of 87% on the test dataset.



Learning Rate	Epochs	Validation Accuracy
0.01	10	96.13%
0.01	20	97.07%
0.01	30	98.34%
0.01	40	50.00%
0.01	50	50.00%
0.001	10	96.73%
0.001	20	98.81%
0.001	30	96.14%
0.001	**40**	**99.06%**
0.001	50	98.27%
0.0001	10	95.46%
0.0001	20	97.09%
0.0001	30	97.43%
0.0001	40	96.68%
0.0001	50	97.06%

Test Accuracy	87.34%

Figure 6: Results of different settings using CNN with Transfer Learning. The best performance is highlighted in red colour where Test Accuracy is 87%.

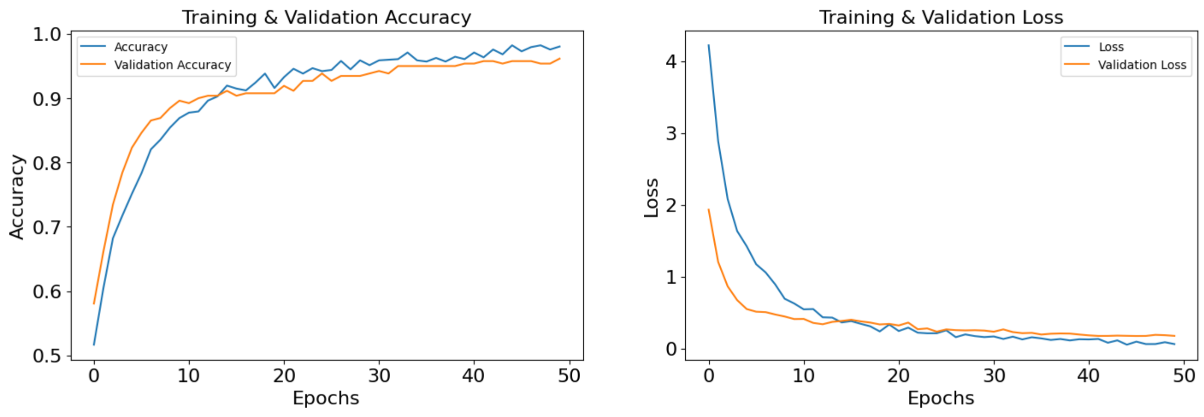
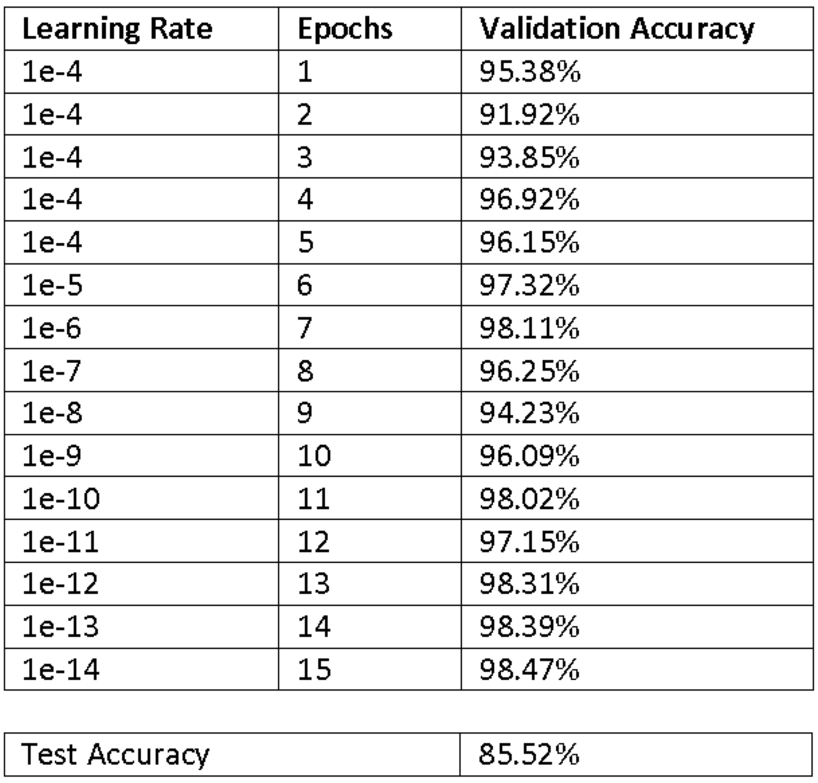


Figure 7: Accuracy and Loss Graph for CNN With Transfer Learning Model. The Training and Validation Accuracy and Loss graph seems to converge well.

5.2 Results using Callback Functions

We used **EarlyStopping** with patience = 10, **ModelCheckpoint** and **LearningRateScheduler** with decerading learning rate. After training the model, we achieved an accuracy of 85% on the test dataset.



Learning Rate	Epochs	Validation Accuracy
1e-4	1	95.38%
1e-4	2	91.92%
1e-4	3	93.85%
1e-4	4	96.92%
1e-4	5	96.15%
1e-5	6	97.32%
1e-6	7	98.11%
1e-7	8	96.25%
1e-8	9	94.23%
1e-9	10	96.09%
1e-10	11	98.02%
1e-11	12	97.15%
1e-12	13	98.31%
1e-13	14	98.39%
1e-14	15	98.47%

Test Accuracy	85.52%

Figure 8: Results of different settings using CNN with Transfer Learning with Callback Functions. We got a Test Accuracy of 85%.

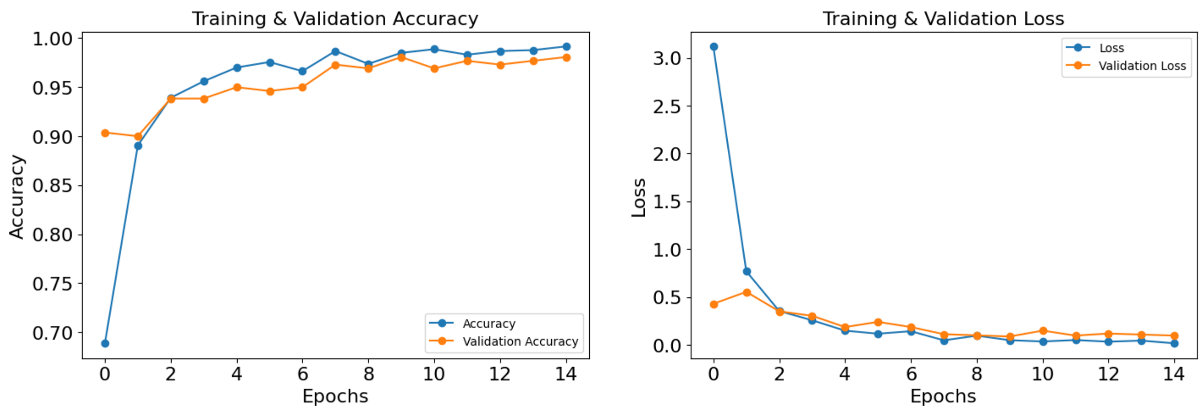


Figure 9: Accuracy and Loss Graph for CNN with Transfer Learning Model with Callback Functions. The Training and Validation Accuracy and Loss graph seems to converge well.

6 Discussion

In our binary image classification task distinguishing between grapes and raisins, we observed that our custom CNN model using transfer learning from VGG16 model was better, with 87.34% (without using Callback Functions) and 85.52% (using Callback Functions) accuracy, compared to our custom CNN model without transfer learning, which achieved 85% accuracy. This difference happened because the VGG16 model already learned a lot from a huge dataset called ImageNet, giving it a significant advantage right from the start. It knows better how to recognize different things in pictures. Our custom CNN model, on the other hand, had to learn everything from scratch, so it might have missed some important details. Additionally, the architectural differences between those two models, such as the number of layers and neurons, can also influence their respective performances. Overall, using the pre-trained VGG16 model with transfer learning gave us better results.

6.1 Effects of using Transfer Learning

1. We were able to train the model faster.

2. We achieved higher accuracy.

3. We were able to train with less training data.

Overall, the model seems to perform well in distinguishing between grapes and raisins in the given context.