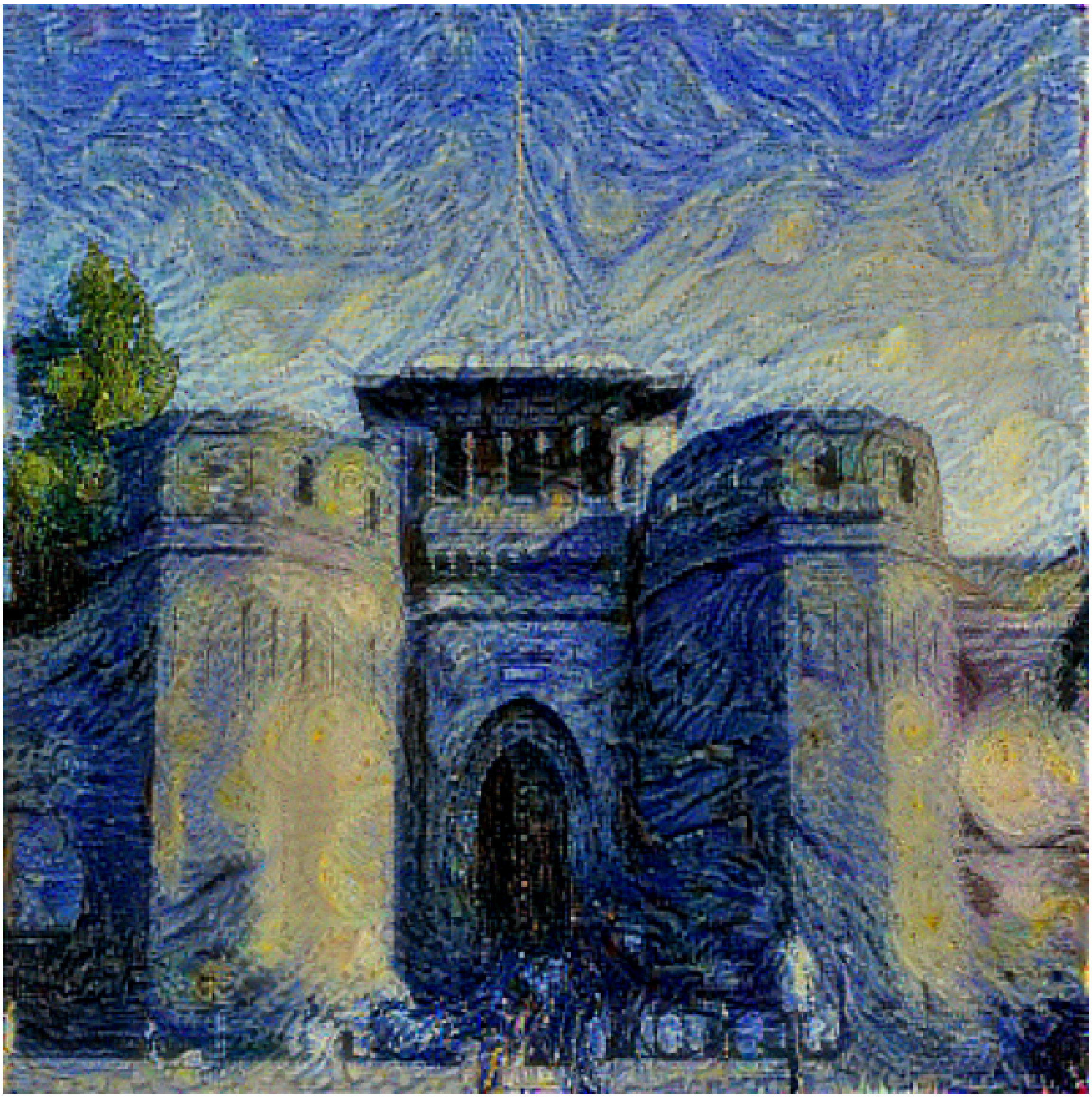
In [1]:

```
import tensorflow as tf
from tensorflow.keras.applications import vgg19
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
import os

# Set paths
content_path = "C:\\Users\\dell\\BE_PRACTICALS\\CL-III\\Images\\content_image.
style_path = "C:\\Users\\dell\\BE_PRACTICALS\\CL-III\\Images\\style_image.jpg"

# Load and preprocess image
def load_and_process_image(path, target_size=(400, 400)):
    img = load_img(path, target_size=target_size)
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)
    return vgg19.preprocess_input(img)

# Deprocess image for display
def deprocess_image(x):
    x = x.reshape((x.shape[1], x.shape[2], 3))
    x[:, :, 0] += 103.939
    x[:, :, 1] += 116.779
    x[:, :, 2] += 123.68
    x = x[:, :, ::-1]  # BGR to RGB
    return np.clip(x / 255.0, 0, 1)

# Load images
content_image = load_and_process_image(content_path)
style_image = load_and_process_image(style_path)

# Define layers
content_layer = 'block5_conv2'
style_layers = [
    'block1_conv1',
    'block2_conv1',
    'block3_conv1',
    'block4_conv1',
    'block5_conv1'
]

# Load VGG19
def get_model():
    vgg = vgg19.VGG19(weights='imagenet', include_top=False)
    vgg.trainable = False
    outputs = [vgg.get_layer(name).output for name in style_layers + [content_
    return Model(inputs=vgg.input, outputs=outputs)

# Compute Gram matrix
def gram_matrix(tensor):
    x = tf.squeeze(tensor)
    x = tf.reshape(x, (-1, x.shape[-1]))
    return tf.matmul(x, x, transpose_a=True)

# Loss function
def compute_loss(model, content_img, style_img, combination_img):
    input_tensor = tf.concat([style_img, content_img, combination_img], axis=0
    features = model(input_tensor)

    style_features = features[:len(style_layers)]
    content_feature = features[-1]

    loss = tf.zeros(shape=())

    # Style loss
    weight_per_style_layer = 1.0 / float(len(style_layers))
    for i in range(len(style_layers)):
        sl = gram_matrix(style_features[i][0])  # style reference
        cl = gram_matrix(style_features[i][2])  # generated
        style_loss = tf.reduce_mean(tf.square(cl - sl))
        loss += weight_per_style_layer * style_loss

    # Content loss
    content_loss = tf.reduce_mean(tf.square(content_feature[2] - content_featu
    loss += content_loss * 1e4

    return loss

# Gradient function
@tf.function()
def compute_grads(cfg):
    with tf.GradientTape() as tape:
        all_loss = compute_loss(**cfg)
    return tape.gradient(all_loss, cfg['combination_img']), all_loss

# Style transfer process
def run_style_transfer(content_path, style_path, iterations=500):
    model = get_model()
    for layer in model.layers:
        layer.trainable = False

    content_img = tf.constant(load_and_process_image(content_path), dtype=tf.f
    style_img = tf.constant(load_and_process_image(style_path), dtype=tf.float
    init_image = tf.Variable(content_img)

    opt = tf.optimizers.Adam(learning_rate=5.0)

    cfg = {
        'model': model,
        'content_img': content_img,
        'style_img': style_img,
        'combination_img': init_image
    }

    best_loss, best_img = float('inf'), None

    for i in range(iterations):
        grads, loss = compute_grads(cfg)
        opt.apply_gradients([(grads, init_image)])
        clipped = tf.clip_by_value(init_image, -128.0, 128.0)
        init_image.assign(clipped)

        if loss < best_loss:
            best_loss = loss
            best_img = init_image.numpy()

        if i % 100 == 0:
            print(f"Iteration {i}, Loss: {loss.numpy():.2f}")

    return best_img
```

```
# Run style transfer
output = run_style_transfer(content_path, style_path, iterations=500)

# Display result
plt.figure(figsize=(10, 8))
plt.imshow(deprocess_image(output))
plt.axis('off')
plt.title("Stylized Image")
plt.show()
```

```
Iteration 0, Loss: 3103515536701521920.00
Iteration 100, Loss: 870474510761984.00
Iteration 200, Loss: 337721969082368.00
Iteration 300, Loss: 199111697498112.00
Iteration 400, Loss: 141400591040512.00
```

Stylized Image



```
In [ ]:
```