

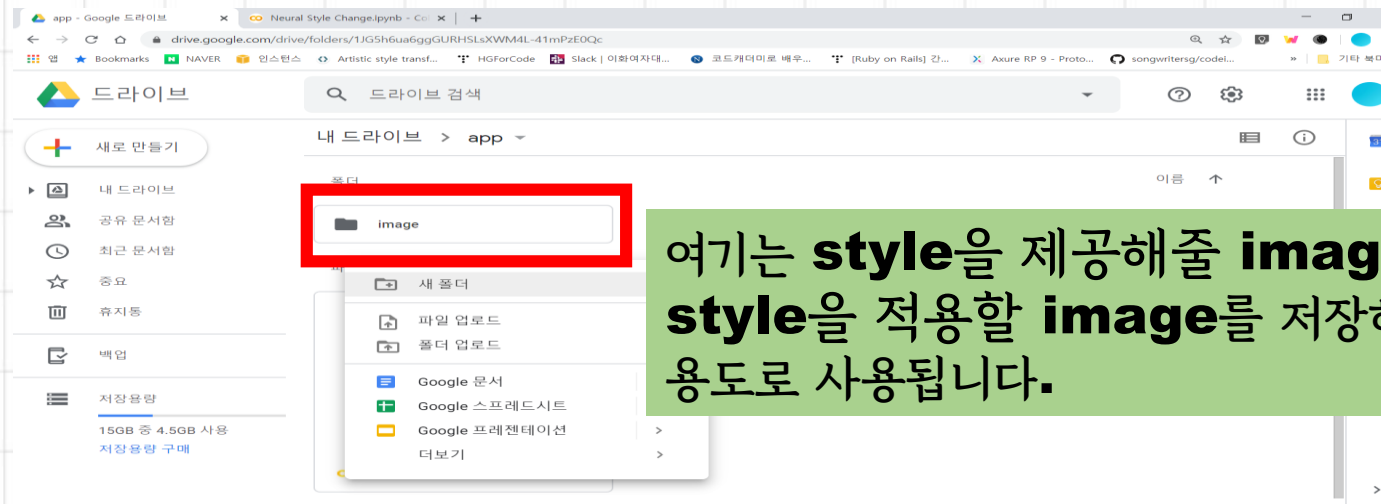
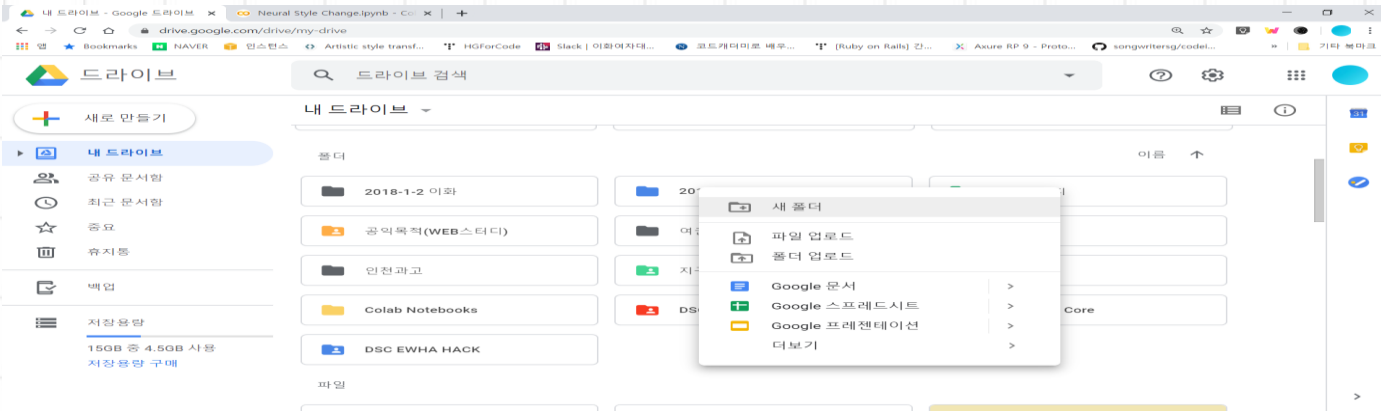
Artistic style transfer & other advanced image editing

이화여자대학교 소프트웨어학부 컴퓨터공학전공 18
이하경

구글 Colab을 사용하기 위한 준비

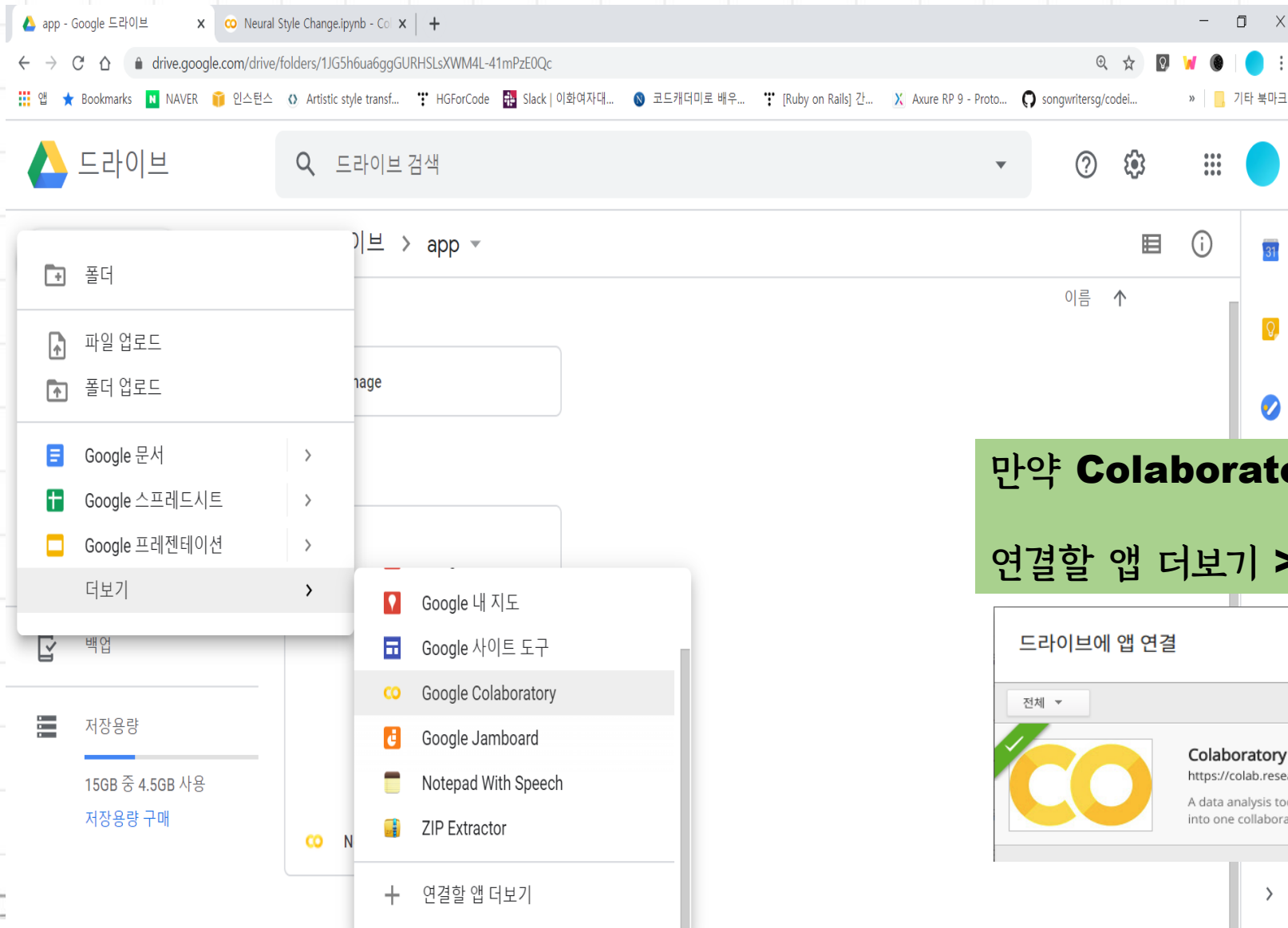
1. 구글 드라이브에서 폴더 생성

새 폴더 → **app**이라는 이름의 새 폴더를 만들었습니다. → 그리고 **app** 안에 **image**라는 폴더를 만들었습니다.

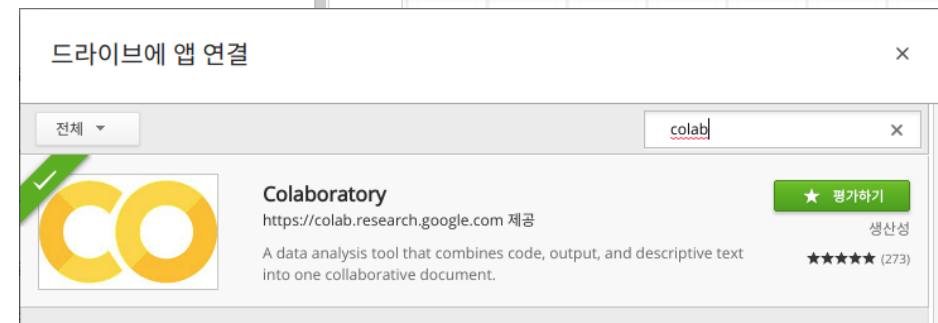


여기는 **style**을 제공해줄 **image**와 **style**을 적용할 **image**를 저장하는 용도로 사용됩니다.

2. Colab Notebook 파일 생성



만약 **Colaboratory**가 보이지 않는다면,
연결할 앱 더보기 > **colab** 검색 > 연결



3. 무료 GPU 설정

수정 → 노트 설정 → 런타임 유형 변경 선택을 통해
하드웨어 가속기를 **GPU**로 변경

구글 드라이브를 colab에서 어떻게 사용하는지 (자세한 건 ppt)

```
[1] from google.colab import drive
drive.mount('/content/drive/')

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=947318989803-6b6668d9f4n4q30fee6491hc0brc41.apps.googleusercontent.com&redirect\_uri=urn%3Aietf%3Awg%3Aoauth%3A2.OX3a0ob&response\_type=code&scope=email%20https%3A%2F%2Fwww.c

Enter your authorization code:
.....
Mounted at /content/drive/

[44] !ls "/content/drive/My Drive/app"

Image: 'Neural Style Change.ipynb'

[4] %matplotlib inline

[5] from __future__ import print_function

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

from PIL import Image
import matplotlib.pyplot as plt

import torchvision.transforms as transforms
import torchvision.models as models

import copy

[6] device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

[29] # desired size of the output image
imgsize = 512 if torch.cuda.is_available() else 128 # use small size if no gpu
```

노트 설정

런타임 유형
Python 3

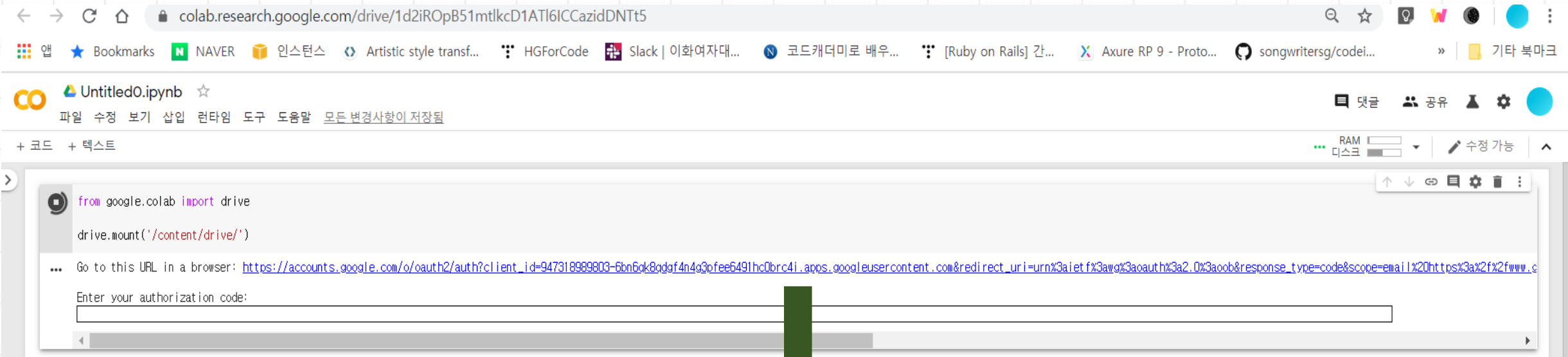
하드웨어 가속기
GPU

☐ 이 노트를 저장할 때 코드 셀 출력 생략

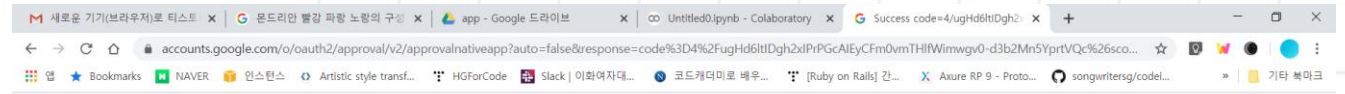
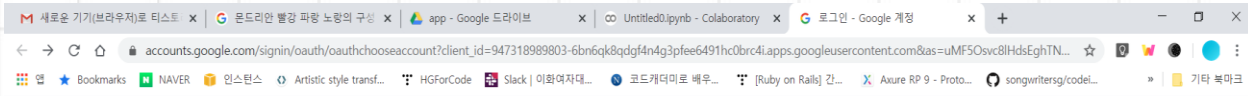
취소 저장

4. 구글 Colab에서 구글 Drive의 파일 가져오기 (디렉토리 설정)

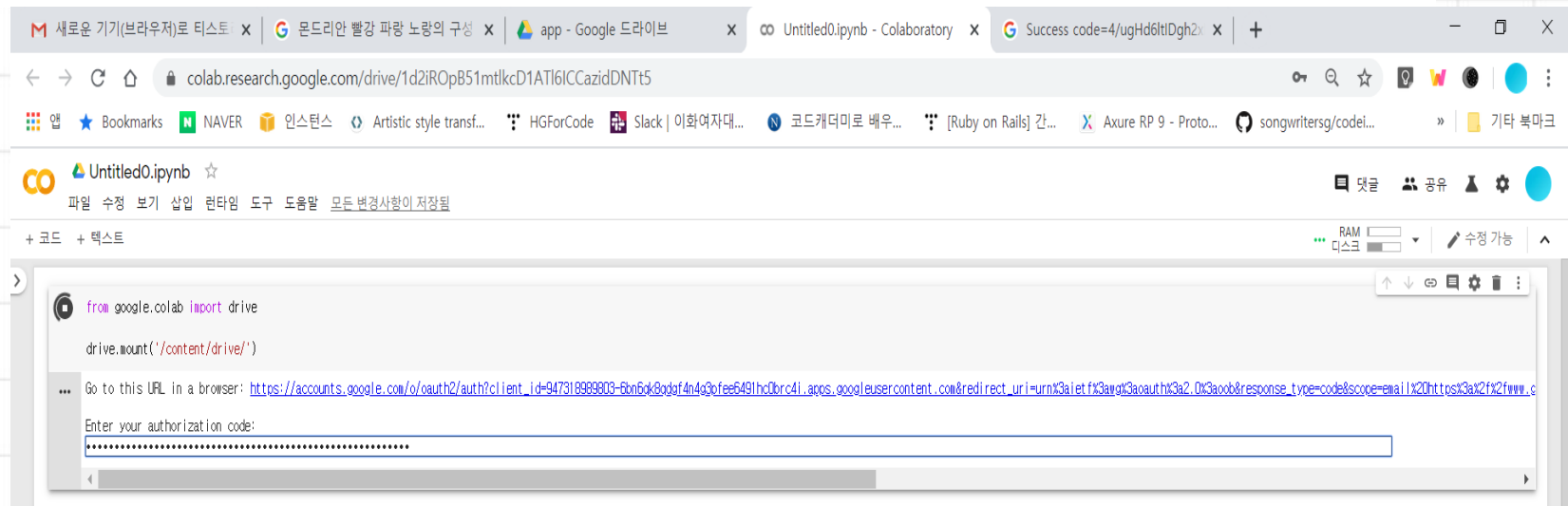
```
from google.colab import drive
drive.mount('/content/drive/')
```



링크 클릭 → 구글 로그인



인증코드를 복사하여 텍스트 박스에 붙여 넣기



```
from google.colab import drive
drive.mount('/content/drive/')

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=9473189989803-6bn6qk8gdg4n4g3ofee6491hc0brc4i.apps.googleusercontent.com&redirect\_uri=urn%3Aietf%3Awg%3Aoauth%3A2.O%3Aaob&response\_type=code&scope=email%20https%3A%2F%2Fwww.c

Enter your authorization code:

Mounted at /content/drive/
```

!ls "/content/drive/My Drive/app"

[44] !ls "/content/drive/My Drive/app"

image 'Neural Style Change.ipynb'

```
!ls "/content/drive/My Drive"

여권          'DSC Ewha Core'
오소플        'DSC EWEA HACK'
지구별        지구.gdoc
인천과고      서명_난쏘공.gdoc
개발스터디    소물발표_이하경.hwp
'도형의방정식답(2015-2019).zip'  활동일지서식.hwp
'2018-1-2 이화'  hwp00001.png
'최종결과보고서서식(2019.06.07).hwp' hwp00002.png
2019-1        hwp00003.png
소물2019_이화여대이하경_.pdf  IMG_0072.jpg
소물2019_이화여대이하경.pptx  IMG_0092.PNG
app           'Sketches 12.PNG'
'Colab Notebooks'  '공익목적(웹스터디)'.
'DSC EWEA'        이하경_wednesday.pdf
```


본격적으로 오늘 함께 할 내용 소개

Leon A. Gatys와 Alexander S. Ecker, Matthias Bethge 가 개발한 알고리즘인 **Neural-Style** 을 직접 실습해보겠습니다!

<https://github.com/jcjohnson/neural-style>

Branch: master New pull request Find file Clone or download

jcjohnson Update README.md Latest commit 07c4b82 on 16 Apr 2017



examples	correct starry stanford result	3 years ago
models	Merged from jcjohnson/neural-style master branch	4 years ago
.gitignore	Merged from jcjohnson/neural-style master branch	4 years ago
INSTALL.md	Update INSTALL.md	4 years ago
LICENSE	add license	4 years ago
README.md	Update README.md	3 years ago
neural_style.lua	fix bug with -normalize_gradient	3 years ago

README.md

neural-style

This is a torch implementation of the paper [A Neural Algorithm of Artistic Style](#) by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge.

The paper presents an algorithm for combining the content of one image with the style of another image using convolutional neural networks. Here's an example that maps the artistic style of [The Starry Night](#) onto a night-time photograph of the Stanford campus:



Cornell University We gratefully acknowledge support from the Simons Foundation and member institutions.

arXiv.org > cs > arXiv:1508.06576 Search All fields Search Help | Advanced Search

Computer Science > Computer Vision and Pattern Recognition

A Neural Algorithm of Artistic Style

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge

(Submitted on 26 Aug 2015 (v1), last revised 2 Sep 2015 (this version, v2))

In fine art, especially painting, humans have mastered the skill to create unique visual experiences through composing a complex interplay between the content and style of an image. Thus far the algorithmic basis of this process is unknown and there exists no artificial system with similar capabilities. However, in other key areas of visual perception such as object and face recognition near-human performance was recently demonstrated by a class of biologically inspired vision models called Deep Neural Networks. Here we introduce an artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images. Moreover, in light of the striking similarities between performance-optimised artificial neural networks and biological vision, our work offers a path forward to an algorithmic understanding of how humans create and perceive artistic imagery.

Subjects: **Computer Vision and Pattern Recognition (cs.CV)**; Neural and Evolutionary Computing (cs.NE); Neurons and Cognition (q-bio.NC)

Cite as: arXiv:1508.06576 [cs.CV]
(or arXiv:1508.06576v2 [cs.CV] for this version)

Submission history

From: Leon Gatys [view email]
[v1] Wed, 26 Aug 2015 17:14:42 UTC (5,474 KB)
[v2] Wed, 2 Sep 2015 08:24:59 UTC (5,474 KB)

Which authors of this paper are endorsers? [Disable MathJax (What is MathJax?)]

Download:

- PDF
- Other formats (download)

Current browse context: **cs.CV**
< prev | next >
new | recent | 1508

Change to browse by:
cs
cs.NE
q-bio
q-bio.NC

References & Citations

- NASA ADS

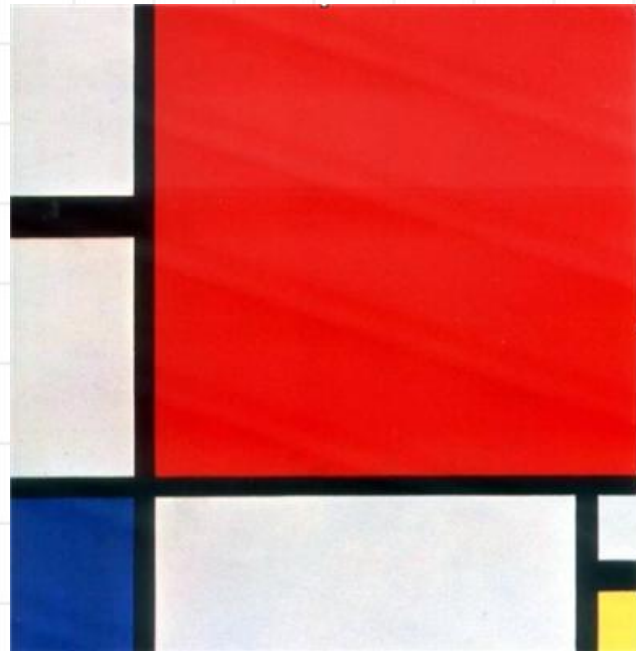
25 blog links (what is this?)

DBLP - CS Bibliography
listing | bibtext
Leon A. Gatys
Alexander S. Ecker
Matthias Bethge

Bookmark
Add to collection

신경망 스타일(Neural-Style)

콘텐츠 이미지(예, 무용수)와 스타일 이미지(예, 몬드리안 작품)을 입력으로 받아
콘텐츠 이미지의 모양대로 스타일 이미지의 '그리는 방식'을 이용해 그린 것처럼 결과를 내는 알고리즘



어떻게 동작할까요?? $\Delta(\triangleleft)\triangleright$

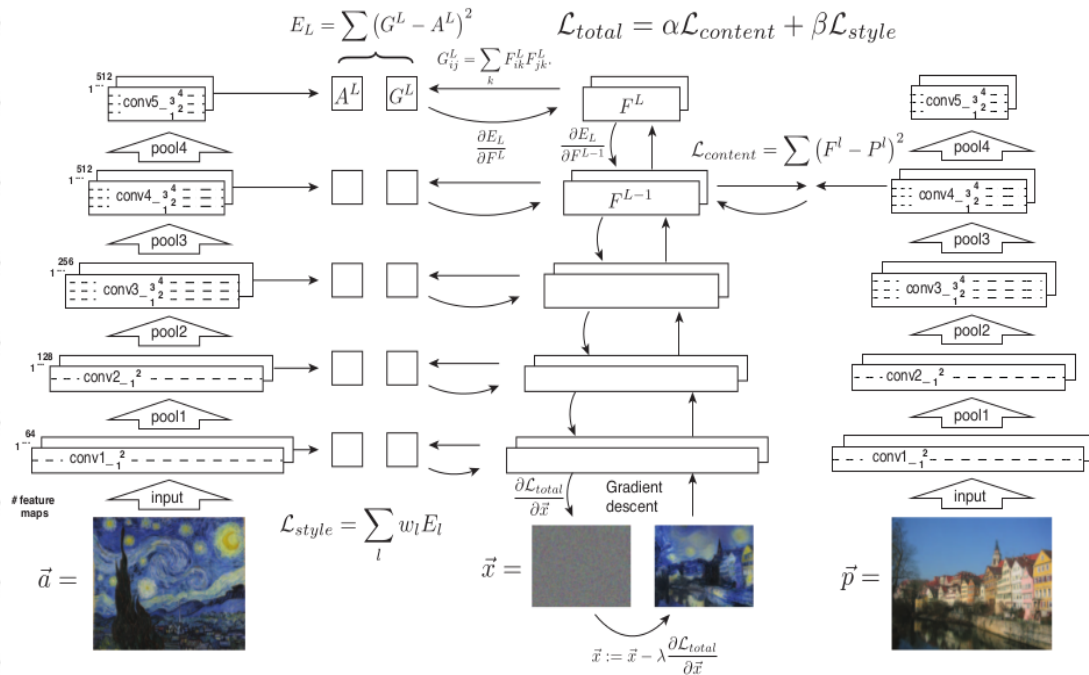
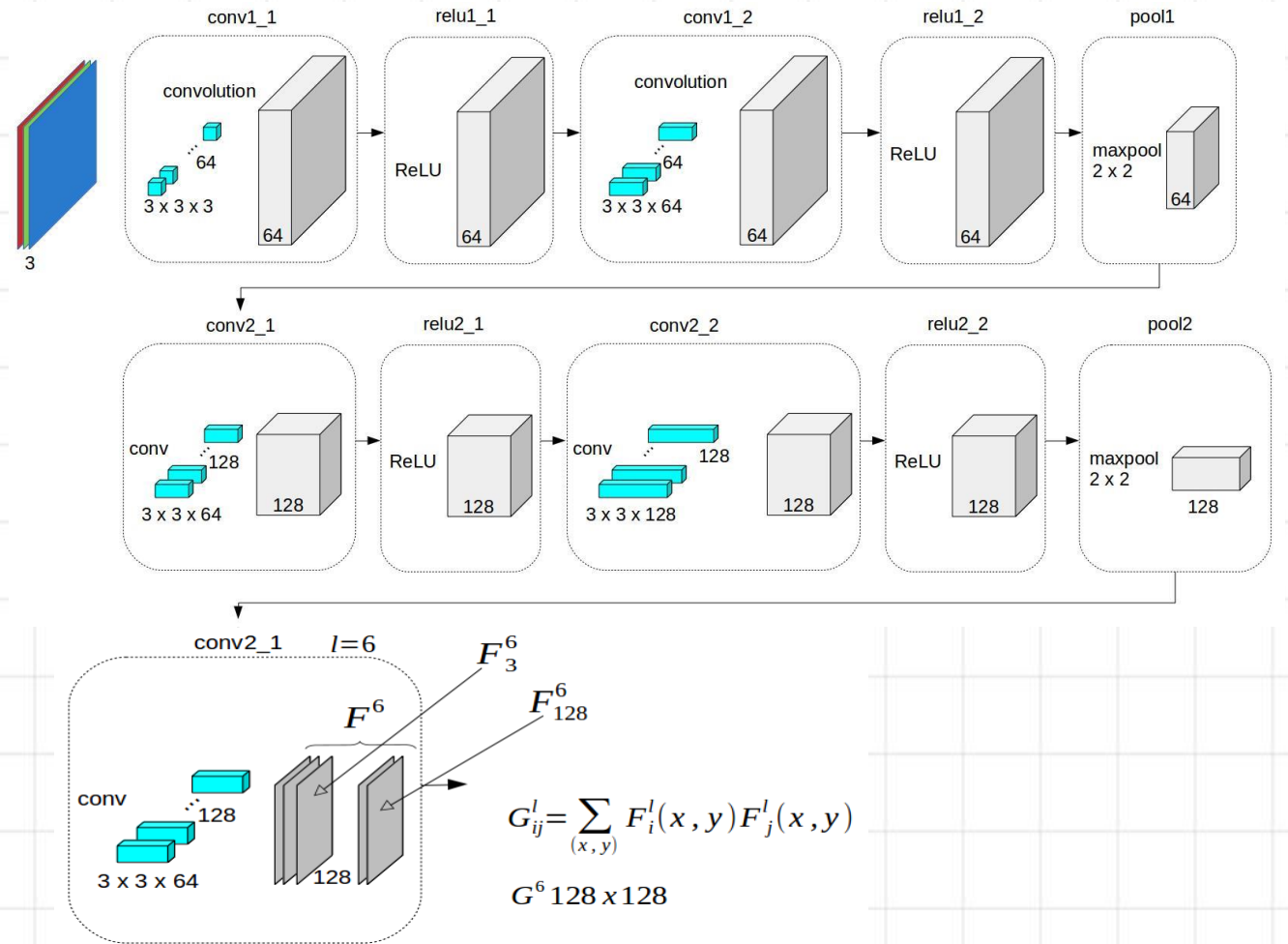


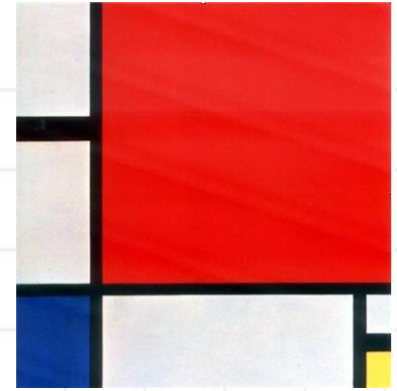
Figure 2. Style transfer algorithm. First content and style features are extracted and stored. The style image \vec{a} is passed through the network and its style representation A^l on all layers included are computed and stored (left). The content image \vec{p} is passed through the network and the content representation P^l in one layer is stored (right). Then a random white noise image \vec{x} is passed through the network and its style features G^l and content features F^l are computed. On each layer included in the style representation, the element-wise mean squared difference between G^l and A^l is computed to give the style loss \mathcal{L}_{style} (left). Also the mean squared difference between F^l and P^l is computed to give the content loss $\mathcal{L}_{content}$ (right). The total loss \mathcal{L}_{total} is then a linear combination between the content and the style loss. Its derivative with respect to the pixel values can be computed using error back-propagation (middle). This gradient is used to iteratively update the image \vec{x} until it simultaneously matches the style features of the style image \vec{a} and the content features of the content image \vec{p} (middle, bottom).



어떻게 동작할까요?? ((˘(˘ω˘)˘))

D_C 콘텐츠 이미지와 스타일 이미지 간의 콘텐츠가 얼마나 차이가 있는지 측정

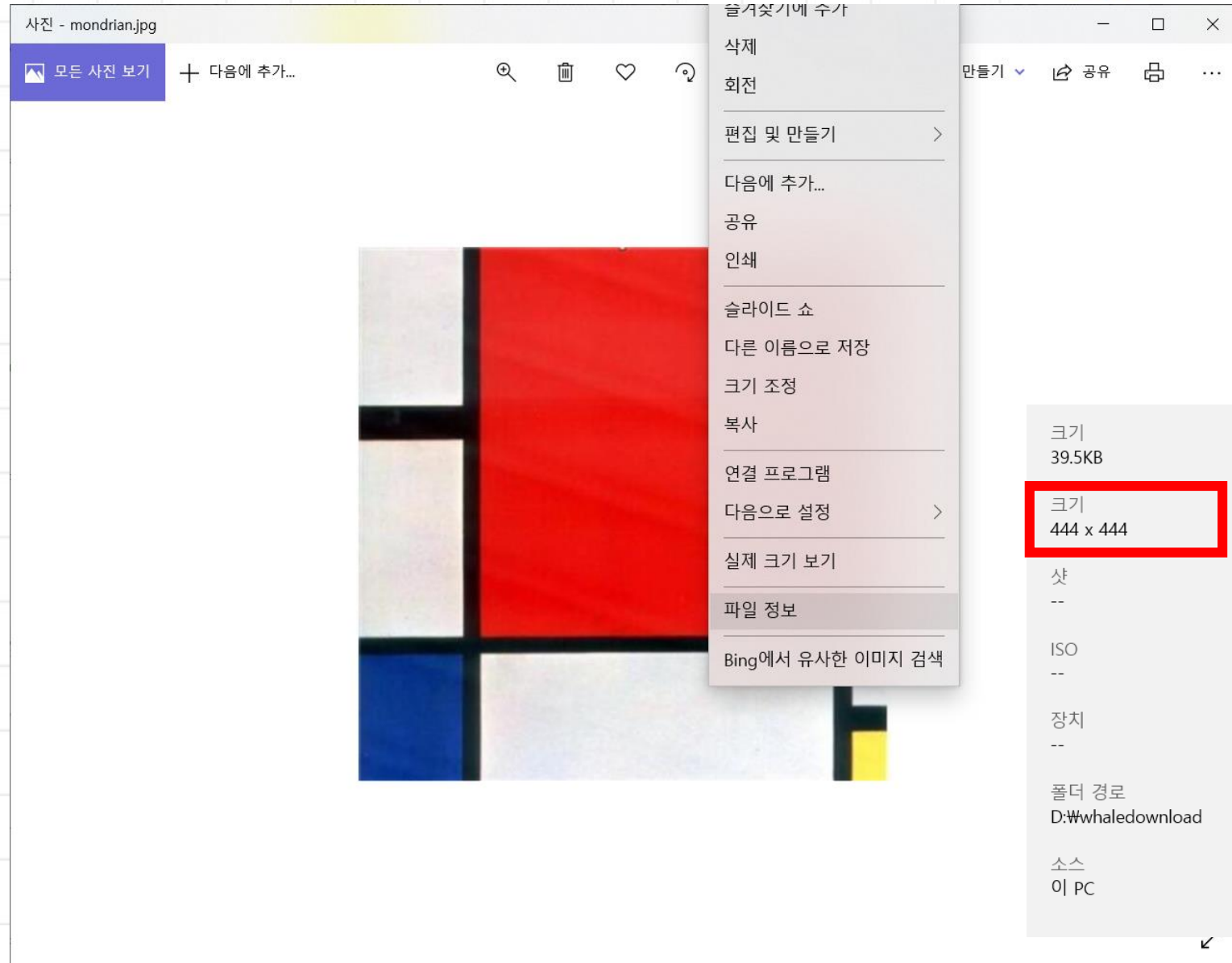
D_S 콘텐츠 이미지와 스타일 이미지 간의 스타일에 얼마나 차이가 있는지를 측정



※ 따라서 콘텐츠 이미지와 스타일 이미지의 크기가 같아야 함!!

세 번째 이미지(우리가 결국 구하고자 하는 스타일 변환된 이미지)를, 콘텐츠 이미지와의 콘텐츠 거리 및 스타일 이미지와의 스타일 거리를 최소화하는 방향으로 세 번째 이미지를 변환





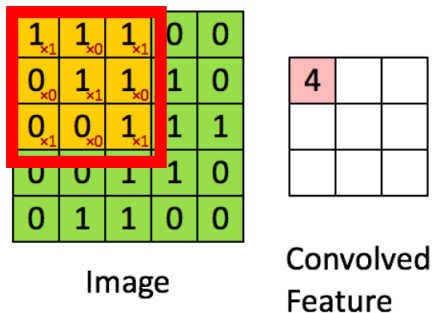
C_{nn} 사전 훈련된 깊은 합성곱 신경망 네트워크

2차원 입력 데이터(Shape: (5,5))를 1개의 필터로 합성곱 연산을 수행하고, 합성곱 처리 결과로 부터 Feature Map을 만든다.

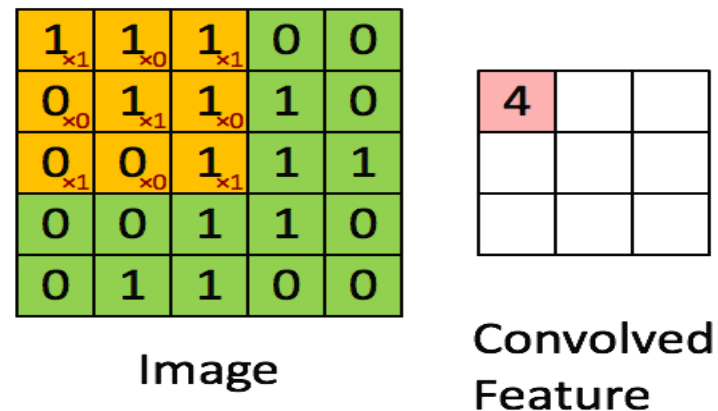
필터는 이미지의 특징을 찾아내기 위한 공용 파라미터

CNN에서 Filter와 Kernel은 같은 의미

필터는 일반적으로 (4, 4)이나 (3, 3)과 같은 정사각 행렬로 정의



$$\begin{aligned}
 &1 \times 1 + 1 \times 0 + 1 \times 1 \\
 &+ 0 \times 0 + 1 \times 1 \\
 &+ 1 \times 0 + 0 \times 1 + \\
 &0 \times 0 + 1 \times 1 \\
 &= 4
 \end{aligned}$$



하나의 Step

Input

1	0	0	0
1	1	1	1
0	1	0	0
1	1	0	1

Filter

1	0	0
0	1	1
1	1	1

Convolution

1	0	0	0
0	1	1	1
0	1	1	0
1	1	0	1

$$\begin{aligned}
 &1*1+0*0+0*0+ \\
 &0*1+1*1+1*1+ \\
 &0*1+1*1+0*1 \\
 &=4
 \end{aligned}$$

Feature Map

4		

<http://taewan.kin>

Step-1

Feature Map

Step-4

Feature Map

Step-2

Feature Map

Step-5

Feature Map

Step-3

Feature Map

Step-9

Feature Map

C_{nn} 사전 훈련된 깊은 합성곱 신경망 네트워크 X, Y 어떤 이미지 (X와 Y는 같은 크기) L 레이어 (의 레벨)



$C_{nn}(X)$ 어떤 이미지 X를 입력으로 해서 CNN 을 통과한 네트워크(모든 레이어들의 특징 맵(feature map)을 포함)



$F_{XL} \in C_{nn}(X)$ 깊이 레벨 L에서의 특징 맵(feature map)을 의미



레이어 L 에 해당하는 **컨텐츠**의 거리를 정의 :

$$D_C^L(X, Y) = \|F_{XL} - F_{YL}\|^2 = \sum_i (F_{XL}(i) - F_{YL}(i))^2$$

스타일 G_{XL} 의 X 레이어에서 L 은 모든 벡터화된 특징 맵(feature map) F_{XL}^k

$$G_{XL}(k, l) = \langle F_{XL}^k, F_{XL}^l \rangle = \sum_i F_{XL}^k(i) \cdot F_{XL}^l(i)$$

벡터화 곱을 의미

F_{XL}^k F_{XL}^l 벡터화(vectorized)되고 연결된(concatenated) 하나의 단일 벡터

레이어 L 에 해당하는 **스타일**의 거리를 정의 :

$$D_S^L(X, Y) = \|G_{XL} - G_{YL}\|^2 = \sum_{k,l} (G_{XL}(k, l) - G_{YL}(k, l))^2$$

$D_C(X, C)$ 의 한 번의 최소화를 위해서, 이미지 변수 X 와 대상 콘텐츠-이미지 C

$D_S(X, C)$ 와 X 와 대상 스타일-이미지 S 둘 다 여러 레이어들에 대해서 계산 되어야 하고,

우리는 원하는 레이어 각각에서의 거리의 그래디언트를 계산하고 더한다.

: 이 과정의 수식은 복잡하므로 생략 ^^;

$$\nabla_{extittotal}(X, S, C) = \sum_{L_C} w_{CL_C} \cdot \nabla_{extitcontent}^{L_C}(X, C) + \sum_{L_S} w_{SL_S} \cdot \nabla_{extitstyle}^{L_S}(X, S)$$

PyTorch

**다행히 pytorch에는 우리가 필요한 모든 함수가 이미 라이브러리로 내장되어 있습니다..!!
(진짜 다행..ㅎㅎ)**

**실제로 PyTorch를 사용하면 라이브러리의 함수를 사용하는 동안 모든 그래디언트가 자동,
동적으로 계산을 해줍니다..!!**

- torch , torch.nn, numpy (PyTorch로 신경망 처리를 위한 필수 패키지)
- torch.optim (효율적인 그라디언트 디센트)
- PIL , PIL.Image , matplotlib.pyplot (이미지를 읽고 보여주는 패키지)
- torchvision.transforms (PIL타입의 이미지들을 토치 텐서 형태로 변형해주는 패키지)
- torchvision.models (사전 훈련된 모델들의 학습 또는 읽기 패키지)
- copy (모델들의 깊은 복사를 위한 시스템 패키지)

```
from __future__ import print_function

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

from PIL import Image
import matplotlib.pyplot as plt

import torchvision.transforms as transforms
import torchvision.models as models

import copy
```

쿠다(CUDA)

컴퓨터에 GPU가 있는 경우, 특히 VGG와 같이 깊은 네트워크를 사용하려는 경우 알고리즘을 CUDA 환경에서 실행

↓
우리는 VGG 19 를 사용할 예정

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
# 출력 이미지의 원하는 크기를 정하기.
imgsize = 512 if torch.cuda.is_available() else 128 # use small size if no gpu

loader = transforms.Compose([
    transforms.Resize(imgsize), # 입력 영상 크기를 맞춤
    transforms.ToTensor()]) # 토치 텐서로 변환

def image_loader(image_name):
    image = Image.open(image_name)
    # 네트워크의 입력 차원을 맞추기 위해 필요한 가짜 배치 차원
    image = loader(image).unsqueeze(0)
    return image.to(device, torch.float)

style_img = image_loader("/content/drive/My Drive/app/image/mondrian.jpg")
content_img = image_loader("/content/drive/My Drive/app/image/dancing.jpg")

assert style_img.size() == content_img.size(), \
    "we need to import style and content images of the same size"
```

- 이미지는 0에서 255 사이의 이미지 픽셀값을 가진다.
- 토치 텐서로 변환하면 0에서 1의 값으로 변환
- 토치 라이브러리의 신경망은 0에서 1의 텐서 이미지로 학습

이미지를 표시하기 위해 plt.imshow 를 이용 / 텐서를 PIL 이미지로 변환

```
unloader = transforms.ToPILImage() # PIL 이미지로 재변환
plt.ion()

def imshow(tensor, title=None):
    image = tensor.cpu().clone() # 텐서의 값에 변화가 적용되지 않도록 텐서를 복제
    image = image.squeeze(0) # 페이크 배치 차원을 제거
    image = unloader(image)
    plt.imshow(image)
    if title is not None:
        plt.title(title)
    plt.pause(0.001) # 그리는 부분이 업데이트 될 수 있게 잠시 정지

plt.figure()
imshow(style_img, title='Style Image')

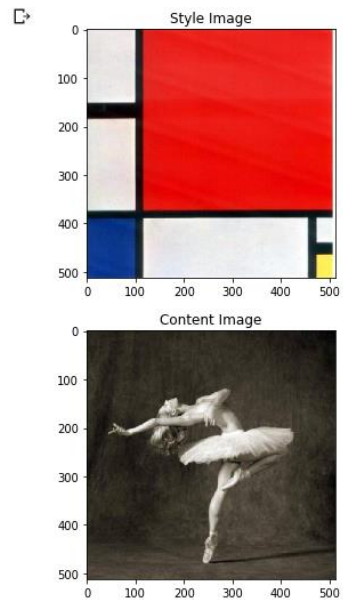
plt.figure()
imshow(content_img, title='Content Image')
```



```
[30] image = image.squeeze() # remove the final batch dimension
      image = unloader(image)
      plt.imshow(image)
      if title is not None:
          plt.title(title)
      plt.pause(0.001) # pause a bit so that plots are updated

      plt.figure()
      imshow(style_img, title='Style Image')

      plt.figure()
      imshow(content_img, title='Content Image')
```



콘텐츠 로스

콘텐츠 로스는 네트워크에서 X 로 입력을 받았을 때 레이어 L 에서 특징 맵(feature map) F_{XL} 을 입력으로 가져 와서 이 이미지와 콘텐츠 이미지 사이의 가중치 콘텐츠 거리 $w_{CL} \cdot D_C^L(X, C)$ 를 반환하는 기능.

```
class ContentLoss(nn.Module):

    def __init__(self, target,):
        super(ContentLoss, self).__init__()
        # 그라디언트를 동적으로 계산하는 데 사용되는 트리에서 대상 콘텐츠를 '분리'
        # :이 값은 변수(variable)가 아니라 명시된 값,
        # 그렇지 않으면 기준의 전달 메소드가 오류를 발생
        self.target = target.detach()

    def forward(self, input):
        self.loss = F.mse_loss(input, self.target)
        return input
```

스타일 로스

```
def gram_matrix(input):  
    a, b, c, d = input.size() # a=배치 크기(=1)    # b=특징 맵의 크기  
    # (c,d)=특징 맵(N=c*d)의 차원  
  
    features = input.view(a * b, c * d) # F_XL을 \hat F_XL로 크기 조정  
  
    G = torch.mm(features, features.t()) # 그램 곱을 수행  
  
    # 그램 행렬의 값을 각 특징 맵의 요소 숫자로 나누는 방식으로 '정규화'를 수행  
    # 그램 행렬은 선형대수학에서 배우게 됩니다..^^  
  
    return G.div(a * b * c * d)
```

특징 맵(feature map) 차원 :math:'N'이 클수록, 그램(Gram) 행렬의 값이 커지는 특징이 있다.

스타일 로스 모듈은 콘텐츠 로스 모듈과 완전히 동일한 방식으로 구현되지만 대상과 입력 간의 그램 매트릭스의 차이를 비교

```
class StyleLoss(nn.Module):  
  
    def __init__(self, target_feature):  
        super(StyleLoss, self).__init__()  
        self.target = gram_matrix(target_feature).detach()  
  
    def forward(self, input):  
        G = gram_matrix(input)  
        self.loss = F.mse_loss(G, self.target)  
        return input
```

새로운 기기(브라우저)로 열기 x | G 몬드리안 빨강 파랑 노랑의 x | image - Google 드라이브 x | Untitled0.ipynb - Colaboratory x | Success code=4/ugHd6ltlDc x | neural_style_tutorial.ipynb x | + - □ ×

← → ↻ 🏠 colab.research.google.com/drive/1d2iOPB51mtlkD1ATl6lCCazidDNTt5#scrollTo=I3bCpkRbSkQq 🔍 ☆ 🔒 W 🌙 | 🔵 ⋮

📱 앱 ★ Bookmarks 📄 NAVER 📁 인스턴스 🔄 Artistic style transf... ⚙️ HGForCode 🗨️ Slack | 이화여자대... 🌐 코드캐더미로 배우... ⚙️ [Ruby on Rails] 간... 📄 Axure RP 9 - Proto... 🔄 songwritersg/codei... » 📁 기타 북마크

🔵 Untitled0.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

+ 코드 + 텍스트

RAM 디스크 수정 가능

```
[13] features = input.view(a + b, c + d) # resize F_XL into what F_XL
      G = torch.mm(features, features.t()) # compute the gram product
      # we 'normalize' the values of the gram matrix
      # by dividing by the number of element in each feature maps.
      return G.div(a + b + c + d)

[14] class StyleLoss(nn.Module):
    def __init__(self, target_feature):
        super(StyleLoss, self).__init__()
        self.target = gram_matrix(target_feature).detach()

    def forward(self, input):
        G = gram_matrix(input)
        self.loss = F.mse_loss(G, self.target)
        return input

cnn = models.vgg19(pretrained=True).features.to(device).eval()

... Downloading: "https://download.pytorch.org/models/vgg19-dcbb9e9d.pth" to /root/.cache/torch/checkpoints/vgg19-dcbb9e9d.pth
82% |██████████| 450M/548M [00:09<00:03, 33.7MB/s]

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

    # create a module to normalize input image so we can easily put it in a
    # nn.Sequential
    class Normalization(nn.Module):
        def __init__(self, mean, std):
            super(Normalization, self).__init__()
            # .view the mean and std to make them [C x 1 x 1] so that they can
            # directly work with image Tensor of shape [B x C x H x W].
            # B is batch size, C is number of channels, H is height and W is width.
            self.mean = torch.tensor(mean).view(-1, 1, 1)
            self.std = torch.tensor(std).view(-1, 1, 1)

        def forward(self, img):
            # normalize img
            return (img - self.mean) / self.std
```

PyTorch의 VGG 구현은 두 개의 하위 순차 모듈로 나뉘어

특징(features) 모듈 : 합성곱과 풀링 레이어들을 포함

분류(classifier) 모듈 : fully connected 레이어들을 포함

```
cnn = models.vgg19(pretrained=True).features.to(device).eval()
```

VGG 네트워크는 평균 = [0.485, 0.456, 0.406] 및 표준편차 = [0.229, 0.224, 0.225]로 정규화 된 각 채널의 이미지에 대해 학습된 모델

```
cnn_normalization_mean = torch.tensor([0.485, 0.456, 0.406]).to(device)
```

```
cnn_normalization_std = torch.tensor([0.229, 0.224, 0.225]).to(device)
```

입력 이미지를 정규화하는 모듈을 만들어 nn.Sequential에 쉽게 입력

nn.Sequential

```
class Normalization(nn.Module):
```

```
    def __init__(self, mean, std):
```

```
        super(Normalization, self).__init__()
```

```
        # .view(텐서의 모양을 바꾸는 함수)로 평균과 표준 편차 텐서를 [C x 1 x 1] 형태로 만들어
```

```
        # 바로 입력 이미지 텐서의 모양인 [B x C x H x W] 에 연산할 수 있도록 만들
```

```
        # B는 배치 크기, C는 채널 값, H는 높이, W는 넓이
```

```
        self.mean = torch.tensor(mean).view(-1, 1, 1)
```

```
        self.std = torch.tensor(std).view(-1, 1, 1)
```

```
    def forward(self, img):
```

```
        # img 값 정규화(normalize)
```

```
        return (img - self.mean) / self.std
```

알고리즘의 저자인 Len Gatys 가 제안한 방식을 그대로 가져다 쓰면 됨.

```
def get_input_optimizer(input_img):  
    # 이 줄은 입력은 그레이던트가 필요한 파라미터라는 것을 보여주기 위해 있습니다.  
    optimizer = optim.LBFGS([input_img.requires_grad_()])  
    return optimizer
```

마지막 단계: 경사 하강의 반복. 각 단계에서 우리는 네트워크의 새로운 로스를 계산하기 위해 업데이트 된 입력을 네트워크에 공급해야 한다. 우리는 그래디언트를 동적으로 계산하고 해당 단계를 수행하기 위해 각 손실의 역방향(backward) 메소드를 실행해야 한다. 옵티마이저는 인수로서 "클로저(closure)"를 필요: 즉, 모델을 재평가하고 로스를 반환 하는 함수.

최적화 된 이미지는 0 과 1 사이에 머물지 않고 무한대 사이의 값을 가질 수도 있다.
→ 따라서 입력 이미지가 올바른 범위의 값을 유지할 수 있도록 제약 조건 하에서 최적화를 수행
→ closure()

```
def run_style_transfer(cnn, normalization_mean, normalization_std,  
                      content_img, style_img, input_img, num_steps=300,  
                      style_weight=1000000, content_weight=1):
```

```
·  
·  
·
```

```
def closure():  
    # 입력 이미지의 업데이트된 값들을 보정  
    input_img.data.clamp_(0, 1)
```

```
·  
·  
·  
·
```

```
# 마지막 보정...  
input_img.data.clamp_(0, 1)
```

```
return input_img
```


마지막으로, 알고리즘을 실행!!

```
output = run_style_transfer(cnn, cnn_normalization_mean, cnn_normalization_std,  
                             content_img, style_img, input_img)  
  
plt.figure()  
imshow(output, title='Output Image')  
  
# sphinx_gallery_thumbnail_number = 4  
plt.ioff()  
plt.show()
```

새로운 기기(브라우저)로 떠... x | G 몬드리안 빨강 파랑 노랑의... x | image - Google 드라이브 x | Untitled0.ipynb - Colaborat... x | Success code=4/ugHd6ltlDc x | neural_style_tutorial.ipynb - x | +

← → ↺ 🏠 colab.research.google.com/drive/1d2iOpB51mtlkC1ATI6ICCazidDNTt5#scrollTo=ZlwZ6kqcSvDF 🔍 ☆ 🔒 W 🌙 | 🔵 ⋮

📱 앱 ★ Bookmarks 📄 NAVER 📦 인스턴스 🔄 Artistic style transf... ⚙️ HGForCode 🗨️ Slack | 이화여자대... 🌐 코드캐더미로 배우... ⚙️ [Ruby on Rails] 간... 🔄 Axure RP 9 - Proto... 🔄 songwritersg/codei... » 📌 기타 북마크

CO Untitled0.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

+ 코드 + 텍스트

RAM 디스크 수정 가능

>

[20] run[0] += 1
if run[0] % 50 == 0:
 print("run {}".format(run))
 print('Style Loss : {:.4f} Content Loss: {:.4f}'.format(
 style_score.item(), content_score.item()))
 print()

 return style_score + content_score

optimizer.step(closure)

a last correction...
input_img.data.clamp_(0, 1)

return input_img

🔍

output = run_style_transfer(cnn, cnn_normalization_mean, cnn_normalization_std,
 content_img, style_img, input_img)

plt.figure()
imshow(output, title='Output Image')

sphinx_gallery_thumbnail_number = 4
plt.ioff()
plt.show()

... Building the style transfer model..
Optimizing..
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
if sys.path[0] == '':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:13: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
del sys.path[0]
run [50]:
Style Loss : 4.287450 Content Loss: 4.239298

#DeveloperStudentClubs

새로운 기기(브라우저) × | 몬드리안 빨강 파랑 노 × | image - Google 드라 × | Untitled0.ipynb - Colab × | Success code=4/ugHd × | neural_style_tutorial.ip × | 파일 다운로드 | iLoveI × | + - □ ×

← → ↺ 🏠 colab.research.google.com/drive/1d2iROpB51mtlkcD1ATl6ICCazidDNTt5#scrollTo=ZlwZ6kqcSvDF 🔍 ☆ 🔒 W ⚫ | 🔵 ⋮

📱 앱 ★ Bookmarks 🟢 NAVER 📁 인스턴스 🔄 Artistic style transf... ⚙️ HGForCode 🗨️ Slack | 이화여자대... 🌐 코드캐더미로 배우... ⚙️ [Ruby on Rails] 간... 🌐 Axure RP 9 - Proto... 🗨️ songwritersg/codei... » | 📁 기타 북마크

🟠 Untitled0.ipynb ☆
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

▶ ▶

```
plt.plot()
plt.show()
```

▶ ▶ Building the style transfer model..
Optimizing..
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
if sys.path[0] == '':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:13: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
del sys.path[0]
run [50]:
Style Loss : 870.882996 Content Loss: 11.984343

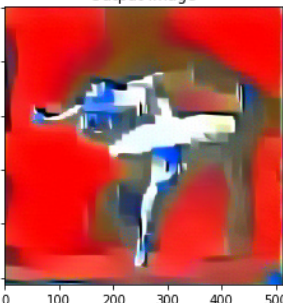
run [100]:
Style Loss : 233.821716 Content Loss: 11.688787

run [150]:
Style Loss : 141.016861 Content Loss: 11.156803

run [200]:
Style Loss : 96.386230 Content Loss: 10.689357

run [250]:
Style Loss : 70.850586 Content Loss: 10.455500

run [300]:
Style Loss : 76.800179 Content Loss: 10.649892

▶ ▶ 

감사합니다 o(*'▽'*)/☆°,'