```
import tensorflow as tf
import numpy as np
import PIL.Image
import time
import functools
import os
path_gambar_konten = '/content/drive/MyDrive/semeter 6/DLD lanjut/dataset/mabuci.jpg'
path_gambar_style = '/content/drive/MyDrive/semeter 6/DLD lanjut/dataset/lukisan.jpg'
def tensor_to_image(tensor):
    tensor = tensor * 255
    tensor = np.array(tensor, dtype=np.uint8)
    if np.ndim(tensor) > 3:
        assert tensor.shape[0] == 1
        tensor = tensor[0]
    return PIL.Image.fromarray(tensor)

def load_img(path_to_img):
    if not os.path.exists(path_to_img):
        raise FileNotFoundError(
            f"\n\n[ERROR FILE TIDAK DITEMUKAN]\n"
            f"Sistem tidak menemukan file bernama: '{path_to_img}'\n"
            f"SOLUSI:\n"
            f"1. Cek panel 'Files' di sebelah kiri. Apakah file sudah terupload?\n"
            f"2. Apakah nama file di kode (path_gambar_konten/style) sudah sama persis dengan nama file
            f"3. Ingat: Python membedakan huruf besar/kecil (Foto.jpg beda dengan foto.jpg).\n"
        )

    max_dim = 512
    img = tf.io.read_file(path_to_img)
    img = tf.image.decode_image(img, channels=3)
    img = tf.image.convert_image_dtype(img, tf.float32)

    shape = tf.cast(tf.shape(img)[:-1], tf.float32)
    long_dim = max(shape)
    scale = max_dim / long_dim

    new_shape = tf.cast(shape * scale, tf.int32)
    img = tf.image.resize(img, new_shape)
    img = img[tf.newaxis, :]
    return img

def vgg_layers(layer_names):

    vgg = tf.keras.applications.VGG19(include_top=False, weights='imagenet')
    vgg.trainable = False

    outputs = [vgg.get_layer(name).output for name in layer_names]
    model = tf.keras.Model([vgg.input], outputs)
    return model
content_layers = ['block5_conv2']
style_layers = ['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1', 'block5_conv1']

num_content_layers = len(content_layers)
num_style_layers = len(style_layers)

class StyleContentModel(tf.keras.models.Model):
    def __init__(self, style_layers, content_layers):
        super(StyleContentModel, self).__init__()
        self.vgg = vgg_layers(style_layers + content_layers)
        self.style_layers = style_layers
        self.content_layers = content_layers
        self.num_style_layers = len(style_layers)
```

```
        self.vgg.trainable = False

    def call(self, inputs):
        inputs = inputs * 255.0
        preprocessed_input = tf.keras.applications.vgg19.preprocess_input(inputs)
        outputs = self.vgg(preprocessed_input)
        style_outputs, content_outputs = (outputs[:self.num_style_layers],
                                          outputs[self.num_style_layers:])

        style_outputs = [gram_matrix(style_output) for style_output in style_outputs]

        content_dict = {content_name: value for content_name, value in zip(self.content_layers, content_
        style_dict = {style_name: value for style_name, value in zip(self.style_layers, style_outputs)}

        return {'content': content_dict, 'style': style_dict}

def gram_matrix(input_tensor):
    result = tf.linalg.einsum('bijc,bijd->bcd', input_tensor, input_tensor)
    input_shape = tf.shape(input_tensor)
    num_locations = tf.cast(input_shape[1]*input_shape[2], tf.float32)
    return result/(num_locations)

extractor = StyleContentModel(style_layers, content_layers)
opt = tf.optimizers.Adam(learning_rate=0.02, beta_1=0.99, epsilon=1e-1)

style_weight = 1e-2
content_weight = 1e4
try:
    print(f"--- Memuat Gambar ---")
    content_image = load_img(path_gambar_konten)
    style_image = load_img(path_gambar_style)
    print("Gambar berhasil dimuat! Sedang menyiapkan model...")
    image = tf.Variable(content_image)
    style_targets = extractor(style_image)['style']
    content_targets = extractor(content_image)['content']

    def style_content_loss(outputs):
        style_outputs = outputs['style']
        content_outputs = outputs['content']

        style_loss = tf.add_n([tf.reduce_mean((style_outputs[name]-style_targets[name])**2)
                               for name in style_outputs.keys()])
        style_loss *= style_weight / num_style_layers

        content_loss = tf.add_n([tf.reduce_mean((content_outputs[name]-content_targets[name])**2)
                                 for name in content_outputs.keys()])
        content_loss *= content_weight / num_content_layers

        loss = style_loss + content_loss
        return loss

    @tf.function()
    def train_step(image):
        with tf.GradientTape() as tape:
            outputs = extractor(image)
            loss = style_content_loss(outputs)

        grad = tape.gradient(loss, image)
        opt.apply_gradients([(grad, image)])
        image.assign(tf.clip_by_value(image, 0.0, 1.0))

    # Loop Training
    print("\n--- Mulai Proses Neural Style Transfer ---")
    print("Mohon tunggu, proses ini memakan waktu (sekitar 1-5 menit)...")
```

```
        epochs = 10
        steps_per_epoch = 100

        start_time = time.time()
        for n in range(epochs):
            for m in range(steps_per_epoch):
                train_step(image)
            print(f"Epoch {n+1}/{epochs} selesai...")

        end_time = time.time()
        print(f"\nSelesai dalam {end_time - start_time:.1f} detik.")

        nama_file_hasil = "hasil_kombinasi.jpg"
        result_image = tensor_to_image(image)
        result_image.save(nama_file_hasil)
        print(f"\nSUKSES! Gambar hasil telah disimpan sebagai '{nama_file_hasil}'.")
        print("Cek panel Files di sebelah kiri, klik kanan pada file tersebut, lalu pilih 'Download'.")

    except FileNotFoundError as fnf:
        print(fnf)
    except Exception as e:
        print("\n[KESALAHAN LAIN]")
        print(f"Detail error: {e}")
        print("Pastikan format gambar adalah JPG atau PNG yang valid.")
```

```
--- Memuat Gambar ---
Gambar berhasil dimuat! Sedang menyiapkan model...

--- Mulai Proses Neural Style Transfer ---
Mohon tunggu, proses ini memakan waktu (sekitar 1-5 menit)...
Epoch 1/10 selesai...
Epoch 2/10 selesai...
Epoch 3/10 selesai...
Epoch 4/10 selesai...
Epoch 5/10 selesai...
Epoch 6/10 selesai...
Epoch 7/10 selesai...
Epoch 8/10 selesai...
Epoch 9/10 selesai...
Epoch 10/10 selesai...

Selesai dalam 91.7 detik.

SUKSES! Gambar hasil telah disimpan sebagai 'hasil_kombinasi.jpg'.
Cek panel Files di sebelah kiri, klik kanan pada file tersebut, lalu pilih 'Download'.
```

You can install Python libraries in Colab using `pip`. The basic syntax is `!pip install <library_name>`. The exclamation mark `!` at the beginning tells Colab to run the command as a shell command.

```
# Example: Install the 'requests' library
!pip install requests
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requ
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requ
```

If you need to install a specific version of a library, you can specify it like this: `!pip install library_name==1.2.3`

Or if you want to upgrade an already installed library: `!pip install --upgrade library_name`