ArIES

Image Colorization Project SpecDoc

Team members:

Ashish Singh

Jinal Shah

Tanmay Gupta

Mentors:

Alakh Verma

Harsh Surya

Image Colorization Project

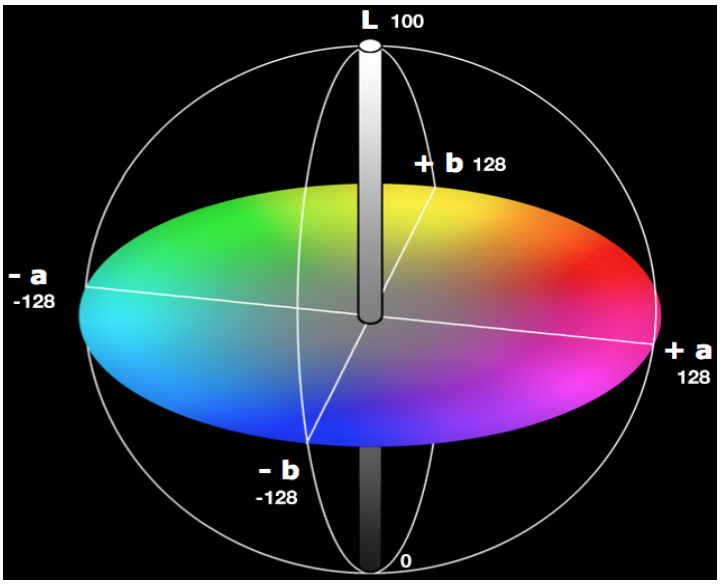
Overview:

Deep Learning is an upcoming subset of machine learning which makes use of artificial neural networks. These are inspired by the human brain and its immense structure and function. Here we make use of it to colorize black and white images.

Methodology:

Taking input as a black and white image, this model tries to produce a colorized image. We are using TensorFlow and Keras API. Our model is trained using Google Colab.

1. To train the network, we start off with image datasets made up of colourful pictures. The datasets used here are CIFAR10 and Landscape Dataset. Then we convert all images from the RGB color space to the *Lab color space*. Just like RGB color space, Lab is an alternate color space in which the three channels represent:-
    - The *L channel* represents light intensity only.
    - The *a channel* encodes green-red color.
    - The *b channel* encodes blue-yellow color.



2. Since the *L channel* can encode intensity, we use it as the input in our network in the grayscale format. We put all ‘L’ values in an array called ‘X’ .

3. We then use our model to let the network predict the respective *a and b channel* values. The ‘ab’ values are divided by 128 (feature scaling) so as to reduce values from [-127, 128] to (-1, 1] to make the learning process quicker and more efficient. These are now put in an array called ’Y’.

```
X =[]
Y =[]
for img in train[0]:
  try:
      lab = rgb2lab(img)
      X.append(lab[:,:,0])
      Y.append(lab[:,:,1:] / 128 )
  except:
     print('error')
X = np.array(X)
Y = np.array(Y)
X = X.reshape(X.shape+(1,))
print(X.shape)
print(Y.shape)
```

```
(2388, 64, 64, 1)
(2388, 64, 64, 2)
```

4. Our model is a Convolutional Neural Network (CNN) which trains on ‘X’ as feature inputs and ‘Y’ as target values.The model consists of a number of convolutional layers. There are no pooling layers in our network. Changes in resolution are achieved using striding and upsampling layers. The kernels are sized 3x3.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 64, 64, 32)        320
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 64, 64, 64)        18496
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 32, 32, 64)        36928
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
dropout_3 (Dropout)          (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 16, 16, 256)       295168
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 16, 16, 128)       295040
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 16, 16, 128)       147584
_________________________________________________________________
up_sampling2d_6 (UpSampling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 32, 32, 64)        73792
_________________________________________________________________
up_sampling2d_7 (UpSampling2 (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 64, 64, 32)        18464
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 64, 64, 16)        4624
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 64, 64, 2)         290
=================================================================
Total params: 1,112,146
Trainable params: 1,112,146
Non-trainable params: 0
_________________________________________________________________
```

5. It uses the *ADAM* optimizer with a *learning rate* of *0.0003*. The loss function used is *mean squared error*. It also includes a Dropout layer to prevent overfitting.

```
opt = tf.keras.optimizers.Adam(learning_rate=0.0003)
model.compile( loss='mse' , metrics=['accuracy'])
```

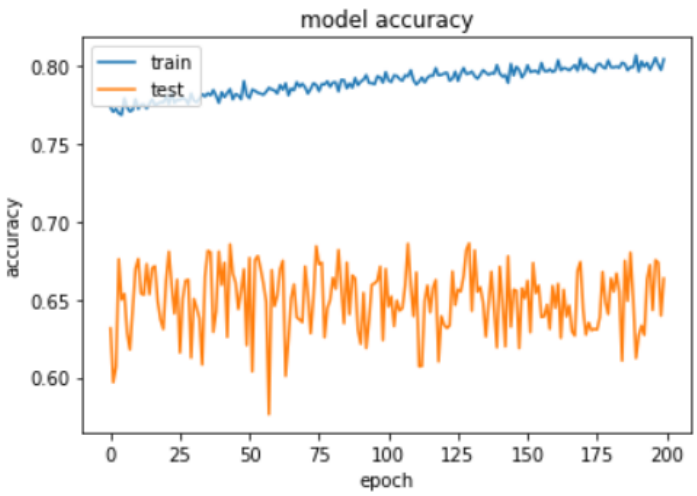
6. 10% of the dataset is used as the validation set. Our network trains for multiple epochs.

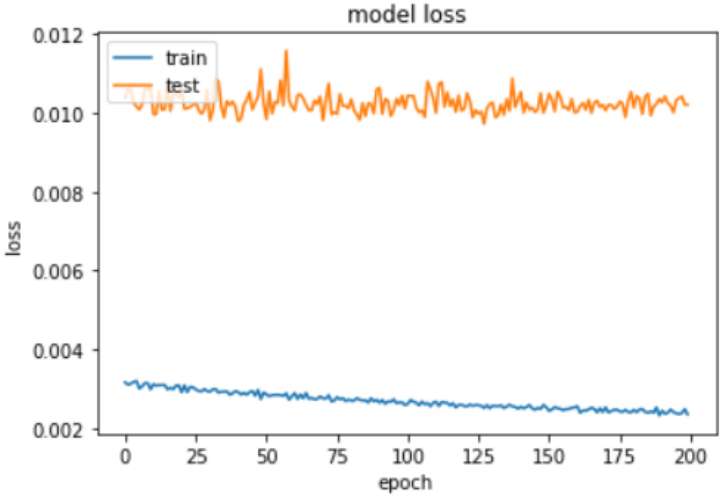
```
model.fit(X,Y,validation_split=0.1, epochs=200, batch_size=100)
```

Finally, we define a function which an image as an input and outputs its original, grayscale and predicted colourized form as predicted by our model.

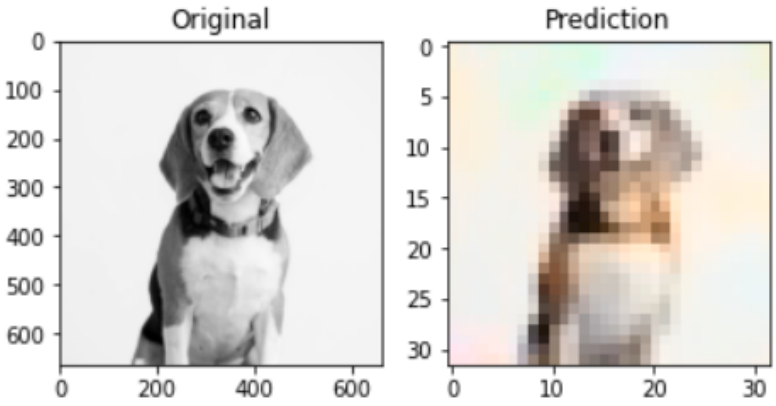
Various Models:

1. Dog:

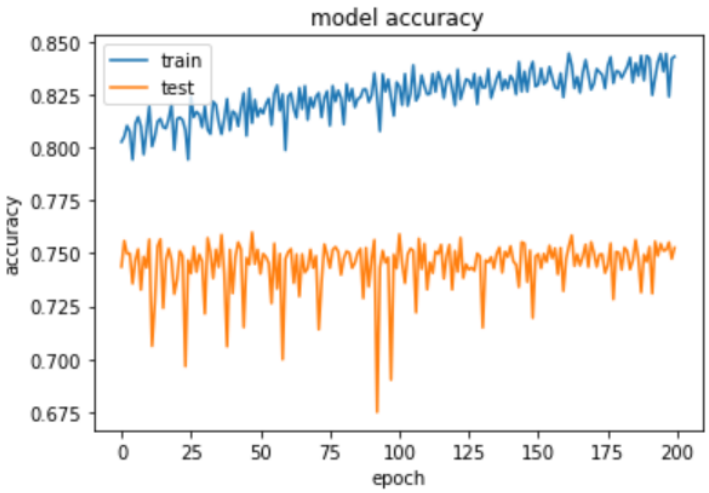


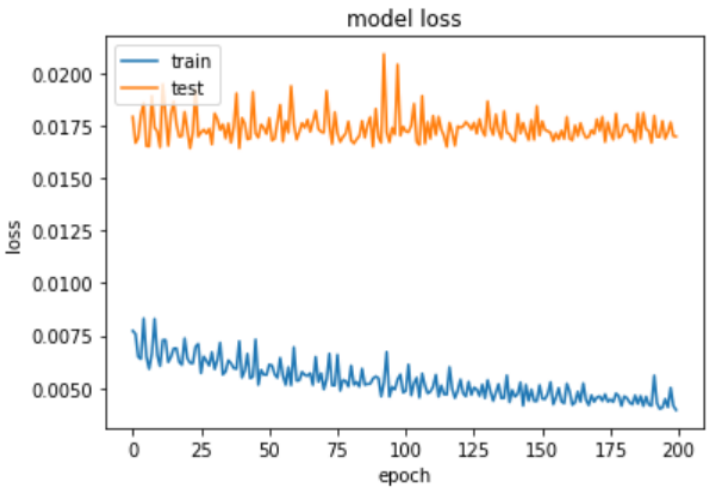



```
predict('/content/Imaghes/imageonline-co-blackandwhiteimage (1).jpg')
```



2. Flower

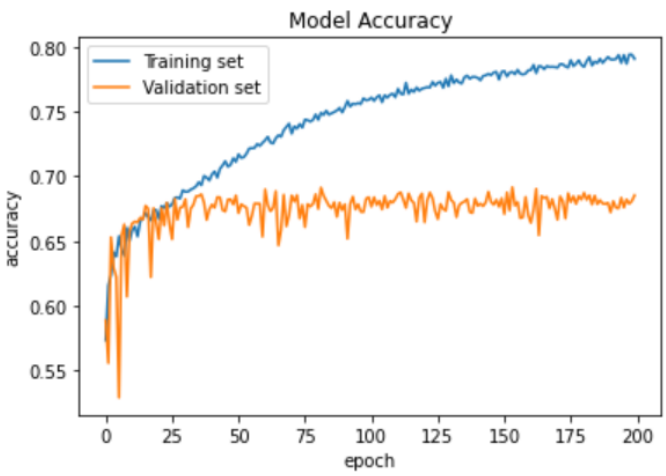


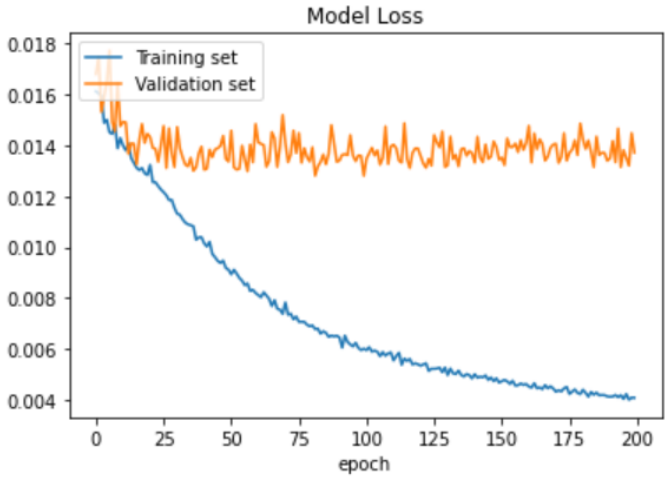



```
predict('/content/fool/test/daisy.jpg')
```

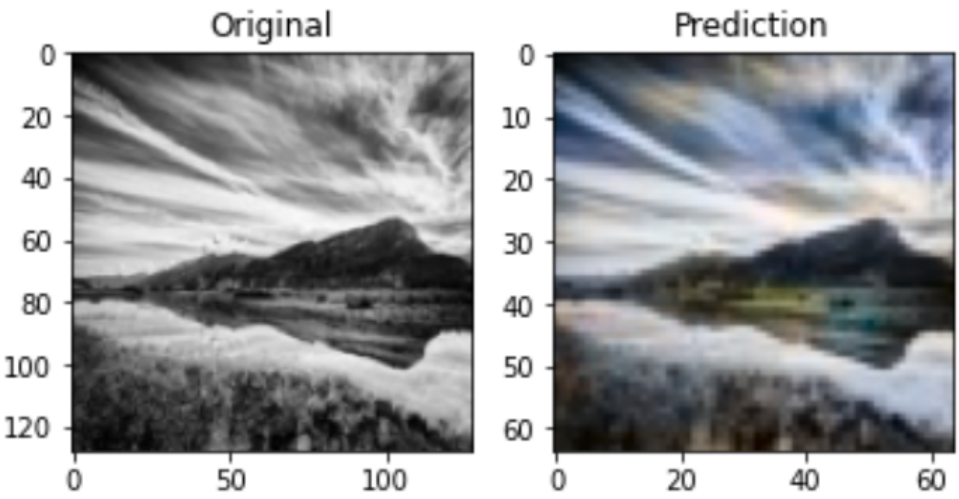
3. Landscape:

Trained on Dataset of 3,000 landscape images sized 64x64.



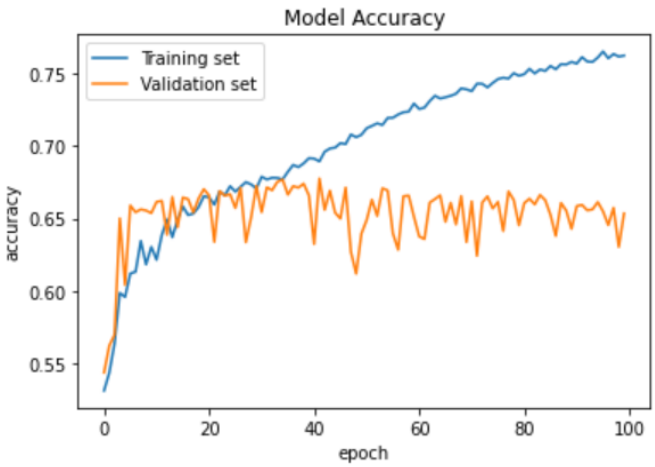


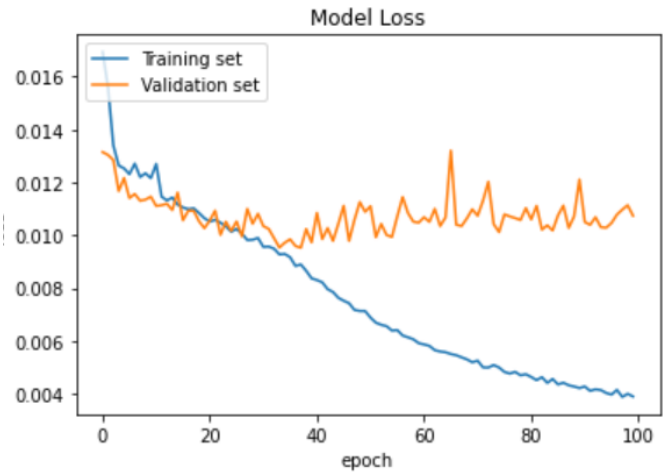

```
predict('/content/Test-images/download.jpg')
```



4. Landscape 2.0

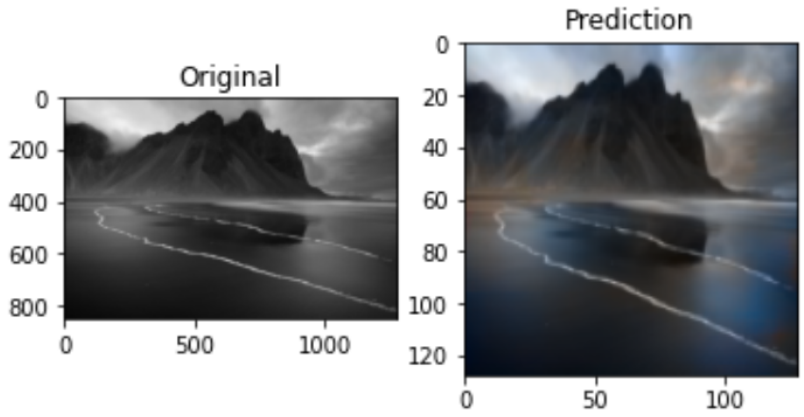
Trained with a Dataset of 8,000 landscape images of size 128x128.



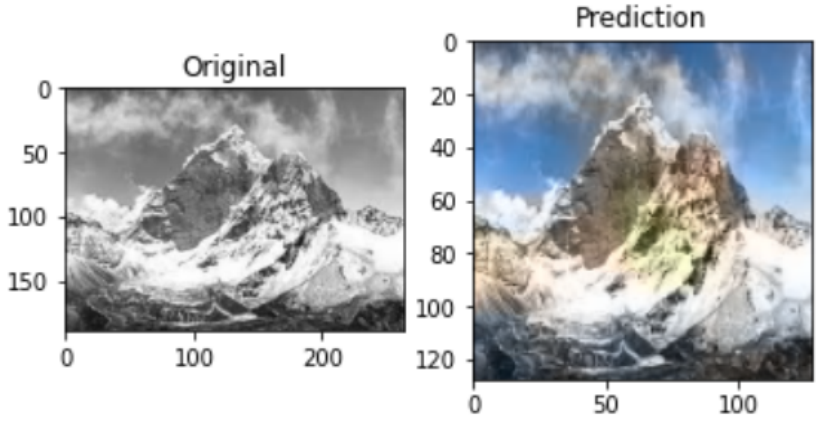


Results:-

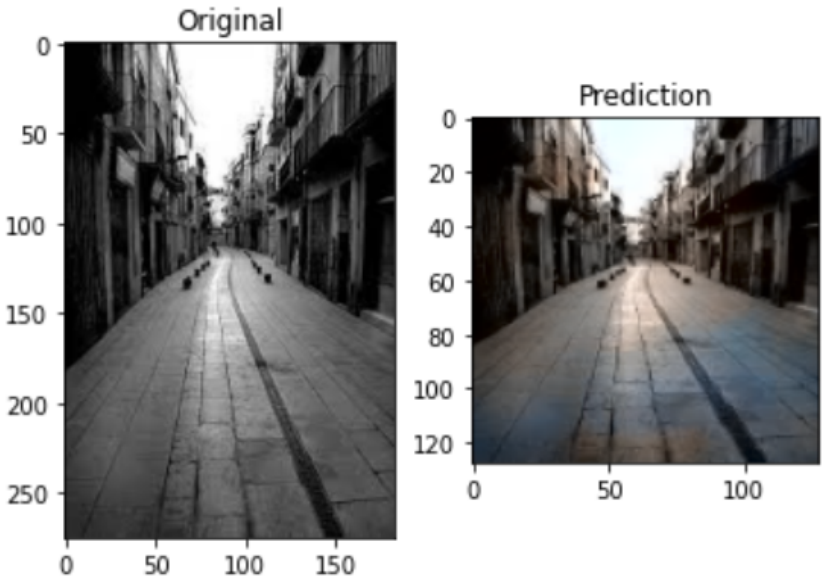
```
predict('/content/Test_Image/Test_Image1.jpeg')
```



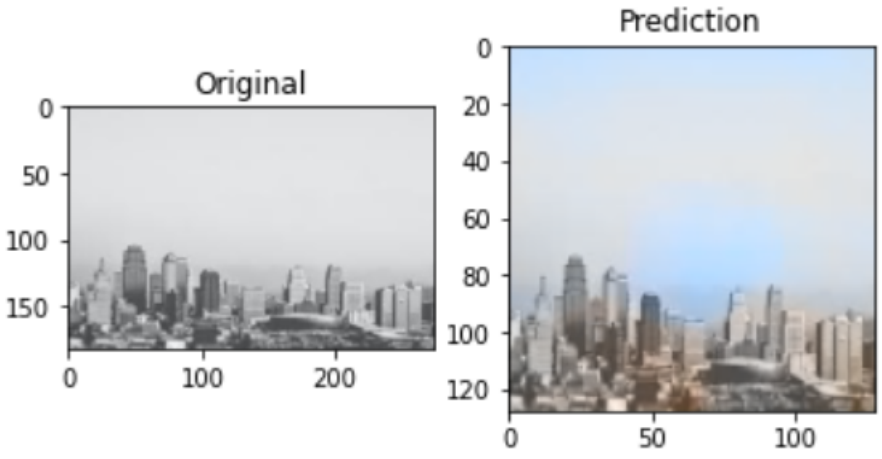

```
[9]  predict('/content/Test_Images/Test_image4.jpg')
```



```
predict('/content/Test_Images/Test_image3.jpg')
```

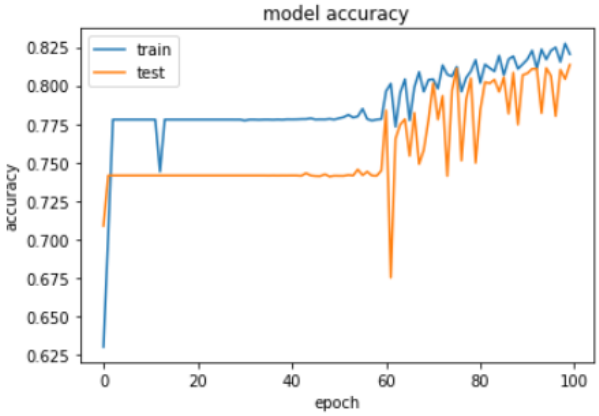


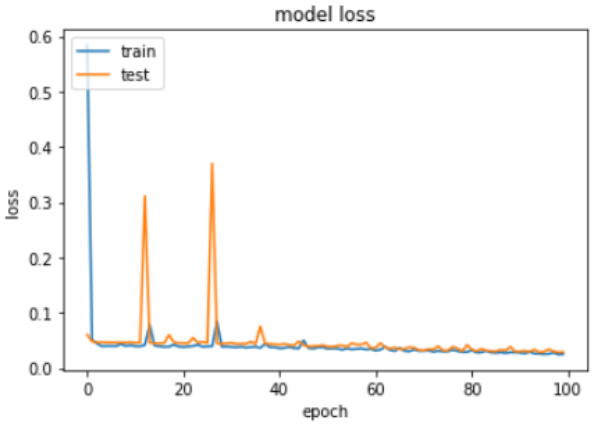

```
predict('/content/Test_Images/Test_Image6.jpg')
```



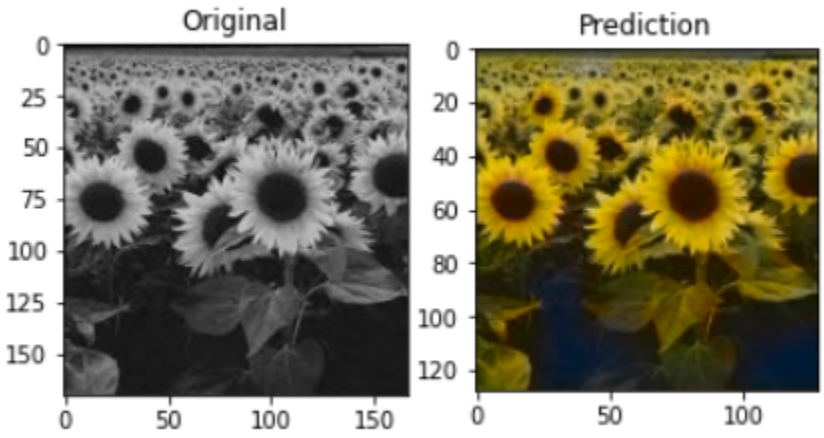
5. **Flowers 2.0**

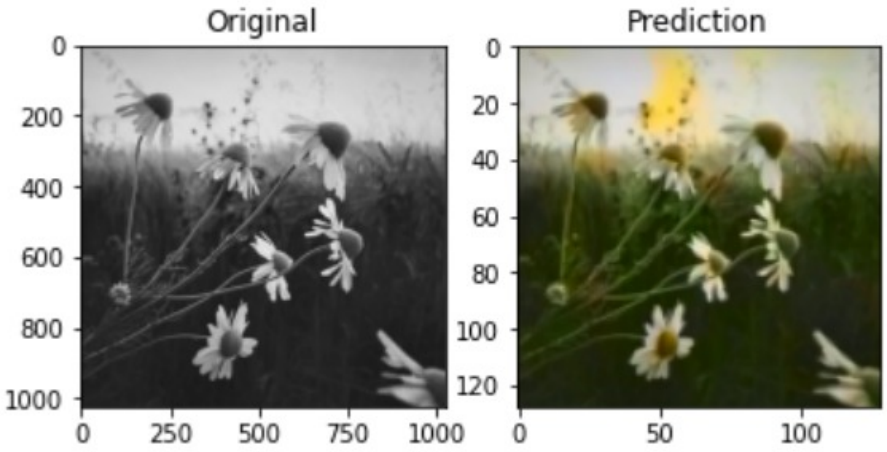
Trained on 3 different datasets to give colourized images of 128x128 pixels.

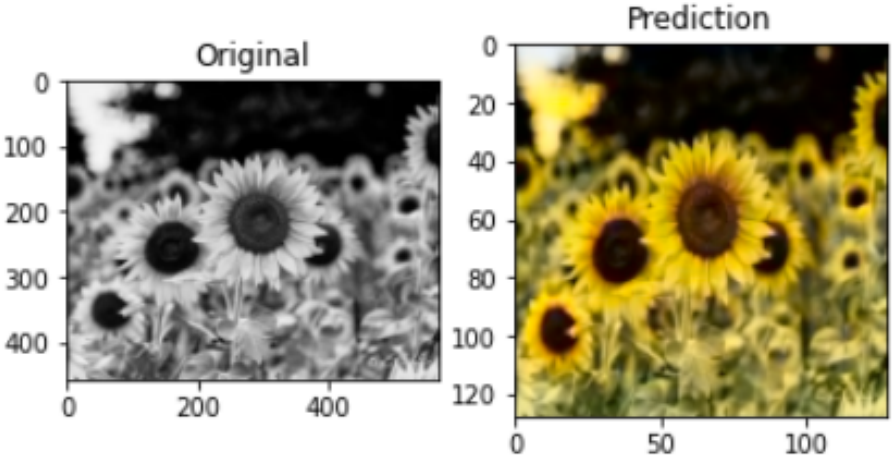


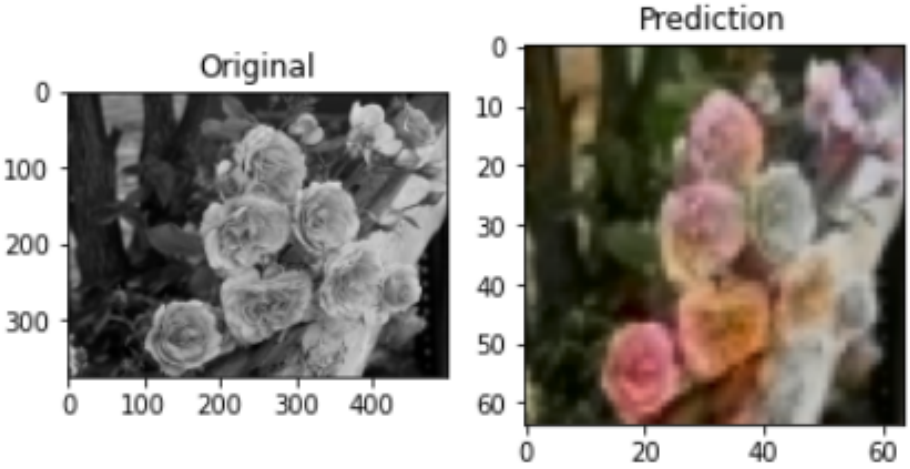


Results:-






Original
Prediction
0
100
200
300
400
0
200
400
0
20
40
60
80
100
120
0
50
100

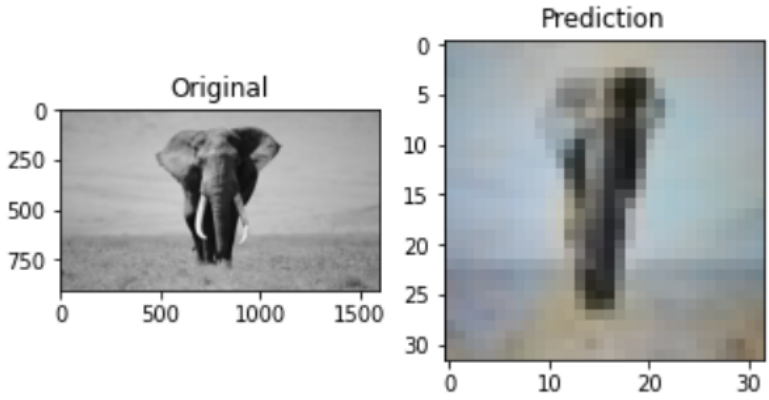

Original
Prediction
0
100
200
300
0
100
200
300
400
0
10
20
30
40
50
60
0
20
40
60

6. **Multiclass**

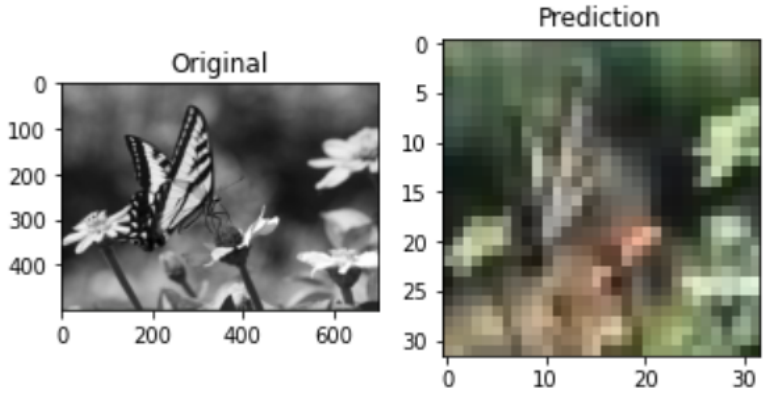
Trained with 10,000 randomly selected images from CIFAR 100 dataset. CIFAR 100 is a dataset of 50,000 images of size 32x32 belonging to 100 different object classes.

Results:-

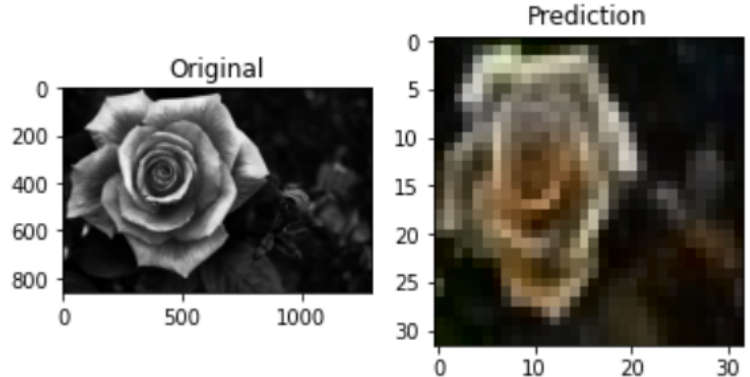
```
[ ]  predict('/content/Test-images/Test-Image1.jpg')
```




```
predict('/content/Test-images/Test-Image2.jpg')
```




```
predict('/content/Test-images/Test-Image3.jpg')
```



References:

1. Colorful Image Colorization Research Paper by *Richard Zhang, Phillip Isola, Alexei A. Efros*: https://arxiv.org/abs/1603.08511

2. Datasets used:
    a. https://www.kaggle.com/arnaud58/landscape-pictures
    b. https://www.kaggle.com/huseynguliyev/landscape-classification
    c. https://www.cs.toronto.edu/~kriz/cifar.html

3. https://www.pyimagesearch.com/2019/02/25/black-and-white-image-colorization-with-opencv-and-deep-learning/

4. https://www.youtube.com/watch?v=VyWAvY2CF9c