In [ ]:

```
import numpy as np
import tensorflow as tf
import cv2
from keras import layers, regularizers
from keras.preprocessing.image import img_to_array
import os
from tqdm import tqdm
import re
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage.metrics import structural_similarity as ssim
```

In [2]:

```
def sorted_alphanumeric(data):
    convert = lambda text: int(text) if text.isdigit() else text.lower()
    alphanum_key = lambda key: [convert(c) for c in re.split('([0-9]+)',key)]
    return sorted(data,key = alphanum_key)
# defining the size of the image
SIZE = 160  # Image resize dimension
color_img_train = []
color_img_test = []
path = '/kaggle/input/landscape-image-colorization/landscape Images/color'
files = os.listdir(path)
files = sorted_alphanumeric(files)
for i in tqdm(files):
    img = cv2.imread(os.path.join(path, i), 1)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (SIZE, SIZE))
    img = img.astype('float32') / 255.0

    if i < '6000.jpg':
        color_img_train.append(img_to_array(img))  # Add to training set
    else:
        color_img_test.append(img_to_array(img))   # Add to test set

# Convert lists to numpy arrays
color_img_train = np.array(color_img_train)
color_img_test = np.array(color_img_test)

print(f"Training set size: {color_img_train.shape}")
print(f"Test set size: {color_img_test.shape}")
```

```
100%|██████████| 7129/7129 [00:48<00:00, 145.98it/s]
```

```
Training set size: (5559, 160, 160, 3)
Test set size: (1570, 160, 160, 3)
```

In [3]:

```
gray_img_train = []
gray_img_test = []
path = '/kaggle/input/landscape-image-colorization/landscape Images/gray'
files = os.listdir(path)
files = sorted_alphanumeric(files)
for i in tqdm(files):
    img = cv2.imread(os.path.join(path, i), 1)
    img = cv2.resize(img, (SIZE, SIZE))
    img = img.astype('float32') / 255.0

    if i < '6000.jpg':
        gray_img_train.append(img_to_array(img))  # Add to training set
    else:
        gray_img_test.append(img_to_array(img))   # Add to test set
```

```
# Convert lists to numpy arrays
gray_img_train = np.array(gray_img_train)
gray_img_test = np.array(gray_img_test)

print(f"Training set size (grayscale): {gray_img_train.shape}")
print(f"Test set size (grayscale): {gray_img_test.shape}")
```

```
100%|██████████| 7129/7129 [00:49<00:00, 144.82it/s]
```

```
Training set size (grayscale): (5559, 160, 160, 3)
Test set size (grayscale): (1570, 160, 160, 3)
```

In [4]:

```
gray_img_train = np.array(gray_img_train)
gray_img_test = np.array(gray_img_test)
```

In [5]:

```
print(f"Color Training set size: {color_img_train.shape}")
print(f"Color Test set size: {color_img_test.shape}")
print(f"Gray Training set size: {gray_img_train.shape}")
print(f"Gray Test set size: {gray_img_test.shape}")
```

```
Color Training set size: (5559, 160, 160, 3)
Color Test set size: (1570, 160, 160, 3)
Gray Training set size: (5559, 160, 160, 3)
Gray Test set size: (1570, 160, 160, 3)
```

In [6]:

```
split_index = int(len(gray_img_train) * 0.95)

train_gray_image = gray_img_train[:split_index]
val_gray_image = gray_img_train[split_index:]
train_color_image = color_img_train[:split_index]
val_color_image = color_img_train[split_index:]

# Reshape arrays to the required dimensions
train_g = np.reshape(train_gray_image, (len(train_gray_image), SIZE, SIZE, 3))
train_c = np.reshape(train_color_image, (len(train_color_image), SIZE, SIZE, 3))
val_g = np.reshape(val_gray_image, (len(val_gray_image), SIZE, SIZE, 3))
val_c = np.reshape(val_color_image, (len(val_color_image), SIZE, SIZE, 3))
test_gray_image = np.reshape(gray_img_test, (len(gray_img_test), SIZE, SIZE, 3))
test_color_image = np.reshape(color_img_test, (len(color_img_test), SIZE, SIZE, 3))

# Check shapes to confirm they are aligned
print('Train color image shape:', train_c.shape)
print('Validation color image shape:', val_c.shape)
print('Test color image shape:', test_color_image.shape)
```

```
Train color image shape: (5281, 160, 160, 3)
Validation color image shape: (278, 160, 160, 3)
Test color image shape: (1570, 160, 160, 3)
```

In [7]:

```
def down(filters, kernel_size, batch_norm):
    def block(x):
        x = layers.Conv2D(filters, kernel_size, strides=2, padding="same",
                          kernel_regularizer=regularizers.l2(0.0000000001))(x)
        if batch_norm:
            x = layers.BatchNormalization()(x)
        x = layers.LeakyReLU()(x)
        return x
    return block

def up(filters, kernel_size, batch_norm):
    def block(x):
        x = layers.Conv2DTranspose(filters, kernel_size, strides=2, padding="same",
                                   kernel_regularizer=regularizers.l2(0.0000000001))(x)
```

```
        if batch_norm:
            x = layers.BatchNormalization()(x)
        x = layers.ReLU()(x)
        return x
    return block
```

In [8]:

```
def recolourization_model():
    inputs = layers.Input(shape=(160, 160, 3))

    # Downsampling
    d1 = down(128, (3, 3), False)(inputs)
    d2 = down(128, (3, 3), False)(d1)
    d3 = down(256, (3, 3), True)(d2)
    d4 = down(512, (3, 3), True)(d3)
    d5 = down(512, (3, 3), True)(d4)

    # Upsampling
    u1 = up(512, (3, 3), False)(d5)
    u1 = layers.concatenate([u1, d4])
    u2 = up(256, (3, 3), False)(u1)
    u2 = layers.concatenate([u2, d3])
    u3 = up(128, (3, 3), False)(u2)
    u3 = layers.concatenate([u3, d2])
    u4 = up(128, (3, 3), False)(u3)
    u4 = layers.concatenate([u4, d1])

    u5 = up(3, (3, 3), False)(u4)
    u5 = layers.concatenate([u5, inputs])

    # Output layer with sigmoid activation for [0, 1] range
    output = layers.Conv2D(3, (3, 3), strides=1, padding='same', activation='sigmoid',
                           kernel_regularizer=regularizers.l2(0.01))(u5)

    return tf.keras.Model(inputs=inputs, outputs=output)
```

In [9]:

```
recolourization_model = recolourization_model()
recolourization_model.summary()
```

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 160, 160, 3)	0	-
conv2d (Conv2D)	(None, 80, 80, 128)	3,584	input_layer[0][0]
leaky_re_lu (LeakyReLU)	(None, 80, 80, 128)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 40, 40, 128)	147,584	leaky_re_lu[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 40, 40, 128)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 20, 20, 256)	295,168	leaky_re_lu_1[0]…
batch_normalization (BatchNormalizatio…	(None, 20, 20, 256)	1,024	conv2d_2[0][0]
leaky_re_lu_2	(None, 20, 20	0	batch_normalizat…

leaky_re_lu_2 (LeakyReLU)	(None, 20, 20, 256)	0	batch_normalizat…
conv2d_3 (Conv2D)	(None, 10, 10, 512)	1,180,160	leaky_re_lu_2[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 10, 10, 512)	2,048	conv2d_3[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 10, 10, 512)	0	batch_normalizat…
conv2d_4 (Conv2D)	(None, 5, 5, 512)	2,359,808	leaky_re_lu_3[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 5, 5, 512)	2,048	conv2d_4[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 5, 5, 512)	0	batch_normalizat…
conv2d_transpose (Conv2DTranspose)	(None, 10, 10, 512)	2,359,808	leaky_re_lu_4[0]…
re_lu (ReLU)	(None, 10, 10, 512)	0	conv2d_transpose…
concatenate (Concatenate)	(None, 10, 10, 1024)	0	re_lu[0][0], leaky_re_lu_3[0]…
conv2d_transpose_1 (Conv2DTranspose)	(None, 20, 20, 256)	2,359,552	concatenate[0][0]
re_lu_1 (ReLU)	(None, 20, 20, 256)	0	conv2d_transpose…
concatenate_1 (Concatenate)	(None, 20, 20, 512)	0	re_lu_1[0][0], leaky_re_lu_2[0]…
conv2d_transpose_2 (Conv2DTranspose)	(None, 40, 40, 128)	589,952	concatenate_1[0]…
re_lu_2 (ReLU)	(None, 40, 40, 128)	0	conv2d_transpose…
concatenate_2 (Concatenate)	(None, 40, 40, 256)	0	re_lu_2[0][0], leaky_re_lu_1[0]…
conv2d_transpose_3 (Conv2DTranspose)	(None, 80, 80, 128)	295,040	concatenate_2[0]…
re_lu_3 (ReLU)	(None, 80, 80, 128)	0	conv2d_transpose…
concatenate_3 (Concatenate)	(None, 80, 80, 256)	0	re_lu_3[0][0], leaky_re_lu[0][0]
conv2d_transpose_4 (Conv2DTranspose)	(None, 160, 160, 3)	6,915	concatenate_3[0]…
re_lu_4 (ReLU)	(None, 160, 160, 3)	0	conv2d_transpose…
concatenate_4 (Concatenate)	(None, 160, 160, 6)	0	re_lu_4[0][0], input_layer[0][0]

conv2d_5 (Conv2D)	(None, 160, 160, 3)	165	concatenate_4[0]…

Total params: 9,602,856 (36.63 MB)

Trainable params: 9,600,296 (36.62 MB)

Non-trainable params: 2,560 (10.00 KB)

In [10]:

```
recolourization_model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001
), loss='mse', metrics=['mae'])
```

In [11]:

```
history = recolourization_model.fit(train_g, train_c, epochs = 105,batch_size = 50,valid
ation_data=(val_g, val_c),verbose = 1)
```

```
Epoch 1/105
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1731071934.140625      94 service.cc:145] XLA service 0x7817c80049c0 initiali
zed for platform CUDA (this does not guarantee that XLA will be used). Devices:
I0000 00:00:1731071934.140816      94 service.cc:153]   StreamExecutor device (0): Tesla
P100-PCIE-16GB, Compute Capability 6.0
```

```
  1/106 ━━━━━━━━━━━━━━━━━━━━ 25:01 14s/step - loss: 0.1312 - mae: 0.2502
```

```
I0000 00:00:1731071944.172903      94 device_compiler.h:188] Compiled cluster using XLA!
This line is logged at most once for the lifetime of the process.
```

```
106/106 ━━━━━━━━━━━━━━━━━━━━ 36s 203ms/step - loss: 0.0964 - mae: 0.1940 - val_loss: 0.08
48 - val_mae: 0.2185
Epoch 2/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0543 - mae: 0.1483 - val_loss: 0.04
69 - val_mae: 0.1598
Epoch 3/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0200 - mae: 0.0818 - val_loss: 0.01
81 - val_mae: 0.0946
Epoch 4/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0119 - mae: 0.0710 - val_loss: 0.01
69 - val_mae: 0.0955
Epoch 5/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0095 - mae: 0.0656 - val_loss: 0.01
25 - val_mae: 0.0807
Epoch 6/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0088 - mae: 0.0641 - val_loss: 0.01
70 - val_mae: 0.0988
Epoch 7/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0080 - mae: 0.0613 - val_loss: 0.01
69 - val_mae: 0.1001
Epoch 8/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0077 - mae: 0.0604 - val_loss: 0.00
89 - val_mae: 0.0675
Epoch 9/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0073 - mae: 0.0583 - val_loss: 0.00
84 - val_mae: 0.0629
Epoch 10/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0069 - mae: 0.0569 - val_loss: 0.00
91 - val_mae: 0.0639
Epoch 11/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0068 - mae: 0.0568 - val_loss: 0.00
79 - val_mae: 0.0617
Epoch 12/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 109ms/step - loss: 0.0064 - mae: 0.0549 - val_loss: 0.01
24 - val_mae: 0.0831
Epoch 13/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0068 - mae: 0.0566 - val_loss: 0.00
87 - val_mae: 0.0646
Epoch 14/105
```

```
Epoch 14/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0062 - mae: 0.0540 - val_loss: 0.00
87 - val_mae: 0.0656
Epoch 15/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0062 - mae: 0.0542 - val_loss: 0.00
81 - val_mae: 0.0631
Epoch 16/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0061 - mae: 0.0537 - val_loss: 0.00
82 - val_mae: 0.0652
Epoch 17/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0059 - mae: 0.0520 - val_loss: 0.00
80 - val_mae: 0.0637
Epoch 18/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0058 - mae: 0.0524 - val_loss: 0.00
68 - val_mae: 0.0544
Epoch 19/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0059 - mae: 0.0536 - val_loss: 0.00
90 - val_mae: 0.0633
Epoch 20/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0058 - mae: 0.0524 - val_loss: 0.00
67 - val_mae: 0.0548
Epoch 21/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0052 - mae: 0.0489 - val_loss: 0.00
68 - val_mae: 0.0559
Epoch 22/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0052 - mae: 0.0496 - val_loss: 0.00
67 - val_mae: 0.0543
Epoch 23/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0048 - mae: 0.0474 - val_loss: 0.00
68 - val_mae: 0.0547
Epoch 24/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0047 - mae: 0.0475 - val_loss: 0.00
93 - val_mae: 0.0689
Epoch 25/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 109ms/step - loss: 0.0046 - mae: 0.0472 - val_loss: 0.00
68 - val_mae: 0.0540
Epoch 26/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0045 - mae: 0.0462 - val_loss: 0.00
67 - val_mae: 0.0544
Epoch 27/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0042 - mae: 0.0448 - val_loss: 0.00
92 - val_mae: 0.0685
Epoch 28/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0042 - mae: 0.0449 - val_loss: 0.00
66 - val_mae: 0.0532
Epoch 29/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 109ms/step - loss: 0.0038 - mae: 0.0426 - val_loss: 0.00
81 - val_mae: 0.0637
Epoch 30/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0042 - mae: 0.0452 - val_loss: 0.00
80 - val_mae: 0.0610
Epoch 31/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0036 - mae: 0.0414 - val_loss: 0.00
80 - val_mae: 0.0608
Epoch 32/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0035 - mae: 0.0414 - val_loss: 0.00
73 - val_mae: 0.0590
Epoch 33/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0032 - mae: 0.0394 - val_loss: 0.00
86 - val_mae: 0.0658
Epoch 34/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0033 - mae: 0.0404 - val_loss: 0.00
66 - val_mae: 0.0529
Epoch 35/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0031 - mae: 0.0383 - val_loss: 0.00
70 - val_mae: 0.0563
Epoch 36/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0028 - mae: 0.0365 - val_loss: 0.00
64 - val_mae: 0.0521
Epoch 37/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0027 - mae: 0.0361 - val_loss: 0.00
70 - val_mae: 0.0549
Epoch 38/105
```

```
Epoch 38/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0027 - mae: 0.0357 - val_loss: 0.00
67 - val_mae: 0.0539
Epoch 39/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0027 - mae: 0.0361 - val_loss: 0.00
76 - val_mae: 0.0611
Epoch 40/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0026 - mae: 0.0351 - val_loss: 0.00
82 - val_mae: 0.0641
Epoch 41/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0025 - mae: 0.0343 - val_loss: 0.00
66 - val_mae: 0.0532
Epoch 42/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0025 - mae: 0.0348 - val_loss: 0.00
66 - val_mae: 0.0528
Epoch 43/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0023 - mae: 0.0332 - val_loss: 0.00
66 - val_mae: 0.0539
Epoch 44/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0023 - mae: 0.0333 - val_loss: 0.00
69 - val_mae: 0.0540
Epoch 45/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0022 - mae: 0.0324 - val_loss: 0.00
75 - val_mae: 0.0599
Epoch 46/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0021 - mae: 0.0314 - val_loss: 0.00
64 - val_mae: 0.0519
Epoch 47/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0020 - mae: 0.0306 - val_loss: 0.00
64 - val_mae: 0.0517
Epoch 48/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0020 - mae: 0.0305 - val_loss: 0.00
68 - val_mae: 0.0529
Epoch 49/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0020 - mae: 0.0309 - val_loss: 0.00
65 - val_mae: 0.0521
Epoch 50/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0020 - mae: 0.0306 - val_loss: 0.00
64 - val_mae: 0.0516
Epoch 51/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0018 - mae: 0.0295 - val_loss: 0.00
65 - val_mae: 0.0526
Epoch 52/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0019 - mae: 0.0300 - val_loss: 0.00
62 - val_mae: 0.0511
Epoch 53/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0018 - mae: 0.0295 - val_loss: 0.00
67 - val_mae: 0.0557
Epoch 54/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0018 - mae: 0.0297 - val_loss: 0.00
66 - val_mae: 0.0528
Epoch 55/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 109ms/step - loss: 0.0018 - mae: 0.0290 - val_loss: 0.00
64 - val_mae: 0.0521
Epoch 56/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0018 - mae: 0.0294 - val_loss: 0.00
65 - val_mae: 0.0520
Epoch 57/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0273 - val_loss: 0.00
64 - val_mae: 0.0513
Epoch 58/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0273 - val_loss: 0.00
62 - val_mae: 0.0512
Epoch 59/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0276 - val_loss: 0.00
67 - val_mae: 0.0539
Epoch 60/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0015 - mae: 0.0265 - val_loss: 0.00
63 - val_mae: 0.0508
Epoch 61/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0273 - val_loss: 0.00
67 - val_mae: 0.0537
Epoch 62/105
```

```
Epoch 62/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0277 - val_loss: 0.00
64 - val_mae: 0.0524
Epoch 63/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0278 - val_loss: 0.00
63 - val_mae: 0.0513
Epoch 64/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0016 - mae: 0.0278 - val_loss: 0.00
62 - val_mae: 0.0505
Epoch 65/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0014 - mae: 0.0261 - val_loss: 0.00
72 - val_mae: 0.0572
Epoch 66/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0015 - mae: 0.0269 - val_loss: 0.00
63 - val_mae: 0.0508
Epoch 67/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0013 - mae: 0.0248 - val_loss: 0.00
63 - val_mae: 0.0514
Epoch 68/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0015 - mae: 0.0266 - val_loss: 0.00
66 - val_mae: 0.0541
Epoch 69/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0014 - mae: 0.0261 - val_loss: 0.00
62 - val_mae: 0.0500
Epoch 70/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 109ms/step - loss: 0.0013 - mae: 0.0245 - val_loss: 0.00
67 - val_mae: 0.0533
Epoch 71/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0014 - mae: 0.0254 - val_loss: 0.00
64 - val_mae: 0.0521
Epoch 72/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0013 - mae: 0.0250 - val_loss: 0.00
71 - val_mae: 0.0566
Epoch 73/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0015 - mae: 0.0274 - val_loss: 0.00
63 - val_mae: 0.0510
Epoch 74/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0013 - mae: 0.0243 - val_loss: 0.00
61 - val_mae: 0.0502
Epoch 75/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0013 - mae: 0.0242 - val_loss: 0.00
62 - val_mae: 0.0511
Epoch 76/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0012 - mae: 0.0241 - val_loss: 0.00
64 - val_mae: 0.0522
Epoch 77/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0013 - mae: 0.0245 - val_loss: 0.00
64 - val_mae: 0.0528
Epoch 78/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0012 - mae: 0.0236 - val_loss: 0.00
65 - val_mae: 0.0541
Epoch 79/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0012 - mae: 0.0240 - val_loss: 0.00
62 - val_mae: 0.0510
Epoch 80/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 21s 110ms/step - loss: 0.0012 - mae: 0.0239 - val_loss: 0.00
61 - val_mae: 0.0499
Epoch 81/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0225 - val_loss: 0.00
63 - val_mae: 0.0513
Epoch 82/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0226 - val_loss: 0.00
62 - val_mae: 0.0501
Epoch 83/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0221 - val_loss: 0.00
61 - val_mae: 0.0497
Epoch 84/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0228 - val_loss: 0.00
60 - val_mae: 0.0500
Epoch 85/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0231 - val_loss: 0.00
61 - val_mae: 0.0497
Epoch 86/105
```

```
Epoch 86/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0228 - val_loss: 0.00
64 - val_mae: 0.0509
Epoch 87/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0228 - val_loss: 0.00
63 - val_mae: 0.0511
Epoch 88/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0225 - val_loss: 0.00
62 - val_mae: 0.0505
Epoch 89/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0229 - val_loss: 0.00
61 - val_mae: 0.0497
Epoch 90/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0010 - mae: 0.0215 - val_loss: 0.00
60 - val_mae: 0.0496
Epoch 91/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.9445e-04 - mae: 0.0216 - val_loss:
0.0062 - val_mae: 0.0504
Epoch 92/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.8137e-04 - mae: 0.0214 - val_loss:
0.0060 - val_mae: 0.0495
Epoch 93/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0010 - mae: 0.0217 - val_loss: 0.00
60 - val_mae: 0.0500
Epoch 94/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0010 - mae: 0.0219 - val_loss: 0.00
61 - val_mae: 0.0499
Epoch 95/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.8219e-04 - mae: 0.0213 - val_loss:
0.0062 - val_mae: 0.0519
Epoch 96/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 0.0011 - mae: 0.0228 - val_loss: 0.00
61 - val_mae: 0.0501
Epoch 97/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.9584e-04 - mae: 0.0216 - val_loss:
0.0061 - val_mae: 0.0497
Epoch 98/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.1110e-04 - mae: 0.0205 - val_loss:
0.0061 - val_mae: 0.0499
Epoch 99/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.4289e-04 - mae: 0.0211 - val_loss:
0.0063 - val_mae: 0.0500
Epoch 100/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 109ms/step - loss: 9.4462e-04 - mae: 0.0212 - val_loss:
0.0059 - val_mae: 0.0488
Epoch 101/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 8.9341e-04 - mae: 0.0202 - val_loss:
0.0062 - val_mae: 0.0499
Epoch 102/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.6895e-04 - mae: 0.0213 - val_loss:
0.0060 - val_mae: 0.0494
Epoch 103/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.3705e-04 - mae: 0.0208 - val_loss:
0.0061 - val_mae: 0.0495
Epoch 104/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 8.8685e-04 - mae: 0.0203 - val_loss:
0.0060 - val_mae: 0.0493
Epoch 105/105
106/106 ━━━━━━━━━━━━━━━━━━━━ 12s 110ms/step - loss: 9.3598e-04 - mae: 0.0209 - val_loss:
0.0060 - val_mae: 0.0496
```

In [12]:

```
recolourization_model.save("/kaggle/working/recolorization_model.keras")
```

In [13]:

```
predicted_color = recolourization_model.predict(test_gray_image)
```

```
50/50 ━━━━━━━━━━━━━━━━━━━━ 3s 39ms/step
```

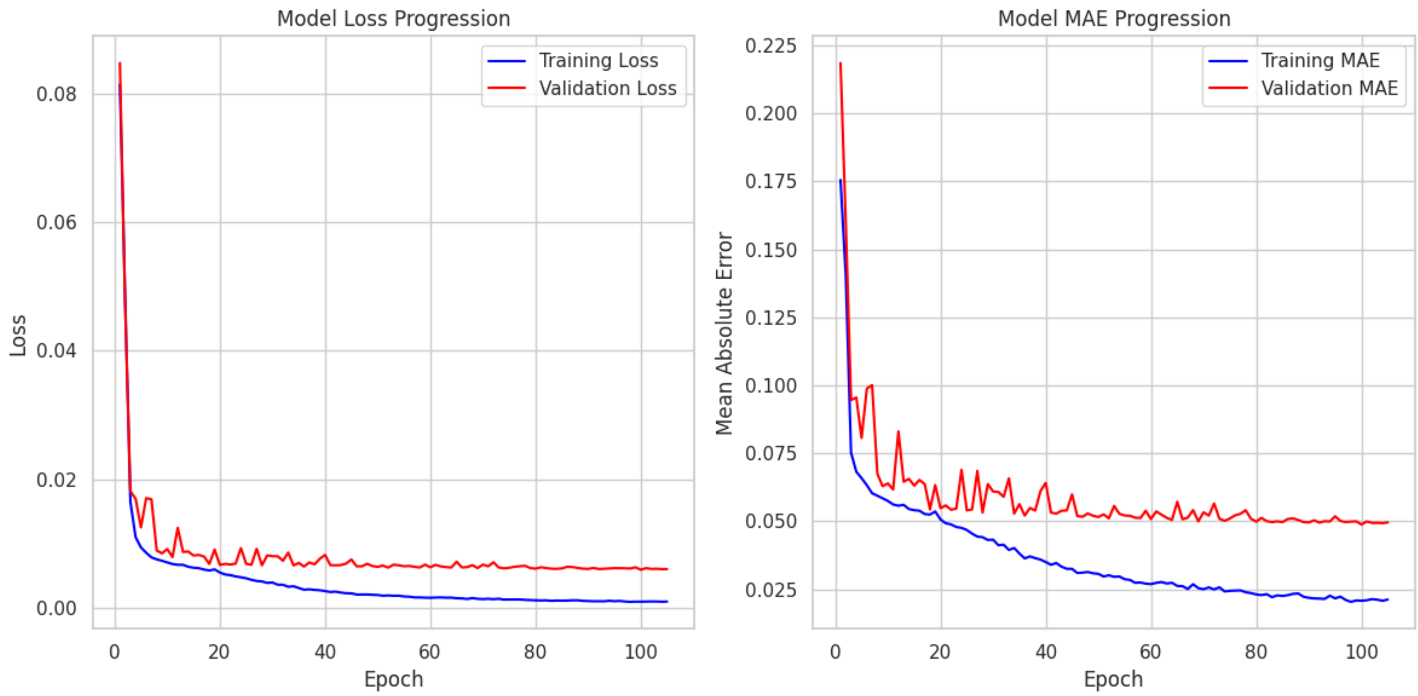
In [14]:

```
# Extract the loss and accuracy values
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_mae = history.history['mae']
val_mae = history.history['val_mae']
epochs = range(1, len(train_loss) + 1)
sns.set(style="whitegrid")

# Plot loss (training vs validation)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)   # Create a 1x2 grid, first subplot for loss
sns.lineplot(x=epochs, y=train_loss, label='Training Loss', color='blue')
sns.lineplot(x=epochs, y=val_loss, label='Validation Loss', color='red')
plt.title('Model Loss Progression')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot mean absolute error (mae) (training vs validation)
plt.subplot(1, 2, 2)   # Second subplot for mae
sns.lineplot(x=epochs, y=train_mae, label='Training MAE', color='blue')
sns.lineplot(x=epochs, y=val_mae, label='Validation MAE', color='red')
plt.title('Model MAE Progression')
plt.xlabel('Epoch')
plt.ylabel('Mean Absolute Error')
plt.legend()

# Show the plots
plt.tight_layout()
plt.show()
```



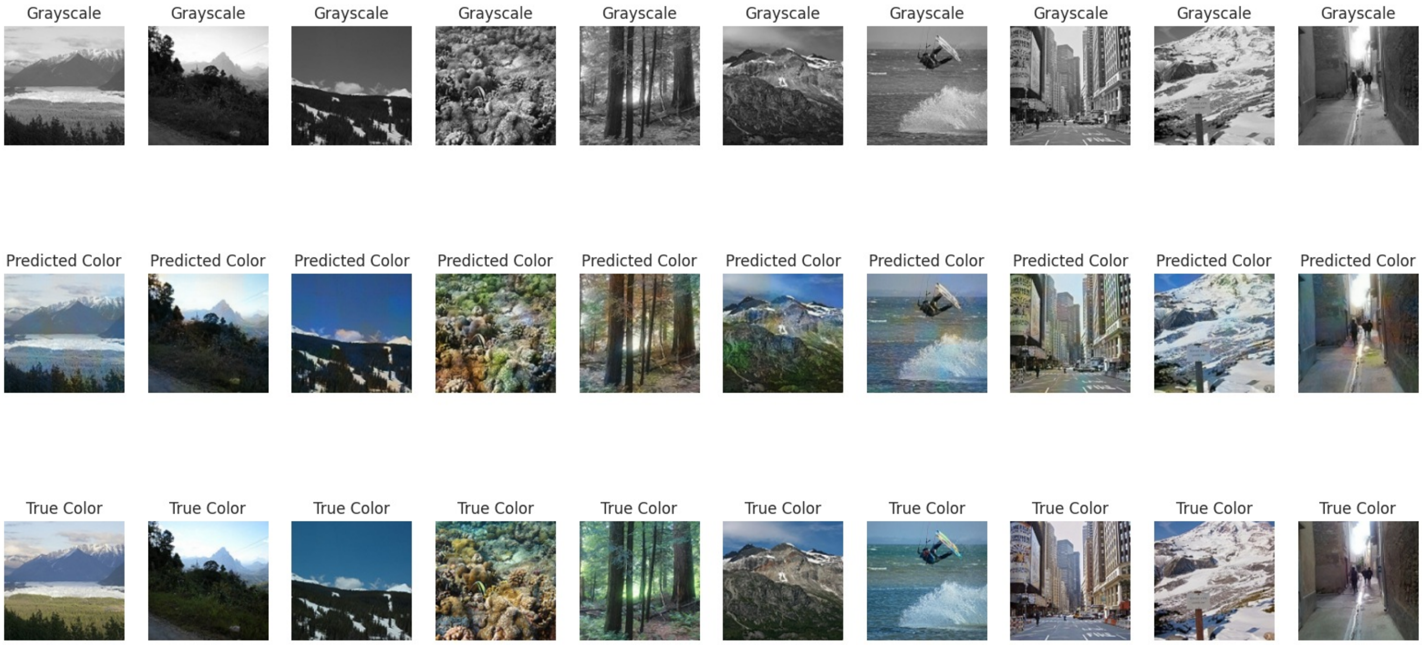
In [15]:

```
n = 10   # Number of images to display
plt.figure(figsize=(20, 10))
for i in range(n):
    # Grayscale input images
    ax = plt.subplot(3, n, i + 1)
    plt.imshow(test_gray_image[i], cmap='gray')
    plt.title("Grayscale")
    plt.axis('off')

    # Predicted color images
    ax = plt.subplot(3, n, i + n + 1)
    plt.imshow(predicted_color[i])
    plt.title("Predicted Color")
    plt.axis('off')
```

```
    # Ground truth color images
    ax = plt.subplot(3, n, i + 2 * n + 1)
    plt.imshow(test_color_image[i])
    plt.title("True Color")
    plt.axis('off')

plt.show()
```



In [16]:

```
psnr_value = psnr(test_color_image[0], predicted_color[0], data_range=1.0)
ssim_value = ssim(test_color_image[0], predicted_color[0], multichannel=True, win_size=3
, channel_axis=-1, data_range=1.0)

print(f"PSNR: {psnr_value}, SSIM: {ssim_value}")
```

```
PSNR: 22.809714345038735, SSIM: 0.9491534233093262
```

In [17]:

```
recolorization_model = tf.keras.models.load_model("/kaggle/working/recolorization_model.k
eras", compile=False)
SIZE = 160
def colorize_image(grayscale_img):
    # Resize and normalize the grayscale image to match the model's input format
    img = cv2.resize(grayscale_img, (SIZE, SIZE))
    img = img.astype('float32') / 255.0  # Normalize
    img = np.stack((img, img, img), axis=-1)  # Convert grayscale to 3-channel RGB forma
t
    img = np.expand_dims(img, axis=0)  # Add batch dimension for the model

    # Predict the colorized version using the model
    colorized_img = recolorization_model.predict(img)
    colorized_img = colorized_img[0]  # Remove batch dimension
    colorized_img = np.clip(colorized_img, 0, 1)  # Clip to valid pixel range

    # Convert colorized image to 0-255 scale for visualization
    colorized_img = (colorized_img * 255).astype(np.uint8)
    return colorized_img
```

In [ ]:

```
import gradio as gr
gr_interface = gr.Interface(
    fn=colorize_image,
    inputs=gr.Image(image_mode='L', label="Upload a grayscale image"),
    outputs=gr.Image(label="Colorized Image"),
    title="Image Recolorization",
```

```
    description="Upload a grayscale image to see it colorized by the trained model."
)

# Launch the Gradio app
gr_interface.launch()
```