CGACH – Artefact – User Guide

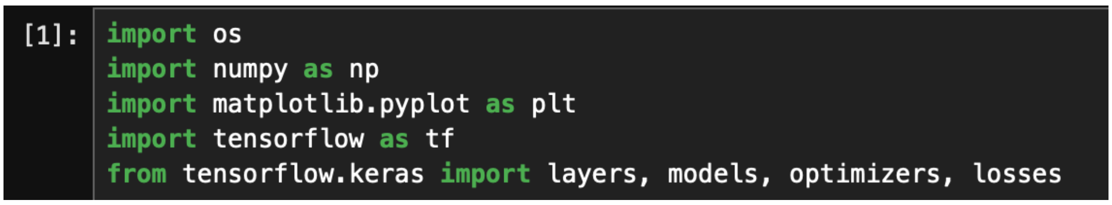
Note – OS independent

Programming Language: Python

Required Libraries: TensorFlow, NumPy, Matplotlib

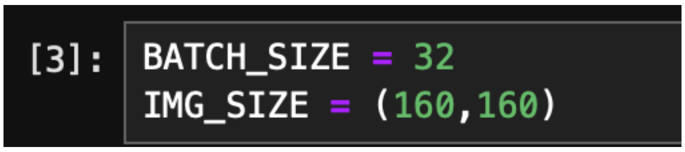
Execute code cell-by-cell:

1. **Importing all required libraries**



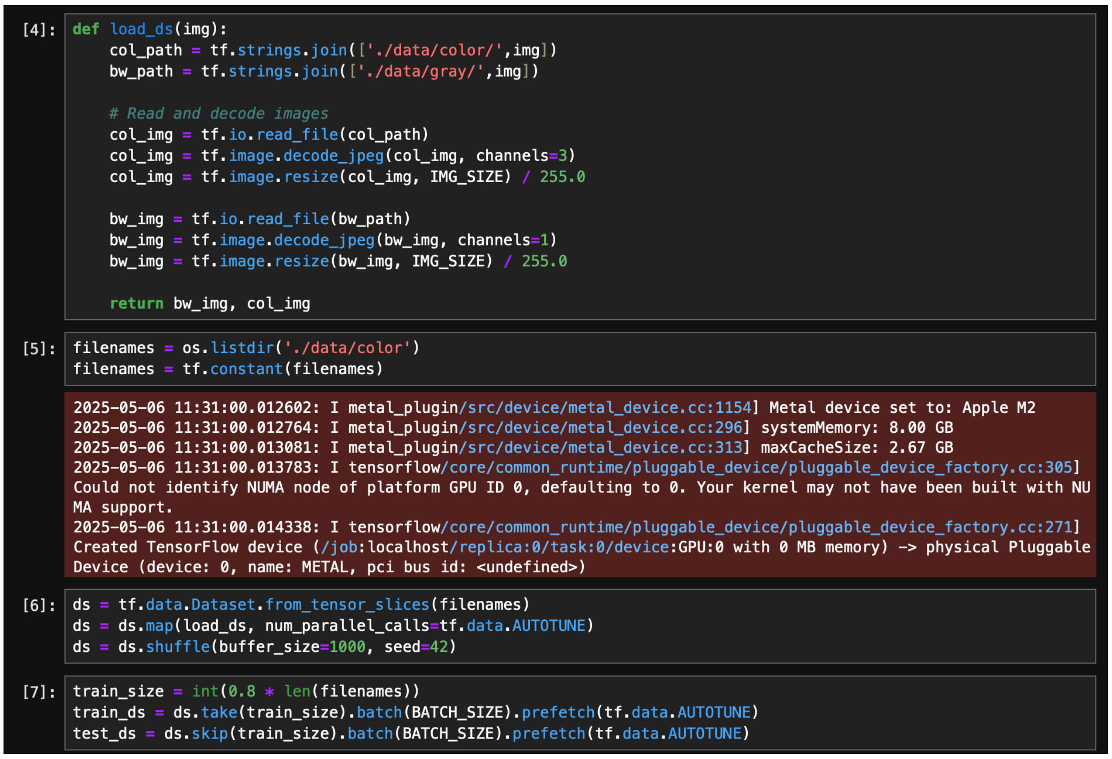
```
[1]: import os
     import numpy as np
     import matplotlib.pyplot as plt
     import tensorflow as tf
     from tensorflow.keras import layers, models, optimizers, losses
```

2. **Declaring image size and size of the data batches**



```
[3]: BATCH_SIZE = 32
     IMG_SIZE = (160,160)
```

3. **Loading dataset**



```
[4]: def load_ds(img):
         col_path = tf.strings.join(['./data/color/',img])
         bw_path = tf.strings.join(['./data/gray/',img])

         # Read and decode images
         col_img = tf.io.read_file(col_path)
         col_img = tf.image.decode_jpeg(col_img, channels=3)
         col_img = tf.image.resize(col_img, IMG_SIZE) / 255.0

         bw_img = tf.io.read_file(bw_path)
         bw_img = tf.image.decode_jpeg(bw_img, channels=1)
         bw_img = tf.image.resize(bw_img, IMG_SIZE) / 255.0

         return bw_img, col_img
```

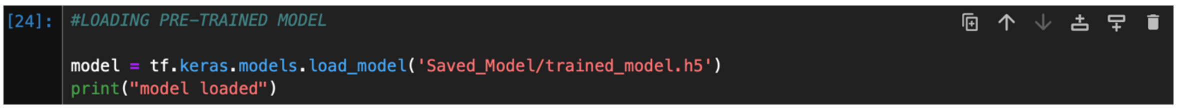
```
[5]: filenames = os.listdir('./data/color')
     filenames = tf.constant(filenames)
```

```
2025-05-06 11:31:00.012602: I metal_plugin/src/device/metal_device.cc:1154] Metal device set to: Apple M2
2025-05-06 11:31:00.012764: I metal_plugin/src/device/metal_device.cc:296] systemMemory: 8.00 GB
2025-05-06 11:31:00.013081: I metal_plugin/src/device/metal_device.cc:313] maxCacheSize: 2.67 GB
2025-05-06 11:31:00.013783: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:305]
Could not identify NUMA node of platform GPU ID 0, defaulting to 0. Your kernel may not have been built with NU
MA support.
2025-05-06 11:31:00.014338: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:271]
Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical Pluggable
Device (device: 0, name: METAL, pci bus id: <undefined>)
```

```
[6]: ds = tf.data.Dataset.from_tensor_slices(filenames)
     ds = ds.map(load_ds, num_parallel_calls=tf.data.AUTOTUNE)
     ds = ds.shuffle(buffer_size=1000, seed=42)
```

```
[7]: train_size = int(0.8 * len(filenames))
     train_ds = ds.take(train_size).batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
     test_ds = ds.skip(train_size).batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
```

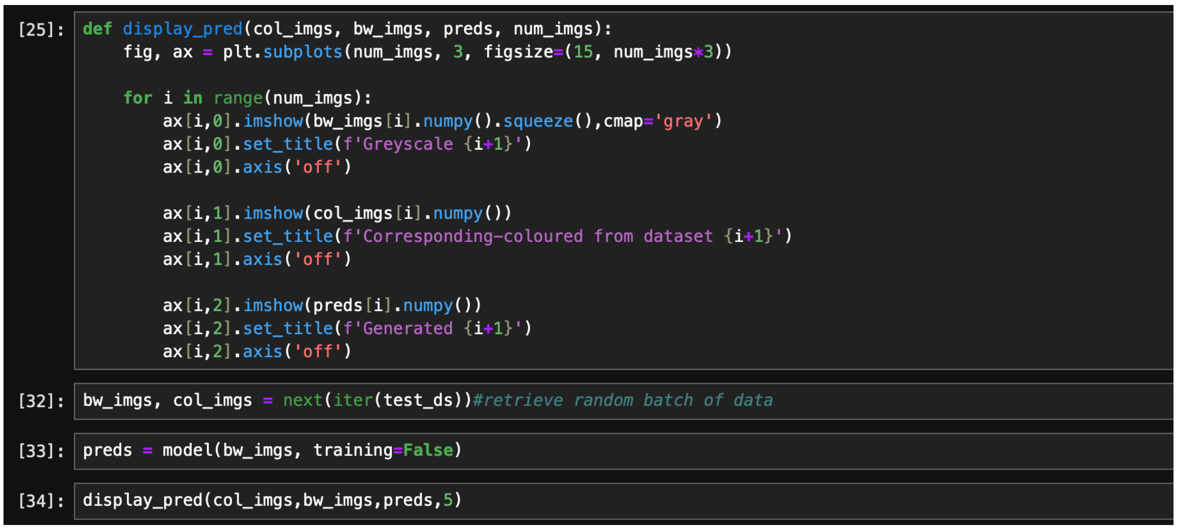
4. Loading pre-trained model



```
[24]: #LOADING PRE-TRAINED MODEL

      model = tf.keras.models.load_model('Saved_Model/trained_model.h5')
      print("model loaded")
```

5. Displaying the generated results



```
[25]: def display_pred(col_imgs, bw_imgs, preds, num_imgs):
          fig, ax = plt.subplots(num_imgs, 3, figsize=(15, num_imgs*3))

          for i in range(num_imgs):
              ax[i,0].imshow(bw_imgs[i].numpy().squeeze(),cmap='gray')
              ax[i,0].set_title(f'Greyscale {i+1}')
              ax[i,0].axis('off')

              ax[i,1].imshow(col_imgs[i].numpy())
              ax[i,1].set_title(f'Corresponding-coloured from dataset {i+1}')
              ax[i,1].axis('off')

              ax[i,2].imshow(preds[i].numpy())
              ax[i,2].set_title(f'Generated {i+1}')
              ax[i,2].axis('off')

[32]: bw_imgs, col_imgs = next(iter(test_ds))#retrieve random batch of data

[33]: preds = model(bw_imgs, training=False)

[34]: display_pred(col_imgs,bw_imgs,preds,5)
```