Тема: **Монтаж художественных образов с помощью нейронных сетей ipynb.**

Цель: научиться использовать нейронные сети для монтажа художественных образов.

Материальное и дидактическое оснащение: Методические рекомендации по выполнению практической работы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

ПРАКТИЧЕСКАЯ ЧАСТЬ РАБОТЫ

В ходе изучения теоретических материалов теоретических сведений данной работы:

I. Ознакомьтесь с теоретическими сведениями.

II. Изучите инструкцию ниже.

III. Предоставьте ссылку на свою работу в https://colab.research.google.com/ онлайн, в процессе выполнения преподавателю на yvmranh48@gmail.com, через вкладку Share (поделиться). Для отслеживания Вашей работы преподавателем онлайн и возможности консультирования.

IV. Оформите работу, используя снимки экрана.

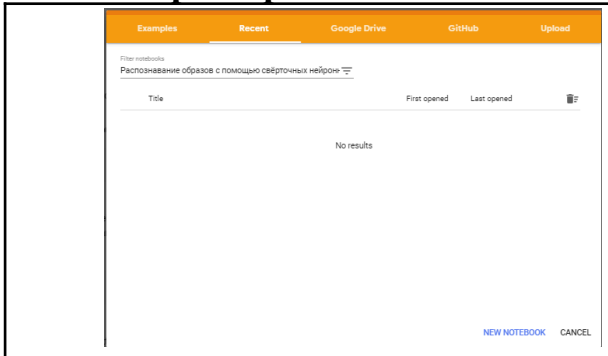
V. Ссылку на работу и отчет по работе выложите в ментальной карте.

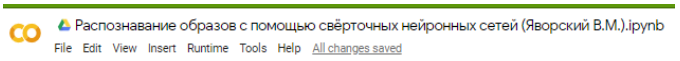
VI. Сделайте алгоритм по Вашей работе в draw.io

VII. Выкладывайте отчет на странице своего сайта.

Ход работы.

Во время работы не забывайте запускать на каждом этапе код.

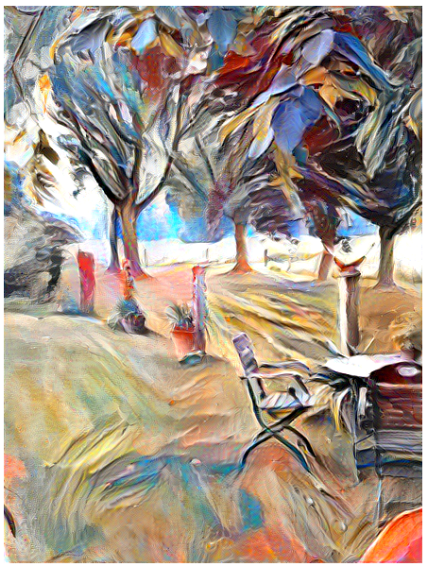
<table>
<tr><td> 

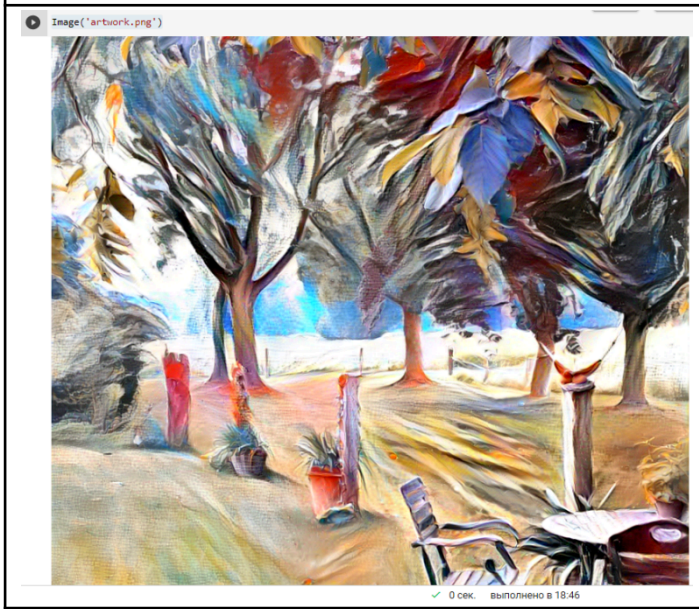

</td><td>Создайте новый ноутбук по ссылке: https://colab.research.google.com/.
Назовите:
Монтаж художественных образов с помощью нейронных сетей (Ф.И.О.).ipynb.
Подпишите в скобках своим Ф.И.О.</td></tr>
<tr><td colspan="2">Ваш новый ноутбук должен содержать комментарии и иметь законченный вид.
Текст для комментариев можем использовать из инструкции</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td> 
</td><td>Рассмотрим, как работают нейронные сети по монтажу художественных образов.
Установим призму.
Добавьте новый код:</td></tr>
</table>

	`!pip install -q git+https://github.com/moritztng/prism.git`
`Building wheel for prism-style-transfer (setup.py) ... done`	Появится надпись: `Building wheel for prism-style-transfer (setup.py) ... done`
`[ ] from IPython.display import Image`	Осуществим импорт библиотек Добавьте новый код: `from IPython.display import Image`

 	Скачаем изображение:
	Добавьте новый код:
	`images_url = 'https://raw.githubusercontent.com/moritztng/prism/master/images'`
	`!wget -q -nc $images_url/content.jpg $images_url/style.jpg`

`Downloading: "https://storage.googleapis.com/prism-weights/vgg19-original.pth" to /root/.cache/torch/hub/checkpoints/vgg19-original.pth`
`100% 76.4M/76.4M [00:00<00:00, 95.3MB/s]`
`2021-05-24 09:27:33.595924: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.11.0`
`Iteration: 50 Time: 0.0512 Loss: 1.0e+06 Content Loss: 4.4e+05 Style Loss: 5.7e+02 Mean Abs Grad: 6.2e-01 Grad Zeros: 0 Grad Scale: 8.0e+00`
`Iteration: 100 Time: 0.0574 Loss: 5.9e+05 Content Loss: 4.0e+05 Style Loss: 1.9e+02 Mean Abs Grad: 9.7e-02 Grad Zeros: 0 Grad Scale: 1.6e+01`
`Iteration: 150 Time: 0.0634 Loss: 5.0e+05 Content Loss: 3.7e+05 Style Loss: 1.3e+02 Mean Abs Grad: 4.0e-02 Grad Zeros: 0 Grad Scale: 3.2e+01`
`Iteration: 200 Time: 0.064 Loss: 4.7e+05 Content Loss: 3.6e+05 Style Loss: 1.1e+02 Mean Abs Grad: 6.1e-02 Grad Zeros: 0 Grad Scale: 6.4e+01`
`Iteration: 250 Time: 0.0643 Loss: 4.5e+05 Content Loss: 3.5e+05 Style Loss: 1.0e+02 Mean Abs Grad: 2.4e-02 Grad Zeros: 0 Grad Scale: 1.3e+02`
`Iteration: 300 Time: 0.0645 Loss: 4.3e+05 Content Loss: 3.4e+05 Style Loss: 9.3e+01 Mean Abs Grad: 2.3e-02 Grad Zeros: 0 Grad Scale: 2.6e+02`
`Iteration: 350 Time: 0.0647 Loss: 4.2e+05 Content Loss: 3.3e+05 Style Loss: 8.8e+01 Mean Abs Grad: 1.5e-02 Grad Zeros: 0 Grad Scale: 5.1e+02`
`Iteration: 400 Time: 0.0649 Loss: 4.2e+05 Content Loss: 3.3e+05 Style Loss: 8.5e+01 Mean Abs Grad: 1.2e-02 Grad Zeros: 0 Grad Scale: 1.0e+03`
`Iteration: 450 Time: 0.0651 Loss: 4.1e+05 Content Loss: 3.3e+05 Style Loss: 8.3e+01 Mean Abs Grad: 1.1e-02 Grad Zeros: 0 Grad Scale: 2.0e+03`
`Iteration: 500 Time: 0.0654 Loss: 4.1e+05 Content Loss: 3.3e+05 Style Loss: 8.1e+01 Mean Abs Grad: 9.2e-03 Grad Zeros: 0 Grad Scale: 4.1e+03`
	Перенос стиля в оболочке
	Добавьте новый код:
	`!style-transfer content.jpg style.jpg --style_weight 1000 --use_amp`

	Выводим полученное изображение Добавьте новый код: `Image('artwork.png')`
`2021-05-30 15:51:08.846854: I tensorflow/stream_executor/platform/default/dso_loader.cc:53] Successfully opened dynamic library libcudart.so.11.0`	Осуществим тонкую настройку Добавьте новый код: `# Use exactly the same parameters as previously, but add "--init_i`

Iteration: 50 Time: 0.192 Loss: 4.8e+05 Content Loss: 4.0e+05 Style Loss: 7.7e+01 Mean Abs Grad: 2.7e-02 Grad Zeros: 0 Grad Scale: 1.0e+03 Iteration: 100 Time: 0.22 Loss: 3.5e+05 Content Loss: 3.0e+05 Style Loss: 4.4e+01 Mean Abs Grad: 1.2e-02 Grad Zeros: 0 Grad Scale: 2.0e+03 Iteration: 150 Time: 0.242 Loss: 3.1e+05 Content Loss: 2.7e+05 Style Loss: 3.8e+01 Mean Abs Grad: 7.9e-03 Grad Zeros: 0 Grad Scale: 4.1e+03 Iteration: 200 Time: 0.247 Loss: 2.9e+05 Content Loss: 2.5e+05 Style Loss: 3.5e+01 Mean Abs Grad: 6.4e-03 Grad Zeros: 0 Grad Scale: 8.2e+03	mg artwork.png --area 1024 --iter 200". !style-transfer content.jpg style.jpg --style_weight 1000 --use_amp --init_img artwork.png --area 1024 --iter 200
	Получим изображение Добавьте новый код: `Image('artwork.png')`

[18] `from PIL import Image`
`from style_transfer.learn import StyleTransfer`
	Передача стилей в Python.
	Импортируем библиотеки
	Добавьте новый код:
	`from PIL import Image`
	`from style_transfer.learn import StyleTransfer`

	Создание экземпляра объекта передачи стиля.
	Добавьте новый код:
	`style_transfer = StyleTransfer(lr=1,`
	`content_weight=1,`
	`style_weight=1e3,`
	`content_weights="{'relu_4_2':1}",`
	`style_weights="{'relu_1_1':1,'relu`

```
2_1':1,'relu_3_1':1,'relu_4_1':1,
'relu_5_1':1}",
                                av
g_pool=False,
                                fe
ature_norm=True,
                                we
ights='original',
                                pr
eserve_color='style',
                                de
vice='auto',
                                us
e_amp=False,
                                ad
am=False,
                                op
tim_cpu=False,
                                lo
gging=50)
```

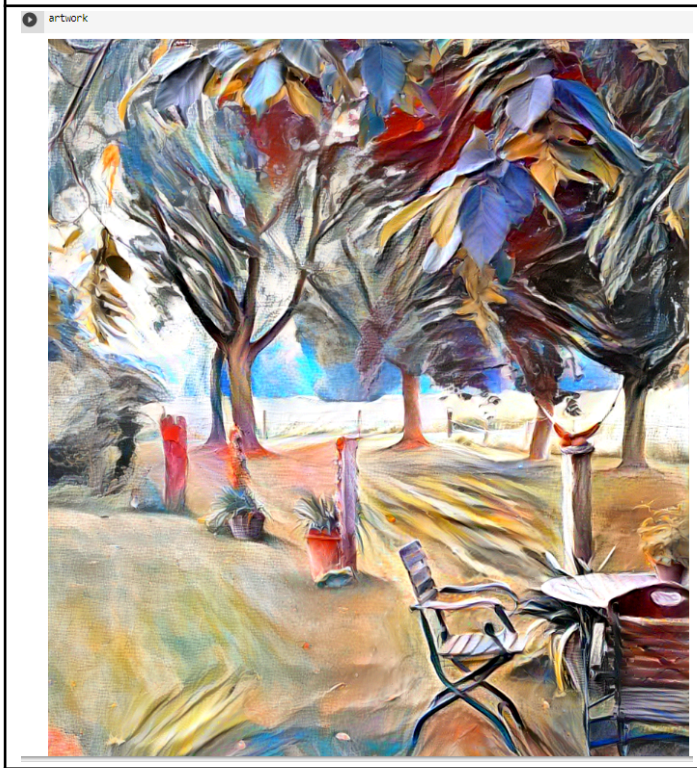
```
Iteration: 50        Time: 0.089
Loss: 8.1e+05        Content Loss:
4.3e+05        Style Loss: 3.8e+02
Mean Abs Grad: 5.7e-01        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 100       Time: 0.0965
Loss: 5.6e+05        Content Loss:
3.9e+05        Style Loss: 1.7e+02
Mean Abs Grad: 8.4e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 150       Time: 0.101
Loss: 4.9e+05        Content Loss:
3.6e+05        Style Loss: 1.3e+02
Mean Abs Grad: 4.2e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 200       Time: 0.102
Loss: 4.6e+05        Content Loss:
3.5e+05        Style Loss: 1.1e+02
Mean Abs Grad: 2.8e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 250       Time: 0.103
Loss: 4.4e+05        Content Loss:
3.4e+05        Style Loss: 9.8e+01
Mean Abs Grad: 2.3e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 300       Time: 0.104
Loss: 4.3e+05        Content Loss:
3.4e+05        Style Loss: 9.1e+01
Mean Abs Grad: 2.9e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 350       Time: 0.105
Loss: 4.2e+05        Content Loss:
3.3e+05        Style Loss: 8.7e+01
Mean Abs Grad: 1.6e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
Iteration: 400       Time: 0.106
Loss: 4.2e+05        Content Loss:
3.3e+05        Style Loss: 8.4e+01
Mean Abs Grad: 2.1e-02        Grad
Zeros: 0    Grad Scale: 6.6e+04
```

Выполним перенос стиля.

Добавьте новый код:

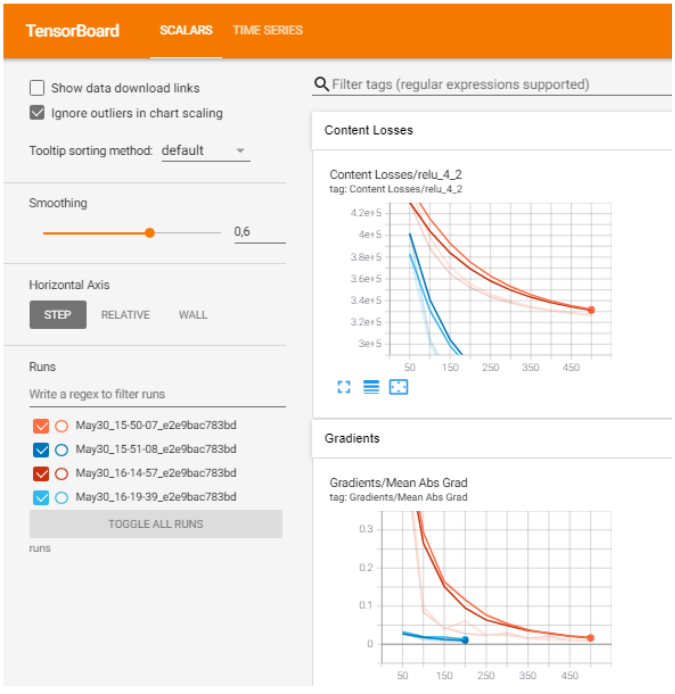
```
artwork = style_transfer(Image.op
en('content.jpg'), Image.open('st
yle.jpg'),
                            area=512
, init_random=False, init_img=Non
e, iter=500)
artwork.save('artwork.png')
```

```
Iteration: 450        Time: 0.107
Loss: 4.1e+05         Content Loss:
3.3e+05         Style Loss: 8.2e+01
Mean Abs Grad: 1.0e-02         Grad
Zeros: 0     Grad Scale: 6.6e+04
    Iteration: 500        Time: 0.108
   Loss: 4.1e+05         Content Loss:
   3.3e+05         Style Loss: 8.0e+01
   Mean Abs Grad: 9.6e-03         Grad
    Zeros: 0     Grad Scale: 6.6e+04
```



Получим изображение.
Добавьте новый код:

```
artwork
```



Определим потери на участке.
Добавьте новый код:

```
%load_ext tensorboard
%tensorboard --logdir runs
```

Мы натренировали модель нейронной сети для преобразования изображений. Самостоятельно загрузим новое изображение в этот блокнот преобразуем изображение и выгрузим новое полученное изображение на Google Drive.

Составьте алгоритм. Работу выполняйте онлайн.
Ссылку на онлайн файл отправьте в чат ивстройте в ментальную карту.
Можете встроить отчет в на страницу своего сайта.