Foundations of Deep Learning

P. Formicola

Purposes and Case Study

This project is developed on Flowers Photo Dataset.

Image Classification Problem.

The aim of this project is to build a Deep Learning Model capable of correctly classify the different species of flowers by analyzing photos.

Python has been used to develop the project.

(Tensorflow – Keras)

Data

Data provided were 3690 flowers photo, each sorted into subfolders with their respective Label.

First of all we splitted the dataset into:

- Training Set: 2955 Photos
- Validation Set: 735 Photos

Consistency of Training and Validation Set

- Check for correct generation of Training and Validation Set.

```
Class: daisy - Size of subsample: 507
Class: dandelion - Size of subsample: 719
Class: roses - Size of subsample: 513
Class: sunflowers - Size of subsample: 560
Class: tulips - Size of subsample: 656
```

```
Class: daisy - Size of subsample: 126
Class: dandelion - Size of subsample: 179
Class: roses - Size of subsample: 128
Class: sunflowers - Size of subsample: 139
Class: tulips - Size of subsample: 163
```

First Model (1.0)

Epoch: 10

Learning Rate: 0.001

Batch Size: 32

1. Conv2D: This convolutional layer has 32 filters with a size of 3x3 and uses the ReLU activation function. The input shape is specified as (224, 224, 3), indicating that the model expects images of size 224x224 with 3 color channels (RGB).

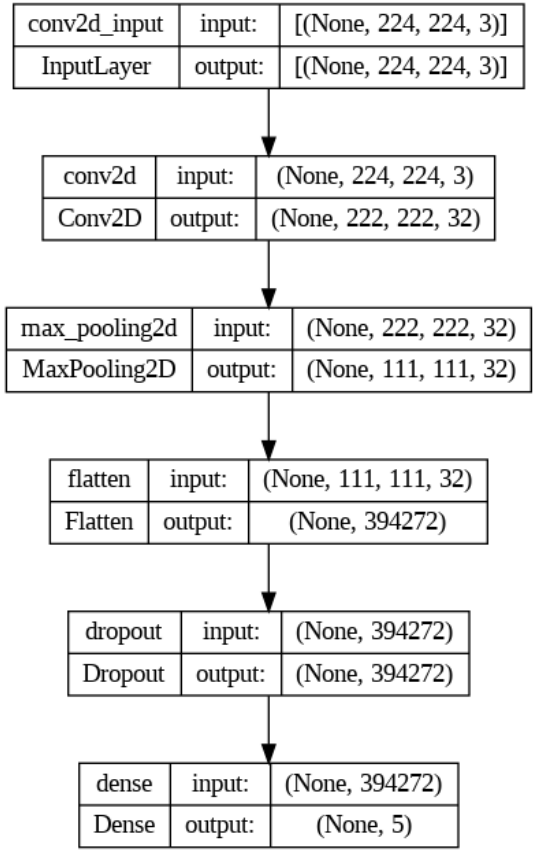
2. MaxPooling2D: This pooling layer reduces the dimensionality of the feature space extracted from the previous layer. In this case, a 2x2 pooling is used.

3. Flatten: This layer transforms the 2D feature space into a 1D vector, allowing it to be connected to the subsequent fully connected layers.

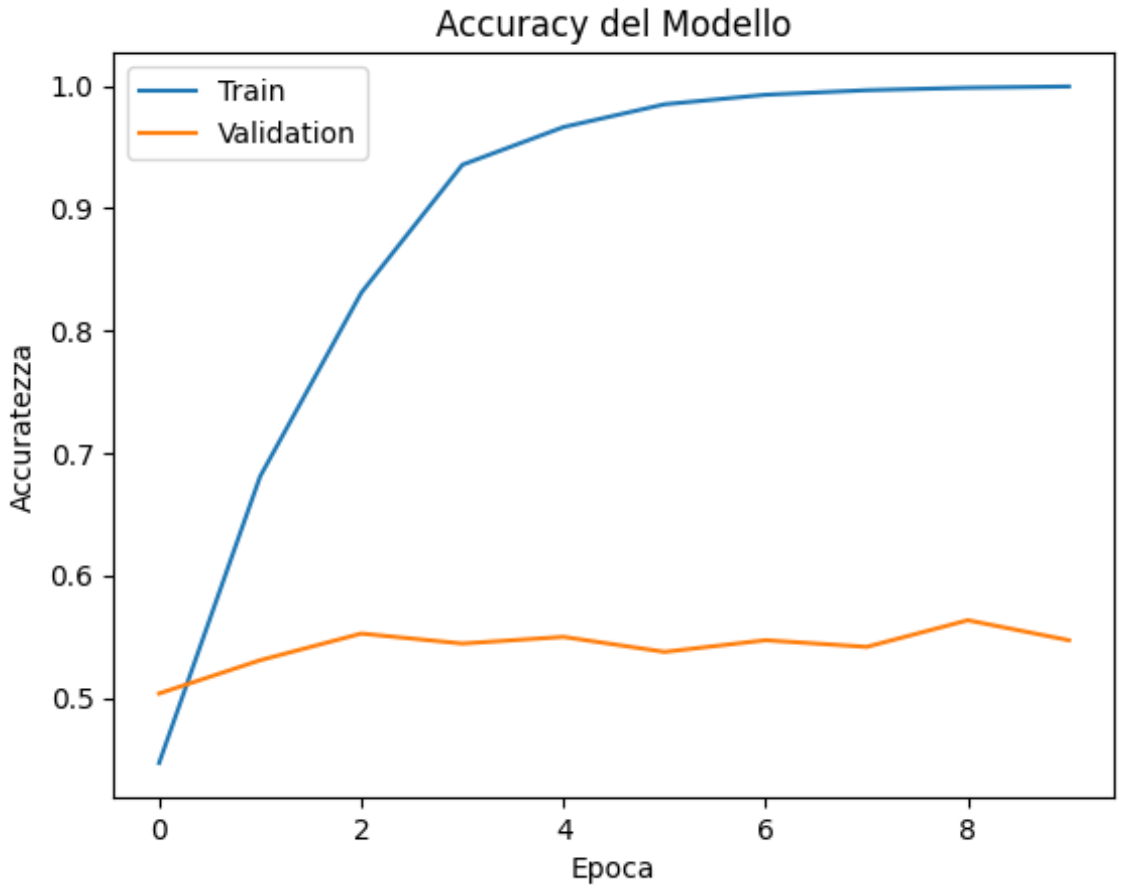
4. Dropout: This dropout layer is used to reduce overfitting during model training.

5. Dense: This fully connected layer has 5 output units, corresponding to the 5 predicted output classes. It uses the softmax activation function, which provides a probability distribution over all classes, allowing for the predicted probabilities for each class.

It takes input images of size 224x224 with 3 color channels and produces a probability distribution over 5 output classes.



Results

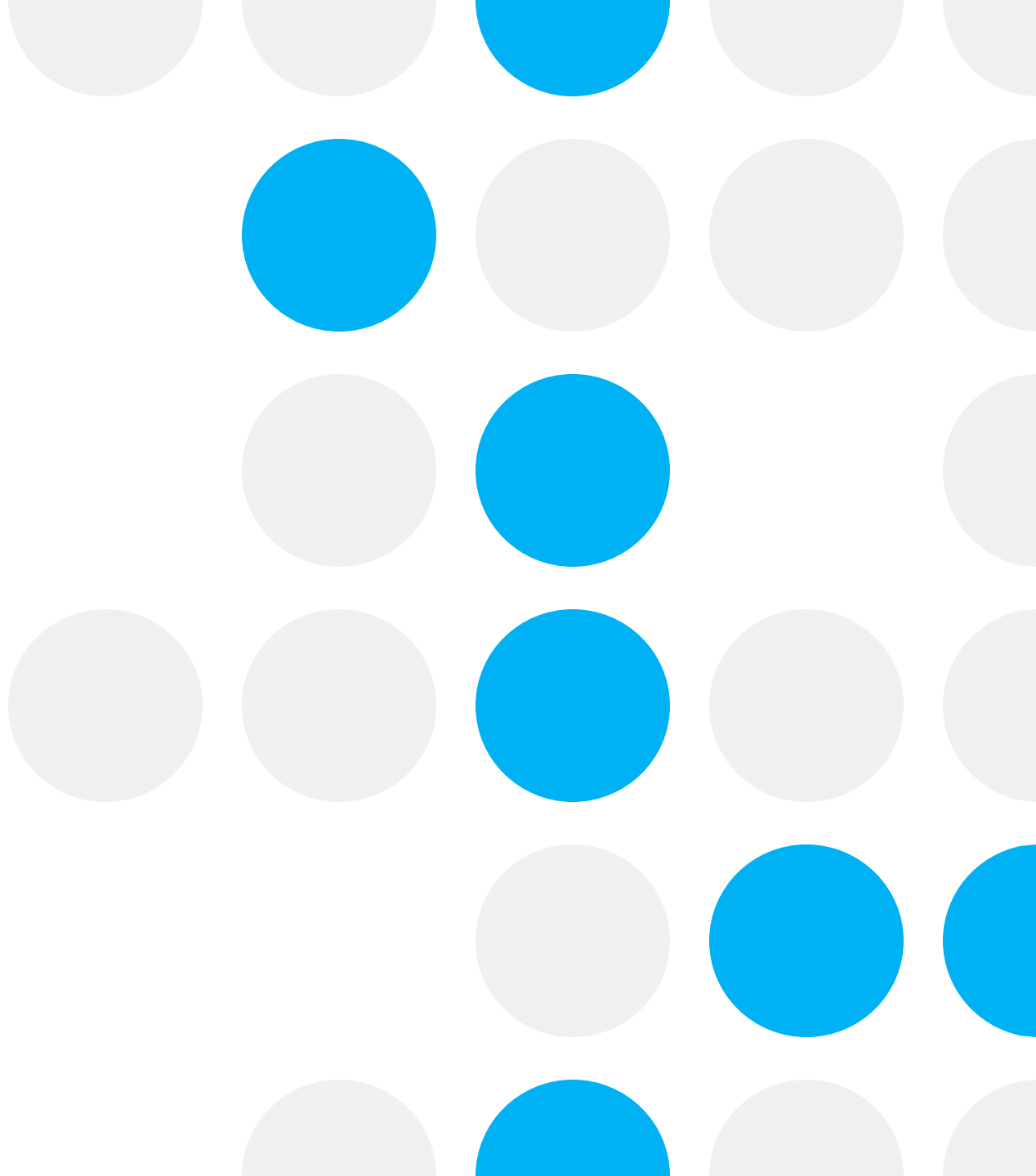


The Accuracy on Validation Set is of **54,69%.**

The model is overfitting maybe due to a low complexity of the model or some imabalances.

How to Improve?

Try to apply some Data Augmentation Techniques and implement a Batch Normalization Layer.

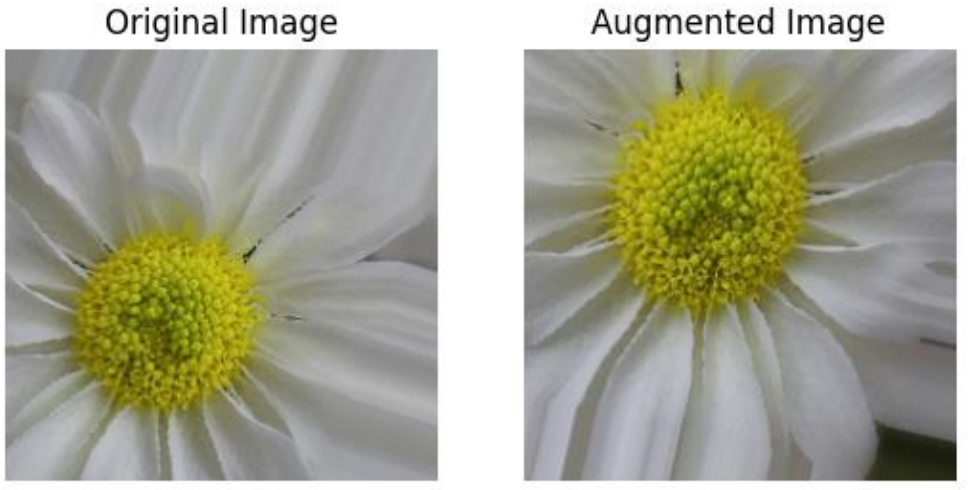


Data Augmentation

Improve model Robustness:

By applying random transformations to the training images, data augmentation exposes the model to various real-world variations and distortions that can occur in the test or real-world scenarios

- Rotation of 20°
- Zoom of 20%
- Width Shift of 20%
- Height Shift of 20%
- Horizontal Flip



Second Model (2.0)

Epoch: 15

Learning Rate: 1 e-4

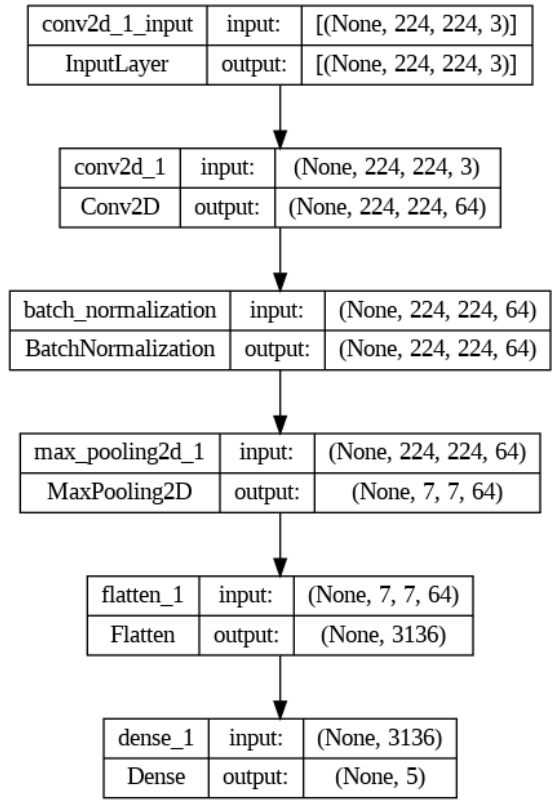
Batch Size: 64

We introduced a Batch Normalization:

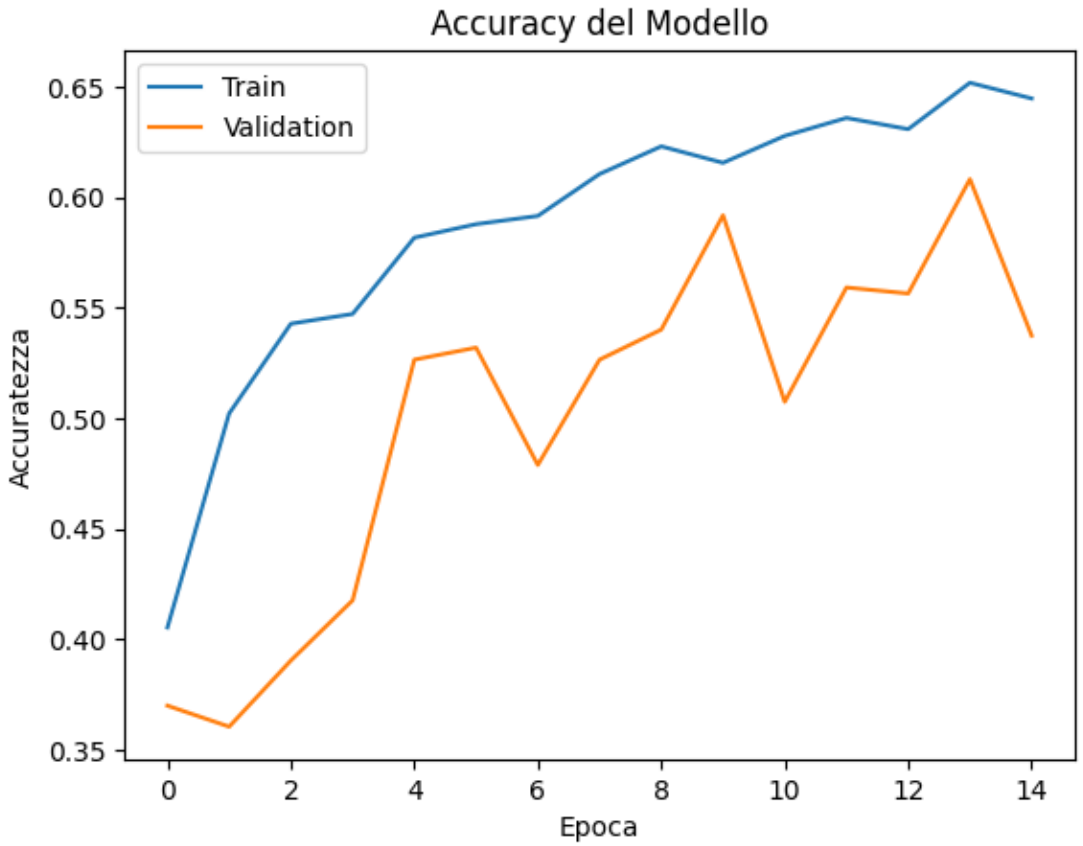
1. **Normalization of input data**: Batch normalization helps in normalizing the input data to each layer of the network. This normalization ensures that the inputs have zero mean and unit variance, which can lead to faster convergence and better performance.

2. **Regularization effect**: Batch normalization introduces a small amount of noise to the layer activations during training, which acts as a regularizer. This noise helps in reducing overfitting and improves generalization performance.

Overall, batch normalization can help improve the training process, accelerate convergence, and enhance the generalization performance of deep neural networks. It is widely used in practice and has become a standard technique for many network architectures.



Results



The Accuracy on Validation Set is of **53,74%**.

As we expected by introducing Batch Normalization, the accuracy on Training Set decreased.

But the accuracy on validation set decreased too.

Further Improvements

We tried a deeper Model, resizing the images to 112x112.

Third Model (3.0)

Epoch: 40

Learning Rate: 1 e-4

Batch Size: 128

This model architecture follows a pattern of:

- **Convolutional** Layers
- **Batch Normalization** Layers
- **Max Pooling** Layers
- **Increasing** filter sizes

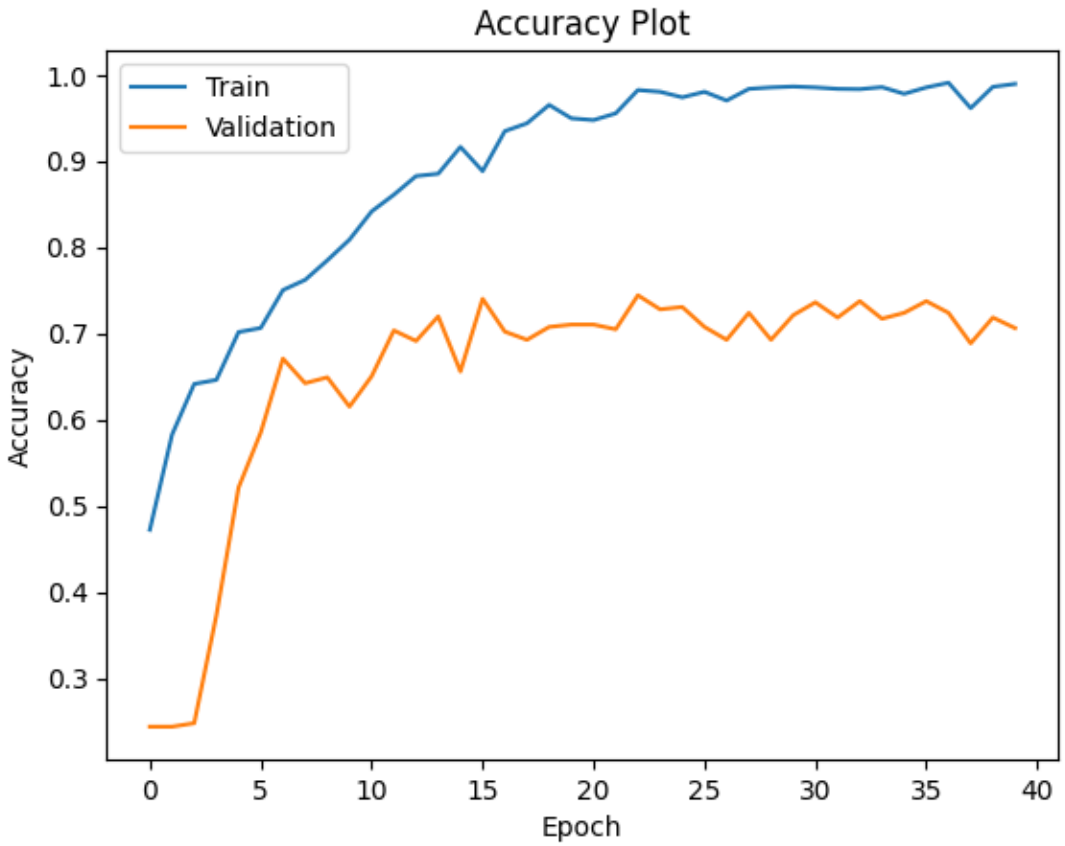
Followed by fully connected layers for classification.

Increased number of Epochs to 40 and a bigger Batch Size (128) to balance speed.

We kept the Adam Optimizer with a learning rate of ***1e-4*** is used to optimize the model's weights.

[Image Available Here]

Model 3.0 - Results

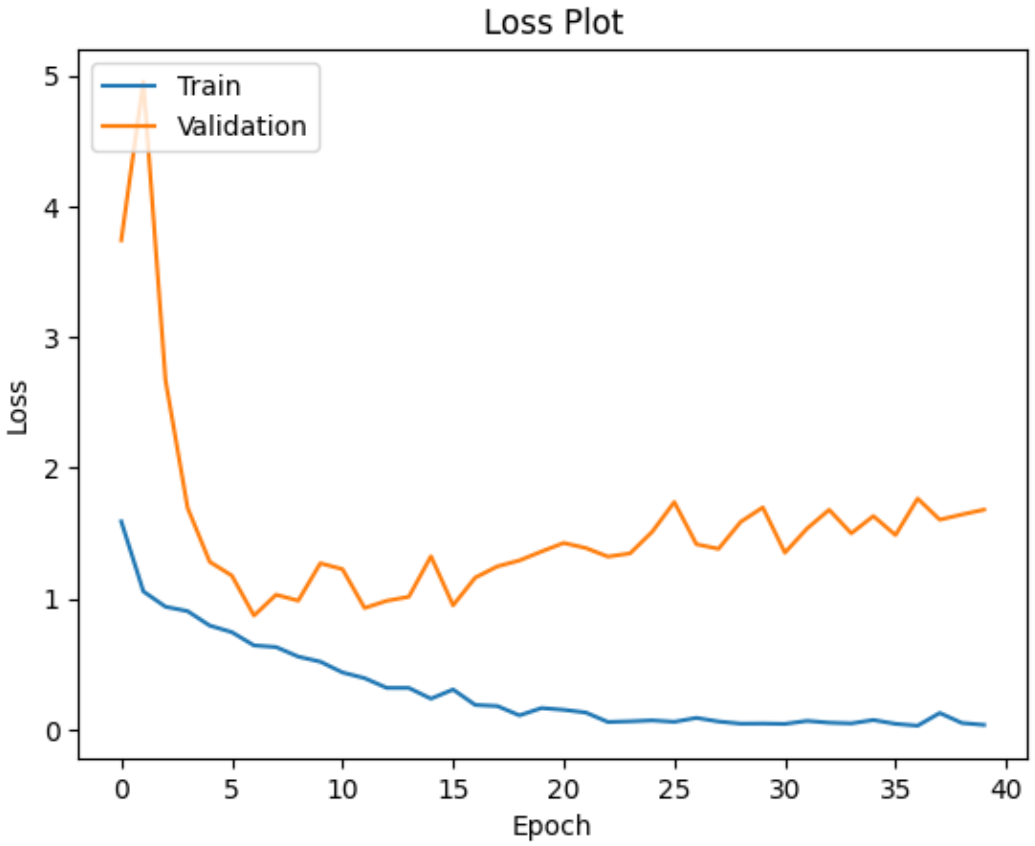


The Accuracy on Validation Set is of **70,61%.**

As we expected with a deeper Net we increased significantly the Accuracy hitting the best result to now.

The main problem is still Overfitting.

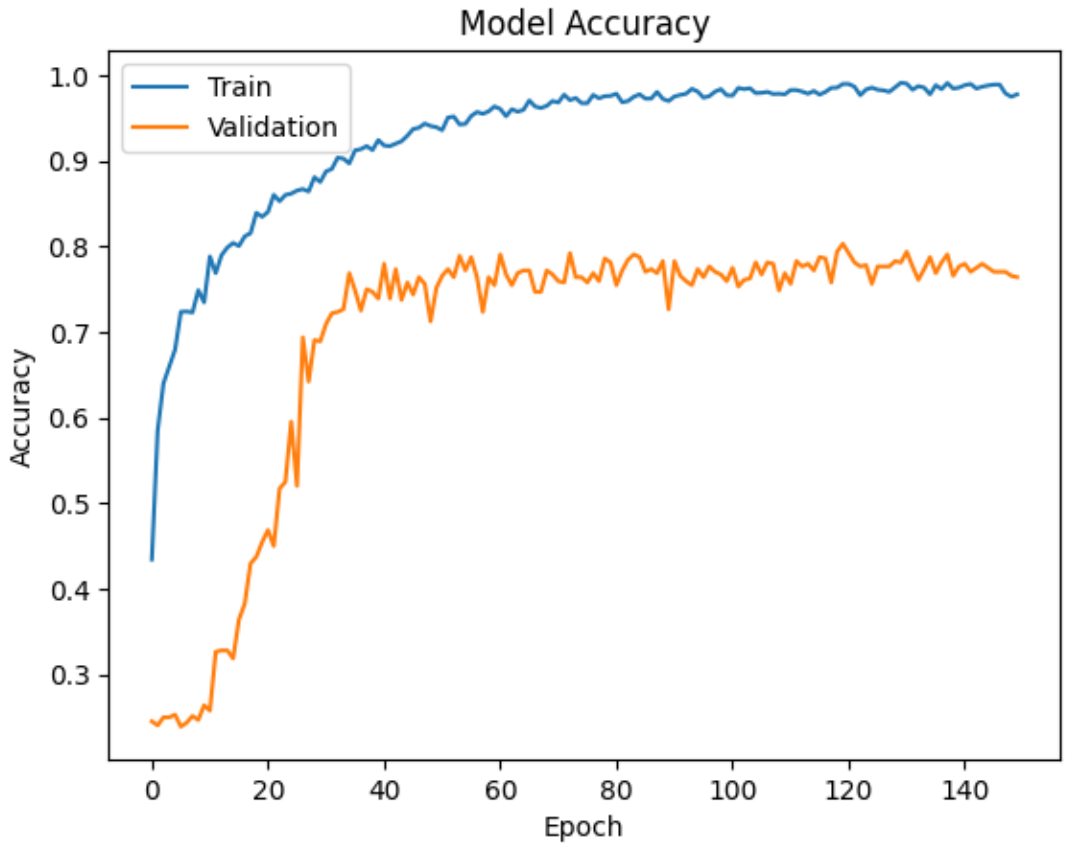
Model 3.0 - Results



The model achieved a very low training loss of **0.0341**, indicating that it was able to effectively minimize the discrepancy between the predicted and actual labels during training.

On the other hand, the validation loss was **1.6799**, which is significantly higher than the training loss. This suggests that the model may have encountered some difficulty in generalizing its predictions to unseen data.

Model 3.1 - Results



We tried the same architecture of the previous model but increasing the number of epochs to 150 and using images of size 224x224.

The Accuracy on Validation Set is of **76,41%.**

Conclusions

- Aimed to improve the accuracy of the initial model, which had an accuracy of **54%.** Various techniques were employed in order to enhance the model's performance.
- However, the second model, despite these modifications, achieved an accuracy of **53%,** which was lower than the initial model.

- Nevertheless, with the development of a third model, implementing a deeper neural network architecture, allowing the model to learn more complex data representations. This deeper and more complex architecture led to a significant improvement, with the third model achieving an accuracy of **70%.**

Conclusions

- In conclusion, the project demonstrated the importance of experimenting with different techniques and architectures to enhance models.

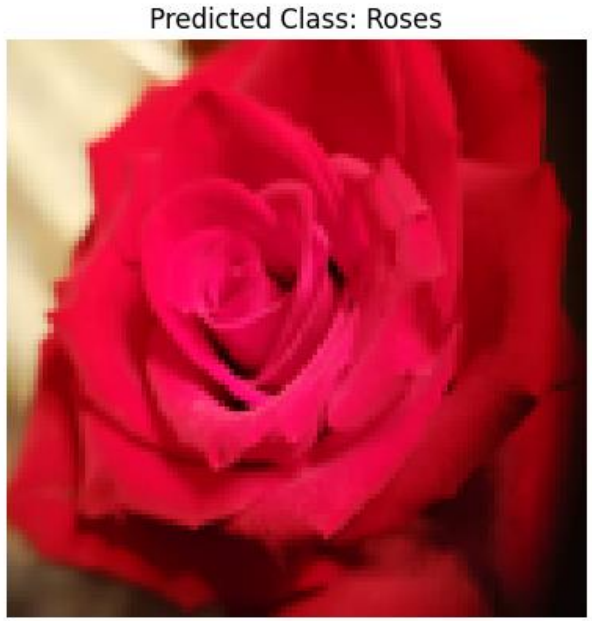
- The 3.1 version of the model, with its **76%** accuracy, represents a successful outcome in enhancing the performance of the initial model, highlighting the value of deeper neural networks and deep learning techniques.

Testing

Obviously I were very curious to test the model on roses in the garden.

So we took a picture of a rose and this is the result:

Picture taken with iPhone 13 Pro



Thanks for Your Attention