Exercise 2: Image Recognition to Identify Species of Flowers

a. The architecture of the CNN model designed for flower classification using the Keras API with Tensorflow. It contains the following segments-

Input Layer: In input layer images of size 224x224 pixels with 3 color channels (RGB) was used.

Convolutional Layers: The CNN model has 4 convolutional layers. In conv1 there are 32 filters, kernel size of (5, 5), padding same and Rectified Linear Unit (relu) activation function. In conv2 there are 64 filters, kernel size of (3, 3), padding same and ReLu as activation function. In conv3 and conv4 there are 96 filters, kernel size of (3, 3), padding same and ReLu activation.

Pooling Layers: Max Pooling in pooling layers have been used. In this CNN model, it has 4 pooling layers, pool size of (2, 2) and a stride of (2, 2).

Activation Function: In Convolutional layer and Dense layer ReLu activation function and in output layer softmax activation function was used.

Flatten Layer: Flattens the output from the pool4 layer into a 1D array.

Dense Layer: The dense layer (dn2) uses 128 neurons and ReLu activation.

Output Layer: 5 neurons (for 5 species of flowers) and softmax activation.

This CNN model designed to identify the species of flowers. The CNN model will provide the best output for this respective task. For better results, the CNN model used several convolution layer, maxpooling layer. The choice of ReLU activation functions in convolution layers and dense layer(dn2) introduces non-linearity, fostering the model's ability to learn complex features. This architecture also balances depth and computational efficiency, aligning with best practices for image classification tasks. The model concludes with a softmax activation in the output layer, facilitating multi-class predictions. Finally, it can be said undoubtedly, this is the best choice for flowers identification (image classification).

b. Regularization methods that are used in the CNN model-

Data Augmentation: Implemented using **ImageDataGenerator** to introduce variety into the training data as well as reducing overfitting.

ReduceLROnPlateau: Adjusting the learning rate during training to improve convergence.

EarlyStopping: Halts training if the validation loss doesn't improve after a certain number of epochs.

Effect on Accuracy of above’s regularization methods**:** Data augmentation helps the model generalize better to unseen data whereas learning rate reduction and early stopping prevent overfitting and improve model performance.

c. The following hyperparameters were adjusted for the CNN model in order to make a better prediction-

batch = 32, epoch = 10: accuracy 68%

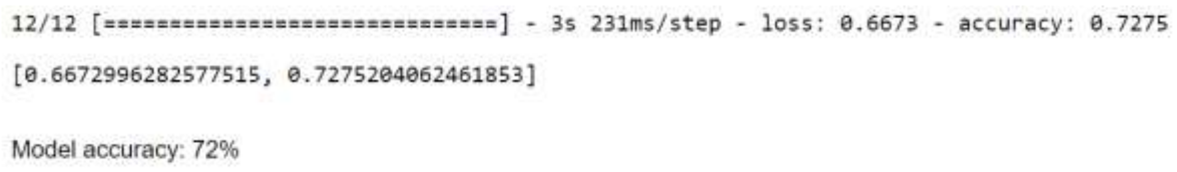
batch = 50, epoch = 10: accuracy 68%

batch = 100, epoch = 10: accuracy 65%

batch = 150, epoch = 10: accuracy 50%

batch = 100, epoch = 15: accuracy 67%

batch = 32, epoch = 30: accuracy 72%



```
12/12 [==============================] - 3s 231ms/step - loss: 0.6673 - accuracy: 0.7275
[0.6672996282577515, 0.7275204062461853]

Model accuracy: 72%
```

Fig: Model accuracy when batch size = 32 and epoch = 30

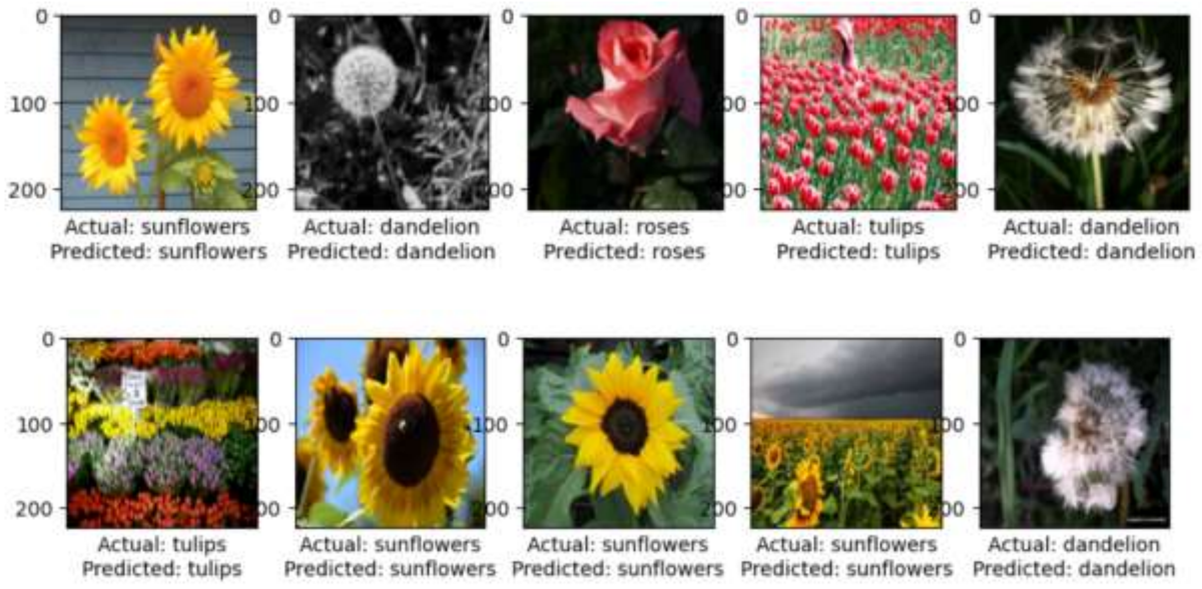


Fig: Comparison between predicted value and actual value

d. Overfitting: Using training and validation accuracy and loss curves one can check overfitted model. If the training accuracy is significantly higher (>10%) than validation accuracy, or if the validation loss increases while training loss decreases, it may indicate overfitting. There is no overfitting in this CNN model. To prevent overfitting, learning rate reduction and early stopping in the CNN model was introduced.

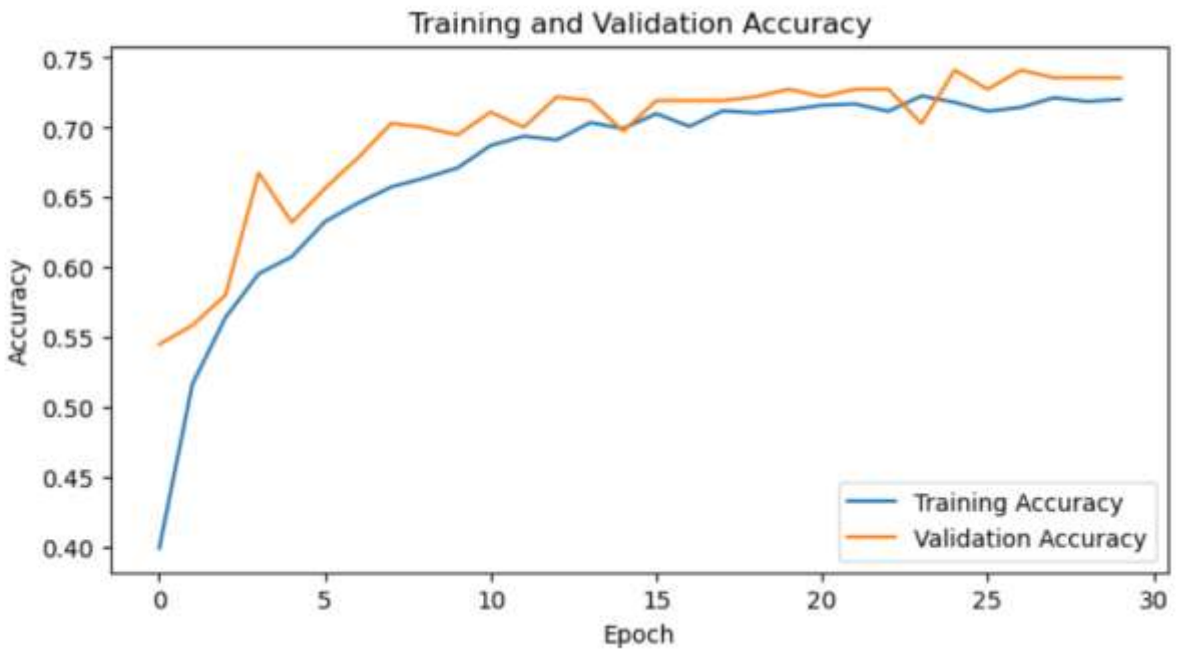


Fig-01: Training accuracy vs validation accuracy curve

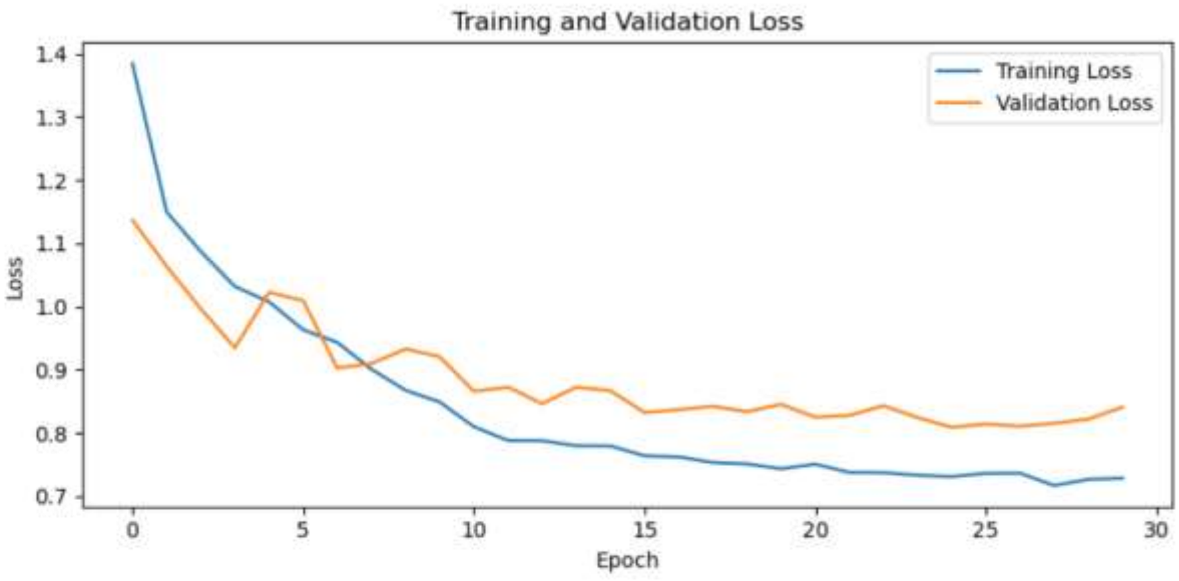


Fig-02: Training loss vs validation loss