Neural Style Transfer

A PROJECT REPORT

Submitted by

Janvi Somani(RA2011042010076)

Chethan Reddy(RA2011042010093)

Bhuvan Doniparthi(RA2011042010102)

Under the Guidance of

Dr. JEEVA

(Assistant Professor, Department of Data Science and Business Systems)

In partial fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603203

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled "**NEURAL STYLE TRANSFER**" is the bonafide work of "**(Bhuvan Doniparti (RA2011042010102, ChethanReddy(RA2011042010093),Janvi Somani(RA2011042010076)"**who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr.Jeeva

SUPERVISOR
AssistantProfessor

Department ofDataScience and BusinessSystems

Dr. M.Laxmi

HEAD OFDEPARTMENT
Department of DataScience and BusinessSystems

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

OUR WORK DECLARATION

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course:B. TECH

Student Name:SHARMISHTHA BHARTI, ANANYA KRISHNA PREETHA, MANOGNYA GATTEREDDI

Registration Number: RA2011042010020, RA2011042010033, RA2011042010046

Title of Work:CHATBOT

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow
- students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalised in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

Janvi Somani	**Chetan Reddy**	**Doniparthi Bhuvan**
RA2011042010076	**RA2011042010093**	**RA2011042010102**

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice Chancellor (I/C), SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her invaluable support.

We wish to thank **Dr. M. Lakshmi**, Professor & Head, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her valuable suggestions and encouragement throughout the period of the project work.

We are extremely grateful to our Academic Advisor **Dr. E. Sasikala**, Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her great support at all the stages of project work.

We register our immeasurable thanks to our Faculty Advisor, **Dr. S Jeeva** Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. S Jeeva** Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for providing me an opportunity to pursue my project under her mentorship. She provided me the freedom and support to explore the research topics of my interest. Her passion for solving real problems and making a difference in the world has always been inspiring.

We sincerely thank the staff and students of the Data Science and Business Systems, SRM Institute of Science and Technology, for their help during my research.

Finally, we would like to thank my parents, our family members and our friends for their unconditional love, constant support and encouragement.

Janvi Somani(RA2011042010076)

Chethan Reddy(RA2011042010093)

Bhuvan Doniparthi(RA2011042010102)

Abstract

This project explores methods for artistic style transfer based on convolutional neural networks. The core idea proposed by Gatys et. al became very popular and with further research Johnson et. al overcame a major limitation to achieve style transfer in real-time. We implement two approaches which are capable of achieving multiple and mixed style transfer, building on top of Johnson's fast style transfer algorithm.

Project Report

Neural Style Transfer

Introduction

As a technique that can combine both artistic aspects and recognition (content) aspects of images, style transfer has always become an interesting topic for researchers in the field of computer vision. Style transfer is essentially combining the style of one image into the content of another. The problem used to be difficult because it was hard to extract texture information using hand-crafted features, but with the advent of CNNs the problem could be tackled.

Dataset and Features

In our implementation of Gatys' method we use weights of VGG-19 , a 19-layer deep convolutional neural network which has been pre-trained on the ImageNet dataset. Before the algorithm begins, pairs of style and content images are cropped and resized to be of matching aspect ratios and sizes. Before images are fed through VGG-19, they are normalized by subtracting the RGB average of the ImageNet dataset.

In artwork generation and algorithm evaluation, we use a variety of content and style images. As content images, we use personal photos, stock images, as well as a few images in the benchmark dataset NPRgeneral proposed by Mould & Rosin (2016). As style images, we use well known artworks of Van Gogh, Claude Monet, Wassily Kandinsky, Pablo Picasso, etc., all of which are in the public domain.

Methods

We first implement the vanilla version of Gatys' method, following Gatys et al. (2015). This method makes use of the fact that a CNN trained on sufficient labelled data for a specific task can learn "generic feature representations" that generalize to other datasets and computer vision tasks.

We also implement a spatial control extension to Gatys, following. Many artworks and photos contain spatially separable style and content regions within the same image. By splitting the style and content images into segments, and applying the style in each specific segment of the style image to the corresponding segment of the content image, we expect to obtain interesting results and achieve more flexibility in the style transfer procedure.

The low-level image features are not sufficiently explored, they are quickly aggregated with shallow network architectures.

The paintings in WikiArt dataset are of different shapes and sizes. To have uniform input image size, we take 224x224 crop of each painting. We also do random horizontal flip and normalize input images. This randomness add variety to train set and avoids over-fitting. For the validation and test sets, we do centre crop and normalize the images.

The ResNet-18 architecture is known to work well for image recognition tasks. Hence, we started with ResNet-18 network initialized with pretrained network weights, trained on the ImageNet dataset, trained for at least 20 epochs with Adam Optimizer. With increase in network depth, the gradients calculated in upper layers slowly degrade before reaching lower layers, hence the accuracy gets saturated. Residual blocks in ResNets make sure that upstream gradients are propagated to lower layers as well. Softmax classifier with cross entropy loss was used for ResNet-18 architecture as well.

Results

We evaluate our implementations taking both speed and quality into account. However, deciding whether or not a particular style transfer was done well is mostly a subjective process and as such we can only rely on human evaluation.

Figure 10. i. Style-1, ii. Style-2, iii. Mixed style result

Figure 11. i. Style-1, ii. Style-2, iii. Mixed style result

Figure 12. i. Style-1, ii. Style-2, iii. Mixed style result

References

[1] Kaggle wikiart dataset. https://www.kaggle.com/c/painterby-numbers.

[2] Y. Bar, N. Levy, and L. Wolf. Classification of artistic styles using binarized features derived from a deep neural network. In ECCV Workshops, 2014.

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. CoRR, abs/1610.07629, 2(4):5, 2016.

[4] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897, 2016. [5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, pages 2414–2423. IEEE, 2016.

CODE

```
importtensorflowastf
fromtensorflowimportkeras
fromtensorflow.keras.applicationsimport vgg19
importnumpyas np
from PIL import Image

base_image_path=
keras.utils.get_file("orange.","https://images.unsplash.com/photo-
1536632901336-0dbfe885db89?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTY3fHxjcmVhdGl2ZXxlbnwwfHwwfHw%3D&auto=
format&fit=crop&w=500&q=60")
style_reference_image_path=
keras.utils.get_file("bg.jpg","https://images.unsplash.com/photo-
1513151233558-d860c5398176?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8MTczfHxjcmVhdGl2ZXxlbnwwfHwwfHw%3D&auto=
format&fit=crop&w=500&q=60")
result_prefix= "uwuwu_gen"

# Weights of the different loss components
total_variation_weight= 1e-6
style_weight= 1e-6
content_weight= 2.5e-8

# Dimensions of the generated picture.
width, height =keras.preprocessing.image.load_img(base_image_path).size
img_nrows= 400
img_ncols=int(width *img_nrows/ height)

defpreprocess_image(image_path):
```

```
# Util function to open, resize and format pictures into appropriate
tensors
img=keras.preprocessing.image.load_img(
image_path, target_size=(img_nrows, img_ncols)
    )
img=keras.preprocessing.image.img_to_array(img)
img=np.expand_dims(img, axis=0)
img= vgg19.preprocess_input(img)
returntf.convert_to_tensor(img)

defdeprocess_image(x):
# Util function to convert a tensor into a valid image
    x =x.reshape((img_nrows, img_ncols, 3))
# Remove zero-center by mean pixel
    x[:, :, 0] += 103.939
    x[:, :, 1] += 116.779
    x[:, :, 2] += 123.68
# 'BGR'->'RGB'
    x = x[:, :, ::-1]
    x =np.clip(x, 0, 255).astype("uint8")
return x

defgram_matrix(x):
        # The gram matrix of an image tensor (feature-wise outer product)
    x =tf.transpose(x, (2, 0, 1))
    features =tf.reshape(x, (tf.shape(x)[0], -1))
    gram =tf.matmul(features, tf.transpose(features))
defstyle_loss(style, combination):
    S =gram_matrix(style)
    C =gram_matrix(combination)
    channels = 3
    size =img_nrows*img_ncols
returntf.reduce_sum(tf.square(S - C)) / (4.0 * (channels ** 2) * (size **
2))

defcontent_loss(base, combination):
returntf.reduce_sum(tf.square(combination - base))

deftotal_variation_loss(x):
    a =tf.square(x[:, : img_nrows- 1, : img_ncols- 1, :] - x[:, 1:, :
img_ncols- 1, :])
    b =tf.square(x[:, : img_nrows- 1, : img_ncols- 1, :] - x[:, :
img_nrows- 1, 1:, :])
returntf.reduce_sum(tf.pow(a + b, 1.25))

# Build a VGG19 model loaded with pre-trained ImageNet weights
model = vgg19.VGG19(weights="imagenet", include_top=False)

# Get the symbolic outputs of each "key" layer (we gave them unique names).
outputs_dict=dict([(layer.name, layer.output) for layer inmodel.layers])

# Set up a model that returns the activation values for every layer in
# VGG19 (as a dict).
feature_extractor=keras.Model(inputs=model.inputs, outputs=outputs_dict)

# List of layers to use for the style loss.
style_layer_names= [
    "block1_conv1",
    "block2_conv1",
    "block3_conv1",
    "block4_conv1",
    "block5_conv1",
```

```
]
# The layer to use for the content loss.
content_layer_name= "block5_conv2"
defcompute_loss(combination_image, base_image, style_reference_image):
input_tensor=tf.concat(
        [base_image, style_reference_image, combination_image], axis=0
    )
    features =feature_extractor(input_tensor)

# Initialize the loss
    loss =tf.zeros(shape=())

# Add content loss
layer_features= features[content_layer_name]
base_image_features=layer_features[0, :, :, :]
combination_features=layer_features[2, :, :, :]
    loss = loss +content_weight*content_loss(
base_image_features, combination_features
    )
# Add style loss
forlayer_nameinstyle_layer_names:
layer_features= features[layer_name]
style_reference_features=layer_features[1, :, :, :]
combination_features=layer_features[2, :, :, :]
sl=style_loss(style_reference_features, combination_features)
        loss += (style_weight/len(style_layer_names)) *sl

# Add total variation loss
    loss +=total_variation_weight*total_variation_loss(combination_image)
return loss
@tf.function
defcompute_loss_and_grads(combination_image,                    base_image,
style_reference_image):
withtf.GradientTape() as tape:
        loss          =compute_loss(combination_image,          base_image,
style_reference_image)
    grads =tape.gradient(loss, combination_image)
return loss, grads
optimizer =keras.optimizers.SGD(
keras.optimizers.schedules.ExponentialDecay(
initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

base_image=preprocess_image(base_image_path)
style_reference_image=preprocess_image(style_reference_image_path)
combination_image=tf.Variable(preprocess_image(base_image_path))

iterations = 1000
foriinrange(1, iterations + 1):
    loss, grads =compute_loss_and_grads(
combination_image, base_image, style_reference_image
    )
optimizer.apply_gradients([(grads, combination_image)])
ifi% 100 == 0:
```

```
print("Iteration %d: loss=%.2f" % (i, loss))
img=deprocess_image(combination_image.numpy())
fname=result_prefix+ "_at_iteration_%d.png" %i
keras.preprocessing.image.save_img(fname, img)
```