2. Flower Species Identification Using Image Recognition

2.1. Introduction

In this work, a Convolution Neural Network (CNN) has been developed to recognize the species of flowers from the images. CNN model has been trained using flowers dataset downloaded from the TensorFlow website. Dataset contain 3670 RGB images of the following flower species:

- Daisy
- Dandelion
- Roses
- Sunflowers
- Tulips

2.2. Convolutional Neural Network (CNN)

CNN is considered as branch of Artificial Neural Network (ANN) that's basically inspired by biological nervous system in which several neurons work in collaboration to crack a problem. A typical CNN system is built up of four layers consisting of a convolutional layer that extract features from input data using feature map, a pooling layer that decreases the size of feature map to eliminate noise, a flattening layer that transform 2D features to 1D vector and a fully connected layers that combine features extracted from previous layers and produces prediction (FatihahSahidan, 2019).

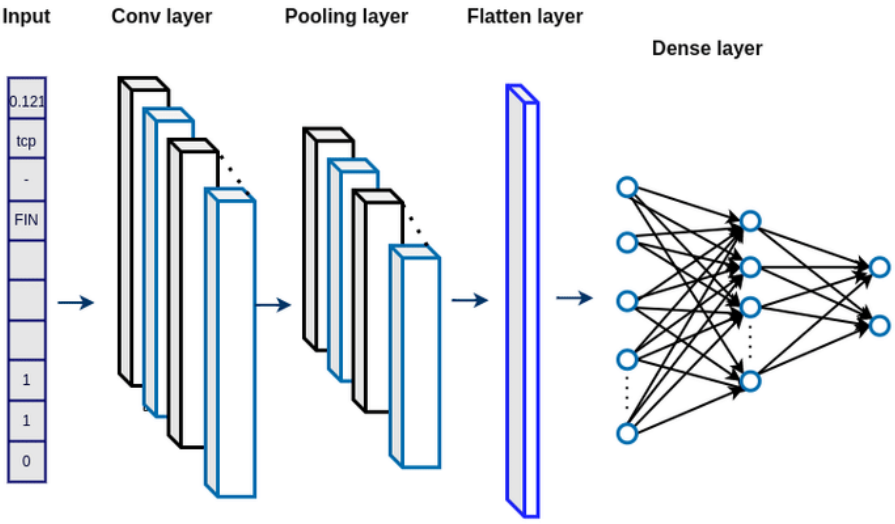


Figure 8: CNN Typical Layers (Meliboyev, n.d.)

2.3. Architecture of CNN Model

2.3.1. Convolutional Layer

Conv #	Filters	Kernel Size	Activation Function
Conv1	32	3 x 3	ReLU
Conv2	64	3 x 3	ReLU
Conv3	128	3 x 3	ReLU

2x2 Max Pooling Layer is used after each convolutional layer to reduce the spatial dimensions for computation ease.

2.3.2. Flatten Layer

To pass the extracted features into the fully connected dense layer, flatten layer is used that convert 3D feature maps to 1D vector.

2.3.3. Fully Connected Dense Layer

Layer	Neurons	Activation Function
First Dense Layer	64	ReLU
Output Layer	5	Softmax

Dropout rate of 20% is applied after first dense layer to prevent overfitting. Also 5 neurons in output layer is referred to the type of flower probability over five flower classes.

2.3.4. Compilation Stage

- Optimizer used was adam with default learning rate
- Batch size of 32 was used
- 20 epochs were used
- To guide the training process, a multi classification loss cross entropy was used

2.4. CNN Model Architecture Operation & Parameter Justification

Input shape of RGB pictures is (128, 128, 3) where 128 x 128 correspond to the pixel size and 3 correspond to three-color channels i-e Red, Green Blue.

CONV1: First convolutional layer applied 32 filters of 3 x 3 matrix to the input image. To perform convolutional operation each filter will slides over the input image capturing features such as edges or corners. ReLU introduces non-linearity allowing model to capture complex patterns and prevent vanishing gradient issue. This transforms the shape from (128, 128, 3) to (126, 126, 32).

POOL1: To reduce the dimensionality of feature map and lower computational load, maximum pooling is used. A 2 x 2 Matrix slides over transformed shaped and select maximum value from its 2 x 2 block. This layer reduces the parameters to (63, 63, 32).

CONV2: Second convolutional layer applied 64 filters of 3 x 3 matrix with ReLU activation function to detect further detailed features like textures. This layer reduces the shape to (61,61,64).

POOL2: Similar to POOL1, POOL2 reduces the feature map size by half i-e (30, 30, 64) by sliding 2 x 2 matrix and selecting maximum value in block.

CONV3: Third convolutional layer uses 128 filters of same matrix size and activation function. This layer operates on even more detailed features like color of flower. This layer transforms the shape to (28, 28, 128).

POOL3: This layer performs final reduction of feature map dimension bringing them to (14, 14, 128).

FLATTEN: Finally, 3D feature map of shape (14, 14, 128) is converted to 1D vector of size 25088 for feeding into the fully connected layer

FC1: Dense layer contains 64 neurons. This layer reduces the dimensionality from 25088 to 64, concentrating features into more manageable size. ReLU activation function is used to maintain non-linearity.

DROP1: To prevent the dependance on particular neurons and prevent overfitting, random 20% neurons are set to zero.

OUTPUT: Final layer with 5 neurons, corresponding to the 5 different flower species. To convert output into probability, softmax activation function is used.

2.5. Initial CNN Model Result

Above discussed architecture yield Training accuracy of 0.7768 and Validation accuracy of 0.7274. No result of overfitting is found as graph of both training and validation accuracy have minimum distance.

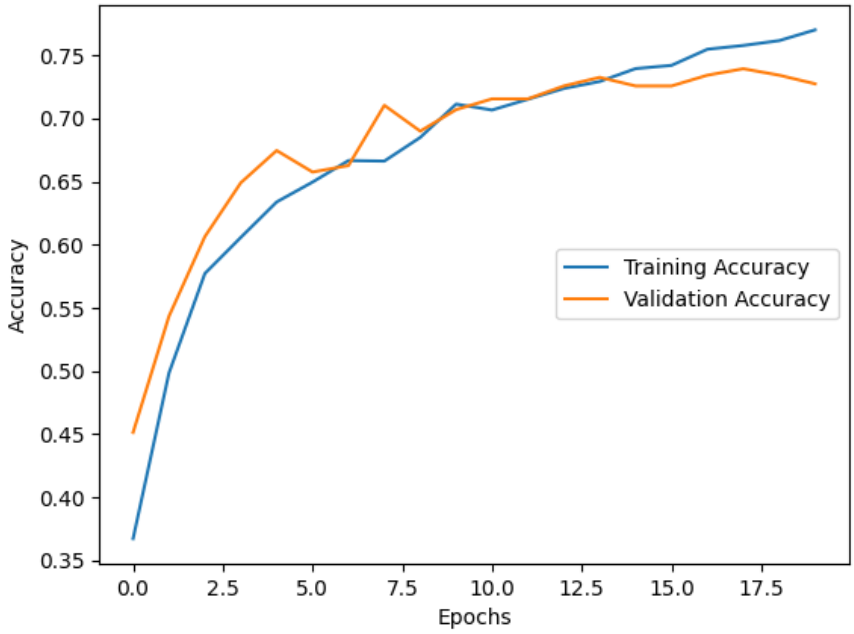


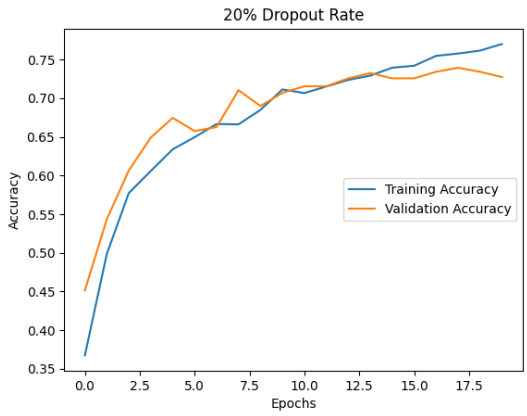
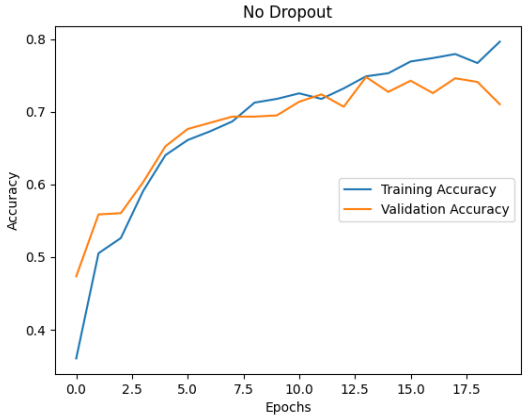
Figure 9: Initial Model Accuracy Graph

2.6. Regularization Techniques & their Impact

For initial start only one regularization technique is used i-e dropout rate.

Dropout: Dropout rate of 20% is used. This randomly ignored half of the neurons during training process so that model doesn't rely on specific neurons and learned all generalized features.

Effect of Dropout on Accuracy

Dropout 20%	No Dropout
Training Accuracy = 0.7768 Validation Accuracy = 0.7274	Training Accuracy = 0.8034 Validation Accuracy = 0.7104
	

When no dropout was used, slight overfitting occur in the model as training accuracy increases from 0.7768 to 0.8034 and validation accuracy decreases from 0.7274 to 0.7104. Model learned detailed patters from training data causes better accuracy for training data but when validation set was used, model struggled as depicted by slight decrease in validation accuracy.

2.6. Hyperparameter Tuning

Hyperparameter tunning was done using some base set parameters due to less computational resources. Throughout tunning same number of convolutional layers and their filters, fully connected layers, activation function, epochs, optimizer and loss measurement metrics were used.

2.6.1. Varying units in Fully Connected Layer

Neurons in fully connected dense layer was altered starting from 64 neurons and then doubled up to 512 neurons. Improvement in validation accuracy was observed. After 256 neurons validation accuracy slightly decreases. Neurons with highest validation accuracy was chosen for further hyperparameter tunning.

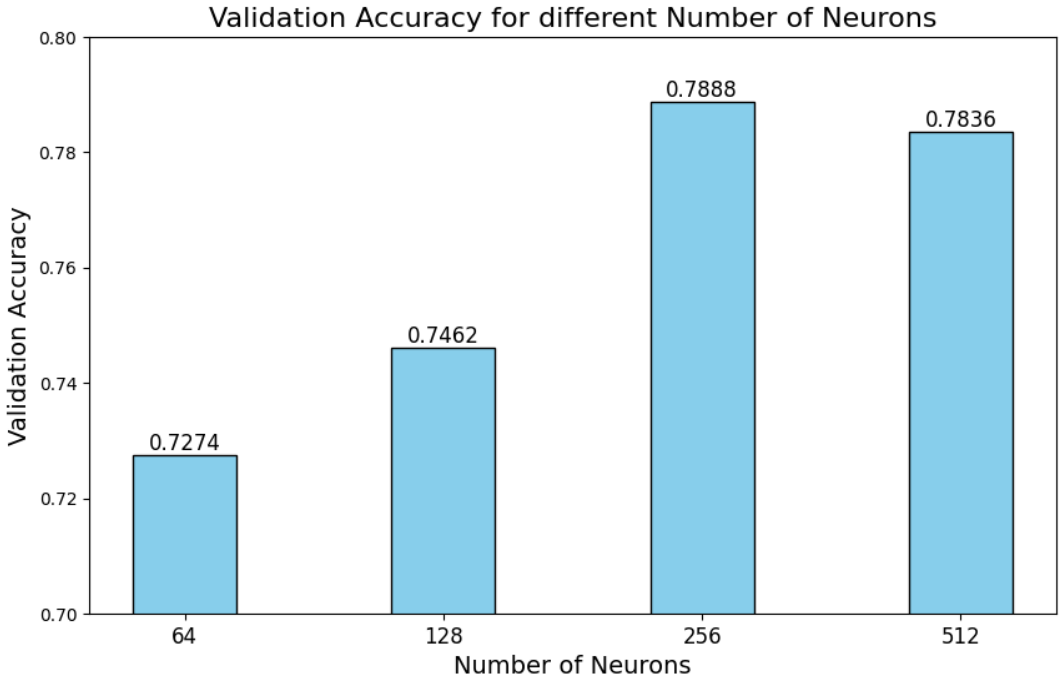


Figure 10: Hyperparameter Tunning (Neurons)

2.6.2. Varying Learning Rate

Learning rate of 0.0001 was checked for same number of epochs. Due to low learning rate convergence speed was low, at the end of 20 epochs validation accuracy of 0.7138 was obtained. Default learning rate was then used for further hyperparameter tunning.

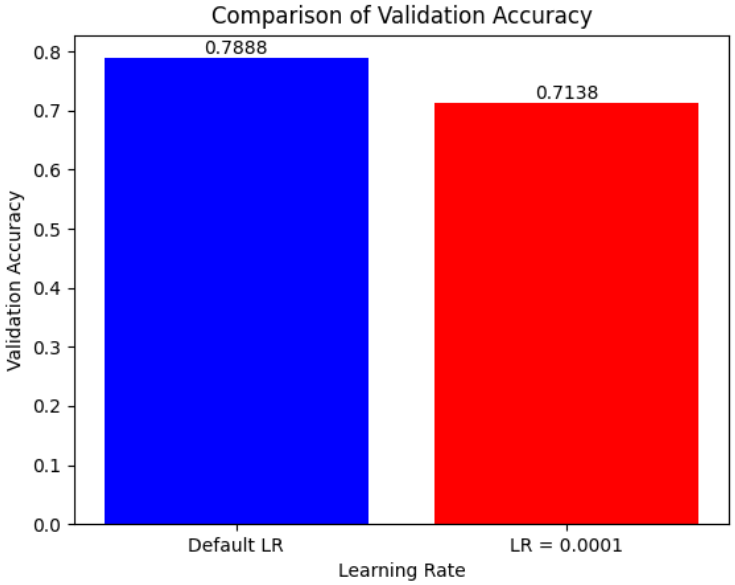


Figure 11: Hyperparameter Tunning (Learning Rate)

2.6.3. Batch Size

Batch size was altered from 32 to 128 but decrease in validation accuracy was observed. With large batch size, sometimes model generalizes poorly. Also, throughout iteration, same learning rate was used which might affect the training dynamics (Keskar, 2017).

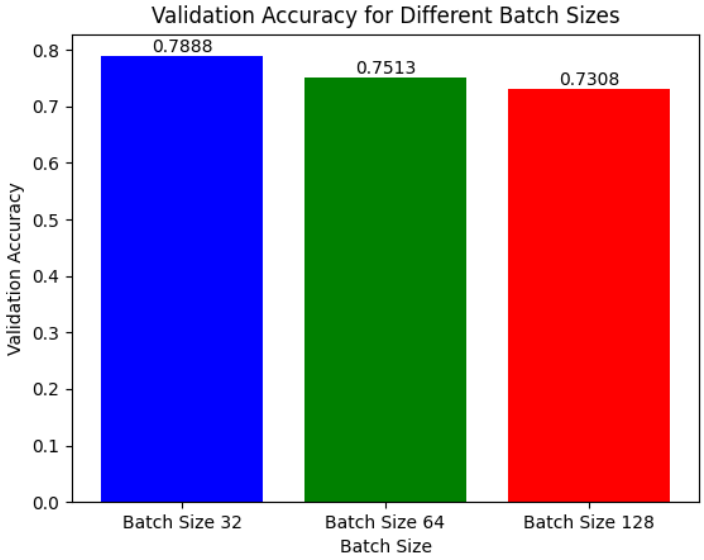


Figure 12: Hyperparameter Tunning (Batch Size)

2.6.4. Kernel size

No improvement in validation accuracy was found when Kernel size changed from 3 x 3 to 5 x 5.

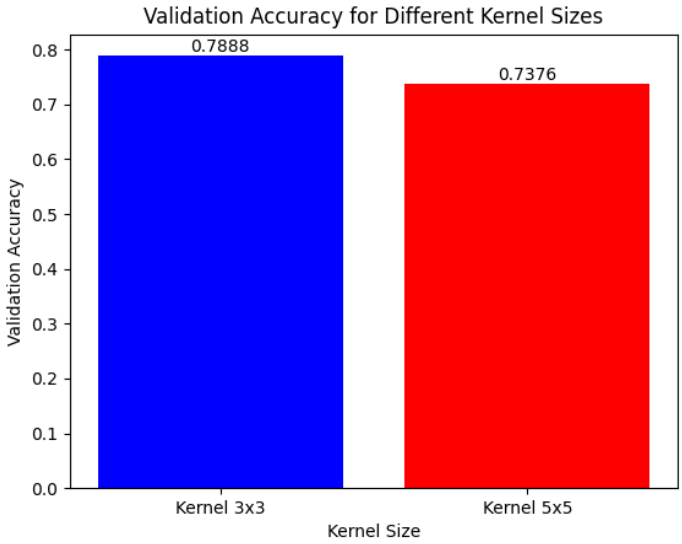


Figure 13: Hyperparameter Tunning (Kernel Size)

2.6.5. Batch Normalization

Batch normalization with momentum of 0.9 was used after every max pooling in convolutional layer. Validation accuracy doesn't improve for same number iterations.

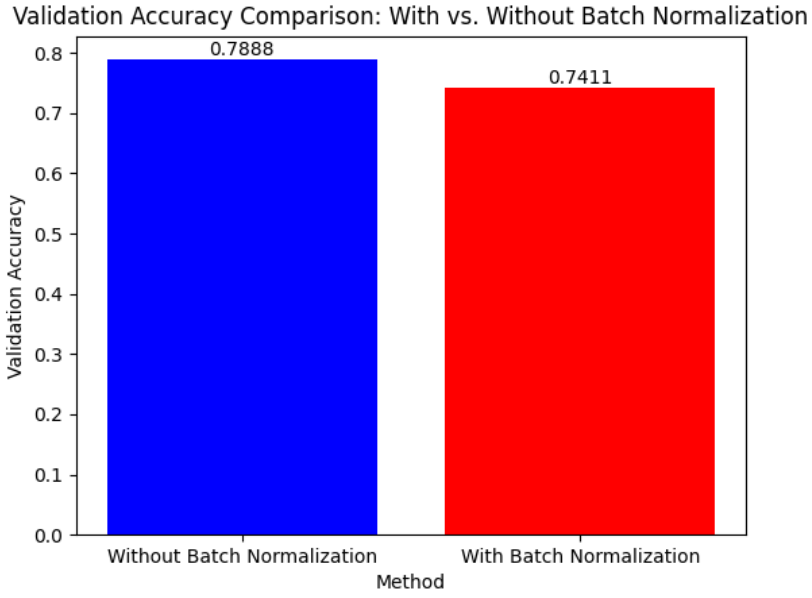


Figure 14: Hyperparameter Tunning (Batch Normalization)

2.6.6. Stride & Padding

Stride of 2 for filter to move 2 pixels at a time in both horizontal and vertical direction and padding = “same” to ensure same spatial dimension of output feature as the input feature map was used. In this case, still no improve in validation accuracy was found.

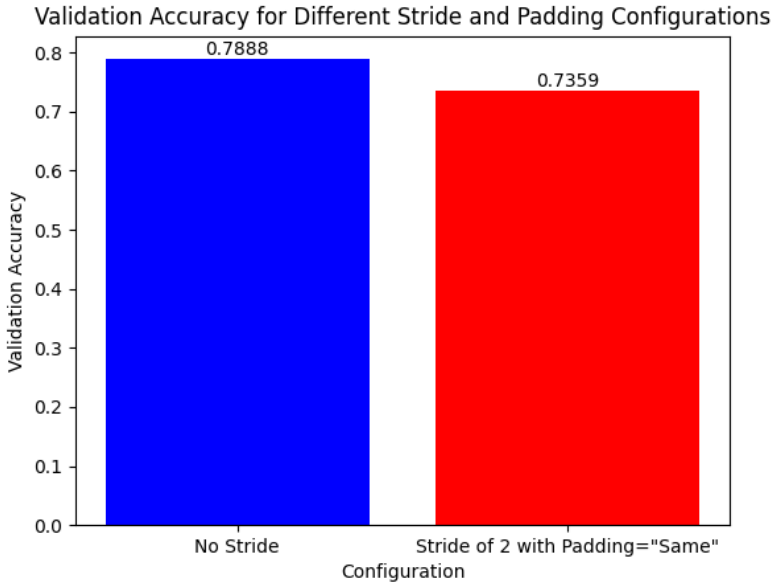


Figure 15: Hyperparameter Tunning (Stride & Padding)

2.6.7. Effect of Hyperparameter on Model

Number of Neurons in fully connected layer, learning rate, kernel size and strides have the strongest effect on the performance of model as variation in these parameters drastically varies the validation accuracy as depicted in figures above.

2.7. Final Model Architecture

2.7.1. Convolutional Layer

Conv #	Filters	Kernel Size	Activation Function
Conv1	32	3 x 3	ReLU
Conv2	64	3 x 3	ReLU
Conv3	128	3 x 3	ReLU

2x2 Max Pooling Layer after each convolutional layer is used to reduce the spatial dimensions for computation ease.

2.7.2. Flatten Layer

To pass the extracted features into the fully connected dense layer, flatten layer is used that convert 3D feature maps to 1D vector.

2.7.3. Fully Connected Dense Layer

Layer	Neurons	Activation Function
First Dense Layer	256	ReLU
Output Layer	5	Softmax

Dropout rate of 20% is applied after first dense layer to prevent overfitting. Also 5 neurons in output layer is referred to the type of flower probability over five flower classes.

2.7.4. Compilation Stage

- Optimizer used was Adam with default learning rate
- Batch size of 32 was used
- 20 epochs were used
- To guide the training process, a multi classification loss cross entropy was used

2.7.5. Results of Final Model

Both training and validation accuracy increases rapidly during the initial epochs with both curves closely tracking each other. Slight dip in the middle for validation accuracy was observed which recover and progress further. Training accuracy continue to increase and ended at 0.82 while validation accuracy also increases slightly but remain lower than training accuracy, reaching 0.78.

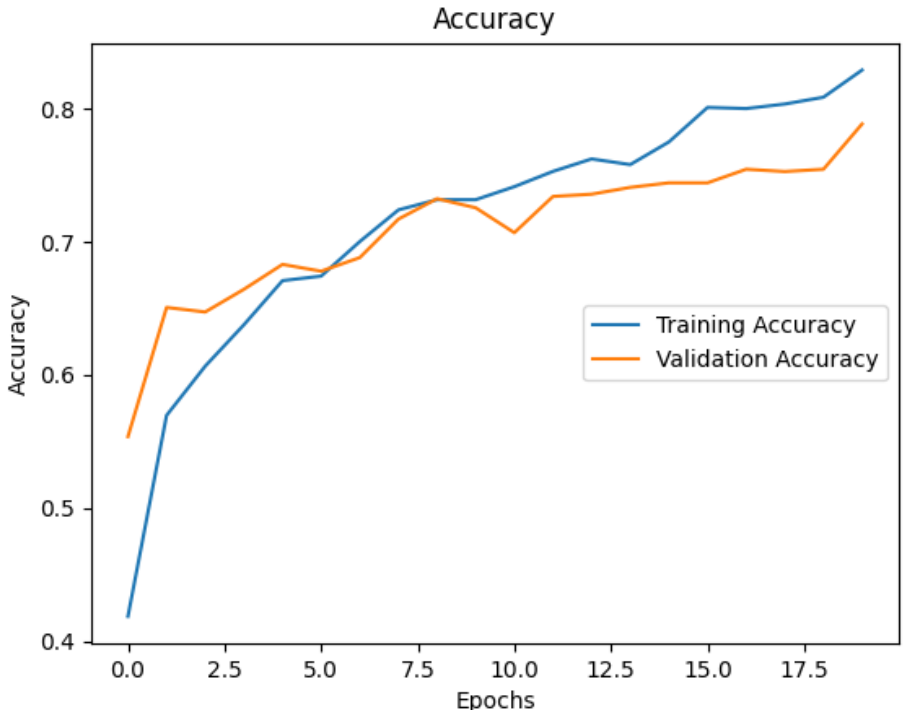


Figure 16: Final Model Accuracy Graph

2.8. Overfitting

In the middle, there's a slight hike where loss drastically increases. This hike suggest that model might starting to overfit the data and that model is beginning to memorize training examples rather than learning general patterns. The gap between training and validation accuracy in later stages also suggest overfitting but relatively mild. Overall, there's small degree of overfitting as evident by the loss graph.

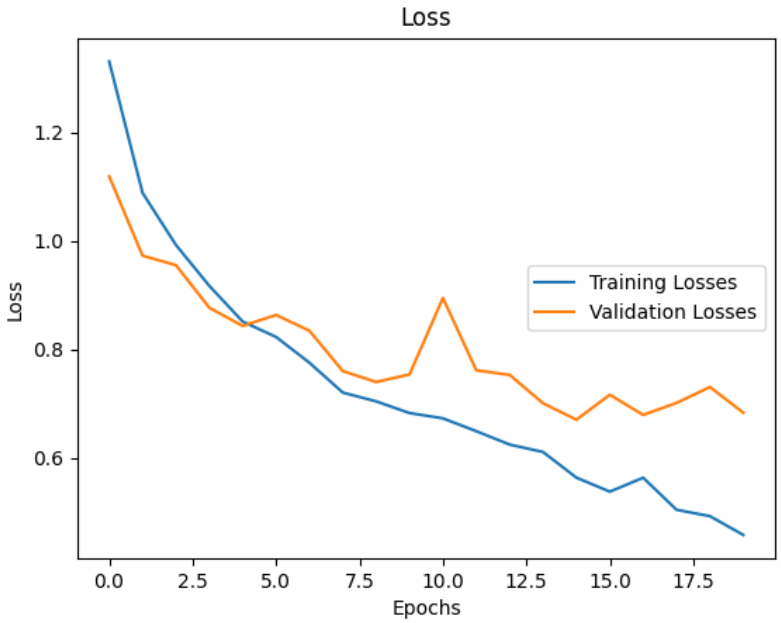


Figure 17: Final Model Loss Graph