Indian Institute of Information Technology, Nagpur

Digital Image Processing (ECL-415)

Project Report

Submitted By :

D.Uday Kiran(BT19ECE007)
B.Varun Rao(BT19ECE061)
Chilla Pravardhan(BT19ECE027)
Semester 6
Electronics and Communication Engineering Dept.

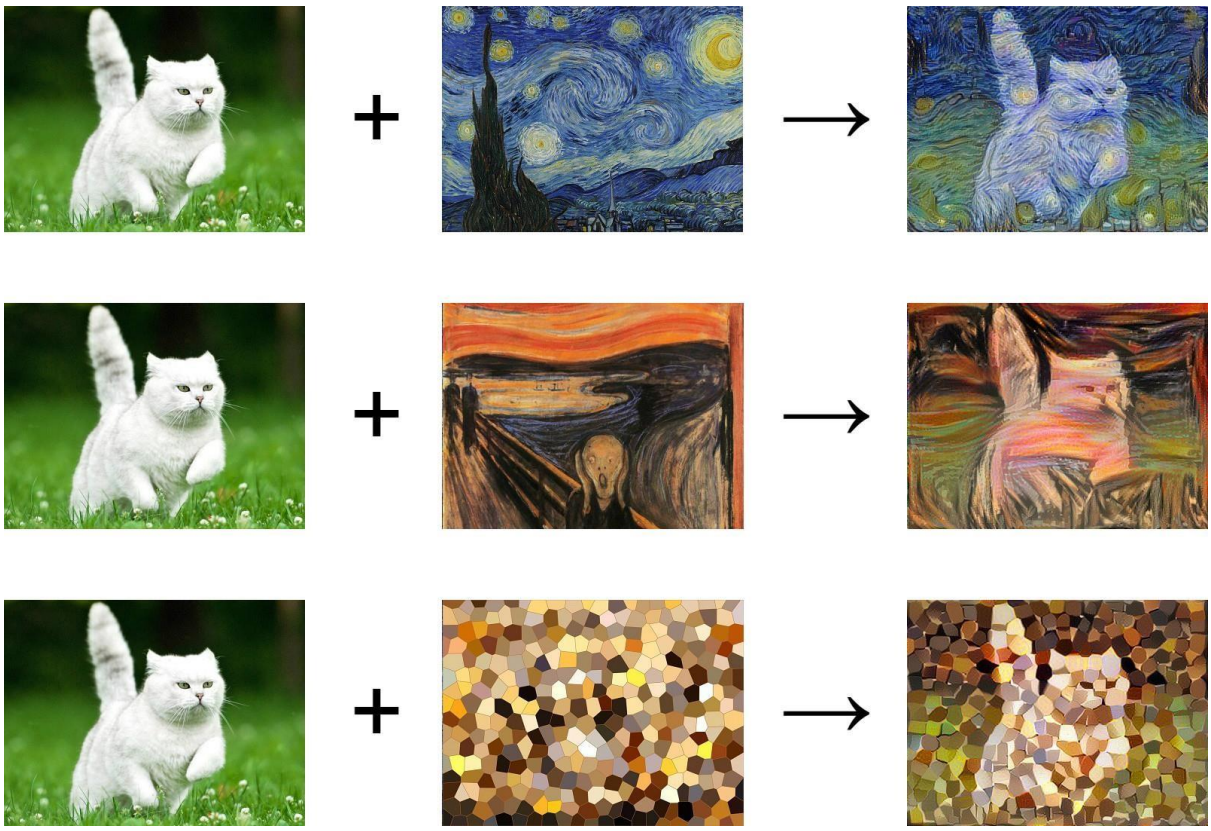
Submitted To :
Dr. Tapan Jain
Course Instructor

Image Styling

The Art of creating Style to any Content

Introduction

Neural Style Transfer is an optimization technique that involves two images - a content image and a style reference image and blends the style from one image to another keeping its content intact.



Environments and technologies used

VS Code

Streamlit framework

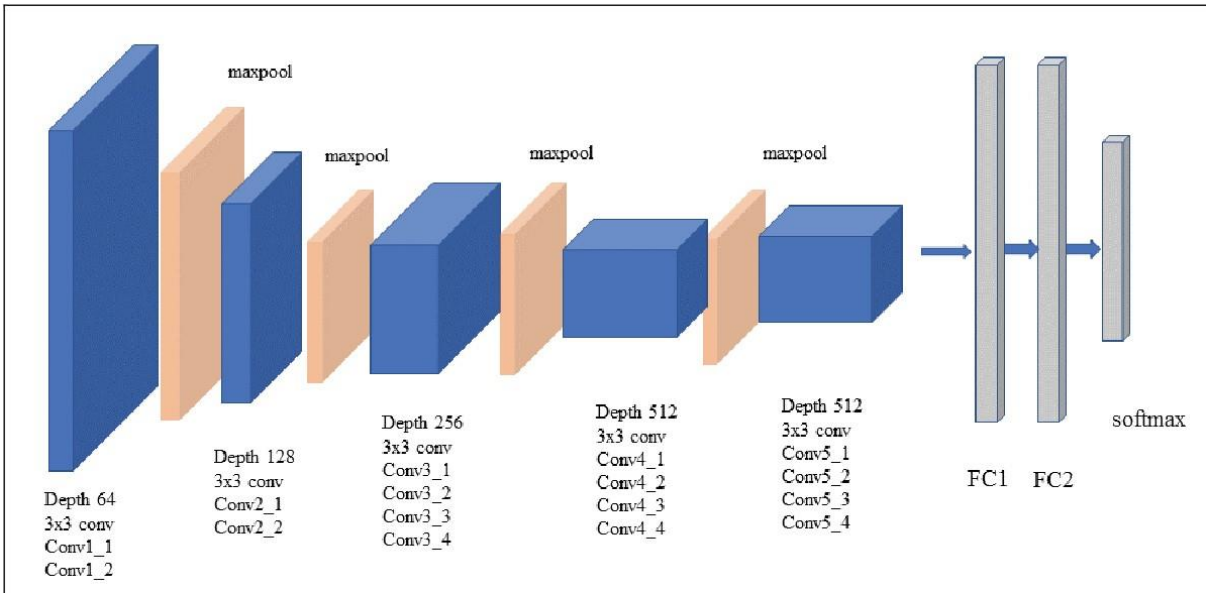
Libraries Used

Torch, PIL, Matplotlib, Numpy

DL model Used

We are using Convolutional neural networks (CNN) and a pre-trained VGG-19 network.

VGG-19 Network



Extraction of content from Image

- ➔ 4_2 Convolutional layer of pre-trained VGG network used as content extractor.
- ➔ Refers to Feature responses in higher layers.
- ➔ Actual content is extracted rather than pixel values.

Extraction of Style

- ➔ Correlations between different filters/Layers used.
- ➔ Texture information is captured rather than global arrangement.
- ➔ Lower layers of the Vgg Network are used.

Features of the web Interface

Content Image (format: png, jpg, jpeg, jfif) can be selected or Uploaded by the user.

Multiple Style Images are available out of which the user can choose one from the selection pane.

And after selecting the content and style images, on clicking the stylize button, the interface outputs the final artistic image.

CODE: main.py

```
from email.mime import image
from fileinput import
filename import imp from
secrets import choice import
streamlit as st from PIL
import Image


import style
 def
main():
     menu = ["Image","About"]     choice =
st.sidebar.selectbox("Menu",menu)
     if choice ==
"Image":
        st.subheader("Image")
 def
load_image(image_file):
    pic = Image.open(image_file)
return pic
 if choice ==
"Image":
    st.subheader("Image")
input_image = st.file_uploader("Upload Content Image",
type=["png","jpg","jpeg"])
 if input_image is not
None:
    file_details = {"filename":input_image.name,
"filetype":input_image.type,"filesize":input_image.size}
st.write(file_details)
st.image(load_image(input_image), width = 250)
```

```
style_name_1=st.sidebar.selectbox(
    'Select Style',
    ('candy', 'mosaic', 'udnie', 'rain_princess')
)


model= "saved_models/" + style_name_1 +".pth"
   output_image = "images/output-images/" + style_name_1 +
".jpg"


clicked=st.button("stylize")
 if
clicked:
    model=style.load_model(model)
style.stylize(model,input_image,output_image)
st.write("### OUTPUT Image:")     image =
Image.open(output_image)     st.image(image,
width=400)
```

style.py

```
import argparse
import os
import sys
import time
import re

import numpy as np import
torch from torch.optim
import Adam
from torch.utils.data import
DataLoader from torchvision import
datasets from torchvision import
transforms import torch.onnx
 import utils from transformer_net import
TransformerNet from vgg import Vgg19
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
 def
load_model(model_path):
with torch.no_grad():
        style_model = TransformerNet()
state_dict = torch.load(model_path)
         # remove saved deprecated running_* keys in InstanceNorm from
the checkpoint         for k in list(state_dict.keys()):             if
re.search(r'in\d+\.running_(mean|var)$', k):
                del state_dict[k]
style_model.load_state_dict(state_dict)
style_model.to(device)
style_model.eval()         return style_model


def stylize(style_model, content_image, output_image):
     content_image =
utils.load_image(content_image)
content_transform = transforms.Compose([
transforms.ToTensor(),
transforms.Lambda(lambda x: x.mul(255))
    ])     content_image =
content_transform(content_image)     content_image =
content_image.unsqueeze(0).to(device)
         with
torch.no_grad():
        output =
style_model(content_image).cpu()
utils.save_image(output_image, output[0])
```

INPUT:

Style image:

stylize

OUTPUT:

CONCLUSION : This technique yields stylized and artistic images with finer and better preserved local details of the content image ,and fewer artifacts introduced from style images