Deriving an algorithmic artist

georges.tod@outlook.com

May 2020

1 Introduction

In a recent book[1], the question of whether Artificial Intelligence (AI) can produce authentic artists is raised. If we follow Maggie Boden's definition, art should be a creation which is "new, surprising and meaningful". Can mere lines of code produce something alike?
There are two aspects we focus on: first *which technologies can help* creating new paintings from past creations. Secondly, we question what could be the *impact of social networks* as a mean to connect an audience with the creations of an algorithm. Creating something *meaningful* as required per Boden's definition is the hardest part of the work and will remain an open question. Nevertheless, an *algorithmic artist* is prototyped see [2]: `https://twitter.com/IaQuinet`. The name is inspired from the Belgian painter *Mig Quinet* (1906-2001).

2 Some techonological aspects

Producing new images based on images observed in the past is problem that is being tackled in the machine learning community by *deep generative models*. Recent results have shown Generative Adversarial Networks (GANs) can impressively produce new images and in particular (unfortunately) deep fakes. A second branch in machine learning useful for artistic creation is called Neural Style Transfer. In this case, it is about trying to transfer the style of a known painter to a photograph. The next paragraph quickly describes both methods. The last subsection is about the interest of using a social network for algorithmic creation using application programming interfaces (APIs).

Generative Adversarial Networks (GANs) A couple of networks compete one against each other. The generator turns noise into an imitation of the data to try to trick the discriminator. The discriminator tries to identify real

[1] Bersini H., *L'intelligence artificielle peut-elle engendrer des artistes authentiques ?*, Academie Royale de Belgique, 2020

[2] The computations behind imply the use of a GPU - to save resources the prototype is not online at all times.

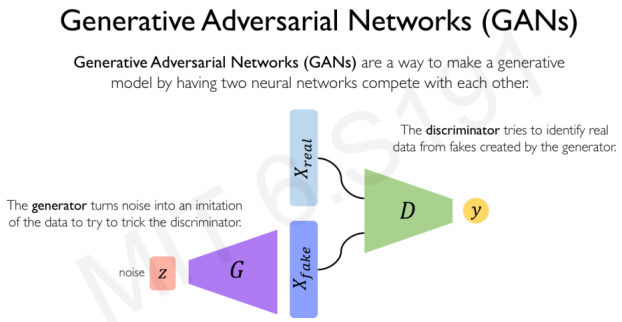


Figure 1: GAN architecture. The generator and the discriminator are the two blocks of an autoencoder: mirrored architectures. - from *MIT online course 6.S191*

data from fakes created by the generator. Early work started using deep neural networks and the architectures as per figure 1. On figure 2, it appears the algorithm is capable of producing hand written digits, without having a hand. Recent works [3] propose to add convolutional layers as well to learn features automatically. **Limitations:** it requires quite some art and a large dataset to train such networks. The classic example that shows hand digits requires 60 000 examples to get reasonable results.

Neural Style Transfer is a lighter approach to creation in terms of data and computational resources - it is as well less impressive. Let u, v and x be the content, style and generated images. The total loss function to be minimized is defined in [4] by,

$$L_{total}(u, v, x) = \alpha \cdot L_{content}(u, x) + \beta \cdot L_{style}(v, x) \tag{1}$$

where $L_{content}(u, x)$ is the squared-loss between feature representations of u and x and L_{style} captures the mean squared distance between intra-image correlations of v and x. On figure 3, an example of neural style transfer is depicted and shows the capability to act at several scales of the image using a convolutional neural network, see figure 4.

Connecting to a social network In a context where a society is confined due to a pandemia for instance, the physical space is more restricted to human beings. As a result they will occupy more virtual spaces such as social networks. And since social networks are merely webpages and code, virtual spaces are very well defined and constrained. As a result, it will be easier for algorithms to pretend to be humans in such digital spaces rather than in physical ones.

[3] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015).

[4] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015)

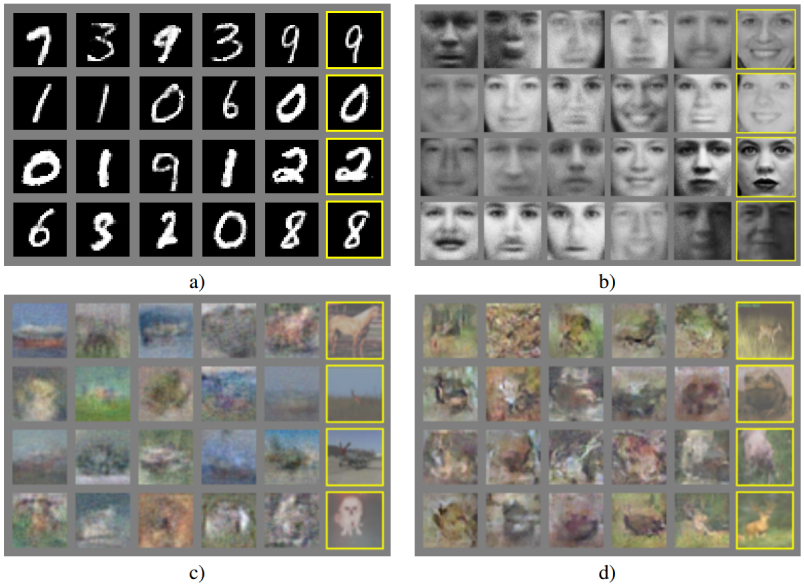


Figure 2: Generated images from 4 datasets using GANs. The images in the yellow squares are the closest image from the training dataset, the rest are new images generated by the GAN, i.e. absent from the training dataset, from *Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.*

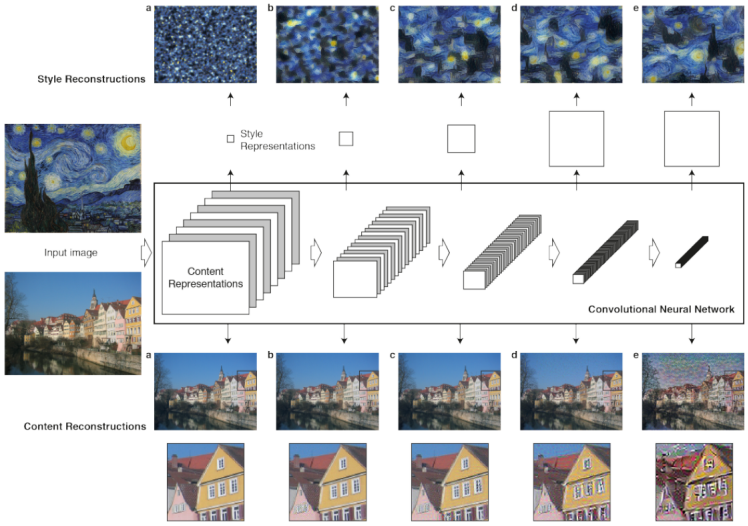


Figure 3: Visualizing the scale impact of neural style transfer on a photograph at different layers from a Convolutional Neural Network, from *Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015)*

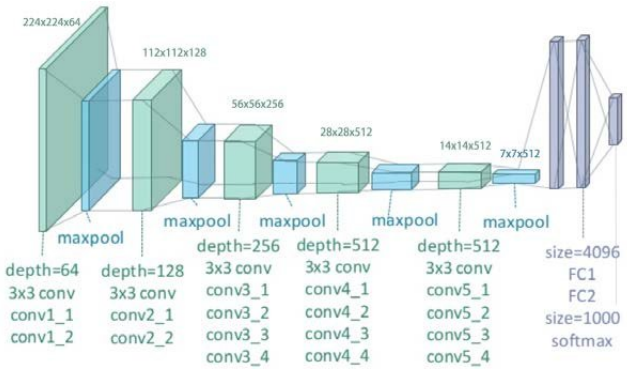


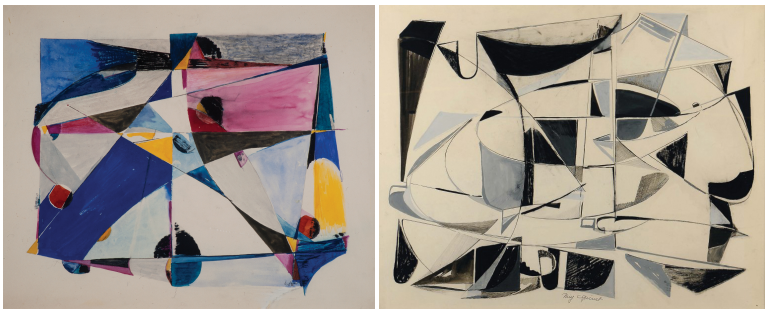
Figure 4: The convolutional neural network used for Neural Style Transfer here is the VGG-19 network: 16 convolution layers for feature extraction and 3 fully connected to build nonlinear relationships between features, from *Zheng, Yufeng, Clifford Yang, and Alex Merkulov. "Breast cancer screening using convolutional neural network and follow-up digital mammography." Computational Imaging, International Society for Optics and Photonics, 2018.*

In order to connect to Twitter we use the API fom Python Twitter Tools [5]. Twitter limits the rate of queries one can make to its network. Some dynamics need to be taken into account while coding, not to be excluded or banned from accessing it. As per Twitter documentation, 15 requests can be made every 15 minutes.

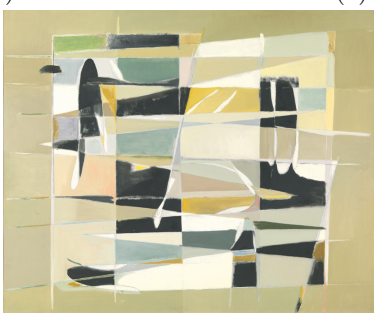
3 Implementation of a prototype

Mig Quinet is a Belgian painter (1906-2001) very active from 1920's till the end of the 90's, see `http://www.migquinet.be/fr/oeuvres`. For the neural style transfer, we choose some of her abstract paintings, see figure 5.
On figure 6, the workflow of our algorithmic artist is depicted and commented.
One can visit the result of the implementation at `https://twitter.com/IaQuinet`.
The code is shared at the end of the report.
DISCLAIMER: the code for the neural style transfer (only) is not mine and comes from *the TensorFlow tutorial*.

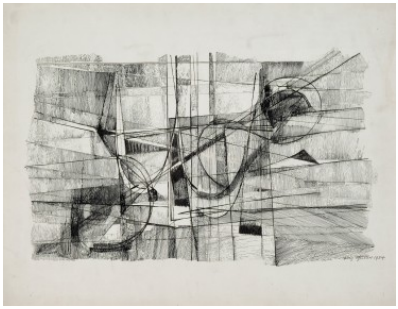
[5] `https://mike.verdone.ca/twitter`



(a) (b)



(c)



(d) (e)

(f)

Figure 5: Mig Quinet's paintings used in the neural style transfer. Paintings will be *randomly* picked - (a) 1946 Géométrie enluminée, (b) 1947 Composition abstraite, (c) 1950 Les oiseaux morts (d) 1952 Angles vifs (e) 1954 Circuit de l'Albertine (f) 1961 Vivace été

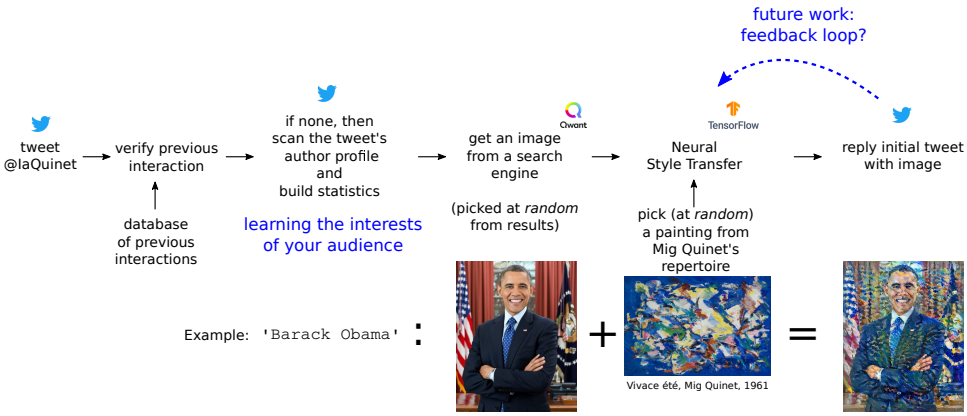


Figure 6: **Workflow of the proposed algorithmic artist.** Here the social network involded is Twitter but any other could be used. In the example depicted, Barack Obama appeared as the entity with the highest number of followers from the person who originally posted the tweet. The neural style transfer step takes as input both Obama's photograph and a Quinet's painting. On a *local* machine a GPU performs the computations ($\sim$ 1000 iterations in $\sim$ 40 to 70 seconds). Finally the synthetized image is attached to a tweet in reply to the original tweet. Once online all the steps are performed *autonomously* by the algorithmic artist. A feedback loop based on the reaction of the audience to the created image could be included in order to amplify the impact on the social network: for instance we could adjust α and β from equation 1 by trying to interpret the satisftication of a human reply to the tweet.

4 Conclusions

To claim that one is an artist, one needs to be considered as such by human beings. When it comes to algorithms, trying to convince human beings they are artists is easier in a digital space such as a social network rather than in a physical space. At the heart of the interaction, application programming interfaces allow to get and post information on social network pages such as Twitter. In theory, an image posted in such a network can gain a large amount of attraction from real human beings.

The prototyped algorithmic artist makes use of Twitter's API, Qwant search engine, tensorflow and performs computations on a GPU to propose new images (using neural style transfer) based on the interets of the audience - *all autonomously*.

Current trends in AI

An AI powered artist - @IaQuinet ¶

by Georges Tod

In [1]:

```
import requests
import random
from PIL import Image
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import IPython.display as display
import PIL.Image
import time
import functools
import os

from twitter import *

import tensorflow as tf
tf.config.experimental.set_memory_growth(tf.config.experimental.list_physical_de
vices('GPU')[0],True)
```

initialization

In [2]:

```
#  (to be done once or to reset robot)
#np.save('answers_list',[],allow_pickle=True)
#np.save('users_answers_list',[],allow_pickle=True)

# to recover from bugs while searching images in Qwant
np.save('log',-1,allow_pickle=True)
```

functions

In [3]:

```
def req_images(query):

    # this functions generates urls of images according to query

    r = requests.get("https://api.qwant.com/api/search/images",
        params={
            'count': 15,
            'q': query,
            't': 'images',
            'imagetype':'photo',
            'safesearch': 1,
            'uiv':20
        },
        headers={
            'User-Agent': 'fff'
        }
    )

    response = r.json().get('data').get('result').get('items')
    urls = [r.get('media') for r in response]

    #plt.figure(figsize=(30,50))
    count=1
    list_valid_urls = []
    for i in urls:

        try:
            response = requests.get(i)
            #img = Image.open(BytesIO(response.content))
            #plt.subplot(10,5,count)
            #plt.imshow(img)
            list_valid_urls.append(i)

            count+=1
        except:
            pass
    return list_valid_urls



def tensor_to_image(tensor):
  tensor = tensor*255
  tensor = np.array(tensor, dtype=np.uint8)
  if np.ndim(tensor)>3:
    assert tensor.shape[0] == 1
    tensor = tensor[0]
  return PIL.Image.fromarray(tensor)

def clip_0_1(image):
    # Since this is a float image, define a function to keep the pixel values be
tween 0 and 1
    return tf.clip_by_value(image, clip_value_min=0.0, clip_value_max=1.0)

def vgg_layers(layer_names):
  """ Creates a vgg model that returns a list of intermediate output values."""
  # Load our model. Load pretrained VGG, trained on imagenet data
  vgg = tf.keras.applications.VGG19(include_top=False, weights='imagenet')
  vgg.trainable = False
```

```
  outputs = [vgg.get_layer(name).output for name in layer_names]

  model = tf.keras.Model([vgg.input], outputs)
  return model

def gram_matrix(input_tensor):
    ## Calculate style
    # The content of an image is represented by the values of the intermediate f
eature maps.
    # It turns out, the style of an image can be described by the means and corr
elations across the different feature maps.
    # Calculate a Gram matrix that includes this information by taking the outer
product of the feature vector with itself at each location,
    # and averaging that outer product over all locations. This Gram matrix can
 be calcualted for a particular layer as:
    # $$G^l_{cd} = \frac{\sum_{ij} F^l_{ijc}(x)F^l_{ijd}(x)}{IJ}$$
    # This can be implemented concisely using the `tf.linalg.einsum` function:
    result = tf.linalg.einsum('bijc,bijd->bcd', input_tensor, input_tensor)
    input_shape = tf.shape(input_tensor)
    num_locations = tf.cast(input_shape[1]*input_shape[2], tf.float32)
    return result/(num_locations)

class StyleContentModel(tf.keras.models.Model):
    ## Extract style and content
    # Build a model that returns the style and content tensors
    def __init__(self, style_layers, content_layers):
        super(StyleContentModel, self).__init__()
        self.vgg =  vgg_layers(style_layers + content_layers)
        self.style_layers = style_layers
        self.content_layers = content_layers
        self.num_style_layers = len(style_layers)
        self.vgg.trainable = False

    def call(self, inputs):
        "Expects float input in [0,1]"
        inputs = inputs*255.0
        preprocessed_input = tf.keras.applications.vgg19.preprocess_input(inputs
)
        outputs = self.vgg(preprocessed_input)
        style_outputs, content_outputs = (outputs[:self.num_style_layers],
                                          outputs[self.num_style_layers:])

        style_outputs = [gram_matrix(style_output)
                         for style_output in style_outputs]

        content_dict = {content_name:value
                        for content_name, value
                        in zip(self.content_layers, content_outputs)}

        style_dict = {style_name:value
                      for style_name, value
                      in zip(self.style_layers, style_outputs)}

        return {'content':content_dict, 'style':style_dict}

def load_img(path_to_img):
    # Define a function to load an image and limit its maximum dimension to 512
 pixels.
    max_dim = 512
    img = tf.io.read_file(path_to_img)
    img = tf.image.decode_image(img, channels=3)
```

```
    img = tf.image.convert_image_dtype(img, tf.float32)

    shape = tf.cast(tf.shape(img)[:-1], tf.float32)
    long_dim = max(shape)
    scale = max_dim / long_dim

    new_shape = tf.cast(shape * scale, tf.int32)

    img = tf.image.resize(img, new_shape)
    img = img[tf.newaxis, :]
    return img

def imshow(image, title=None):
    if len(image.shape) > 3:
        image = tf.squeeze(image, axis=0)

    plt.imshow(image)
    if title:
        plt.title(title)

def style_content_loss(outputs):
    style_outputs = outputs['style']
    content_outputs = outputs['content']
    style_loss = tf.add_n([tf.reduce_mean((style_outputs[name]-style_targets[nam
e])**2)
                           for name in style_outputs.keys()])
    style_loss *= style_weight / num_style_layers

    content_loss = tf.add_n([tf.reduce_mean((content_outputs[name]-content_targe
ts[name])**2)
                             for name in content_outputs.keys()])
    content_loss *= content_weight / num_content_layers
    loss = style_loss + content_loss
    return loss


# Load a VGG19
vgg = tf.keras.applications.VGG19(include_top=False, weights='imagenet')

# Choose intermediate layers from the network to represent the style and content
of the image
content_layers = ['block5_conv2']  # Content layer where will pull our feature m
aps
style_layers = ['block1_conv1',
                'block2_conv1',
                'block3_conv1',
                'block4_conv1',
                'block5_conv1']

num_content_layers = len(content_layers)
num_style_layers = len(style_layers)
```

twitter authentication

In [4]:

```
h = np.load('authen.npy',allow_pickle=True) # to mask secrets...
t = Twitter(auth=OAuth(h[0], h[1], h[2], h[3]))
```

Start robot online

In [5]:

```
life_is_too_short = 3 # time that I will be online in minutes



# twitter API dynamic limitations: we limit our artist to 1 API request per minu
te!
lag = 60

toc0 = lag
toc1 = lag



for yolo in range(life_is_too_short):
    print(yolo)

    # API dynamics
    rest0 = lag-toc0

    if rest0 > 0:
        print('I am going too fast, I will wait a bit!')
        time.sleep(rest0)
    else:
        print('I am late, I can go!')

    tic0 = time.time()
    # checking what is new under the hood
    y = t.statuses.mentions_timeline()


    # scanning for mentions
    mentions_list = []

    for i in y:
        if 'user' in i:
            print(i['id'],i['user']['name'],i['user']['screen_name'],i['user'][
'id'],i['in_reply_to_status_id'])
            mentions_list.append(i['id'])
        else:
            print('no')


    answers_list = np.load('answers_list.npy',allow_pickle= True)

    # scanning for unanswered
    needs_to_be = []
    for i in mentions_list:
        if i in answers_list:
            print('I already produced art for the author of this tweet!')
        else:
            print('I might be missing this person? I need to do more checks.')
            needs_to_be.append(i)

    l_users_to_answered = []
    # scanning for the user ane of potentially unanswered tweets
    for k in needs_to_be:
        for i in y:
            if i['id']== k:
```

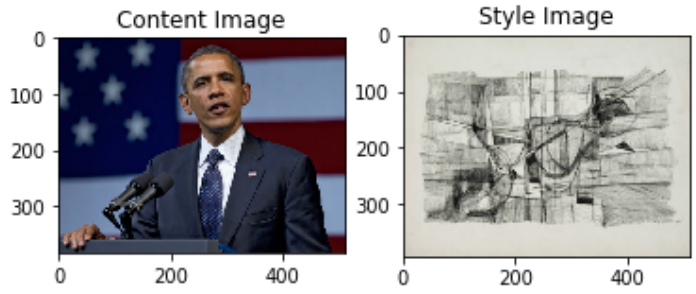
```
                print(i['user']['id'],i['user']['screen_name'],i['text'])
                l_users_to_answered.append(i['user']['id'])

    # filtering tweet to answer
    users_answers_list = np.load('users_answers_list.npy',allow_pickle= True)

    tweet_user_to_answer = []
    for k in set(l_users_to_answered):
        count = 1
        for i in y:
            if 'user' in i:
                if i['user']['id']== k and count ==1 and not i['user']['id'] in
users_answers_list:
                    tweet_user_to_answer.append([i['id'],i['user']['id'],i['use
r']['screen_name']])
                    count += 1
    tweet_user_to_answer



    ibug = np.load('log.npy',allow_pickle=True)

    tweet_user_to_answerB = tweet_user_to_answer[ibug+1:]

    # this is going to happen per tweet
    count=0
    for j in tweet_user_to_answerB:
        print(j)
        rest = lag-toc1

        if rest > 0:
            print('I am going too fast, I will wait a bit!')
            time.sleep(rest)
            tic1 = time.time()

        else:
            print('I am late, I can go!')
            tic1 = time.time()

        c = t.friends.list(screen_name=j[2])


        # extracting info from account
        try:
            # try to find an image linked to interests of person, otherwise: Dar
th Vader!
            max_f = 1
            for i in c['users']:
                if i['followers_count'] > max_f:
                    print(i['name'])
                    max_f = i['followers_count']
                    id_max_f = i['name']



            # finding an image with some randomness
            list_images = req_images(id_max_f)
            index_image = random.randint(1,len(list_images))
            link_IM = list_images[index_image-1]
        except:
            link_IM = 'https://upload.wikimedia.org/wikipedia/commons/thumb/1/1
```

```
3/Darth_vader_hot_air_balloon_1.jpg/470px-Darth_vader_hot_air_balloon_1.jpg'
            id_max_f = 'Darth Vader'

        url_content = link_IM
        filename_content = link_IM[link_IM.rfind('/')+1:]

        print(url_content)
        response = requests.get(url_content, stream = True)


        # Neural style transfer needs to happen
        # this piece of code comes from a Tensorflow tutorial, it is not mine !

        ############### INPUTS ###############
        # loading content
        content_path = tf.keras.utils.get_file(filename_content,url_content)

        # loading a painting as style from amazing Mig Quinet
        basepath = './ref_paintings'
        dirList=[]
        for fname in os.listdir(basepath):
            path = os.path.join(basepath, fname)
            if os.path.isdir(path):
                # skip directories
                continue
            else:
                dirList.append(fname)
        nFiles = len(dirList)
        filename_style = dirList[random.randint(0,nFiles-1)]
        style_path = os.path.join('./ref_paintings/',filename_style)

        # load them
        content_image = load_img(content_path)
        style_image = load_img(style_path)

        # plotting
        plt.subplot(1, 2, 1)
        imshow(content_image, 'Content Image')

        plt.subplot(1, 2, 2)
        imshow(style_image, 'Style Image')

        plt.show()



        ############### OPTIMIZER ###############

        # loss function parameters
        style_weight=1e-2
        content_weight=1e4

        # some optimization options
        epochs = 10
        steps_per_epoch = 100
        opt = tf.optimizers.Adam(learning_rate=0.02, beta_1=0.99, epsilon=1e-1)


        ############### COMPUTATION ###############
        # Create the model
        style_extractor = vgg_layers(style_layers)
```

```
        style_outputs = style_extractor(style_image*255)

        # When called on an image, this model returns the gram matrix (style) of
the `style_layers` and content of the `content_layers`:
        extractor = StyleContentModel(style_layers, content_layers)
        results = extractor(tf.constant(content_image))
        style_results = results['style']

        # With this style and content extractor, you can now implement the style
transfer algorithm.
        style_targets = extractor(style_image)['style']
        content_targets = extractor(content_image)['content']

        # Define a `tf.Variable` to contain the image to optimize
        image = tf.Variable(content_image)

        # Use `tf.GradientTape` to update the image.
        @tf.function()
        def train_step(image):
            with tf.GradientTape() as tape:
                outputs = extractor(image)
                loss = style_content_loss(outputs)

            grad = tape.gradient(loss, image)
            opt.apply_gradients([(grad, image)])
            image.assign(clip_0_1(image))

        @tf.function()
        def train_step(image):
            with tf.GradientTape() as tape:
                outputs = extractor(image)
                loss = style_content_loss(outputs)

            grad = tape.gradient(loss, image)
            opt.apply_gradients([(grad, image)])
            image.assign(clip_0_1(image))

        # Compute
        import time
        start = time.time()
        step = 0
        for n in range(epochs):
            for m in range(steps_per_epoch):
                step += 1
                train_step(image)
                print(".", end='')
            #display.clear_output(wait=True)

            print("Train step: {}".format(step))

        display.display(tensor_to_image(image))
        end = time.time()
        print("Total time: {:.1f}".format(end-start))


        file_name = 'stylized-image.png'
        tensor_to_image(image).save(file_name)

        ##############################################################################
####
```

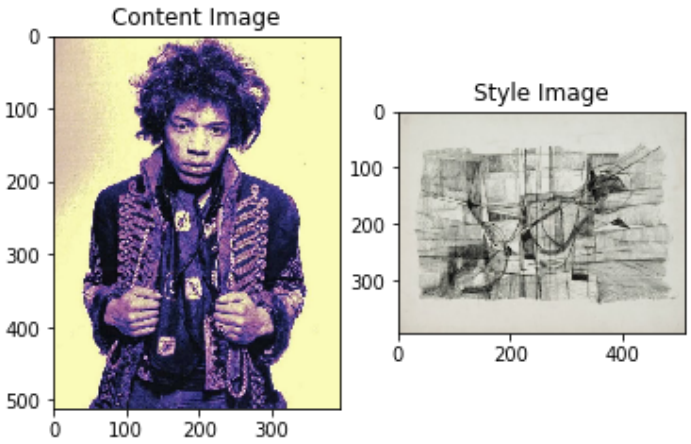
```
        # as an output we need to produce an image, perhaps reload it from disk
        with open("stylized-image.png", "rb") as imagefile:
            imagedata = imagefile.read()




        params = {"media[]": imagedata, "status": "Merci de t'intéresser à moi!
 Je ne suis que des lignes de codes mais je pense que tu portes un certain intér
êt pour "+id_max_f+", je me trompe? @"+j[2], "in_reply_to_status_id ": j[0] }
        try:
            t.statuses.update_with_media(**params)
            print('I have written on twitter!')
            answers_list = np.append(answers_list,j[0])
            users_answers_list = np.append(users_answers_list,j[1])
            np.save('answers_list',answers_list, allow_pickle= True)
            np.save('users_answers_list',users_answers_list, allow_pickle= True)


            # logging success in case of bug
            np.save('log',count,allow_pickle=True)

        except:
            print('I couldnt write on twitter!')


        count+=1

        toc1 = time.time()-tic1
        print('------------------------------------------')

    toc0 = time.time()-tic0



    print('Im done with this round of tweets!')
    print('##############################################')

print('I have worked, will rest now.')
```

```
0
I am late, I can go!
1263796322727014401 IA Quinet IaQuinet 749616853698355201 None
1261931485315072000 Greg Greg53838158 960981999711326208 12617038485
50371328
1261721612010782720 Stéphane Lefèvre cagou007 2883285551 None
1261721447321554945 Stéphane Lefèvre cagou007 2883285551 12617038485
50371328
I might be missing this person? I need to do more checks.
I might be missing this person? I need to do more checks.
I might be missing this person? I need to do more checks.
I might be missing this person? I need to do more checks.
749616853698355201 IaQuinet Can I produce art for myself? @IaQuinet
960981999711326208 Greg53838158 @mIAtisse Hello petit robot ! Fais m
oi un tableau!
2883285551 cagou007 Fais moi un tableau @mIAtisse !
2883285551 cagou007 @mIAtisse Salut l'artiste !
[1261931485315072000, 960981999711326208, 'Greg53838158']
I am late, I can go!
France Inter
Libération
Agence France-Presse
Barack Obama
https://cbsnews1.cbsistatic.com/hub/i/r/2012/06/06/a6be8bb1-a644-11e
2-a3f0-029118418759/thumbnail/620x465/5b1e7136fa8e2fa352be65e06d0fbd
99/barack_obama_AP120604060539.jpg#
```




```
.................................................................
Train step: 100
.................................................................
Train step: 200
.................................................................
Train step: 300
.................................................................
Train step: 400
.................................................................
Train step: 500
.................................................................
Train step: 600
.................................................................
Train step: 700
.................................................................
Train step: 800
.................................................................
Train step: 900
.................................................................
Train step: 1000
```

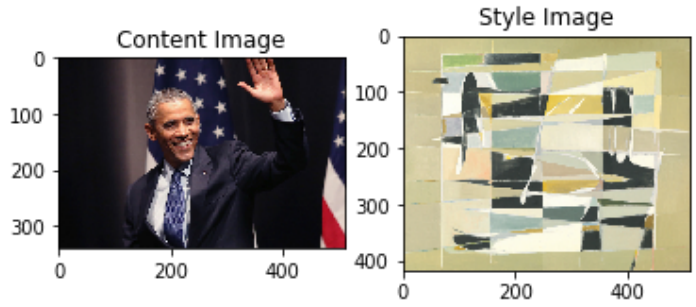
```
Total time: 44.5
I have written on twitter!
------------------------------------------
[1263796322727014401, 749616853698355201, 'IaQuinet']
I am going too fast, I will wait a bit!
Jimi Hendrix
http://1.bp.blogspot.com/-HqyjMigsgKE/UKlBZVhvisI/AAAAAAAAhMI/iiO2Cg
tBOtk/s1600/Jimi_Hendrix-quotes.jpg
Downloading data from http://1.bp.blogspot.com/-HqyjMigsgKE/UKlBZVhv
isI/AAAAAAAAhMI/iiO2CgtBOtk/s1600/Jimi_Hendrix-quotes.jpg
147456/146582 [==============================] - 0s 1us/step
```




```
.........................................................................
Train step: 100
.........................................................................
Train step: 200
.........................................................................
Train step: 300
.........................................................................
Train step: 400
.........................................................................
Train step: 500
.........................................................................
Train step: 600
.........................................................................
Train step: 700
.........................................................................
Train step: 800
.........................................................................
Train step: 900
.........................................................................
Train step: 1000
```

Total time: 50.6
I have written on twitter!

[1261721612010782720, 2883285551, 'cagou007']
I am going too fast, I will wait a bit!
Aubervilliers en commun
Barack Obama
https://image.cnbcfm.com/api/v1/image/104917714-GettyImages-456540086rr.jpg?v=1532563708




```
.....................................................................
Train step: 100
.....................................................................
Train step: 200
.....................................................................
Train step: 300
.....................................................................
Train step: 400
.....................................................................
Train step: 500
.....................................................................
Train step: 600
.....................................................................
Train step: 700
.....................................................................
Train step: 800
.....................................................................
Train step: 900
.....................................................................
Train step: 1000
```

```
Total time: 41.8
I have written on twitter!
------------------------------------------
Im done with this round of tweets!
################################################
1
I am late, I can go!
1263797816708734978 IA Quinet IaQuinet 749616853698355201 1263796322
727014401
1263796322727014401 IA Quinet IaQuinet 749616853698355201 None
1261931485315072000 Greg Greg53838158 960981999711326208 12617038485
50371328
1261721612010782720 Stéphane Lefèvre cagou007 2883285551 None
1261721447321554945 Stéphane Lefèvre cagou007 2883285551 12617038485
50371328
I might be missing this person? I need to do more checks.
I already produced art for the author of this tweet!
I already produced art for the author of this tweet!
I already produced art for the author of this tweet!
I might be missing this person? I need to do more checks.
749616853698355201 IaQuinet Merci de t'intéresser à moi! Je ne suis
que des lignes de codes mais je pense que tu portes un certain intér
êt pour… https://t.co/m8ir4rErtp
2883285551 cagou007 @mIAtisse Salut l’artiste !
Im done with this round of tweets!
################################################
2
I am going too fast, I will wait a bit!
1263797816708734978 IA Quinet IaQuinet 749616853698355201 1263796322
727014401
1263796322727014401 IA Quinet IaQuinet 749616853698355201 None
1261931485315072000 Greg Greg53838158 960981999711326208 12617038485
50371328
1261721612010782720 Stéphane Lefèvre cagou007 2883285551 None
1261721447321554945 Stéphane Lefèvre cagou007 2883285551 12617038485
50371328
I might be missing this person? I need to do more checks.
I already produced art for the author of this tweet!
I already produced art for the author of this tweet!
I already produced art for the author of this tweet!
I might be missing this person? I need to do more checks.
749616853698355201 IaQuinet Merci de t'intéresser à moi! Je ne suis
que des lignes de codes mais je pense que tu portes un certain intér
êt pour… https://t.co/m8ir4rErtp
2883285551 cagou007 @mIAtisse Salut l’artiste !
Im done with this round of tweets!
################################################
I have worked, will rest now.
```