```
! pip install torchvision
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-whe
Requirement already satisfied: torchvision in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.7
Requirement already satisfied: torch==1.12.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dis
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/l
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/di
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import math
import copy
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision.models as models
import torchvision.transforms as tranforms
import torch.optim as optim

from PIL import Image

import glob
import matplotlib.image as mpimg
```

```
df = pd.read_csv('/content/drive/MyDrive/artists.csv')
```

```
df.head()
```

	id	name	years	genre	nationality	
0	0	Amedeo Modigliani	1884 - 1920	Expressionism	Italian	Amede
1	1	Vasiliy Kandinskiy	1866 - 1944	Expressionism,Abstractionism	Russian	Wassily Was
2	2	Diego Rivera	1886 - 1957	Social Realism,Muralism	Mexican	Diego

```
df.set_index('id', inplace=True)
```

	3	Claude Monet	1926	Impressionism	French	

```
df.head()
```

id	name	years	genre	nationality	
0	Amedeo Modigliani	1884 - 1920	Expressionism	Italian	Amede
1	Vasiliy Kandinskiy	1866 - 1944	Expressionism,Abstractionism	Russian	Wassily Was
2	Diego Rivera	1886 - 1957	Social Realism,Muralism	Mexican	Diego
		1840 -			Oscar-Claude Mo

```
model= models.vgg19(pretrained = True).features
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:209: UserWarn
  f"The parameter '{pretrained_param}' is deprecated since 0.13 and will be remov
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWarn
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/vgg19-dcbb9e9d.pth" to /root/.c
100%                                  548M/548M [00:31<00:00, 29.5MB/s]
```

```
print(model)
```

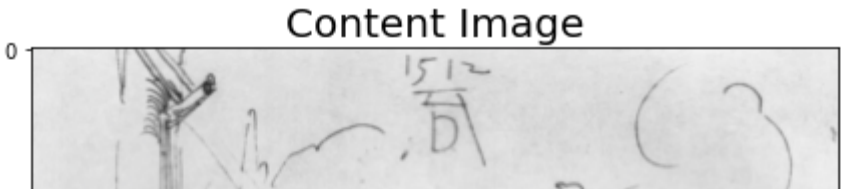
```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): ReLU(inplace=True)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=Fals
    (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): ReLU(inplace=True)
    (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (26): ReLU(inplace=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=Fals
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): ReLU(inplace=True)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=Fals
  )


for param in model.parameters():
  param.requires_grad_(False)


dev = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(dev)

  Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
  ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
  ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
    (17): ReLU(inplace=True)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
  ceil_mode=False)
    (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): ReLU(inplace=True)
    (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (26): ReLU(inplace=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
  ceil_mode=False)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): ReLU(inplace=True)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
  ceil_mode=False)
  )
```

```
from torchvision.ops.poolers import torchvision
def load_img(path, max_size = 400, shape = None):
  image = Image.open(path).convert('RGB')

  if max(image.size) > max_size:
    size = max_size
  else:
    size = max(image.size)


  if shape is not None:
    size = shape


  in_trans = tranforms.Compose([tranforms.Resize(size),
                                tranforms.ToTensor(),
                                tranforms.Normalize((0.485, 0.456, 0.406),
                                                    (0.229, 0.224, 0.225))])
  image = in_trans(image)[:3,:,:].unsqueeze(0)

  return image


from PIL import Image
import glob
image_lst = []
for filename in glob.glob('/content/drive/MyDrive/Dataset/*.jpg'):
```

```
    im = Image.open(filename)
    image_lst.append(im)

data = load_img('/content/drive/MyDrive/Dataset/Albrecht_Dürer_102.jpg').to(dev)
style_data = load_img('/content/drive/MyDrive/Dataset/Amedeo_Modigliani_109.jpg').to(d



def convert(tensor):
  image = tensor.to("cpu").clone().detach()
  image = image.numpy().squeeze()
  image = image.transpose(1,2,0)
  image = image * np.array((0.229, 0.224, 0.225)) + np.array((0.485, 0.456, 0.406))
  image = image.clip(0, 1)

  return image


fig ,( ax1, ax2) = plt.subplots(1, 2, figsize = (20, 10))

ax1.imshow(convert(data))
ax1.set_title("Content Image",fontsize = 20)
ax2.imshow(convert(style_data))
ax2.set_title("Style Image", fontsize = 20)
plt.show()
```



Model

```
def 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',
                  '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
def gram_matrix(tensor):
  _, d, h, w = tensor.size()

  tensor = tensor.view(d, h*w)
  gram = torch.mm(tensor, tensor.t())
  return gram


cont_features = features(data, model)
style_feat = features(style_data, model)

style_grams = {layer: gram_matrix(style_feat[layer])for layer in style_feat}
target = data.clone().requires_grad_(True).to(dev)


# Loss and Weights

style_weights = {'conv1_1': 1.,
                 'conv2_1': 0.75,
                 'conv3_1': 0.2,
                 'conv4_1': 0.2,
                 'conv5_1': 0.2}
```

```
data_weights= 1
style_weight = 1e9

#Total loss

show_every = 400

optimizer = optim.Adam([target], lr = 0.003)
steps = 2000

for ii in range(1, steps+1):
  target_feat = features(target, model)
  cont_loss = torch.mean((target_feat['conv4_2'] - cont_features['conv4_2'])**2)

  style_loss = 0
  for layer in style_weights:
    target_feats = target_feat[layer]
    target_gram = gram_matrix(target_feats)
    _, d, h, w = target_feats.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 = data_weights * cont_loss + style_weight * style_loss

  optimizer.zero_grad()
  total_loss.backward()
  optimizer.step()

  if ii % show_every == 0:
    print('Total Loss: ', total_loss.item())
    plt.imshow(convert(target))
    plt.axis('off')
    plt.grid(False)
    plt.figure(figsize = (7, 7))
    plt.show()
```

```
Total Loss:  91320958976.0
```

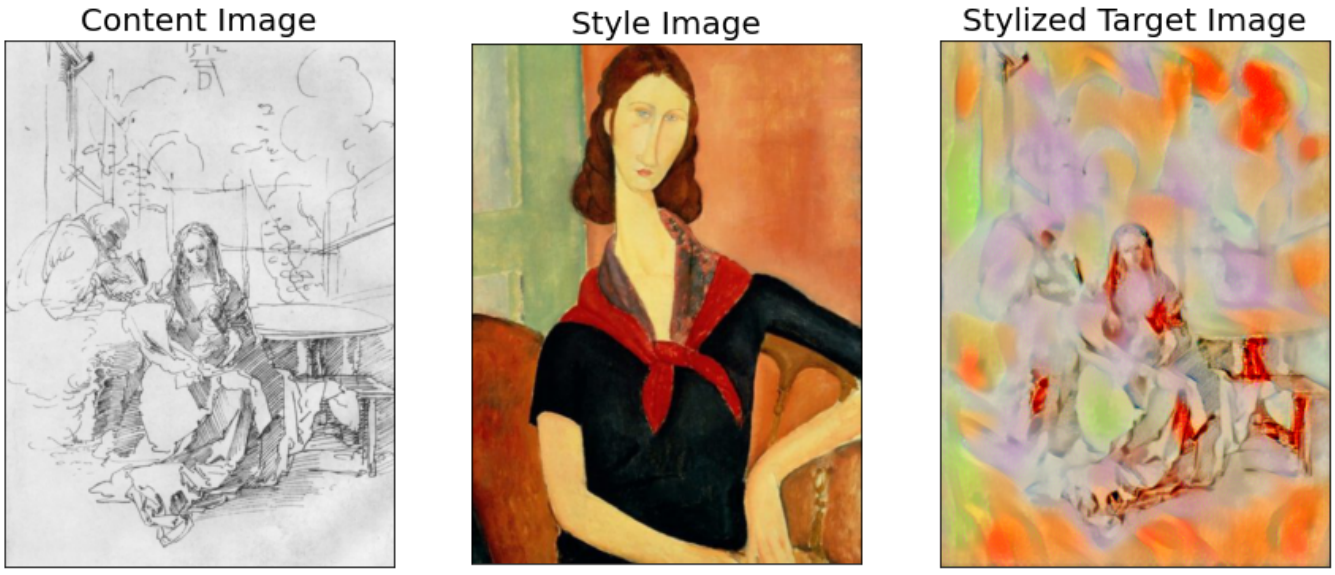
```
<Figure size 504x504 with 0 Axes>
Total Loss:  63734923264.0
```

```
<Figure size 504x504 with 0 Axes>
Total Loss:  39968530432.0
```

```
<Figure size 504x504 with 0 Axes>
Total Loss:  24250875904.0
```

```
<Figure size 504x504 with 0 Axes>
Total Loss:  14284768256.0
```

```
fig, (ax1, ax2,ax3) = plt.subplots(1, 3, figsize=(15, 15))
ax1.imshow(convert(data))
ax1.set_title("Content Image", fontsize = 20)
ax2.imshow(convert(style_data))
ax2.set_title("Style Image", fontsize = 20)
ax3.imshow(convert(target))
ax3.set_title("Stylized Target Image", fontsize = 20)
ax1.grid(False)
ax2.grid(False)
ax3.grid(False)

ax1.set_xticks([])
ax1.set_yticks([])
ax2.set_xticks([])
ax2.set_yticks([])
ax3.set_xticks([])
ax3.set_yticks([])

plt.show()
```



Colab paid products - Cancel contracts here