Flower Recognition CNN Keras

Tianai Tang 6219703

Contents

Approach

- This project is about recogeniting the types of flowers.
- This project trained Convolutional Neural Network written in Keras to predict the type of flower on the validation set.
- Also used ImageDataGenerator to augment the training set and avoid overfitting problem and a LR annealer to schedule the learning rate.

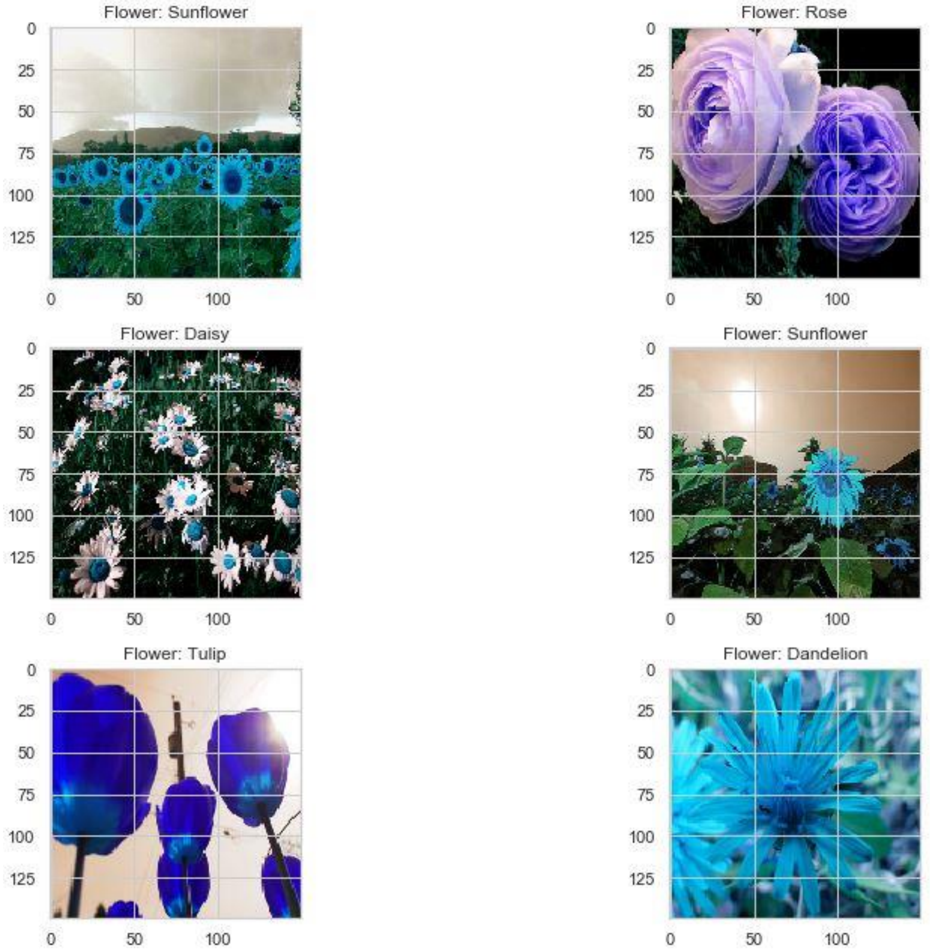
Preparing the Data

Dataset

- This dataset contains 4242 images of flowers. The data collection is based on scraped data from flickr, google images, and yandex images.
- The pictures are divided into five classes: chamomile, tulip, rose, sunflower, dandelion. For each class there are about 800 photos. Photos are not high resolution, about 320x240 pixels. Photos are not reduced to a single size, they have different proportions.

Preparing the Data

Dataset



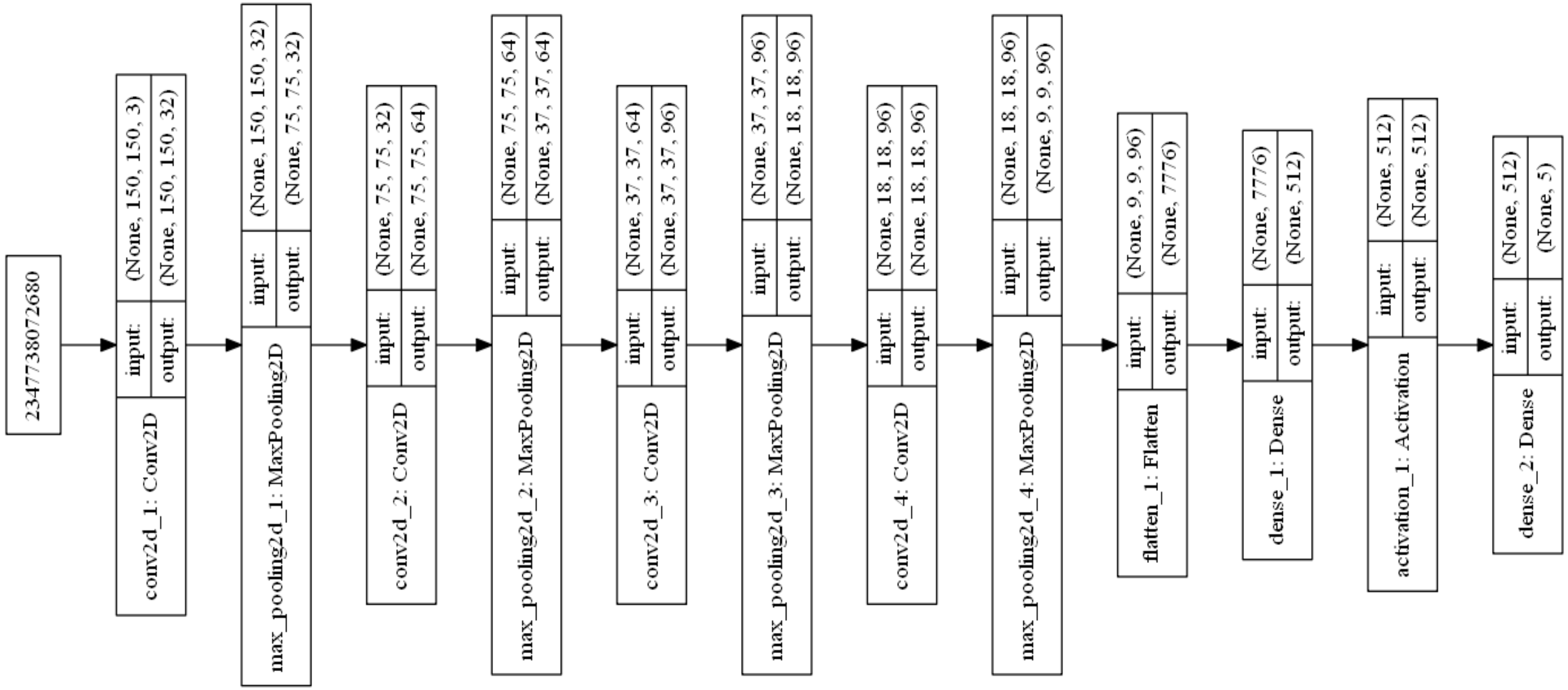
Preparing the Data

Resize

- Smaller and constant size of all images are required for CNN to do image classification, because the model requires a constant input dimensionality and low resolution will speed up the model training.
- In the project, reasonable resolution of 150*150 pixels is applied to each image.

Model Design

Build CNN architecture



Using plot_model to visualize the model

Build CNN architecture

- 4x(conv2d+max-pooling)+flatten+dense+activation+dense

Conv2D layer

- This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

Model Design

Conv2D layer

```
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation ='relu', input_shape = (1
```

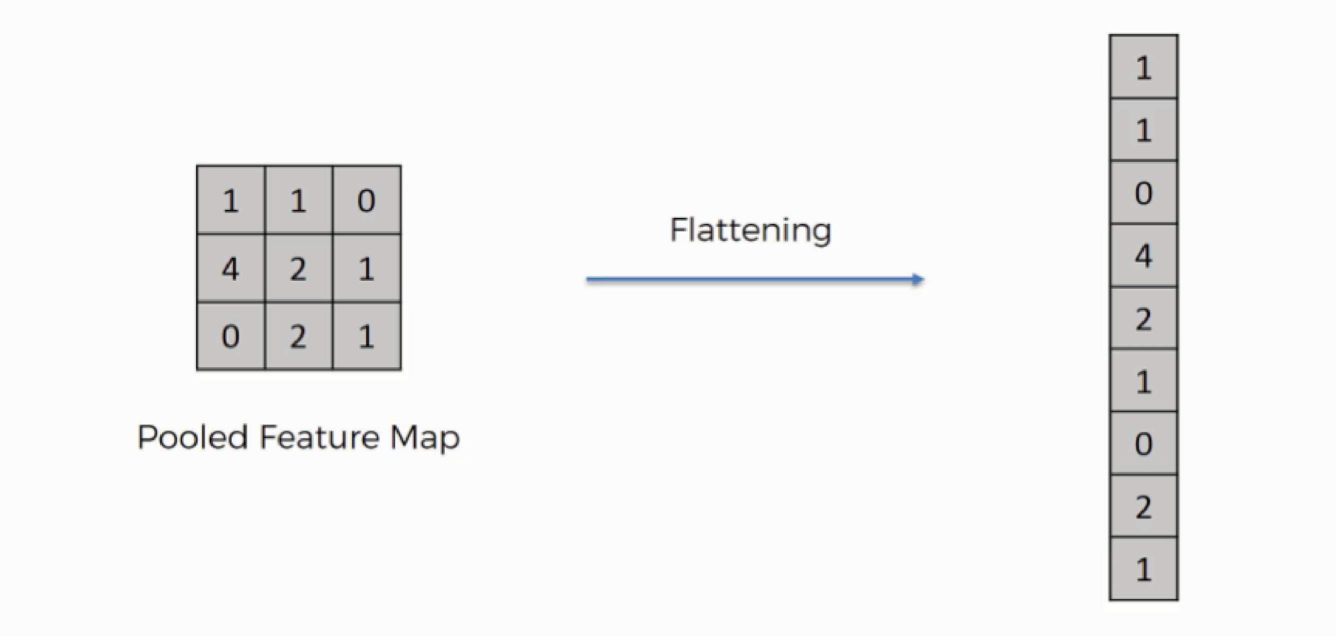
- When using this layer as the first layer in a model, provide the keyword argument input_shape (tuple of integers, does not include the batch axis), e.g. input_shape=(128, 128, 3) for 128x128 RGB pictures in data_format="channels_last".
- The ordering of the dimensions in the inputs. "channels_last" corresponds to inputs with shape (batch, height, width, channels).

Pooling layer

- In pooling layers, features are extracted and compressed into a small map, which simplifies the neural network computation complexity, leading to the decrease of the volume of parameters and computation.
- In this project, we use max pooling.

Flatten layer

- A flatten layer collapses the spatial dimensions of the input into the channel dimension.

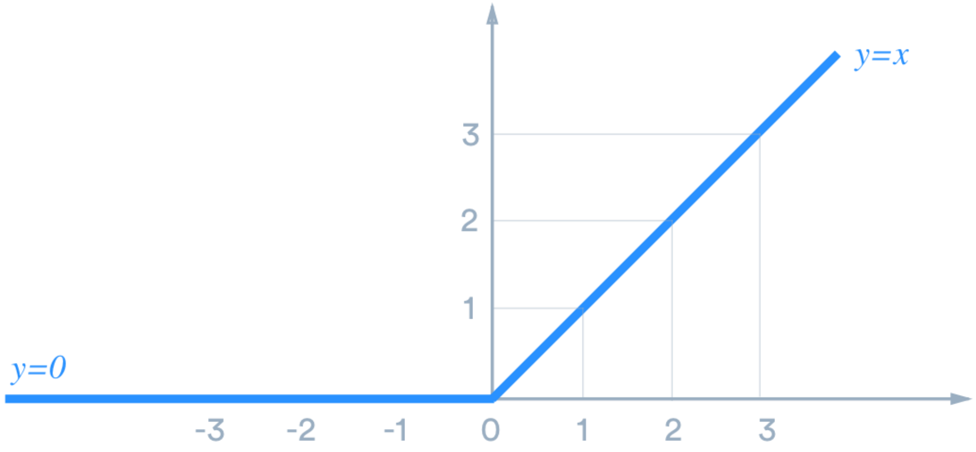


Model Design

Activation layer(ReLU)

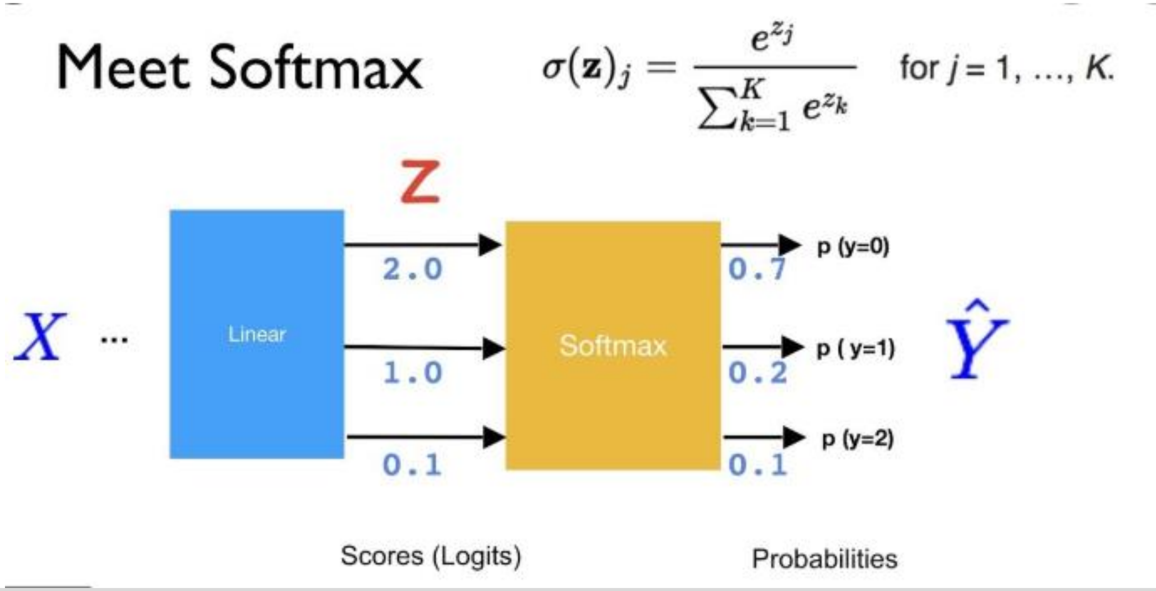
- Rectified Linear Unit (ReLU) is a piecewise linear function implemented in this model. The ReLU activation function is given by : different proportions.

$$ReLU = \begin{cases} x, & \text{if } n > 0 \\ 0, & \text{if } x \leq 0 \end{cases}$$



Softmax Function

- The softmax function is used in neural networks when we want to build a multi-class classifier which solves the problem of assigning an instance to one class when the number of possible classes is larger than two.



Softmax Function

Properties of Softmax Function:

- The calculated probabilities will be in the range of 0 to 1.
- The softmax function generates high probability for a high value
- The sum of all the probabilities is equals to 1.

Learning rate annealing(ReduceLROnPlateau)

- Reduce learning rate when a metric has stopped improving.
- Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

Data Augmentation(ImageDataGenerator class)

the Keras ImageDataGenerator class actually works by:

- Accepting a batch of images used for training.
- Taking this batch and applying a series of random transformations to each image in the batch (including random rotation, resizing, shearing, etc.).
- Replacing the original batch with the new, randomly transformed batch.
- Training the CNN on this randomly transformed batch (i.e., the original data itself is not used for training).

Model Training

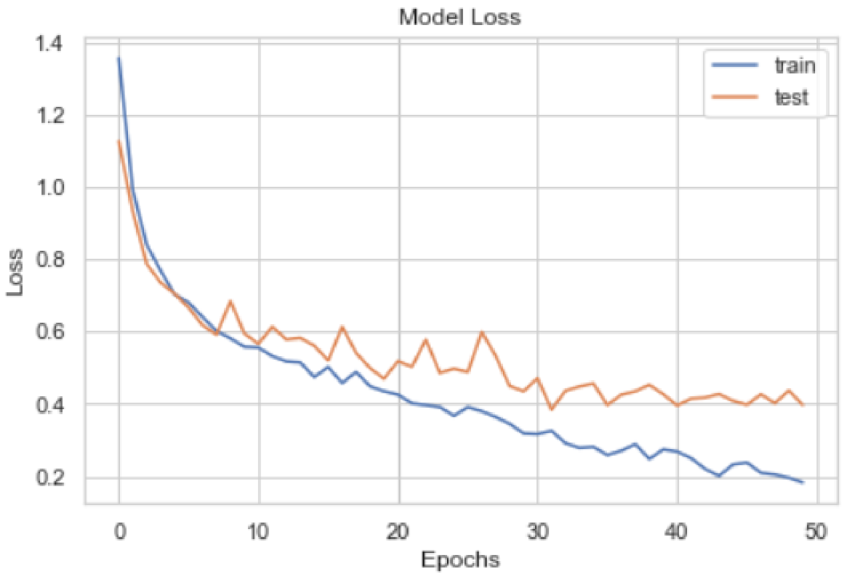
Model architecture-4

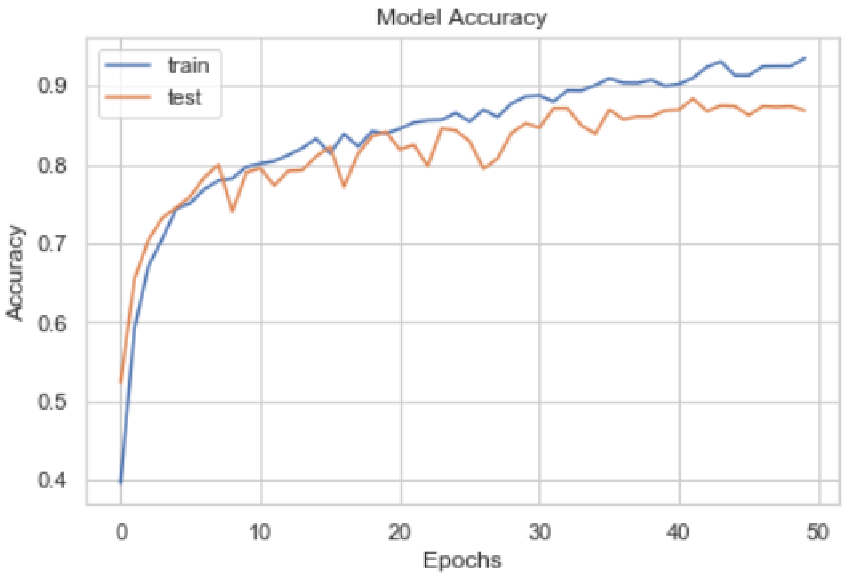
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 150, 150, 32)	2432
max_pooling2d_1 (MaxPooling2	(None, 75, 75, 32)	0
conv2d_2 (Conv2D)	(None, 75, 75, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 37, 37, 64)	0
conv2d_3 (Conv2D)	(None, 37, 37, 96)	55392
max_pooling2d_3 (MaxPooling2	(None, 18, 18, 96)	0
conv2d_4 (Conv2D)	(None, 18, 18, 96)	83040
max_pooling2d_4 (MaxPooling2	(None, 9, 9, 96)	0
flatten_1 (Flatten)	(None, 7776)	0
dense_1 (Dense)	(None, 512)	3981824
activation_1 (Activation)	(None, 512)	0
dense_2 (Dense)	(None, 5)	2565

Total params: 4,143,749
Trainable params: 4,143,749
Non-trainable params: 0

Model Training

Result





Finally the accuracy on the validation set using the self-laid ConvNet is over 85%.

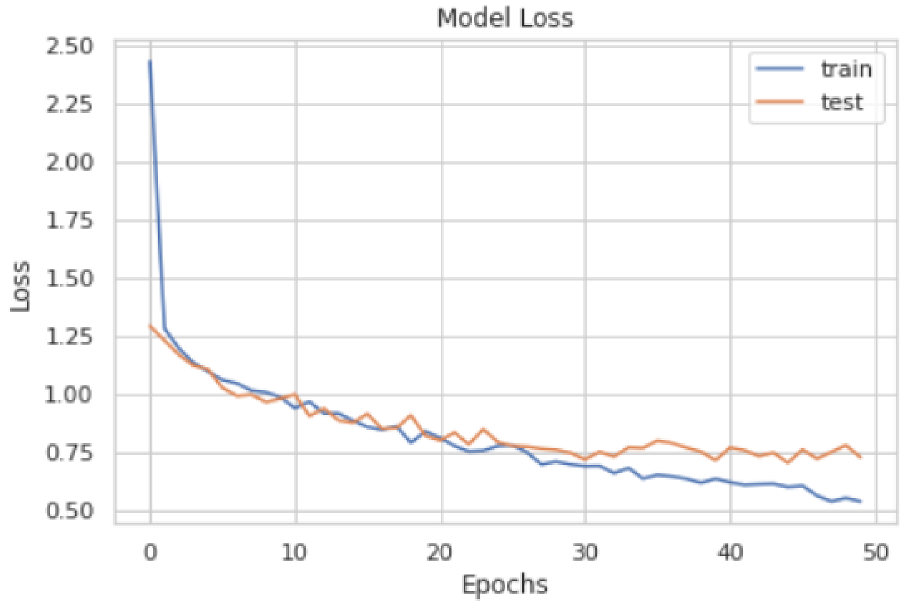
Model Training

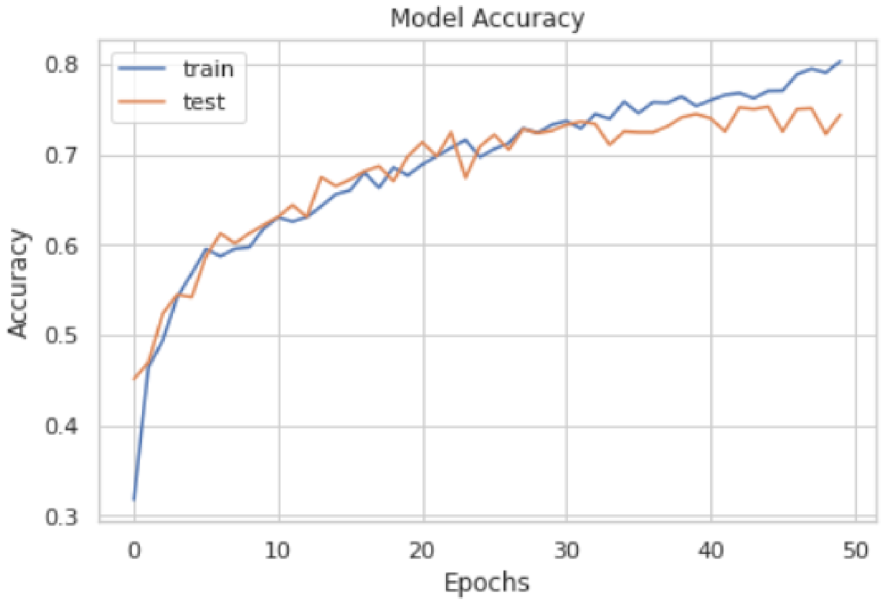
Model architecture-2

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 150, 150, 32)      2432
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 75, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 37, 37, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 87616)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               44859904
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 2565
=================================================================
Total params: 44,883,397
Trainable params: 44,883,397
Non-trainable params: 0
_________________________________________________________________
```

Model Training

Result-2





Finally the accuracy on the validation set using the self-laid ConvNet is close to 75%.

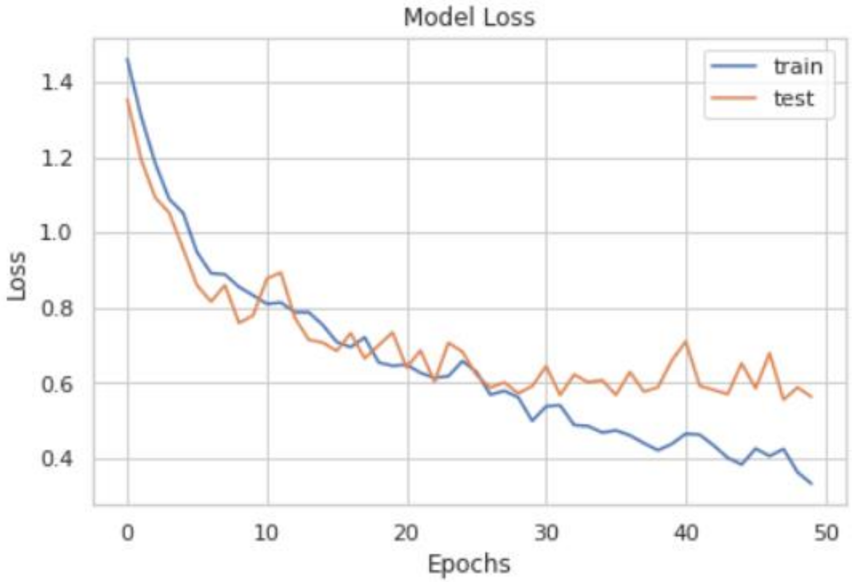
Model Training

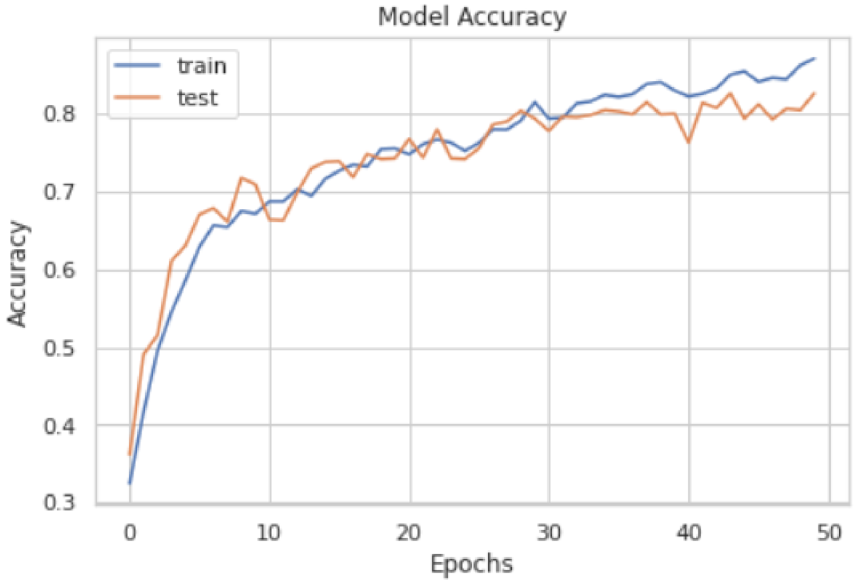
Model architecture-6

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 150, 150, 32)      2432
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 75, 75, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 37, 37, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 37, 37, 96)        55392
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 18, 18, 96)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 18, 18, 96)        83040
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 9, 9, 96)          0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 9, 9, 96)          83040
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 96)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 4, 4, 96)          83040
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 2, 2, 96)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 384)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               197120
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 2565
=================================================================
Total params: 525,125
Trainable params: 525,125
Non-trainable params: 0
```

Model Training

Result-6





Finally the accuracy on the validation set using the self-laid ConvNet is around 80%.

Model Training

Why

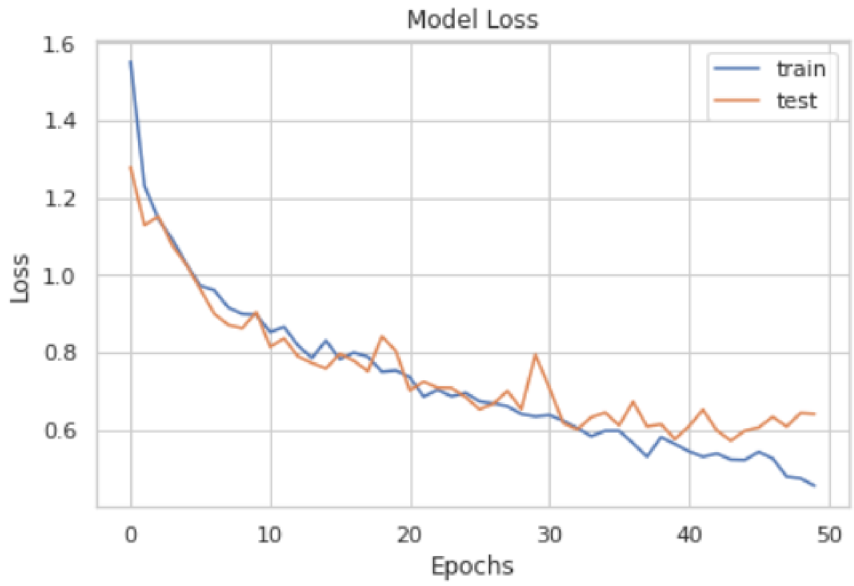
When there are too many hidden layers, the accuracy may decline.

This is because the more hidden layers the Gradient in the Back Propagation algorithm goes through, the smaller it will be and gradually approaches zero.

This phenomenon is called Vanishing Gradient Problem.

Model Training

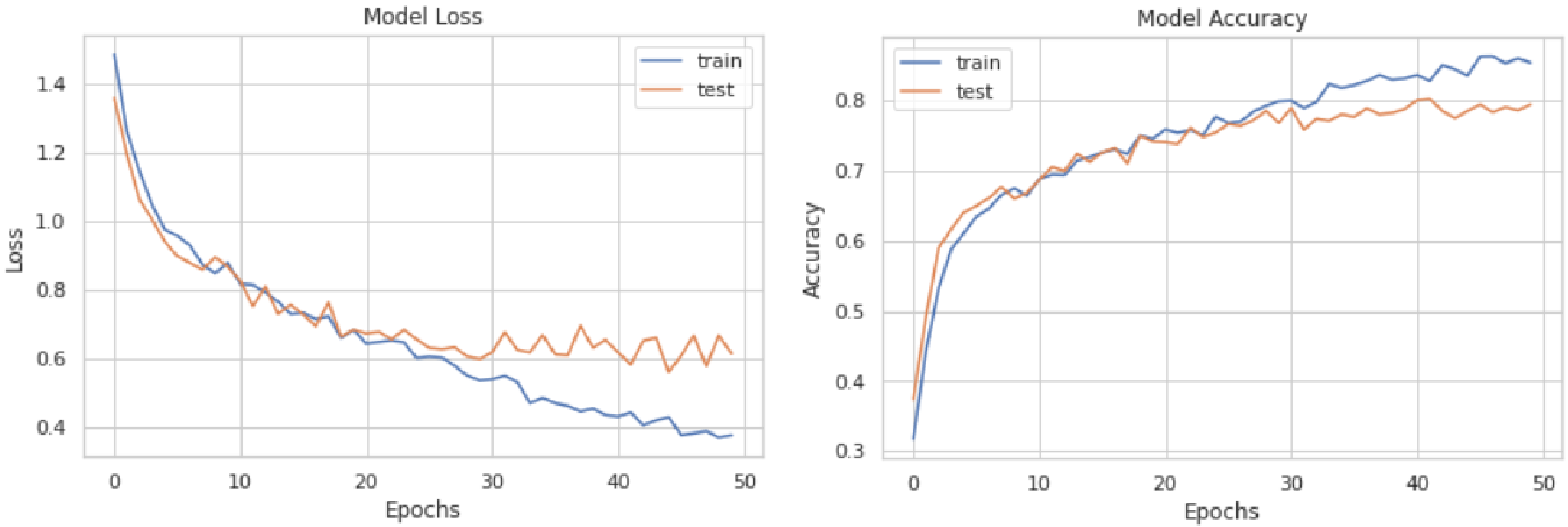
Result-4x32



Finally the accuracy on the validation set using the self-laid ConvNet is around 80%.

Model Training

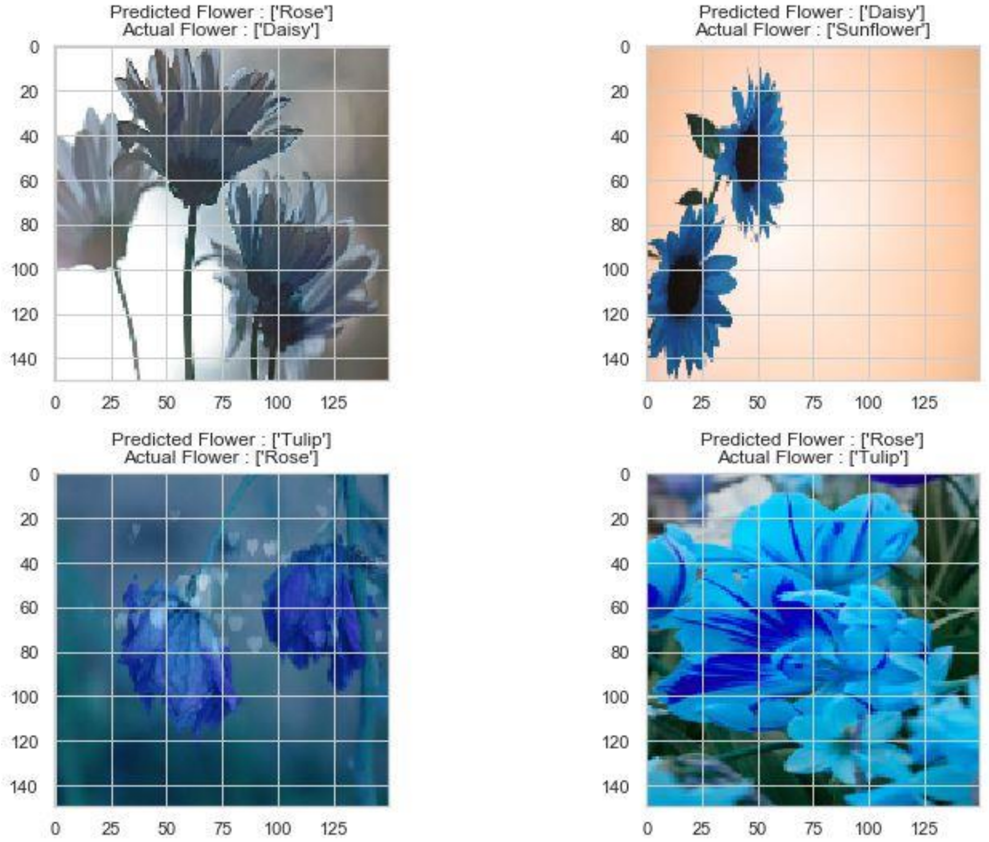
Result-4x64



Finally the accuracy on the validation set using the self-laid ConvNet is around 80%.

Model Training

Misclassified images of flowers



- The reason why the model misclassified could be because the flowers are not front facing ,too big , too small and so on.