

▼ Нейронный перенос стиля с Pytorch

Автор: Alexis Jacq <<https://alexis-jacq.github.io>>

Адаптивный перевод: Zueva Nadya <<https://github.com/nestyme>>

Введение

В этом ноутбуке объясняется и показывается, как работает алгоритм переноса стиля

Neural-Style <<https://arxiv.org/abs/1508.06576>>

Леона А. Гатиса, Александра С. Эккера и Маттиаса Бетге.

Нейронный перенос стиля – это алгоритм, который принимает контент-изображение (напр (например, картинку известного художника) и возвращает изображение, которое будто бы

Как это работает?

Всего есть три картинки: вход, стиль и контент. Определим два расстояния:

- D_S - оно определяет на сколько разные стили у двух произвольных картинок.
- D_C - оно определяет на сколько разнится контент у двух произвольных картинок.

задача сети - минимизировать D_S от входной картинки до стиля и D_C от входной картинки

В качестве входа обычно берется зашумленная к артинка контента.

Это все что нам понадобится:

```
1 !pip3 install torch torchvision
2 #!pip3 install pillow==4.1.1
```

```
↳ Requirement already satisfied: torch in /usr/local/lib/python3.6/dist-packages (1.5.0)
Requirement already satisfied: torchvision in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (from torch)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from torch)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.6/dist-packages
```

```
1 !wget https://www.dropbox.com/s/afqhab9zc9r8otb/images.zip?dl=1
2 !mv images.zip?dl=1 images.zip
3 !unzip images.zip
```



```
--2020-06-08 12:38:43-- https://www.dropbox.com/s/afqhab9zc9r8otb/images.zip?dl=1
Resolving www.dropbox.com (www.dropbox.com)... 162.125.67.1, 2620:100:6023:1::a27d:43
Connecting to www.dropbox.com (www.dropbox.com).|162.125.67.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/dl/afqhab9zc9r8otb/images.zip [following]
--2020-06-08 12:38:44-- https://www.dropbox.com/s/dl/afqhab9zc9r8otb/images.zip
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com/cd/0/get/A5F
--2020-06-08 12:38:44-- https://uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com/cd/0/get/A5F
Resolving uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com (uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com) (uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com) (uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com)
Connecting to uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com (uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com) (uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com) (uc2837e4903a11abfc58eba6f62d.dl.dropboxusercontent.com)
HTTP request sent, awaiting response... 200 OK
Length: 1057322 (1.0M) [application/binary]
Saving to: 'images.zip?dl=1'
```

```
1 %matplotlib inline
2 from PIL import Image
3
4
5 import torch
6 import torch.nn as nn
7 import torch.nn.functional as F
8 import torch.optim as optim
9
10 import matplotlib.pyplot as plt
11
12
13 import torchvision.transforms as transforms
14 import torchvision.models as models
15
16 import copy

1 !ls images/
⇨ content.jpg  first_style.jpg  second_style.jpg
```

Загрузка изображений

Нам понадобятся картинки стиля и контента, так что загрузим их.

Чтобы упростить реализацию, начнем с контента и стиля одного размера. Затем мы масштабируем выходного изображения.

Примеры изображений лежат в папке `Images` на гуглдиске

Вы можете добавить туда свои собственные изображения -- главное, чтобы они были однотипными

```
1 imsize = 512
2
3 loader = transforms.Compose([
4     transforms.Resize(imsize), # нормируем размер изображения
5     transforms.CenterCrop(imsize),
6     transforms.ToTensor()]) # превращаем в удобный формат
```

```
1 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
2 def image_loader(image_name):
3     image = Image.open(image_name)
4     image = loader(image).unsqueeze(0)
5     return image.to(device, torch.float)
6
7
8 #content_img = image_loader("images/picasso.jpg")# as well as here
9 #style_img = image_loader("images/lisa.jpg")#измените путь на тот который у вас.
10 #style_img = image_loader("images/picasso.jpg")# as well as here
11 #style_img_second = image_loader("images/second.jpg")# as well as here
12 #content_img = image_loader("images/lisa.jpg")#измените путь на тот который у вас.

1 #!wget https://museum-design.ru/wp-content/uploads/i-scott-05.jpg -O /content/images/fi
2 # ТУТ КРАСНЫЙ КВАДРАТ
3 #!wget https://art-holst.com.ua/wp-content/uploads/thumb_1_28551.jpg -O /content/images
4 #!wget https://storage.pic2.me/c/1360x800/721/59aea1186a1bf.jpg -O /content/images/conten
5
6 #!wget https://vestikavkaza.ru/upload/2019-12-18/15767022705dfa913e619de3.24454359.jpg
```



```
--2020-06-08 10:41:40-- https://museum-design.ru/wp-content/uploads/i-scott-05.jpg
Resolving museum-design.ru (museum-design.ru)... 5.9.57.100
Connecting to museum-design.ru (museum-design.ru)|5.9.57.100|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 561477 (548K) [image/jpeg]
Saving to: '/content/images/first_style.jpg'

/content/images/fir 100%[=====] 548.32K --.-KB/s in 0.1s
```

1

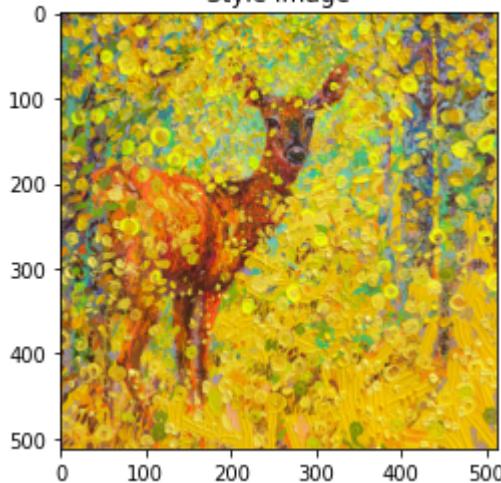
```
--2020-06-08 10:41:43-- https://art-holst.com.ua/wp-content/uploads/thumb\_1\_28551.jpg
1 #content_img = image_loader("images/picasso.jpg")# as well as here
2 #style_img = image_loader("images/lisa.jpg")#измените путь на тот который у вас.
3 style_img = image_loader("images/first_style.jpg")# as well as here
4 style_img_second = image_loader("images/second_style.jpg")# as well as here
5 content_img = image_loader("images/content.jpg")#измените путь на тот который у вас.
```

Выведем то, что было загружено

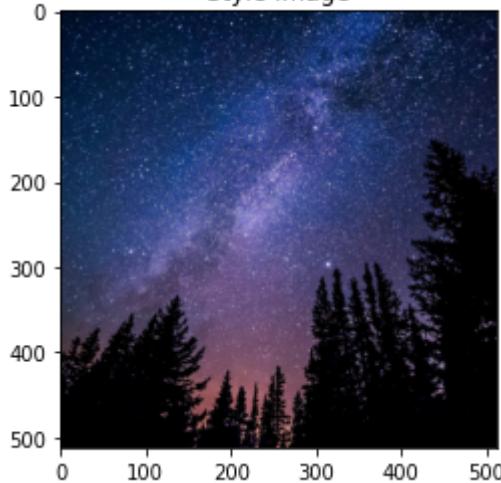
```
--2020-06-08 10:41:45-- https://storage.pics.ma/c/1260x800/721/5022118621bf.jpg
1 unloader = transforms.ToPILImage() # тензор в кратинку
2
3 plt.io()
4
5 def imshow(tensor, title=None):
6     image = tensor.cpu().clone()
7     image = image.squeeze(0)      # функция для отрисовки изображения
8     image = unloader(image)
9     plt.imshow(image)
10    if title is not None:
11        plt.title(title)
12    plt.pause(0.001)
13
14 def tensor_to_matrix(tensor):
15    # plt.figure(figsize=(10,10))
16    image = tensor.cpu().clone()
17    image = image.squeeze(0)      # функция для отрисовки изображения
18    image = unloader(image)
19    return image
20
21 # отрисовка
22
23 plt.figure()
24 imshow(style_img, title='Style Image')
25
26 plt.figure()
27 imshow(style_img_second, title='Style Image')
28
29
30 plt.figure()
31 imshow(content_img, title='Content Image')
32
```



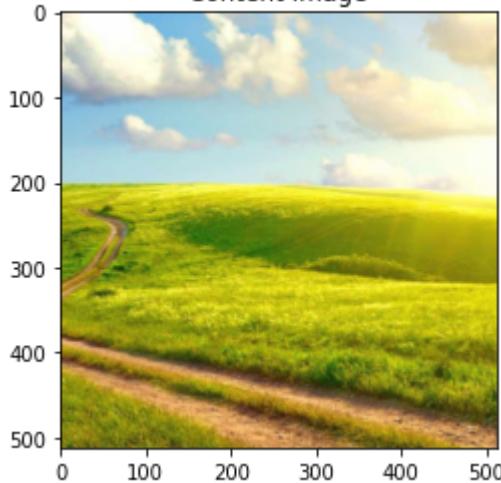
Style Image



Style Image



Content Image



Теперь нужно создать функции, которые будут вычислять расстояния (D_C и D_S). Они будут выполнены в виде слоев, чтобы брать по ним автоградиент.

D_S - средняя квадратичная ошибка input'a и target'a

```

1 class ContentLoss(nn.Module):
2
3     def __init__(self, target,):
4         super(ContentLoss, self).__init__()
5         # we 'detach' the target content from the tree used
6         # to dynamically compute the gradient: this is a stated value.

```

```

7      # not a variable. Otherwise the forward method of the criterion
8      # will throw an error.
9      self.target = target.detach()#это константа. Убираем ее из дерева вычислени
10     self.loss = F.mse_loss(self.target, self.target )#to initialize with sometk
11
12     def forward(self, input):
13         self.loss = F.mse_loss(input, self.target)
14         return input

```

Матрица грама позволяет учесть не только сами значения feature map'a, но и кореляцию ф. Это нужно для того, чтобы сделать акцент на встречаемость фич с друг другом, а не на их |. Полное понимание этого момента можно получить с помощью [этого](#) и [этого](#).

Таким образом:

$$D_S = \sum (G_{ij}(img_1) \cdot G_{ij}(img_2))^2$$

Сначала задаем способ подсчета матрицы грама: Это просто тензорное тензорное произ себя.

Однако наш выход - не вектор. В этом случае операция тоже возможна, но мы получим тензор третьего ранга. Поэтому перед перемножением выход нужно приве

```

1 def gram_matrix(input):
2     batch_size , f_map_num, h, w = input.size()  # batch size(=1)
3     #print("gram_matrix =" , batch_size, f_map_num, h, w)
4     # b=number of feature maps
5     # (h,w)=dimensions of a feature map (N=h*w)
6
7     features = input.view(batch_size * f_map_num, w * h)  # resise F_XL into \hat F
8
9     G = torch.mm(features, features.t())  # compute the gram product
10
11    # we 'normalize' the values of the gram matrix
12    # by dividing by the number of element in each feature maps.
13    return G.div(batch_size * h * w * f_map_num)

```

Матрица грама готова, теперь нужно лишь реализовать MSE

```

1
2
3
4
5
6
7
8

```

```

1 class StyleLoss(nn.Module):
2     def __init__(self, target_feature):
3         super(StyleLoss, self).__init__()
4         self.target = gram_matrix(target_feature).detach()
5         self.loss = F.mse_loss(self.target, self.target)# to initialize with sometk
6
7         def forward(self, input):
8             G = gram_matrix(input)

```

```

9     self.loss = F.mse_loss(G, self.target)
10    return input

```

При тренировке VGG каждое изображение на котором она обучалась было нормировано и использовать ее для нашей модели, то мы должны реализовать нормировку и для наших изображений.

```

1 cnn_normalization_mean = torch.tensor([0.485, 0.456, 0.406]).to(device)
2 cnn_normalization_std = torch.tensor([0.229, 0.224, 0.225]).to(device)

1 class Normalization(nn.Module):
2     def __init__(self, mean, std):
3         super(Normalization, self).__init__()
4         # .view the mean and std to make them [C x 1 x 1] so that they can
5         # directly work with image Tensor of shape [B x C x H x W].
6         # B is batch size. C is number of channels. H is height and W is width.
7         self.mean = torch.tensor(mean).view(-1, 1, 1)
8         self.std = torch.tensor(std).view(-1, 1, 1)
9
10    def forward(self, img):
11        # normalize img
12        return (img - self.mean) / self.std

```

Теперь соберем это все в одну функцию, которая отдаст на выходе модель и две функции для вычисления ошибок стиля и содержания.

Определим после каких уровней мы будем считать ошибки стиля, а после каких ошибки содержания.

```

1 content_layers_default = ['conv_4']
2 style_layers_default = ['conv_1', 'conv_2', 'conv_3', 'conv_4', 'conv_5']

```

Определим предобученную модель

```

1 cnn = models.vgg19(pretrained=True).features.to(device).eval()
2
3 Downloading: "https://download.pytorch.org/models/vgg19-dcb9e9d.pth" to /root/.cache/torch/
4 100%                                548M/548M [00:29<00:00, 19.5MB/s]

```

```

1 def get_style_model_and_losses(cnn, normalization_mean, normalization_std,
2                               style_img, style_img_second, content_img,
3                               content_layers=content_layers_default,
4                               style_layers=style_layers_default):
5     cnn = copy.deepcopy(cnn)
6
7     # normalization module
8     normalization = Normalization(normalization_mean, normalization_std).to(device)
9
10    # just in order to have an iterable access to or list of content/style

```

```
11     # losses
12     content_losses = []
13     style_losses = []
14     style_losses_second = []
15
16     # assuming that cnn is a nn.Sequential, so we make a new nn.Sequential
17     # to put in modules that are supposed to be activated sequentially
18     model = nn.Sequential(normalization)
19
20     i = 0 # increment every time we see a conv
21     for layer in cnn.children():
22         if isinstance(layer, nn.Conv2d):
23             i += 1
24             name = 'conv_{}'.format(i)
25         elif isinstance(layer, nn.ReLU):
26             name = 'relu_{}'.format(i)
27             # The in-place version doesn't play very nicely with the ContentLoss
28             # and StyleLoss we insert below. So we replace with out-of-place
29             # ones here.
30             #Переопределим relu уровень
31             layer = nn.ReLU(inplace=False)
32         elif isinstance(layer, nn.MaxPool2d):
33             name = 'pool_{}'.format(i)
34         elif isinstance(layer, nn.BatchNorm2d):
35             name = 'bn_{}'.format(i)
36         else:
37             raise RuntimeError('Unrecognized layer: {}'.format(layer.__class__.__name__))
38
39         model.add_module(name, layer)
40
41         if name in content_layers:
42             # add content loss:
43             target = model(content_img).detach()
44             content_loss = ContentLoss(target)
45             model.add_module("content_loss_{}".format(i), content_loss)
46             content_losses.append(content_loss)
47
48         if name in style_layers:
49             # add style loss:
50             target_feature = model(style_img).detach()
51             style_loss = StyleLoss(target_feature)
52             model.add_module("style_loss_{}".format(i), style_loss)
53             style_losses.append(style_loss)
54
55             # add style loss:
56             target_feature_second = model(style_img_second).detach()
57             style_loss_second = StyleLoss(target_feature_second)
58             model.add_module("style_loss_second_{}".format(i), style_loss_second)
59             style_losses_second.append(style_loss_second)
60
61         # now we trim off the layers after the last content and style losses
62         #выбрасываем все уровни после последнего style loss или content loss
63         for i in range(len(model) - 1, -1, -1):
64             if isinstance(model[i], ContentLoss) or isinstance(model[i], StyleLoss):
65                 break
```

```
66
67     model = model[:i + 1]
68
69     return model, style_losses, style_losses_second, content_losses

1 def get_input_optimizer(input_img):
2     # this line to show that input is a parameter that requires a gradient
3     #добавляет содержимое тензора катринки в список изменяемых оптимизатором параметров
4     #optimizer = optim.LBFGS([input_img.requires_grad_()])
5     #optimizer = optim.LBFGS([input_img.requires_grad_()])
6     optimizer = torch.optim.Adam([input_img.requires_grad_()])
7     return optimizer
```

1

Дальше стандартный цикл обучения, но что это за closure?

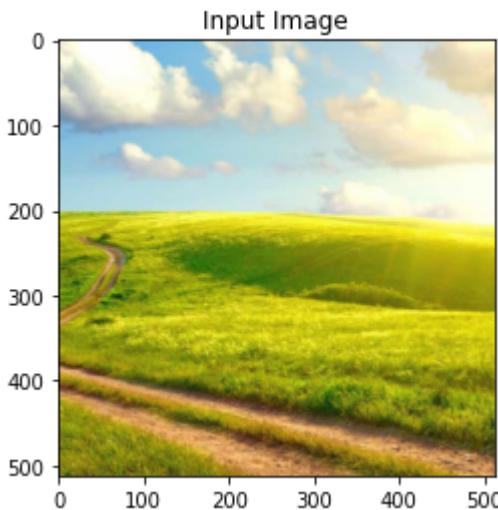
Это функция, которая вызывается во время каждого прохода, чтобы пересчитать loss. Без функции ошибки

```
1 def run_style_transfer(cnn, normalization_mean, normalization_std,
2                         content_img, style_img, style_img_second, input_img, num_steps=
3                             style_weight=100000, content_weight=1, alpha_coefficient=0.1):
4                             #style_weight=1, content_weight=100000):
5     """Run the style transfer."""
6     print('Building the style transfer model..')
7     model, style_losses, style_losses_second, content_losses = get_style_model_and_
8         normalization_mean, normalization_std, style_img, style_img_second, content_
9         optimizer = get_input_optimizer(input_img)
10
11
12     print('Optimizing..')
13     run = [0]
14     while run[0] <= num_steps:
15
16         def closure():
17             # correct the values
18             # это для того, чтобы значения тензора картинки не выходили за пределы
19             input_img.data.clamp_(0, 1)
20
21             optimizer.zero_grad()
22
23             model(input_img)
24
25             style_score = 0
26             style_score_second = 0
27             content_score = 0
28
29             for sl in style_losses:
30                 style_score += sl.loss
31             for sl_second in style_losses_second:
32                 style_score_second += sl_second.loss
33
34             if style_score > 0:
35                 style_score = style_score / num_steps
36             if style_score_second > 0:
37                 style_score_second = style_score_second / num_steps
38
39             if content_score > 0:
40                 content_score = content_score / num_steps
41
42             if style_score > 0 and style_score_second > 0:
43                 style_score = style_score * alpha_coefficient + style_score_second * (1 - alpha_coefficient)
44
45             if style_score > 0:
46                 style_score = style_score * content_weight + content_score * (1 - content_weight)
47
48             if style_score > 0:
49                 style_score = style_score / num_steps
50
51             if style_score > 0:
52                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
53
54             if style_score > 0:
55                 style_score = style_score / num_steps
56
57             if style_score > 0:
58                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
59
60             if style_score > 0:
61                 style_score = style_score / num_steps
62
63             if style_score > 0:
64                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
65
66             if style_score > 0:
67                 style_score = style_score / num_steps
68
69             if style_score > 0:
70                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
71
72             if style_score > 0:
73                 style_score = style_score / num_steps
74
75             if style_score > 0:
76                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
77
78             if style_score > 0:
79                 style_score = style_score / num_steps
80
81             if style_score > 0:
82                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
83
84             if style_score > 0:
85                 style_score = style_score / num_steps
86
87             if style_score > 0:
88                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
89
90             if style_score > 0:
91                 style_score = style_score / num_steps
92
93             if style_score > 0:
94                 style_score = style_score * style_weight + style_score_second * (1 - style_weight)
95
96             if style_score > 0:
97                 style_score = style_score / num_steps
98
99             if style_score > 0:
100                style_score = style_score * style_weight + style_score_second * (1 - style_weight)
101
102                if style_score > 0:
103                    style_score = style_score / num_steps
104
105                if style_score > 0:
106                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
107
108                if style_score > 0:
109                    style_score = style_score / num_steps
110
111                if style_score > 0:
112                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
113
114                if style_score > 0:
115                    style_score = style_score / num_steps
116
117                if style_score > 0:
118                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
119
120                if style_score > 0:
121                    style_score = style_score / num_steps
122
123                if style_score > 0:
124                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
125
126                if style_score > 0:
127                    style_score = style_score / num_steps
128
129                if style_score > 0:
130                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
131
132                if style_score > 0:
133                    style_score = style_score / num_steps
134
135                if style_score > 0:
136                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
137
138                if style_score > 0:
139                    style_score = style_score / num_steps
140
141                if style_score > 0:
142                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
143
144                if style_score > 0:
145                    style_score = style_score / num_steps
146
147                if style_score > 0:
148                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
149
150                if style_score > 0:
151                    style_score = style_score / num_steps
152
153                if style_score > 0:
154                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
155
156                if style_score > 0:
157                    style_score = style_score / num_steps
158
159                if style_score > 0:
160                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
161
162                if style_score > 0:
163                    style_score = style_score / num_steps
164
165                if style_score > 0:
166                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
167
168                if style_score > 0:
169                    style_score = style_score / num_steps
170
171                if style_score > 0:
172                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
173
174                if style_score > 0:
175                    style_score = style_score / num_steps
176
177                if style_score > 0:
178                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
179
180                if style_score > 0:
181                    style_score = style_score / num_steps
182
183                if style_score > 0:
184                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
185
186                if style_score > 0:
187                    style_score = style_score / num_steps
188
189                if style_score > 0:
190                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
191
192                if style_score > 0:
193                    style_score = style_score / num_steps
194
195                if style_score > 0:
196                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
197
198                if style_score > 0:
199                    style_score = style_score / num_steps
200
201                if style_score > 0:
202                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
203
204                if style_score > 0:
205                    style_score = style_score / num_steps
206
207                if style_score > 0:
208                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
209
210                if style_score > 0:
211                    style_score = style_score / num_steps
212
213                if style_score > 0:
214                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
215
216                if style_score > 0:
217                    style_score = style_score / num_steps
218
219                if style_score > 0:
220                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
221
222                if style_score > 0:
223                    style_score = style_score / num_steps
224
225                if style_score > 0:
226                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
227
228                if style_score > 0:
229                    style_score = style_score / num_steps
230
231                if style_score > 0:
232                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
233
234                if style_score > 0:
235                    style_score = style_score / num_steps
236
237                if style_score > 0:
238                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
239
240                if style_score > 0:
241                    style_score = style_score / num_steps
242
243                if style_score > 0:
244                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
245
246                if style_score > 0:
247                    style_score = style_score / num_steps
248
249                if style_score > 0:
250                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
251
252                if style_score > 0:
253                    style_score = style_score / num_steps
254
255                if style_score > 0:
256                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
257
258                if style_score > 0:
259                    style_score = style_score / num_steps
260
261                if style_score > 0:
262                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
263
264                if style_score > 0:
265                    style_score = style_score / num_steps
266
267                if style_score > 0:
268                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
269
270                if style_score > 0:
271                    style_score = style_score / num_steps
272
273                if style_score > 0:
274                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
275
276                if style_score > 0:
277                    style_score = style_score / num_steps
278
279                if style_score > 0:
280                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
281
282                if style_score > 0:
283                    style_score = style_score / num_steps
284
285                if style_score > 0:
286                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
287
288                if style_score > 0:
289                    style_score = style_score / num_steps
290
291                if style_score > 0:
292                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
293
294                if style_score > 0:
295                    style_score = style_score / num_steps
296
297                if style_score > 0:
298                    style_score = style_score * style_weight + style_score_second * (1 - style_weight)
299
299
```

```
(MFTI)[HomeWork]style_transfer.ipynb - Colaboratory
...
34         content_score += cl.loss
35
36
37     #взвешивание ошибки
38     style_score *= (style_weight * (1 - alpha_coefficient))
39     style_score_second *= (style_weight * alpha_coefficient)
40     content_score *= content_weight
41
42     loss = style_score + style_score_second + content_score
43     loss.backward()
44
45     run[0] += 1
46     if run[0] % 50 == 0:
47         print("run {}".format(run))
48         print('Style Loss : {:.4f} Style-Second Loss : {:.4f} Content Loss: {:.4f}'.format(
49             style_score.item(), style_score_second.item(), content_score.item()))
50         print()
51         plt.figure()
52         imshow(input_img, title='Output Image')
53         plt.ioff()
54         plt.show()
55
56     return style_score + style_score_second + content_score
57
58     optimizer.step(closure)
59
60     # a last correction...
61     input_img.data.clamp_(0, 1)
62
63     return input_img
```

```
1 input_img = content_img.clone()
2 # if you want to use white noise instead uncomment the below line:
3 #input_img = torch.randn(content_img.data.size(), device=device)
4
5 # add the original input image to the figure:
6 plt.figure()
7 imshow(input_img, title='Input Image')
8 output = run_style_transfer(cnn, cnn_normalization_mean, cnn_normalization_std,
9                             content_img, style_img, style_img_second, input_img, alpha_
10
11 plt.figure()
12 imshow(output, title='Output Image')
13 #plt.imsave(output, 'output.png')
14 # sphinx_gallery_thumbnail_number = 4
15 plt.ioff()
16 plt.show()
```





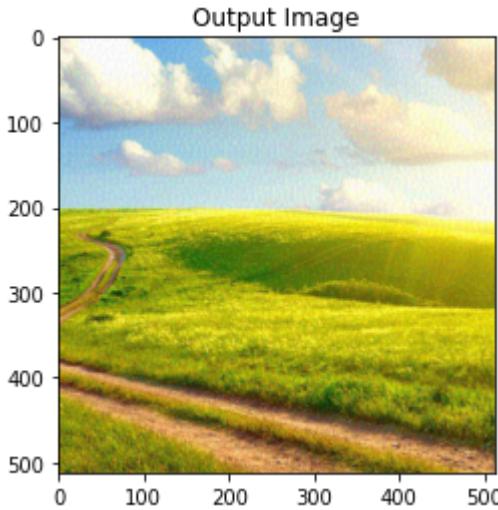
Building the style transfer model..

Optimizing..

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: UserWarning: To copy  
    import sys  
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: UserWarning: To copy
```

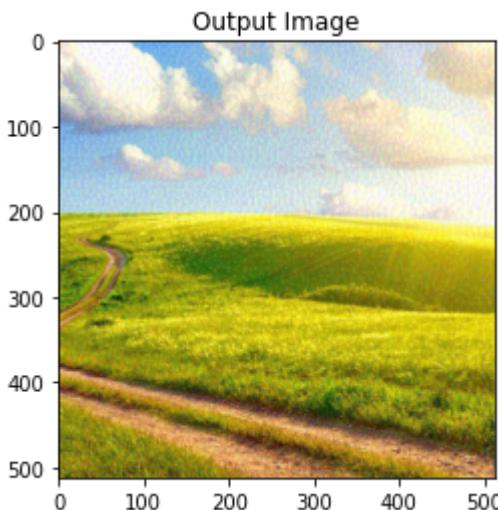
run [50]:

```
Style Loss : 1385.941162 Style-Second Loss : 991.702942 Content Loss: 8.912428
```



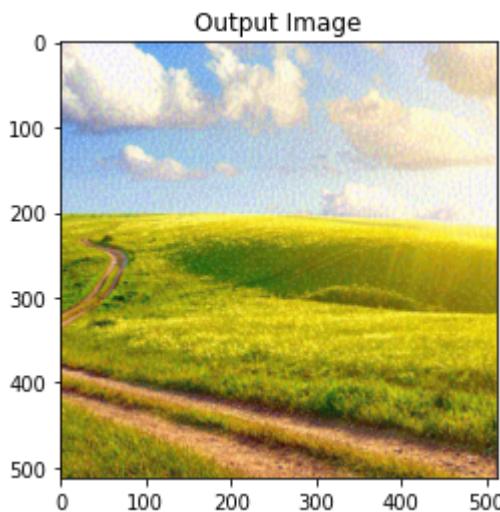
run [100]:

```
Style Loss : 1323.689087 Style-Second Loss : 900.548035 Content Loss: 11.816435
```



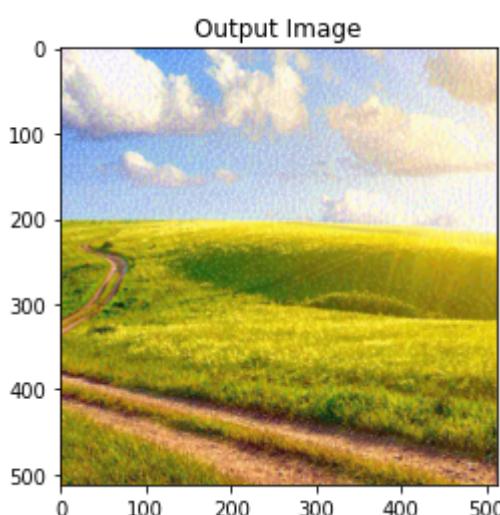
run [150]:

```
Style Loss : 1316.747070 Style-Second Loss : 836.502747 Content Loss: 13.192642
```



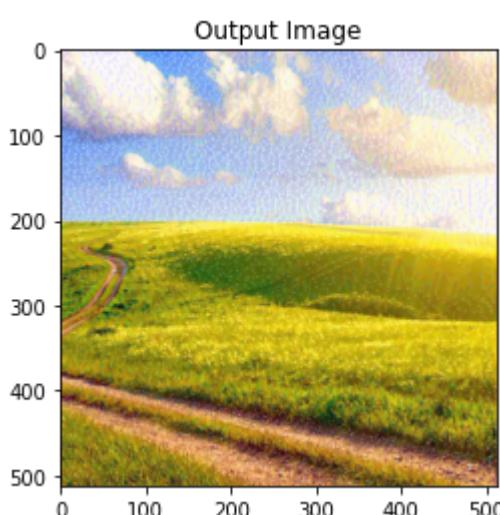
run [200]:

Style Loss : 1317.272339 Style-Second Loss : 793.015808 Content Loss: 14.105225



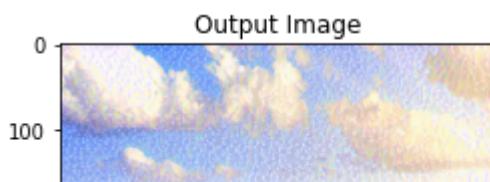
run [250]:

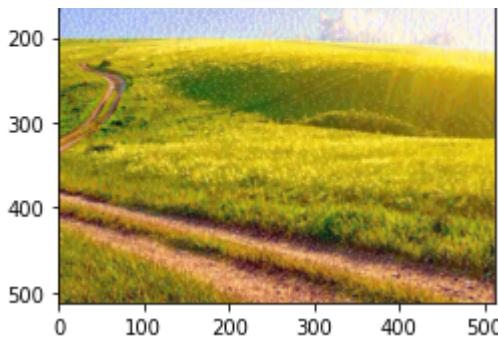
Style Loss : 1320.058350 Style-Second Loss : 761.951172 Content Loss: 14.747772



run [300]:

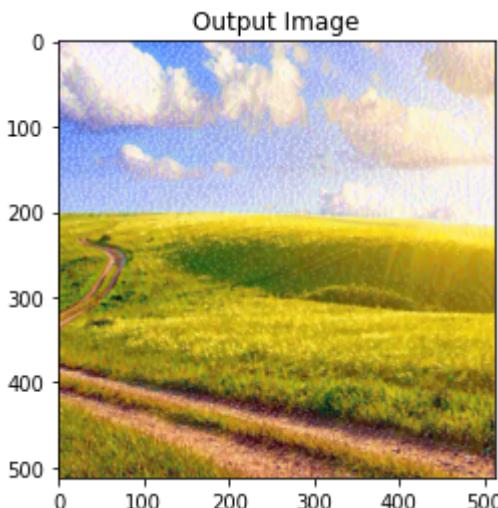
Style Loss : 1323.468750 Style-Second Loss : 738.644165 Content Loss: 15.224835





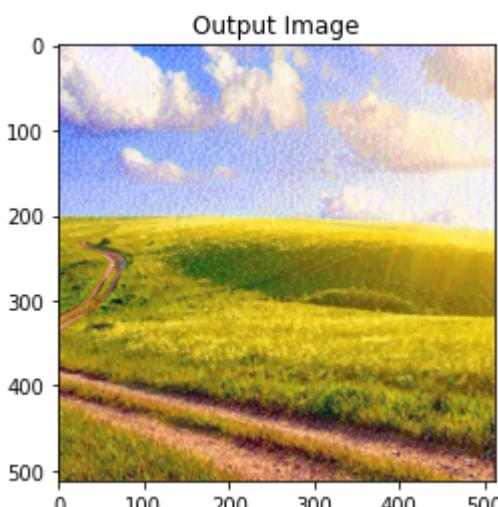
run [350]:

Style Loss : 1327.009888 Style-Second Loss : 720.307129 Content Loss: 15.594793



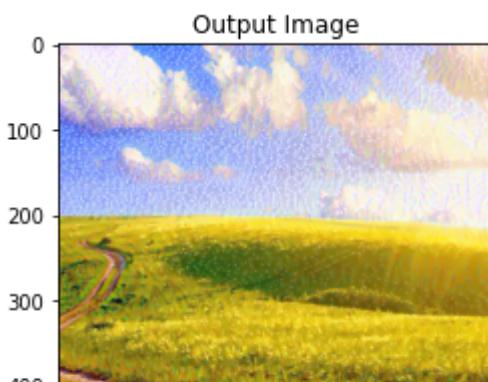
run [400]:

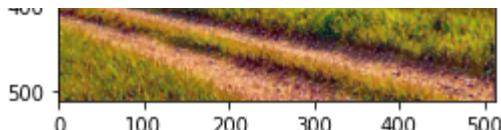
Style Loss : 1330.464233 Style-Second Loss : 705.446411 Content Loss: 15.890959



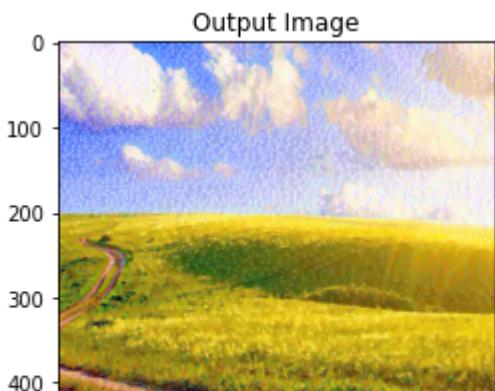
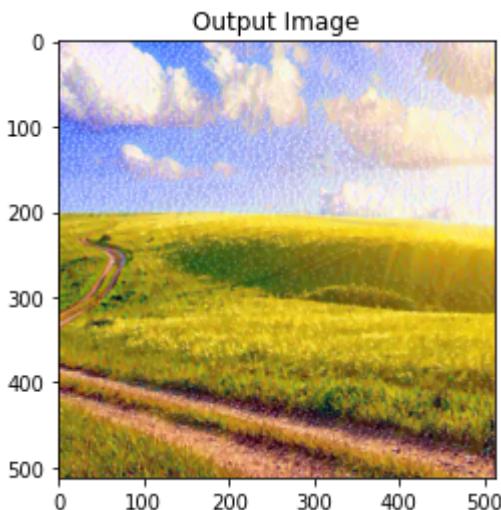
run [450]:

Style Loss : 1333.833252 Style-Second Loss : 693.079529 Content Loss: 16.129791



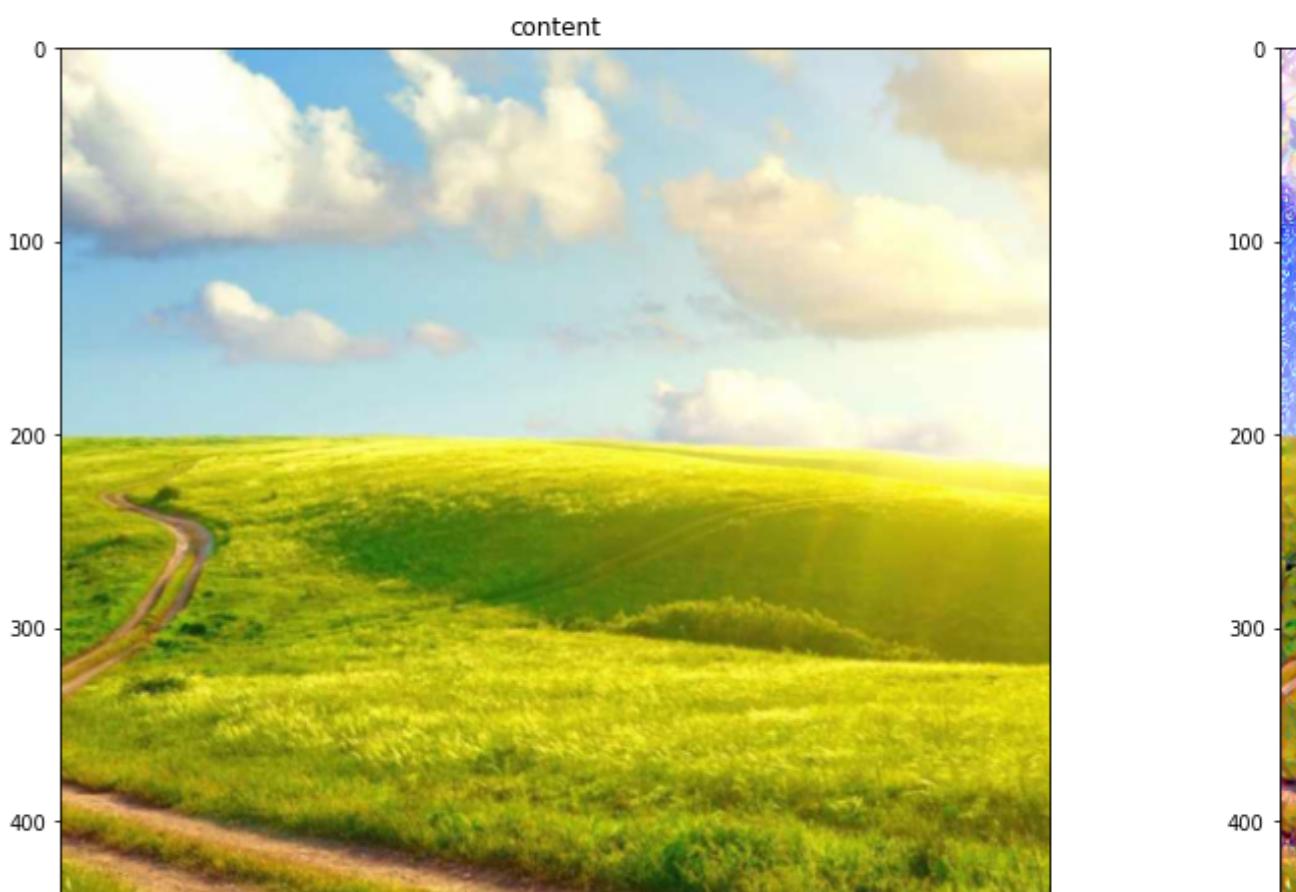
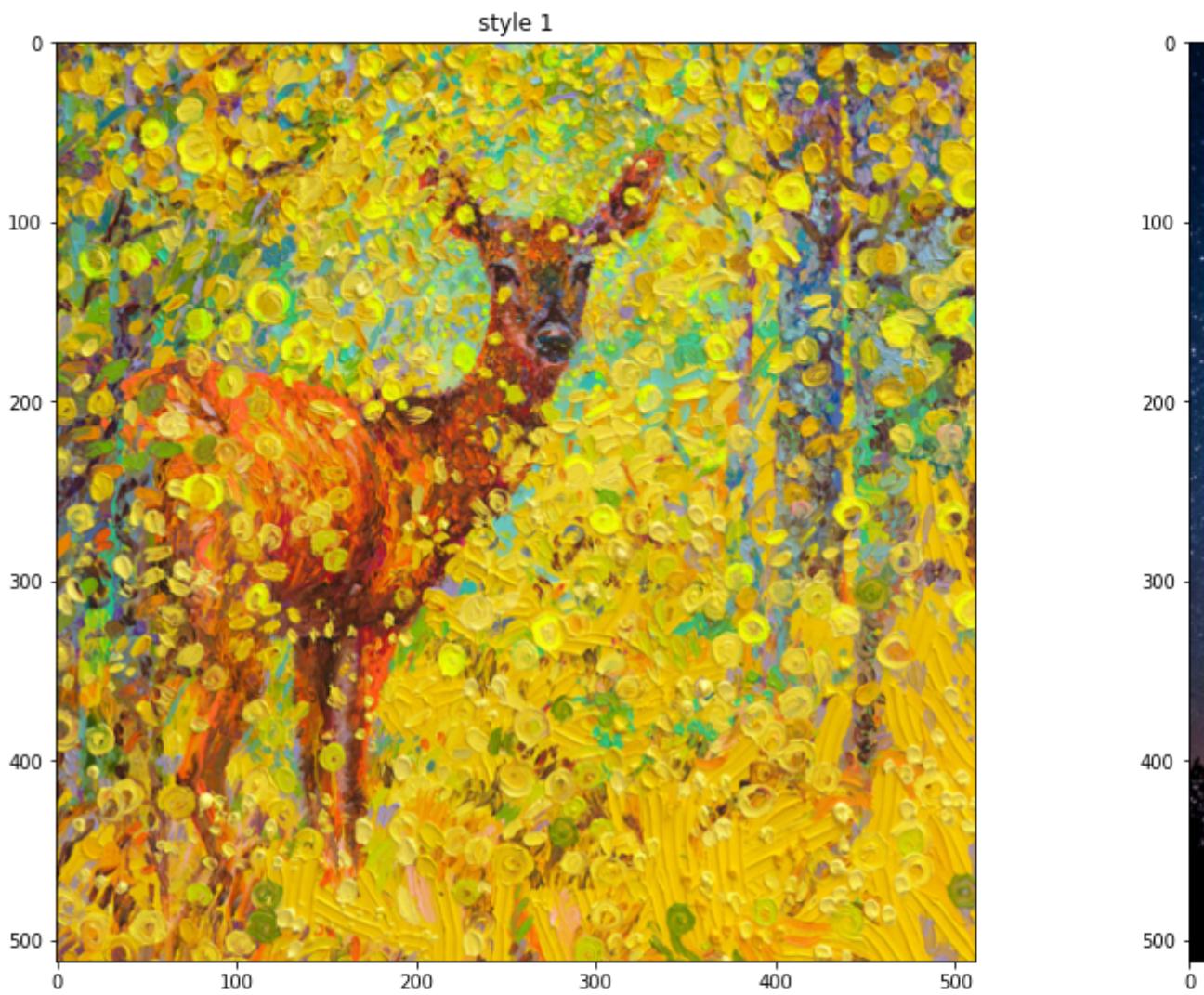


```
run [500]:  
Style Loss : 1337.081299 Style-Second Loss : 682.613037 Content Loss: 16.322048
```



```
1 fig = plt.figure(figsize=(20,20), )  
2  
3 plt.subplot(2, 2, 1)  
4 plt.title('style 1')  
5 plt.imshow(tensor_to_matrix(style_img))  
6  
7 plt.subplot(2, 2, 2)  
8 plt.title('style second')  
9 plt.imshow(tensor_to_matrix(style_img_second))  
10  
11 plt.subplot(2, 2, 3)  
12 plt.title('content')  
13 plt.imshow(tensor_to_matrix(content_img))  
14  
15 plt.subplot(2, 2, 4)  
16 plt.title('outout')  
17 plt.imshow(tensor_to_matrix(output))  
18  
19  
20 plt.ioff()  
21 plt.show()
```

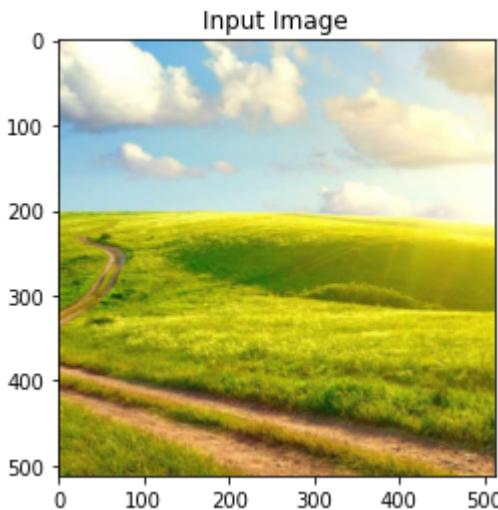






```
1 input_img = content_img.clone()
2 # if you want to use white noise instead uncomment the below line:
3 #input_img = torch.randn(content_img.data.size(), device=device)
4
5 # add the original input image to the figure:
6 plt.figure()
7 imshow(input_img, title='Input Image')
8 output = run_style_transfer(cnn, cnn_normalization_mean, cnn_normalization_std,
9                             content_img, style_img, style_img_second, input_img, alpha_
10
11 plt.figure()
12 imshow(output, title='Output Image')
13 #plt.imsave(output, 'output.png')
14 # sphinx_gallery_thumbnail_number = 4
15 plt.ioff()
16 plt.show()
```





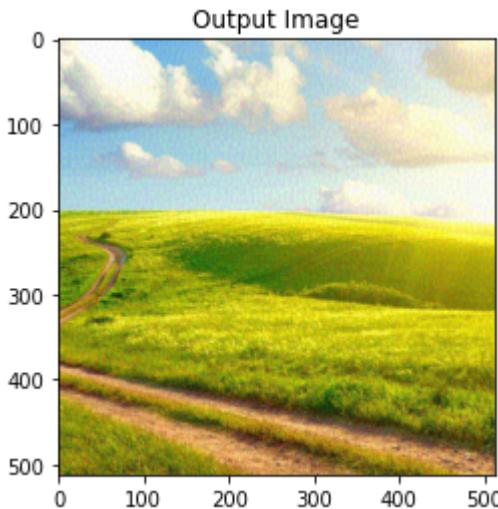
Building the style transfer model..

Optimizing..

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: UserWarning: To copy  
    import sys  
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: UserWarning: To copy
```

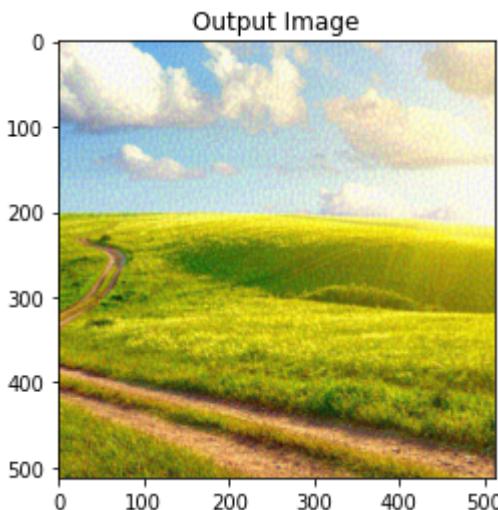
run [50]:

```
Style Loss : 1407.555054 Style-Second Loss : 1372.163330 Content Loss: 10.413445
```



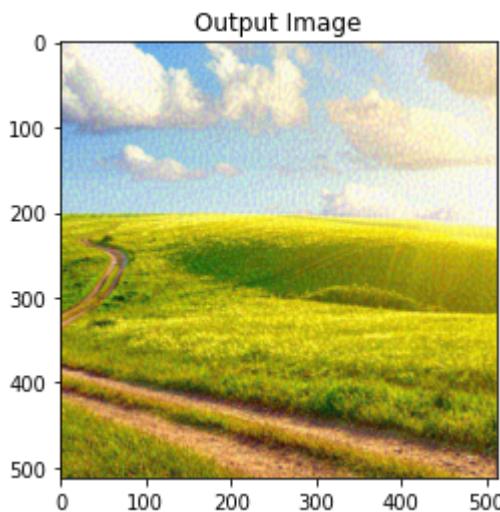
run [100]:

```
Style Loss : 1225.127075 Style-Second Loss : 1351.886108 Content Loss: 14.330048
```



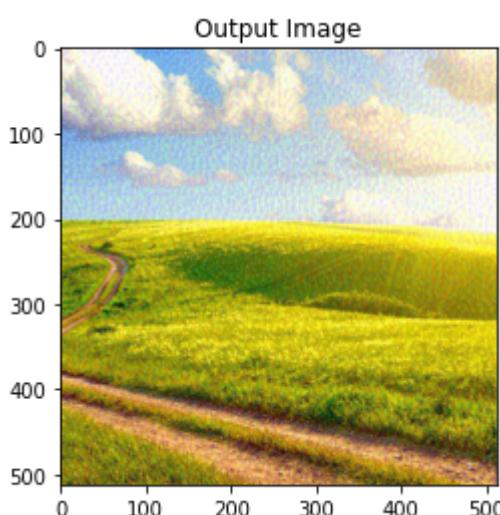
run [150]:

```
Style Loss : 1185.430664 Style-Second Loss : 1324.331177 Content Loss: 15.677208
```



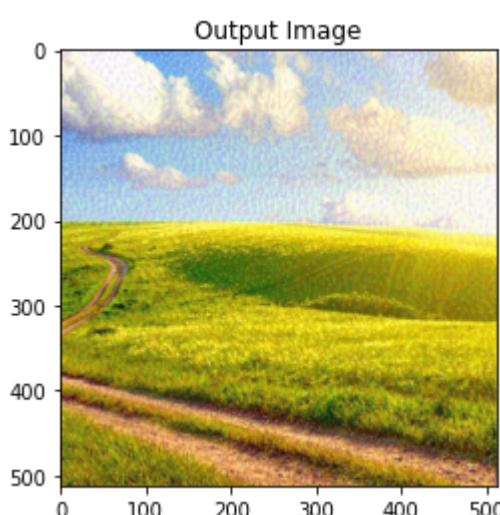
run [200]:

Style Loss : 1167.370728 Style-Second Loss : 1301.530151 Content Loss: 16.625723



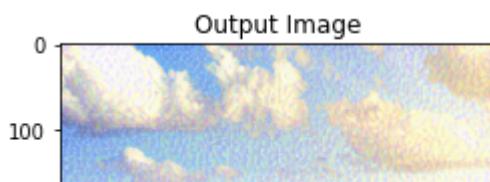
run [250]:

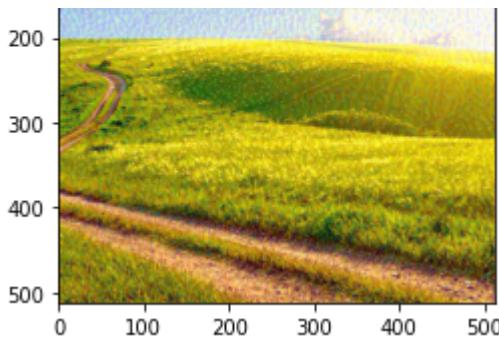
Style Loss : 1157.950439 Style-Second Loss : 1283.749390 Content Loss: 17.306702



run [300]:

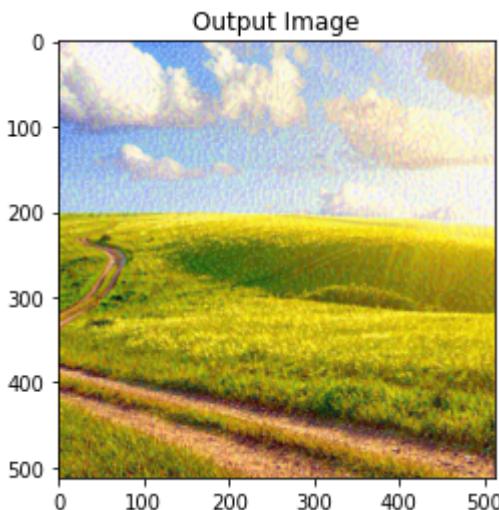
Style Loss : 1153.011719 Style-Second Loss : 1269.721191 Content Loss: 17.807995





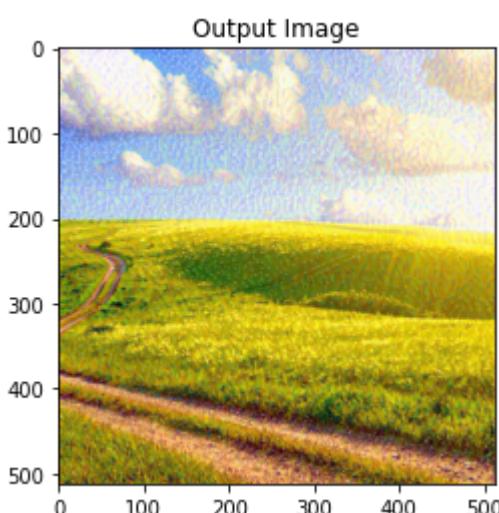
run [350]:

Style Loss : 1150.686768 Style-Second Loss : 1258.341431 Content Loss: 18.183350



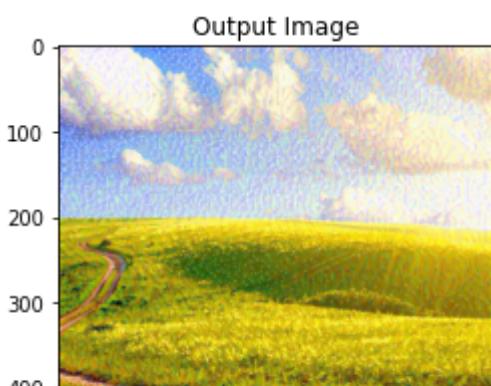
run [400]:

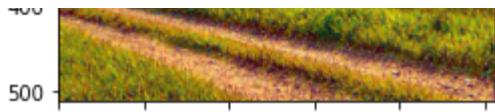
Style Loss : 1149.900513 Style-Second Loss : 1248.874390 Content Loss: 18.467407



run [450]:

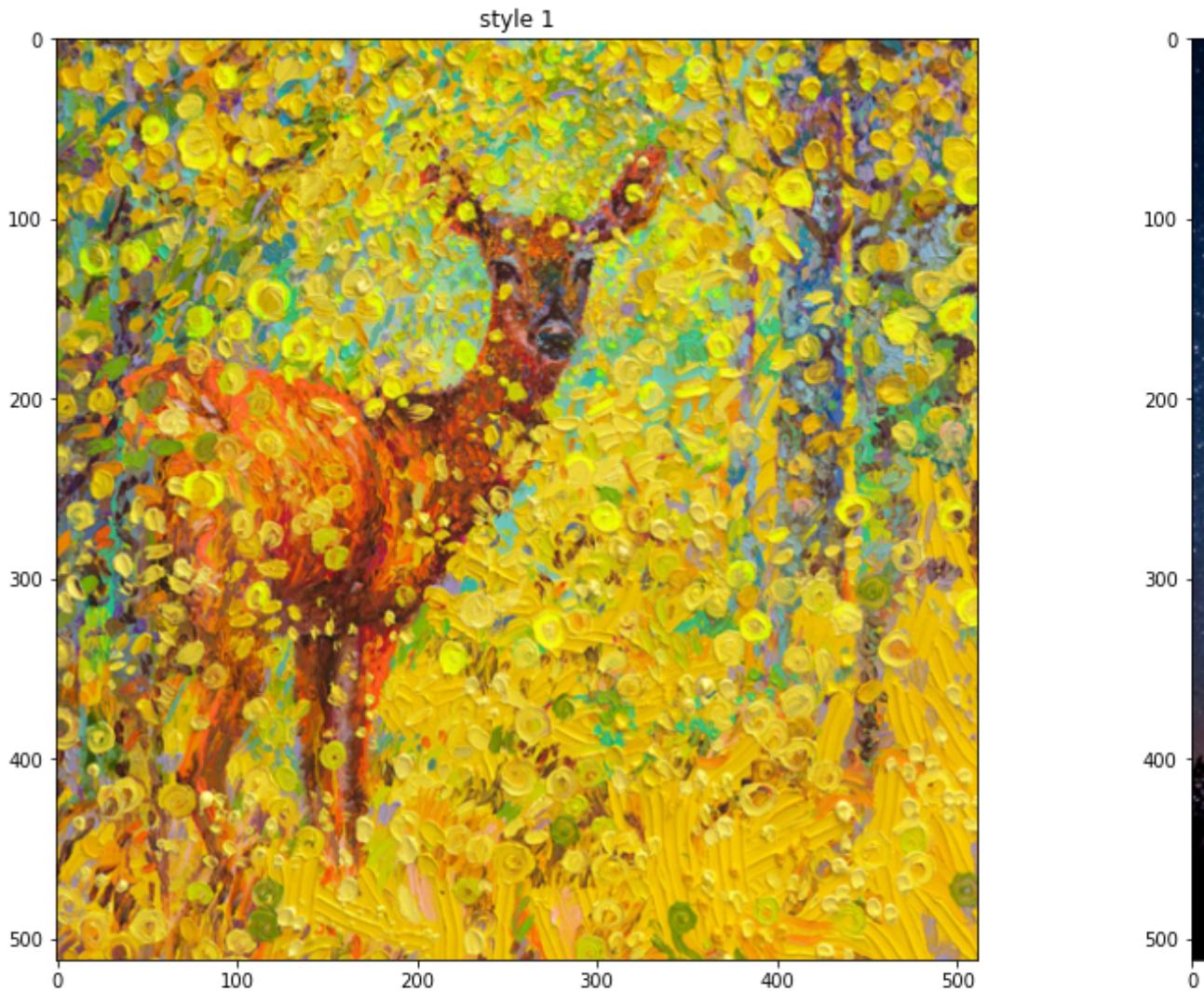
Style Loss : 1149.979736 Style-Second Loss : 1240.857910 Content Loss: 18.684111





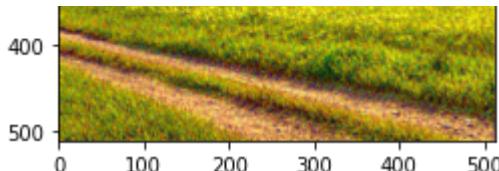
```
1 fig = plt.figure(figsize=(20,20), )
2
3 plt.subplot(2, 2, 1)
4 plt.title('style 1')
5 plt.imshow(tensor_to_matrix(style_img))
6
7 plt.subplot(2, 2, 2)
8 plt.title('style second')
9 plt.imshow(tensor_to_matrix(style_img_second))
10
11 plt.subplot(2, 2, 3)
12 plt.title('content')
13 plt.imshow(tensor_to_matrix(content_img))
14
15 plt.subplot(2, 2, 4)
16 plt.title('outout')
17 plt.imshow(tensor_to_matrix(output))
18
19
20 plt.ioff()
21 plt.show()
```





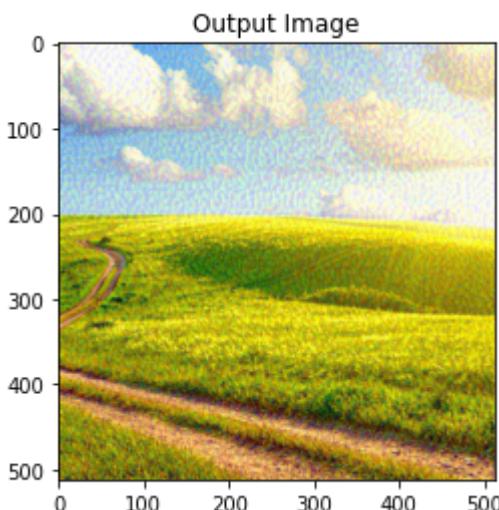
```
1 input_img = content_img.clone()
2 # if you want to use white noise instead uncomment the below line:
3 #input_img = torch.randn(content_img.data.size(), device=device)
4
5 # add the original input image to the figure:
6 plt.figure()
7 imshow(input_img, title='Input Image')
8 output = run_style_transfer(cnn, cnn_normalization_mean, cnn_normalization_std,
9                             content_img, style_img, style_img_second, input_img, alpha_
10
11 plt.figure()
12 imshow(output, title='Output Image')
13 #plt.imsave(output, 'output.png')
14 # sphinx_gallery_thumbnail_number = 4
15 plt.ioff()
16 plt.show()
```





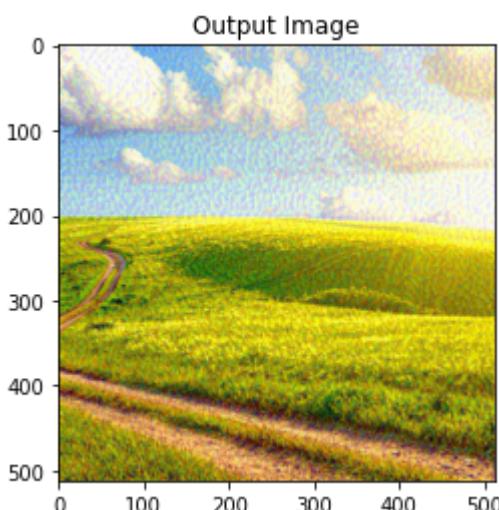
run [300]:

Style Loss : 409.905457 Style-Second Loss : 1190.224976 Content Loss: 23.982220



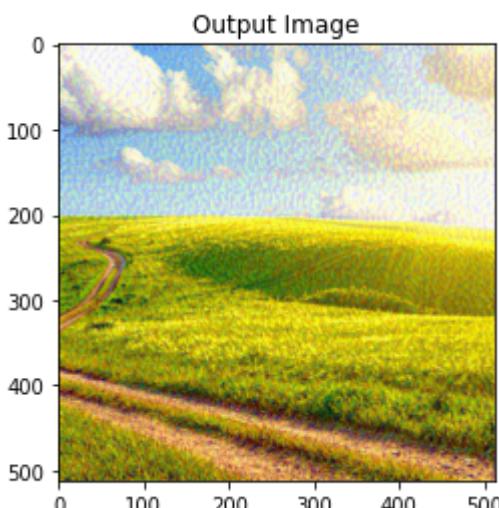
run [350]:

Style Loss : 389.535156 Style-Second Loss : 1192.620972 Content Loss: 24.453384



run [400]:

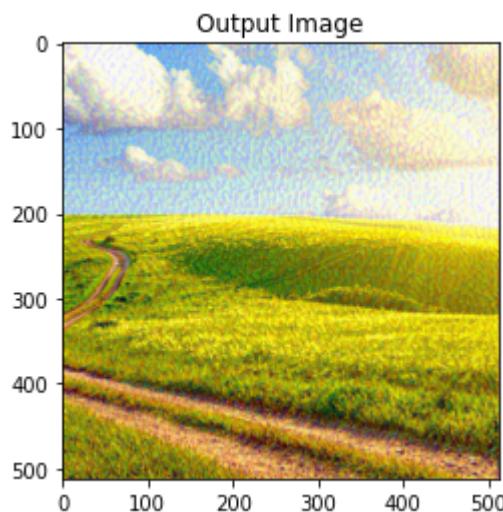
Style Loss : 374.901550 Style-Second Loss : 1193.886963 Content Loss: 24.804613



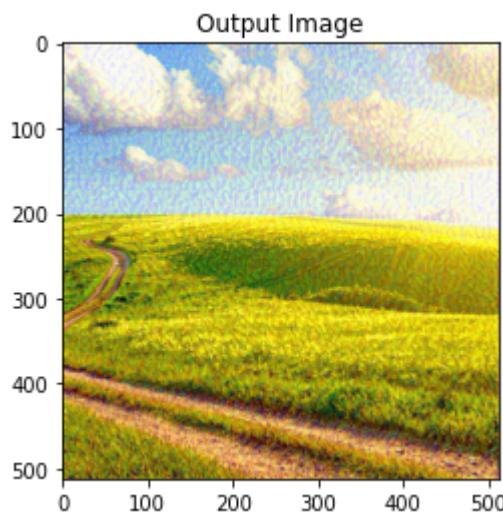
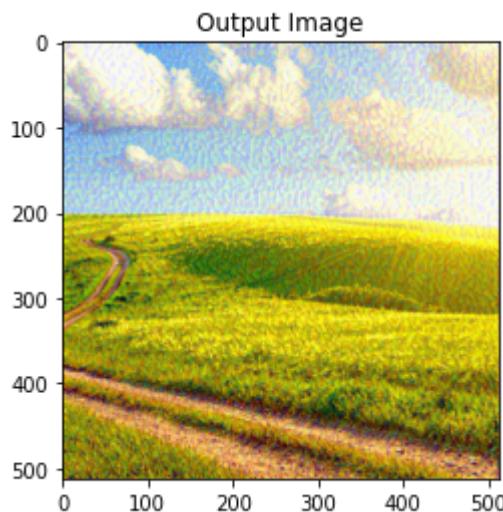
run [450]:

<https://colab.research.google.com/drive/1YeCr3XaGZIGj9sxDs49UgD2TwnMwzVfH#scrollTo=oR2aemLH6I5T&printMode=true>

```
run [500].  
Style Loss : 364.102570 Style-Second Loss : 1194.523560 Content Loss: 25.067974
```

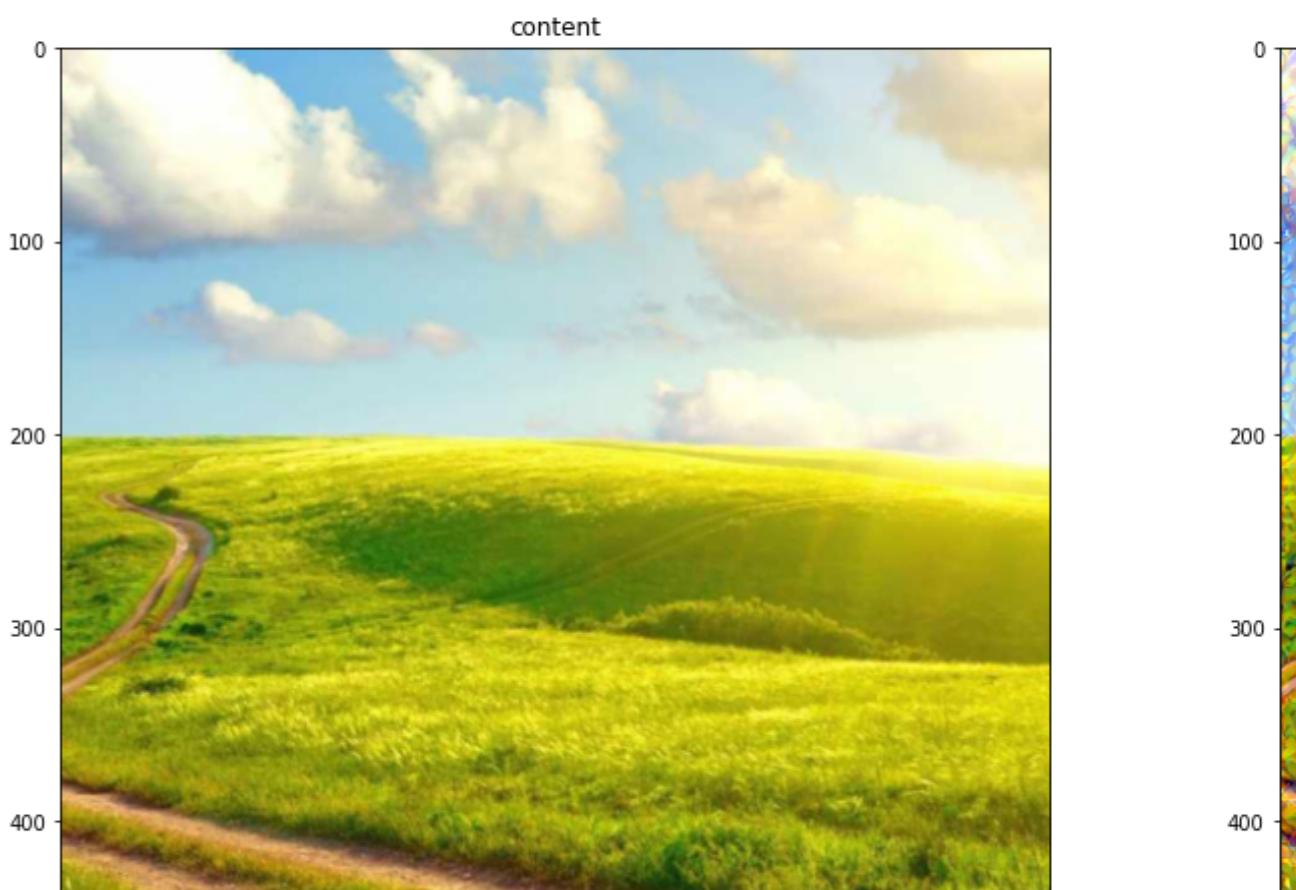
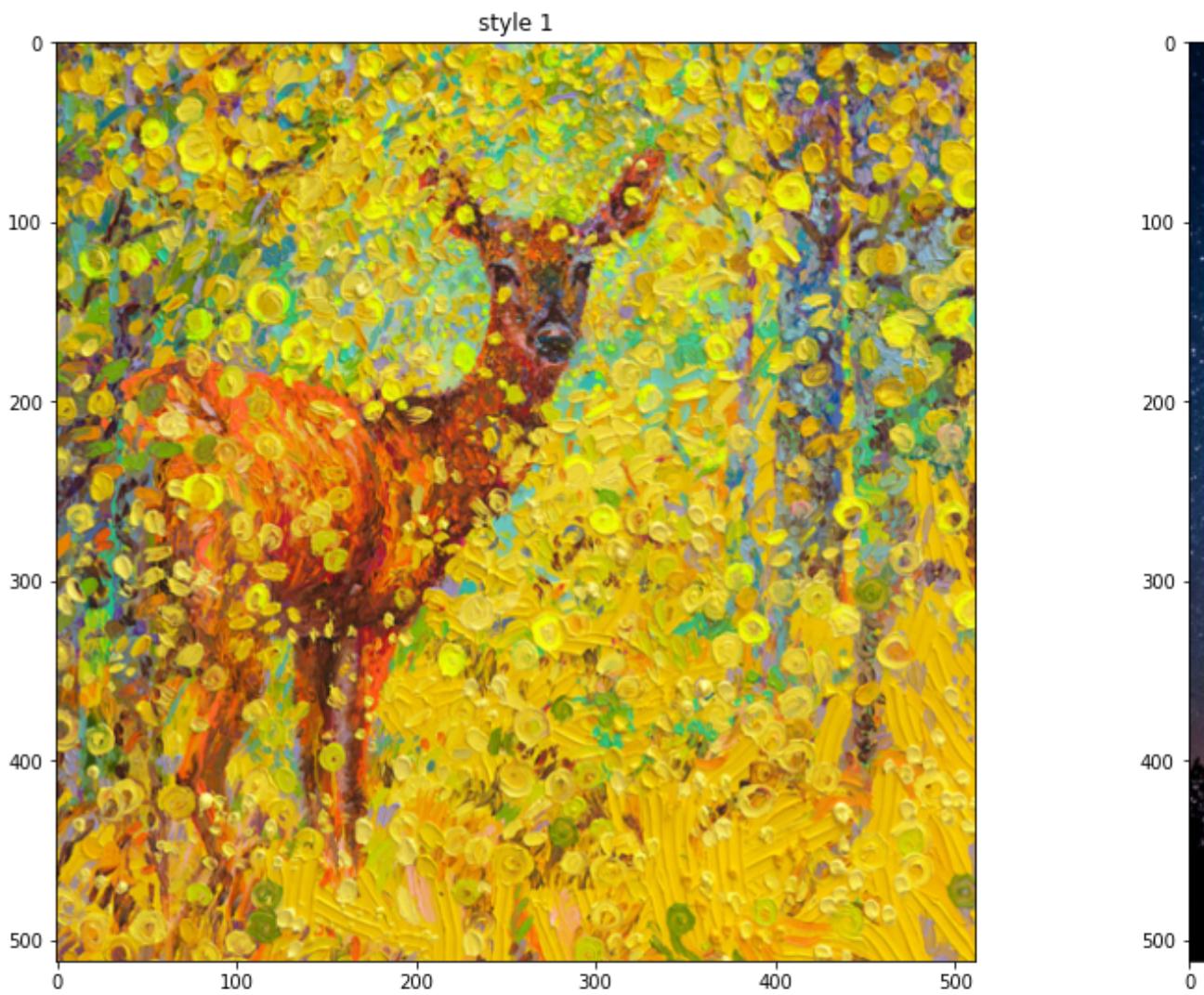


```
run [500]:  
Style Loss : 356.037628 Style-Second Loss : 1194.721802 Content Loss: 25.264849
```




```
1 fig = plt.figure(figsize=(20,20), )
2
3 plt.subplot(2, 2, 1)
4 plt.title('style 1')
5 plt.imshow(tensor_to_matrix(style_img))
6
7 plt.subplot(2, 2, 2)
8 plt.title('style second')
9 plt.imshow(tensor_to_matrix(style_img_second))
10
11 plt.subplot(2, 2, 3)
12 plt.title('content')
13 plt.imshow(tensor_to_matrix(content_img))
14
15 plt.subplot(2, 2, 4)
16 plt.title('outout')
17 plt.imshow(tensor_to_matrix(output))
18
19
20 plt.ioff()
21 plt.show()
```

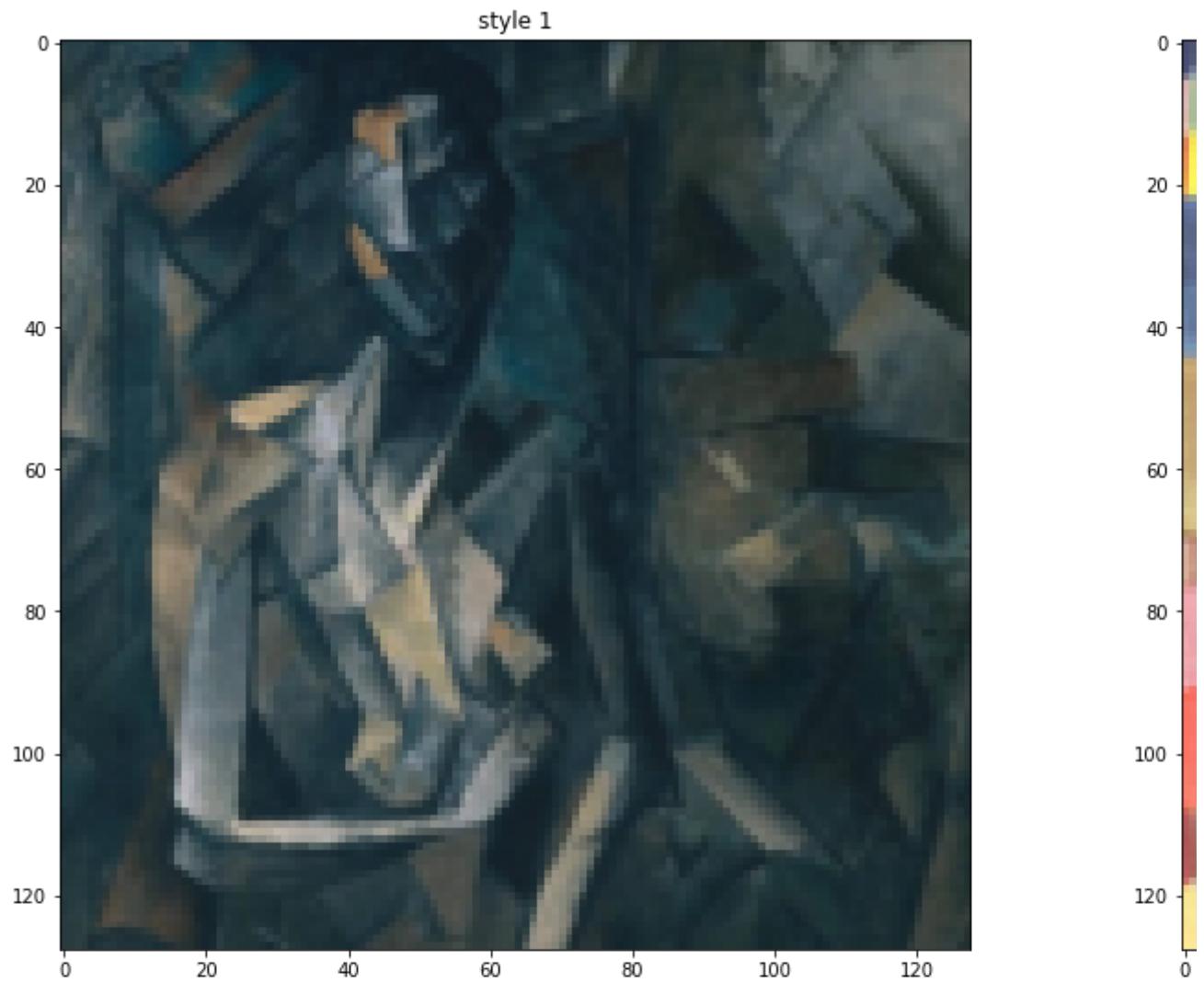






```
1 fig = plt.figure(figsize=(20,20), )
2
3 plt.subplot(2, 2, 1)
4 plt.title('style 1')
5 plt.imshow(tensor_to_matrix(style_img))
6
7 plt.subplot(2, 2, 2)
8 plt.title('style second')
9 plt.imshow(tensor_to_matrix(style_img_second))
10
11 plt.subplot(2, 2, 3)
12 plt.title('content')
13 plt.imshow(tensor_to_matrix(content_img))
14
15 plt.subplot(2, 2, 4)
16 plt.title('outout')
17 plt.imshow(tensor_to_matrix(output))
18
19
20 plt.ioff()
21 plt.show()
```

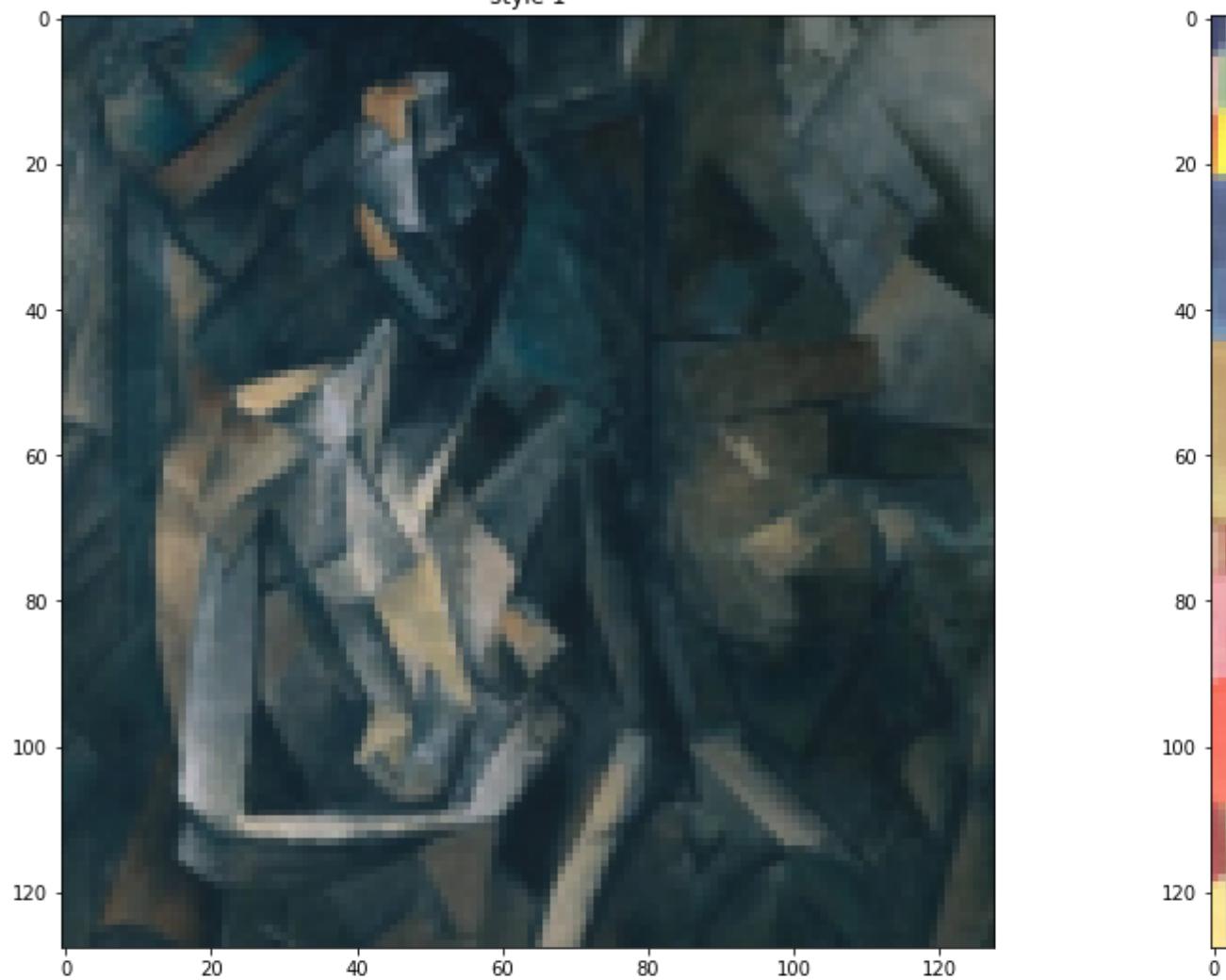




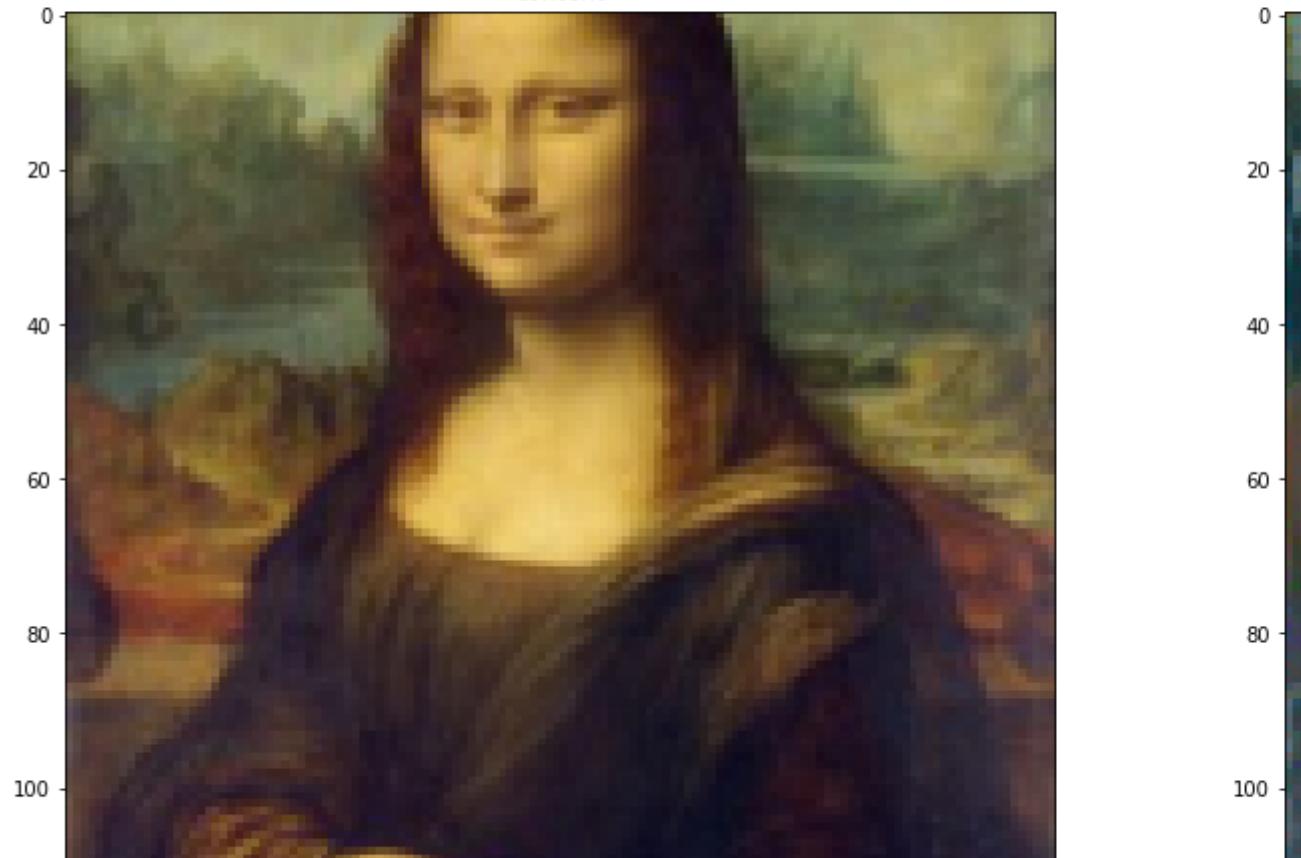
```
1 fig = plt.figure(figsize=(20,20), )
2
3 plt.subplot(2, 2, 1)
4 plt.title('style 1')
5 plt.imshow(tensor_to_matrix(style_img))
6
7 plt.subplot(2, 2, 2)
8 plt.title('style second')
9 plt.imshow(tensor_to_matrix(style_img_second))
10
11 plt.subplot(2, 2, 3)
12 plt.title('content')
13 plt.imshow(tensor_to_matrix(content_img))
14
15 plt.subplot(2, 2, 4)
16 plt.title('outout')
17 plt.imshow(tensor_to_matrix(output))
18
19
20 plt.ioff()
21 plt.show()
```

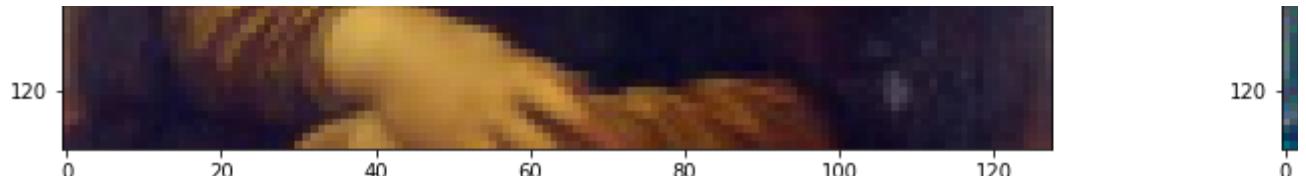


style 1



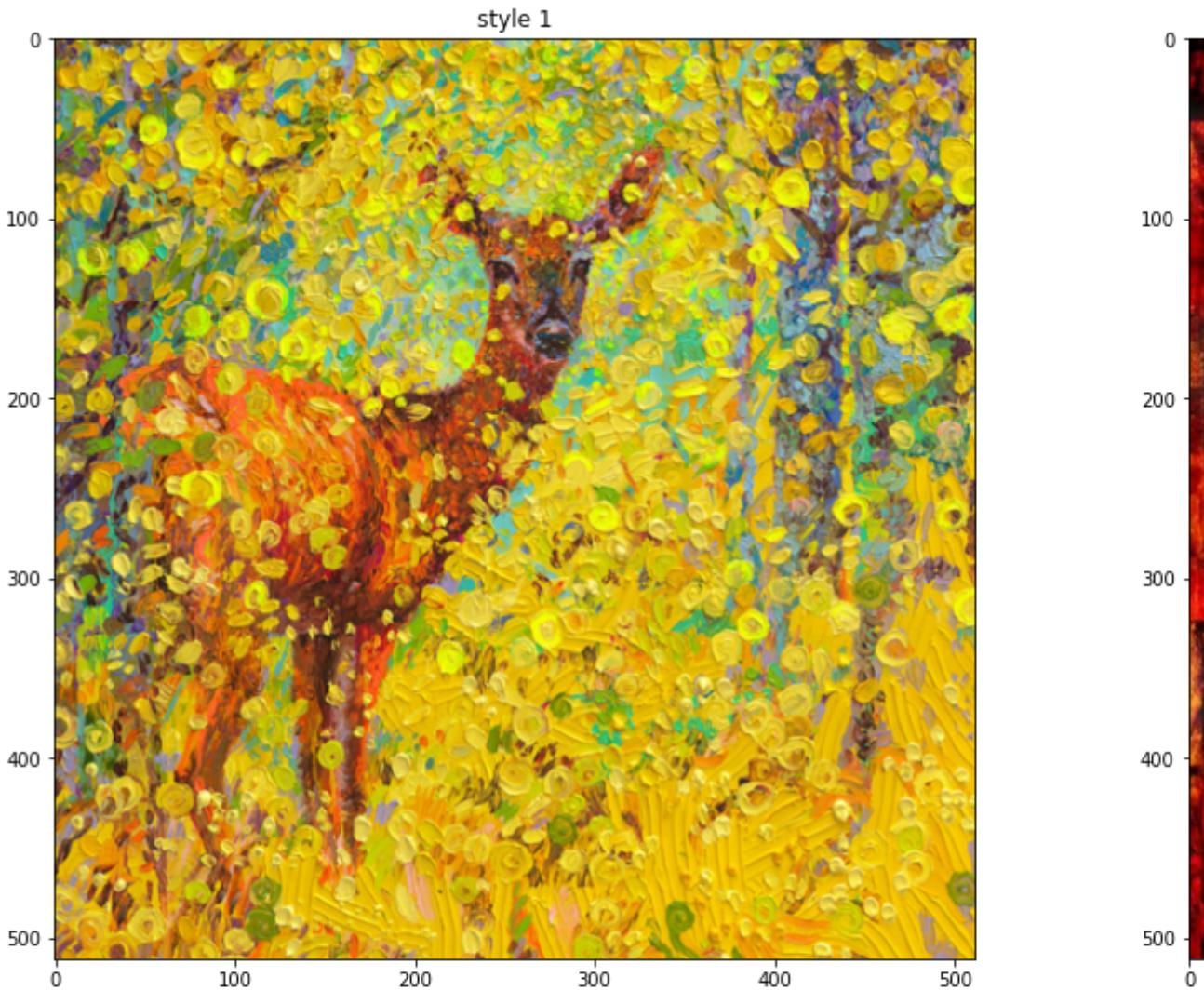
content





```
1 fig = plt.figure(figsize=(20,20), )
2
3 plt.subplot(2, 2, 1)
4 plt.title('style 1')
5 plt.imshow(tensor_to_matrix(style_img))
6
7 plt.subplot(2, 2, 2)
8 plt.title('style second')
9 plt.imshow(tensor_to_matrix(style_img_second))
10
11 plt.subplot(2, 2, 3)
12 plt.title('content')
13 plt.imshow(tensor_to_matrix(content_img))
14
15 plt.subplot(2, 2, 4)
16 plt.title('outout')
17 plt.imshow(tensor_to_matrix(output))
18
19
20 plt.ioff()
21 plt.show()
```





content

```

1 !wget https://art-holst.com.ua/wp-content/uploads/thumb_1_28551.jpg -O /content/images/
2 #content_img = image_loader("images/picasso.jpg")# as well as here
3 #style_img = image_loader("images/lisa.jpg")#измените путь на тот который у вас.
4 style_img = image_loader("images/first_style.jpg")# as well as here
5 style_img_second = image_loader("images/second_style.jpg")# as well as here
6 content_img = image_loader("images/content.jpg")#измените путь на тот который у вас.

```

```

↳ --2020-06-08 12:47:48-- https://art-holst.com.ua/wp-content/uploads/thumb\_1\_28551.jpg
Resolving art-holst.com.ua (art-holst.com.ua)... 104.26.2.77, 104.26.3.77, 172.67.73.
Connecting to art-holst.com.ua (art-holst.com.ua)|104.26.2.77|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 132670 (130K) [image/jpeg]
Saving to: '/content/images/second_style.jpg'

/content/images/sec 100%[=====] 129.56K --.-KB/s in 0.01s

```

2020-06-08 12:47:48 (9.22 MB/s) - '/content/images/second_style.jpg' saved [132670/132670]



```

1 input_img = content_img.clone()
2 # if you want to use white noise instead uncomment the below line:
3 #input_img = torch.randn(content_img.data.size(), device=device)

```