International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 10, Issue 10, October-2022, Impact Factor: 7.429, Available online at: www.ijaresm.com

Image Colorization using AI

Vangala Bhanu Prakash[1], Abdul Mannan Khan[2], Sagar Sujith Somepalli[3],
Rakesh Chigurupati[4], Pratham Shah[5], Shubham Nandlal Vishwakarma[6]

---*****************--

ABSTRACT

Colourization is a PC helped procedure of adding shading to a monochrome picture or film. The procedure includes typically fragmenting pictures into areas and followingthese districts crosswise over picture successions. Neither of these undertakings canbe performed dependably by and by; thus, colourization requires extensive client mediation and stays a monotonous, tedious, and costly assignment.

Colourization is a term presented by Wilson Markle in 1970 to portray the PC helped process he created for including shading.Colourizing highly contrasting movies is an old thought going back to 1902. For a considerable length of time, numerous filmmakers restricted colourizing their high contrast motion pictures and thought of it as vandalism of their craft. Today it is acknowledged as an upgrade to the artistic expression.

The innovation itself has moved from meticulous hand colourization to the present to a great extent, robotized strategy. In India, the film Mughal-e-Azam, a blockbuster discharged in 1960 was remastered in shading in 2004. Individuals from different ages swarmed the performance centres to see it in shading, and the motion picture was an immense hit for the subsequent time!

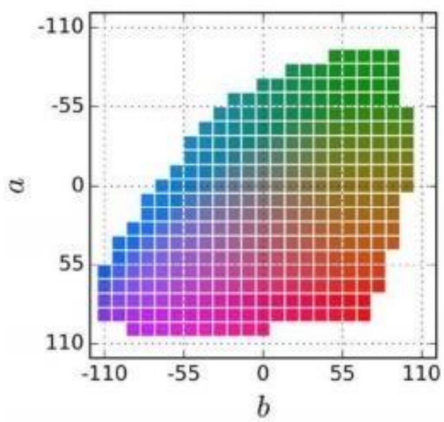
Keywords: AI, Deep Learning, Convolution Neural Network, Image Processing, Generative Adversarial Network.

INTRODUCTION

Let us first define the colourization problem in terms of the CIE Lab colour space. Like the RGB colour space, it is a 3-channel colour space, but unlike the RGB colour space, colour information is encoded only in the a (green-red component) and b (blue-yellow component) channels. The L (lightness) channel encodes intensity information only.
The grayscale image we want to colour can be thought as the L-channel of the image in the Lab colour space and our objective to find the a and b components.

The Lab image so obtained can be transformed to the RGB colour space using standard colour space transforms.
To simplify calculations, the ab space of the Lab colour space is quantized into 313 bins, as shown in Figure below. Instead of finding the a and b values forevery pixel, because of this quantization, we simply need to find a bin number between 0 and 312.

However, another way of thinking about the problem is that we already have the Lchannel that takes values from 0 to 255, and we need to find the ab channel that takes values between 0 to 312. So, the colour prediction task is now turned into a multinomial classification problem where for every grey pixel, there are 313 classes to choose from.



The objective of the project:

This project is a basic auto-encoder for image colourization. We have usedfeature extraction and fused it with a layer that is obtained after downsampling the input layer. We have used a convolutional neural network to up sample and predict the colour of the input image.

We have researched and surveyed various methods that have been applied for image colourization using various technologies on artificial intelligence. As part of this project, we have developed an understanding of Convolution Neural Network,Image Colorization and a bit of Image Processing.

We have utilized a dataset from Kaggle, which has pairs of pictures- B&W and colour.

The purpose of this project is not just the fulfilment of the J component for this course; instead, it is to learn and experiment with the tools that are available to usin the field of AI.

LITERATURE REVIEW SUMMARY TABLE

Authors and Year(Reference)	***Title (Study)***	***Concept / Theoretical model/ Framework***	***Methodology used/ Implementation***	***Dataset details/ Analysis***	***RelevantFinding***	***Limitations/ Future Research/Gaps identified***
You Zhou, Jeff Hwang 2016	*Image Colorization with Deep Convolutional Neural Networks*	*Deep Convolution Neural Network*	*Build a learning pipeline that comprisesa neural network and an image pre-processing front-end.*	*The MIT CVCL Urban and Natural Scene Categories dataset*	*It uses the RGB to CIELUV colorspace to train its model*	*It will take a very large time to train themodel and it's practically impossible in a laptop. It producesunder coloredProjects*
Mark J. Huiskes, Michael S. Lew 2008	*The MIR Flickr Retrieval Evaluation*	*Image Retrieval through API*	*Create a redistributable image set from a social networking site.*	*Flickr retrieved images*	*Uses metadata and tags found to group the images*	*It is not reliable to train a model based on this small dataset obtained.*
Anat Levin, Dani Lischinski, Yair Weiss	*Colorization using Optimization*	*Image Colourization without precise segmentation*	*Cost effective colourization technique with minimal user input*	*ACM SIGGRAPH 2004*	*Reducing manual inputfor colourization process*	*Doesn't distinguish betweenhue and saturation, optimization can be improved.*

Authors and Year (Reference)	***Title (Study)***	***Concept / Theoretical model/ Framework***	***Methodology used/ Implementation***	***Dataset details / Analysis***	***RelevantFinding***	***Limitations/ Future Research/Gaps identified***
Bei Tang, Guillermo Sapiro, Vicent Caselles	*Color Image Enhancement via Chromaticity Diffusion*	*Image Enhancement by color image denoising*	*Seperating color data into brightness and chromaticity*	*Internet obtained images*	*Coupling brightness and chromaticity helps in segmentation*	*To find the optimal coupling is challenging*
Tomihisa Welsh, Michael Ashikhmin, Klaus Mueller	*Transferring Color to Greyscale Images*	*Transfering chromatic information by matching luminance and texture information*	*Using an example color image to colourize a greyscale image*	*Internet obtained images*	*Helpful in creating color indexable image collections*	*Can be further improved by using more sophisticated measure of texture similarity*

Innovation component in the project

The actual ab colour space is not discrete, it is continuous, and for a CNNmodel to train in such a data would have taken a very long time.

Nevertheless, as we have made it to be divided into 313 spaces, thetraining can be done in a standard laptop itself.

Work was done and implementation

Methodology:
Use dataset for feature extraction.Use epoc to train our CNN

Once trained, use the model for predicting the colourized image.

Hardware and software requirements:
8GB RAM

Minimum Quad-core Intel i5 Kabylake ProcessorInternet Connection (High-Speed Broadband)
M.2 PCIe SSD with 256GB of storage or more

Premium Graphics Processing Card, GTX 1050Ti or higher.Python

Libraries like sklearn, pandas, tensorflow, keras and matplotlib

Dataset used:

a. Where from you are taking your dataset?

We are taking the data set from Kaggle.
Image Colorization (25kX224X224 grayscale and typicalimages)
https://www.kaggle.com/shravankumar9892/image-colorization

b. Is your project based on any other reference project?

Yes, Stanford University, which was trained on an AWS instancerunning on a NVIDIA GRID K520 GPU.

c. How does your project differ from the reference project?

They have used a cieluv colour space, and we are using an ab colourspace, which is easier to train.

Tools used:

1. GPU GTX 1050Ti
2. 8GB RAM
3. Jupiter Notebook
4. Python 3
5. NVIDIA Nsight HUD 2019.4 to enhance the GPU functionality
6. Keras
7. Tensorflow
8. Tensor Board

SCREENSHOT AND DEMO

```
#Importing the Libraries
%matplotlib inline
import matplotlib.pyplot as plt
import cv2
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.inception_resnet_v2 import␣
↪InceptionResNetV2, decode_predictions, preprocess_input
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Conv2D, Conv2DTranspose, Reshape
import tensorflow as tf
```

```
#Importing the images as np array
images_gray = np.load('l/
gray_scale.npy') images_lab =
```

```
#Function to convert the lab images imported above to rgb images
def get_rbg_from_lab(gray_imgs, ab_imgs, n
    = 10): imgs = np.zeros((n, 224, 224,
    3))
    imgs[:, :, :, 0] = gray_imgs[0:n:]
    imgs[:, :, :, 1:] =

    ab_imgs[0:n:] imgs =

    imgs.astype("uint8")

    imgs_ = []
    for i in range(0, n):
        imgs_.append(cv2.cvtColor(imgs[i],

    cv2.COLOR_LAB2RGB)) imgs_ = np.array(imgs_)

    print(imgs_.shape)
```

```
#Function to pipe line the gray scale images that we imported in cell
```

```
def pipe_line_img(gray_scale_imgs, batch_size = 100, preprocess_f =␣
↪preprocess_input):
    imgs = np.zeros((batch_size, 224, 224, 3))
    for i in range(0, 3):
        imgs[:batch_size, :, :,i] = gray_scale_imgs[:batch_size]
    return preprocess_f(imgs)
```

```
#TensorBoard is a visualization tool provided with TensorFlow
tbCallBack = tf.keras.callbacks.TensorBoard(log_dir='./
folder_to_save_graph_3',␣
```

```
#Pipeline the images
imgs_for_input = pipe_line_img(images_gray, batch_size = 300)
```

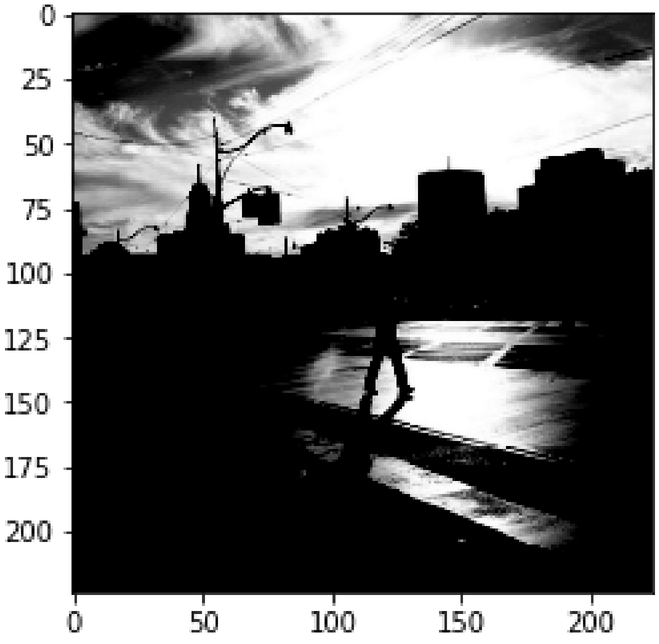
```
#Obtaining the rgb of the lab images
imgs_for_output = preprocess_input(get_rbg_from_lab(gray_imgs =
images_gray,␣
```

(300, 224, 224, 3)

```
#Outputting the rgb image we obtained in cell 7
plt.imshow(imgs_for_output[10])
```

Clipping input data to the valid range for imshow with RGBdata ([0..1]for floats or [0..255] for integers).

<matplotlib.image.AxesImage at 0x25ace381c88>




```
imgs_for_input_train, imgs_for_input_test,␣
 ↪imgs_for_output_train,imgs_for_output_test =␣
 ↪train_test_split(imgs_for_input, imgs_for_output, test_size=0.10,␣
 ↪random_state=42)
```

```
#Making the imple CNN network using Keras (4 layers)
model_simple = Sequential()
model_simple.add(Conv2D(strides = 1, kernel_size = 3, filters = 12,
use_bias =␣
 ↪True, bias_initializer =
 tf.keras.initializers.RandomUniform(minval=-0.05,␣
 ↪maxval=0.05) , padding = "valid", activation = tf.nn.relu))
model_simple.add(Conv2D(strides = 1, kernel_size = 3, filters = 12,
use_bias =␣
 ↪True, bias_initializer =
 tf.keras.initializers.RandomUniform(minval=-0.05,␣
 ↪maxval=0.05) , padding = "valid", activation = tf.nn.relu))
model_simple.add(Conv2DTranspose(strides = 1, kernel_size = 3, filters
= 12,␣
 ↪use_bias = True, bias_initializer = tf.keras.initializers.
 ↪RandomUniform(minval=-0.05, maxval=0.05) , padding = "valid",
```

2

International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 10, Issue 10, October-2022, Impact Factor: 7.429, Available online at: www.ijaresm.com

```
#Adding the optimiser to optimise our neural networn and Loss funcion to see␣
↪the loss while training our model and compiling it
model_simple.compile(optimizer = tf.keras.optimizers.Adam(epsilon =
```

```
imgs_for_s = np.zeros((300, 224,
224, 1)) imgs_for_s[:, :, :, 0] =
```

```
#Chekcing if we have added the number of input and output nodes properly by␣
↪running a sample image for errors on its size
```

```
prediction.shape
```

(300, 224, 224, 3)

```
#Training our model with 100 epochs and batch size 16
model_simple.fit(imgs_for_input_train, imgs_for_output_train, epochs =
100,␣
```

Train on 270 samples Epoch 1/100
270/270 [==============================] - 3s 10ms/sample - loss:
0.3860

Epoch 2/100
270/270 - 5ms/ - 0.34
[==============================] 1s sample loss: 13
Epoch 3/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 60
Epoch 4/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 38
Epoch 5/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 23
Epoch 6/100
270/270 - 3 5ms/ - 0.33

```
[==============================]   1s   sample   loss:   18
Epoch 7/100
270/270                            -    5ms/         -  0.33
[==============================]   1s   sample   loss:   15
```

International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 10, Issue 10, October-2022, Impact Factor: 7.429, Available online at: www.ijaresm.com

Epoch 8/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 09
Epoch 9/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 11
Epoch 10/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 09
Epoch 11/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 02
Epoch 12/100
270/270 - 5ms/ - 0.33
[==============================] 1s sample loss: 01
Epoch 13/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 99
Epoch 14/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 98
Epoch 15/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 96
Epoch 16/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 95
Epoch 17/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 96
Epoch 18/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 94
Epoch 19/100
270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 92
Epoch 20/100 4
270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 94

Epoch 21/100

270/270 - 5ms/ - 0.32
[==============================] 1s sample loss: 93

Epoch 22/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 91

Epoch 23/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 89

Epoch 24/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 88

Epoch 25/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 89

Epoch 26/100
270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 89

Epoch 27/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 89

Epoch 28/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 89

Epoch 29/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 88

Epoch 30/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 86

Epoch 31/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 87

Epoch 32/100

270/270 - 5ms/ - 0.32

[==============================] 1s sample loss: 86

Epoch 33/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3286

Epoch 34/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3286

Epoch 35/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3287

Epoch 36/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3284

Epoch 37/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3284

Epoch 38/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3284

Epoch 39/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3286

Epoch 40/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3287

Epoch 41/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3284

Epoch 42/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3286

Epoch 43/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3286

Epoch 44/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3283

Epoch 45/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 46/100

270/270

[==============================] - 1s 5ms/sample - loss: 0.3282

Epoch 47/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3284

Epoch 48/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3282

Epoch 49/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3285

Epoch 50/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3283

Epoch 51/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 52/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 53/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3280

Epoch 54/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 55/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3280

Epoch 56/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 57/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3280

Epoch 58/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 59/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3279

Epoch 60/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3278

Epoch 61/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3281

Epoch 62/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3279

Epoch 63/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3279

Epoch 64/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3279

Epoch 65/100

270/270 [==============================] - 1s 5ms/sample - loss: 0.3276

Epoch 66/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3276

Epoch 67/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3277

Epoch 68/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3276

Epoch 69/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3278

Epoch 70/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3279

Epoch 71/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3275

Epoch 72/100270/270

[==============================] - 1s 5ms/sample - loss: 0.3276

Epoch 73/100

270/270

[==============================] - 1s 5ms/sample - loss: 0.3278

Epoch 74/100

270/270 [==============================]

Epoch 75/100

270/270 [==============================]

Epoch 76/100270/270

[==============================]

Epoch 77/100270/270

[==============================]

Epoch 78/100270/270

[==============================]

Epoch 79/100270/270

[==============================]

Epoch 80/100270/270

[==============================]

Epoch 81/100270/270

[==============================]

Epoch 82/100270/270

[==============================]

Epoch 83/100270/270

[==============================]

Epoch 84/100270/270

[==============================]

Epoch 85/100270/270

[==============================]

Epoch 86/100

270/270 [==============================]

Epoch 87/100

```
- 1s 5ms/sample - loss: 0.3277
- 1s 5ms/sample - loss: 0.3279
- 1s 5ms/sample - loss: 0.3277
- 1s 5ms/sample - loss: 0.3275
- 1s 5ms/sample - loss: 0.3274
- 1s 5ms/sample - loss: 0.3276
- 1s 5ms/sample - loss: 0.3276
- 1s 5ms/sample - loss: 0.3275
- 1s 5ms/sample - loss: 0.3274
- 1s 5ms/sample - loss: 0.3274
- 1s 5ms/sample - loss: 0.3275
- 1s 5ms/sample - loss: 0.3274
- 1s 5ms/sample - loss: 0.3275
```

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 75

Epoch 88/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 74

Epoch 89/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 73

Epoch 90/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 73

Epoch 91/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 80

Epoch 92/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 76

Epoch 93/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 75

Epoch 94/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 74

Epoch 95/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 74

Epoch 96/100

270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 73

Epoch 97/100

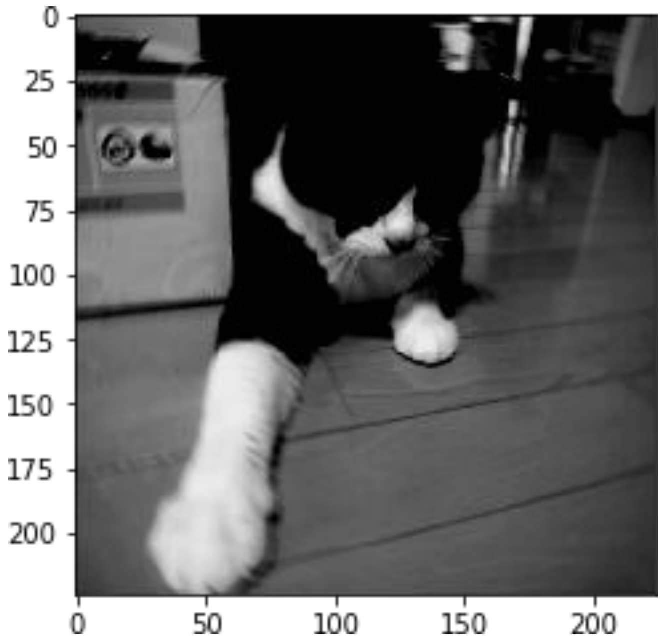
270/270 [==============================]	- 1s	5ms/ sample	- loss:	0.32 75

Epoch 98/100
270/270 [==============================] - 1s 5ms/sample - loss:
0.3273
Epoch 99/100
270/270 [==============================] - 1s 5ms/sample - loss:
0.3272
Epoch 100/100
270/270 [==============================] - 1s 5ms/sample - loss:
0.3272

<tensorflow.python.keras.callbacks.History at0x25ad0b9db88>

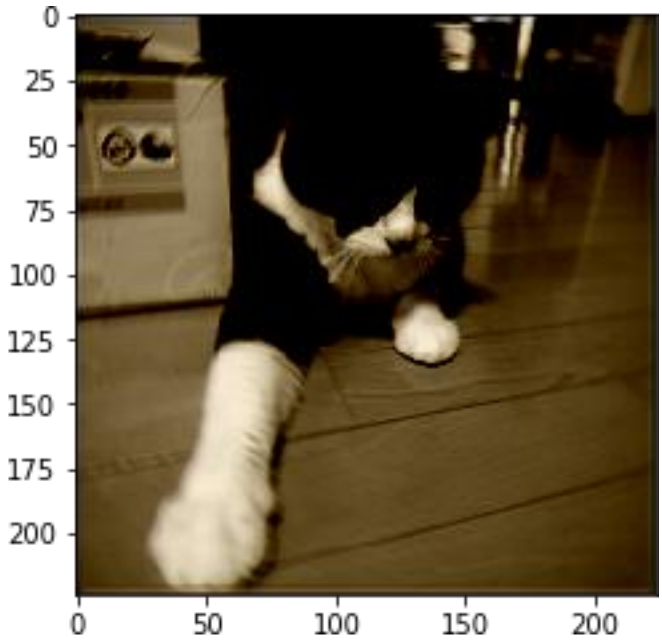
```
#Predicting the output of the input test images using our model which the model␣
↪has never seen before
```

Clipping input data to the valid range for imshow with RGBdata ([0..1]for floats or [0..255] for integers).
: <matplotlib.image.AxesImage at 0x25bc39950c8>



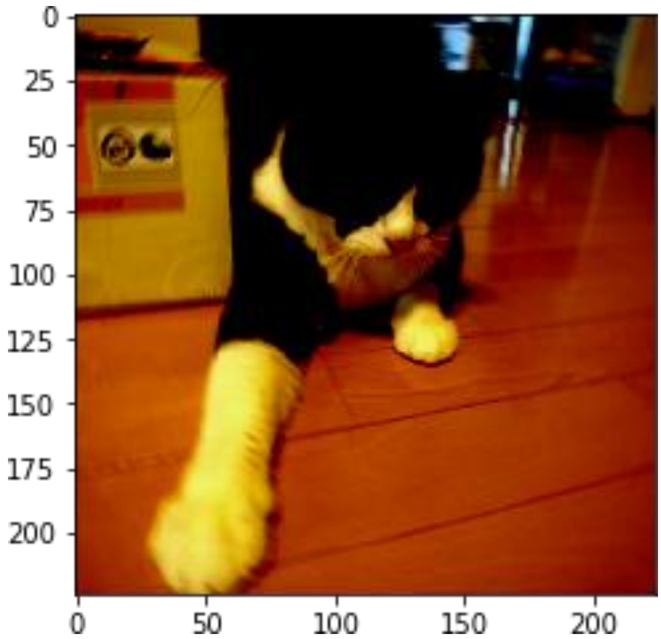

```
#Image Colorised by our model
plt.imshow(out[16,:]) # Ouput
```

: <matplotlib.image.AxesImage at 0x25bc4b3f588>



```
#The original Colorised image
plt.imshow(np.squeeze(imgs_for_output_test[16,:])) # Expected
```

Clipping input data to the valid range for imshow with RGBdata ([0..1]for floats or [0..255] for integers).
: <matplotlib.image.AxesImage at 0x25bc4ba1808>



RESULTS AND DISCUSSION

We obtain a satisfactorily coloured image that has been obtained from agreyscale image via our trained model.

Upon surveying various papers on the topic of *Image Colorization*, we realizethat the most predominantly used methods for colourizing images are the use of manual labour to enhance images even after the images are colourized by the program using various AI technologies. This is the case since a typical computer does not have high floating-point operations computability.

The usage of such programs to obtain images are widely used by researchers and is still developing. There is still a long way to go to achieveperfection in this task using a learning model or a neural network.

Moreover, since an image is a much more complicated thing when compared to numeric or textual data, the program finds it hard to distinguishbetween several parameters such as hue and saturation. We need more computation power in typical computers to be able to achieve the high quality output of coloured images from B&W images. We may never know, several years from now, we could have the capability of scanning a 100- year old picture and colourizing it in an instant.

Such things are not very far away, since Google in 2018, teased an upcoming feature in a smartphone that had the capability of converting ablack and white picture to a coloured one.

REFERENCES

[1] http://cs231n.stanford.edu/reports/2016/pdfs/
[2] 219_Report.pdf https://docs.opencv.org/3.4/de/d25/
[3] imgproc_color_conversions.html https://www.cs.huji.ac.il/~yweiss/
[4] Colorization/colorization-siggraph04.pdf https://www.kaggle.com/
[5] shravankumar9892/image-colorization http://videolectures.net/
[6] eccv2016_zhang_image_colorization/ https://press.liacs.nl/mirflickr/
[7] mirflickr.pdf
[8] https://pdfs.semanticscholar.org/8369/73ca8c06e9cab22d3ab77c95ef77fea758c d.pdf
[9] https://www.cs.drexel.edu/~david/Classes/Papers/colorize-sig02.pdf