Objective: Implement a neural style transfer model to apply artistic styles to photographs.

Deliverable: A Python script or notebook with examples of styled images.

Python Script: Neural Style Transfer

Below is a complete implementation using PyTorch to perform neural style transfer:

Install Required Libraries

# Uncomment and run this command in your terminal if not installed

# pip install torch torchvision matplotlib

Code Implementation

Import torch

Import torch.optim as optim

From torchvision import transforms, models

From PIL import Image

Import matplotlib.pyplot as plt

# Function to load and preprocess images

Def load\_image(img\_path, max\_size=400, shape=None):

Image = Image.open(img\_path).convert(“RGB”)

# Resize the image

If max(image.size) > max\_size:

Size = max\_size

Else:

Size = max(image.size)

If shape:

Size = shape

In\_transform = transforms.Compose([

Transforms.Resize((size, size)),

Transforms.ToTensor(),

Transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))

])

# Add batch dimension

Image = in\_transform(image).unsqueeze(0)

Return image

# Display the image

Def im\_convert(tensor):

Image = tensor.to(“cpu”).clone().detach()

Image = image.numpy().squeeze()

Image = image.transpose(1, 2, 0)

Image = image \* 0.5 + 0.5

Return image.clip(0, 1)

# Load content and style images

Content = load\_image(“content.jpg”) # Replace with path to your content image

Style = load\_image(“style.jpg”, shape=content.shape[-2:]) # Replace with path to your style image

# Use GPU if available

Device = torch.device(“cuda” if torch.cuda.is\_available() else “cpu”)

Content = content.to(device)

Style = style.to(device)

# Load the VGG19 model

Vgg = models.vgg19(pretrained=True).features

For param in vgg.parameters():

Param.requires\_grad\_(False)

Vgg.to(device)

# Define content and style layers

Def get\_features(image, model, layers=None):

If layers is None:

Layers = {

‘0’: ‘conv1\_1’,

‘5’: ‘conv2\_1’,

‘10’: ‘conv3\_1’,

‘19’: ‘conv4\_1’,

‘21’: ‘conv4\_2’, # Content layer

‘28’: ‘conv5\_1’

}

Features = {}

X = image

For name, layer in model.\_modules.items():

X = layer(x)

If name in layers:

Features[layers[name]] = x

Return features

# Gram matrix calculation

Def gram\_matrix(tensor):

\_, d, h, w = tensor.size()

Tensor = tensor.view(d, h \* w)

Gram = torch.mm(tensor, tensor.t())

Return gram

# Get content and style features

Content\_features = get\_features(content, vgg)

Style\_features = get\_features(style, vgg)

# Calculate style gram matrices

Style\_grams = {layer: gram\_matrix(style\_features[layer]) for layer in style\_features}

# Initialize target image

Target = content.clone().requires\_grad\_(True).to(device)

# Define weights and hyperparameters

Style\_weights = {

‘conv1\_1’: 1.0,

‘conv2\_1’: 0.8,

‘conv3\_1’: 0.5,

‘conv4\_1’: 0.3,

‘conv5\_1’: 0.1

}

Content\_weight = 1 # Alpha

Style\_weight = 1e6 # Beta

Optimizer = optim.Adam([target], lr=0.003)

Steps = 2000

# Style transfer loop

For step in range(1, steps+1):

Target\_features = get\_features(target, vgg)

Content\_loss = torch.mean((target\_features[‘conv4\_2’] – content\_features[‘conv4\_2’])\*\*2)

Style\_loss = 0

For layer in style\_weights:

Target\_feature = target\_features[layer]

Target\_gram = gram\_matrix(target\_feature)

\_, d, h, w = target\_feature.shape

Style\_gram = style\_grams[layer]

Layer\_style\_loss = style\_weights[layer] \* torch.mean((target\_gram – style\_gram)\*\*2)

Style\_loss += layer\_style\_loss / (d \* h \* w)

Total\_loss = content\_weight \* content\_loss + style\_weight \* style\_loss

Optimizer.zero\_grad()

Total\_loss.backward()

Optimizer.step()

If step % 500 == 0:

Print(f”Step {step}, Total Loss: {total\_loss.item()}”)

# Display the result

Final\_image = im\_convert(target)

Plt.imshow(final\_image)

Plt.axis(“off”)

Plt.show()

How It Works

1. Content Image: The image you want to stylize.
2. Style Image: The artistic style to apply.
3. VGG19 Model: Extracts content and style features.
4. Optimization: Iteratively adjusts a target image to minimize content and style losses.