

Transfer Learning & Style Transfer

2017.11.25

최건호

01

Transfer Learning

- 정의
- 이유
- 활용

02

Style Transfer

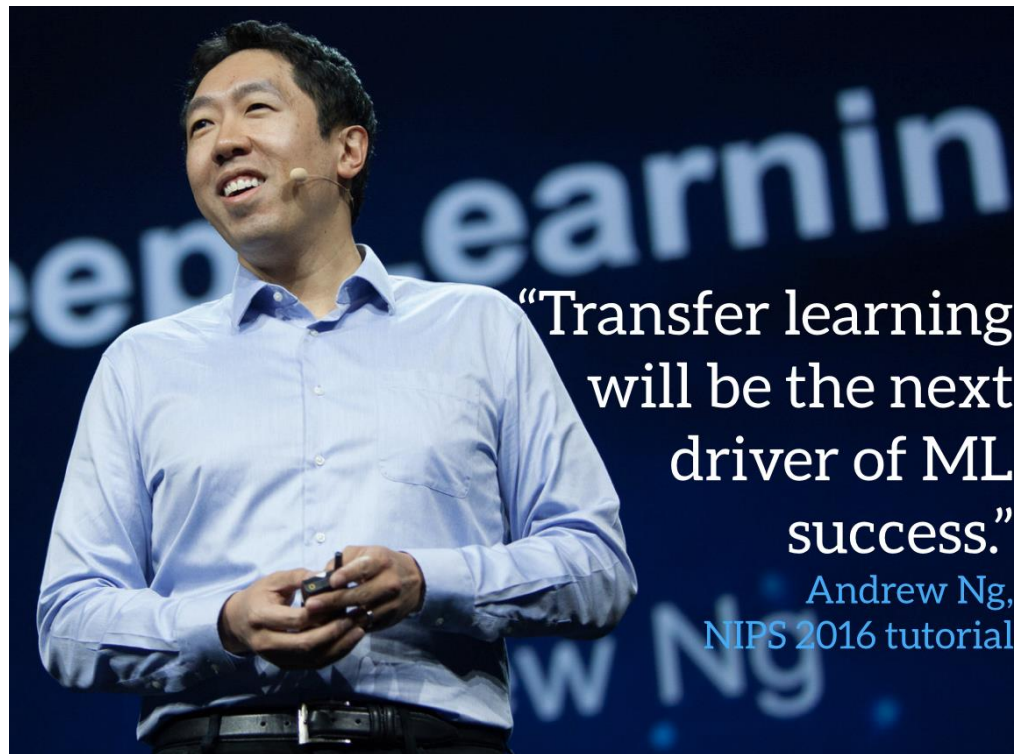
- 소개
- 원리
- 구현

03

t-SNE Visualization

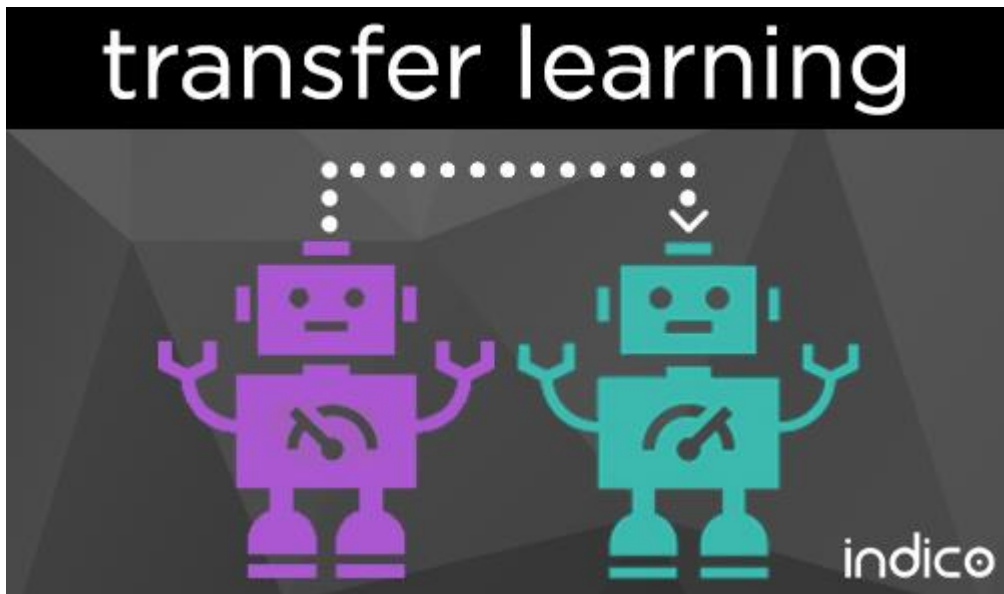
- 소개
 - 활용 사례
-

Transfer Learning



<https://www.youtube.com/watch?v=F1ka6a13S9I>

Transfer Learning



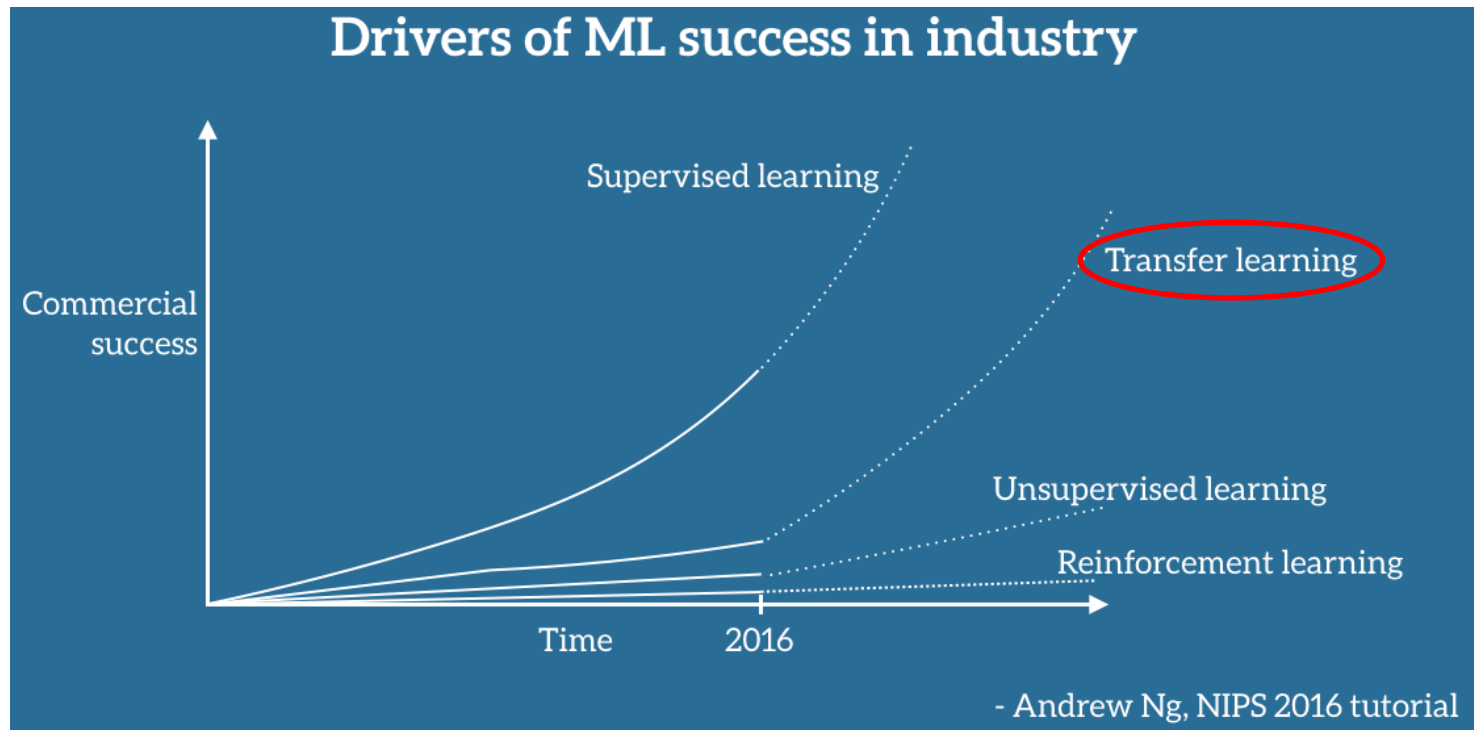
매번 모든 태스크를 처음부터
학습 시키기에는 데이터도 시
간도 부족하다



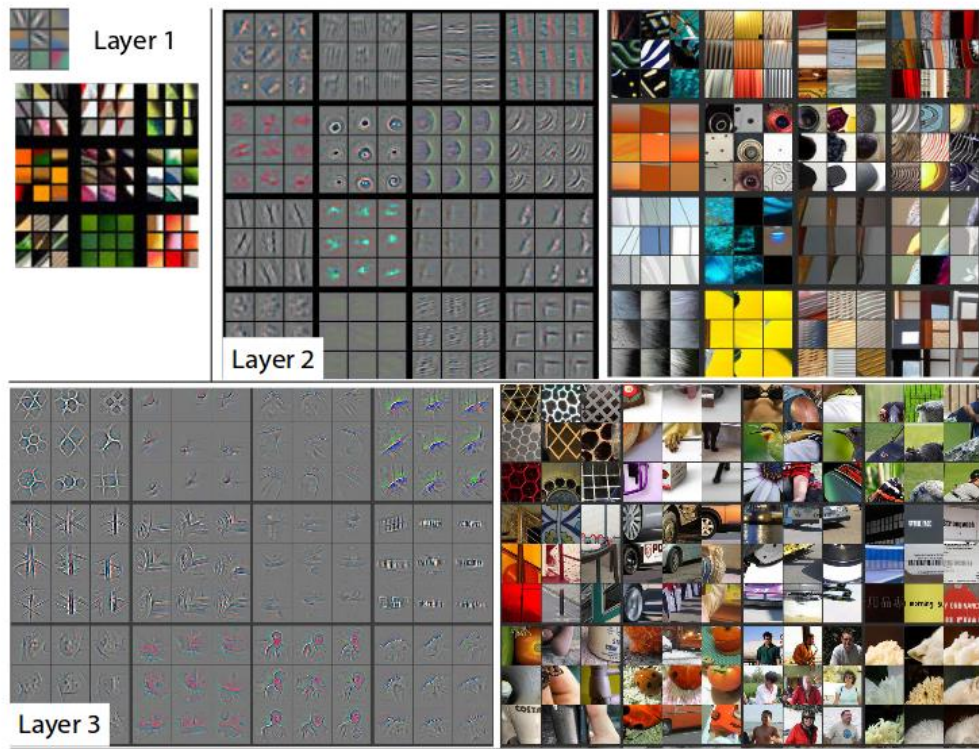
누군가 학습 시켜놓은 모델을
활용하여 새로운 데이터, 태
스크에 적용할 수 있을까?

Transfer Learning!!

Transfer Learning



Transfer Learning



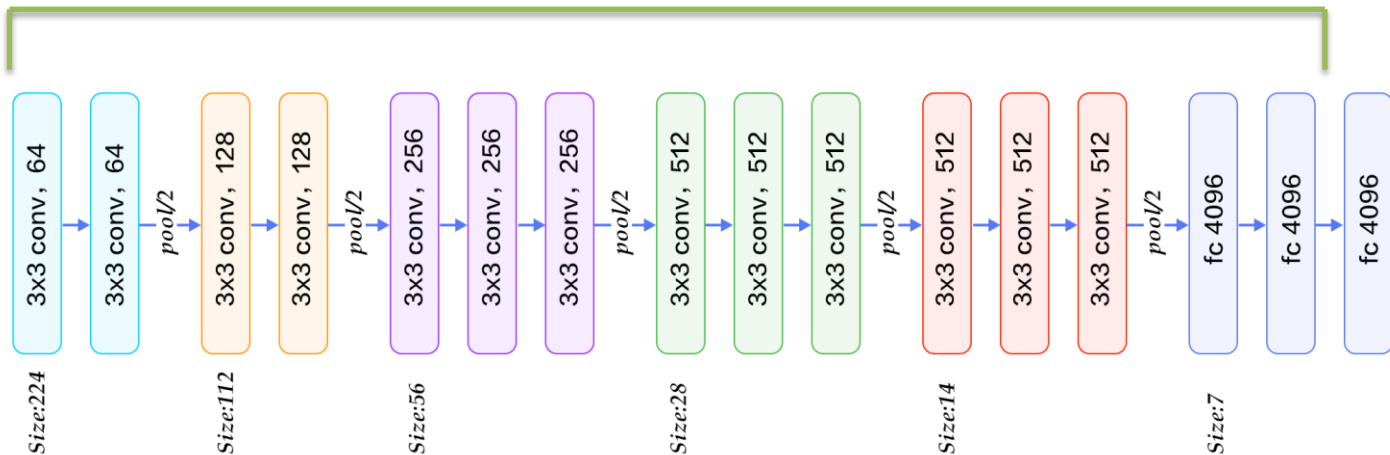
ImageNet 데이터로 학습된 모델들은 주어진 라벨 카테고리를 구분하기 위해 간단한 형태부터 복잡한 형태의 필터들을 생성함



하지만 이 필터들은 사실 universal한 형태들이기 때문에 다른 이미지 데이터에서도 사용할 수 있음

Transfer Learning

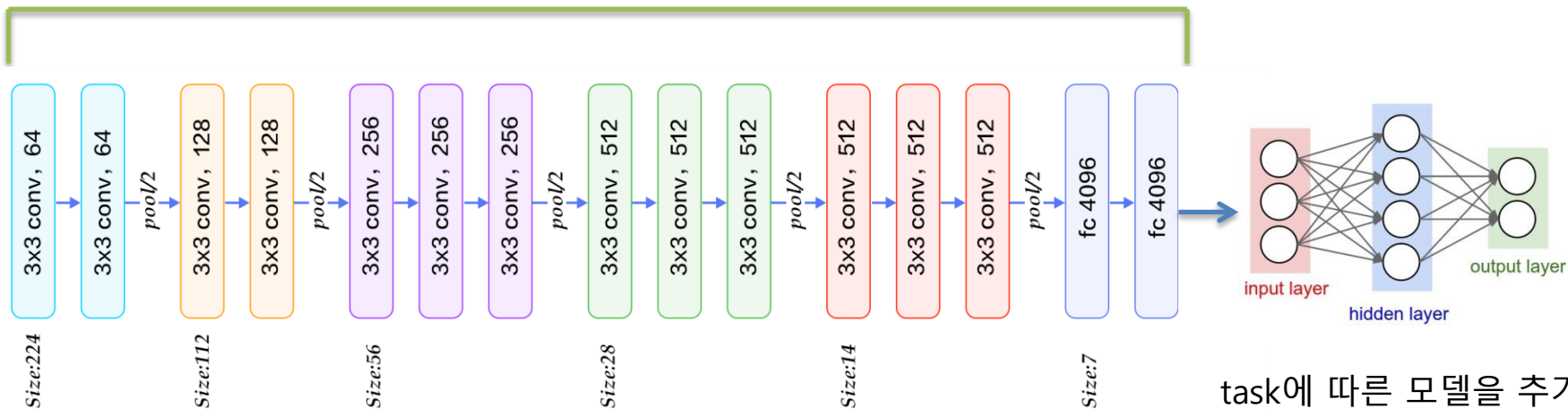
feature extractor



VGGNet

Transfer Learning

feature extractor



VGGNet

task에 따른 모델을 추가

학습의 범위는?

Transfer Learning

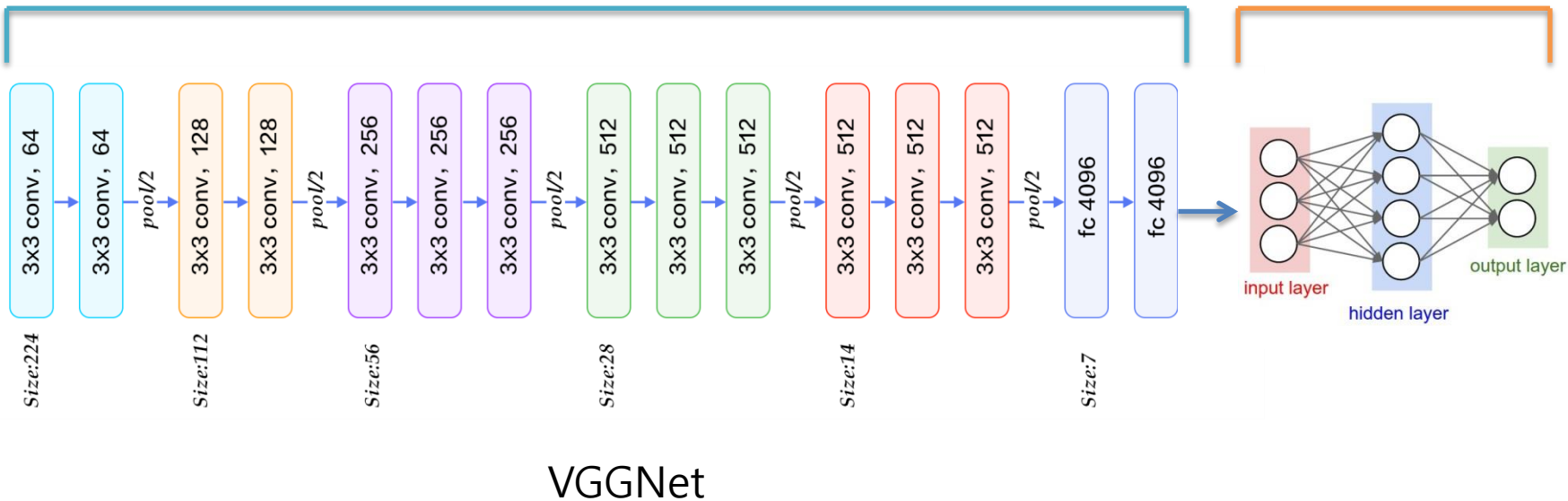
데이터 수가 적을 때

```
resnet = Resnet().cuda()

for param in resnet.parameters():
    param.requires_grad = False
```

Freeze

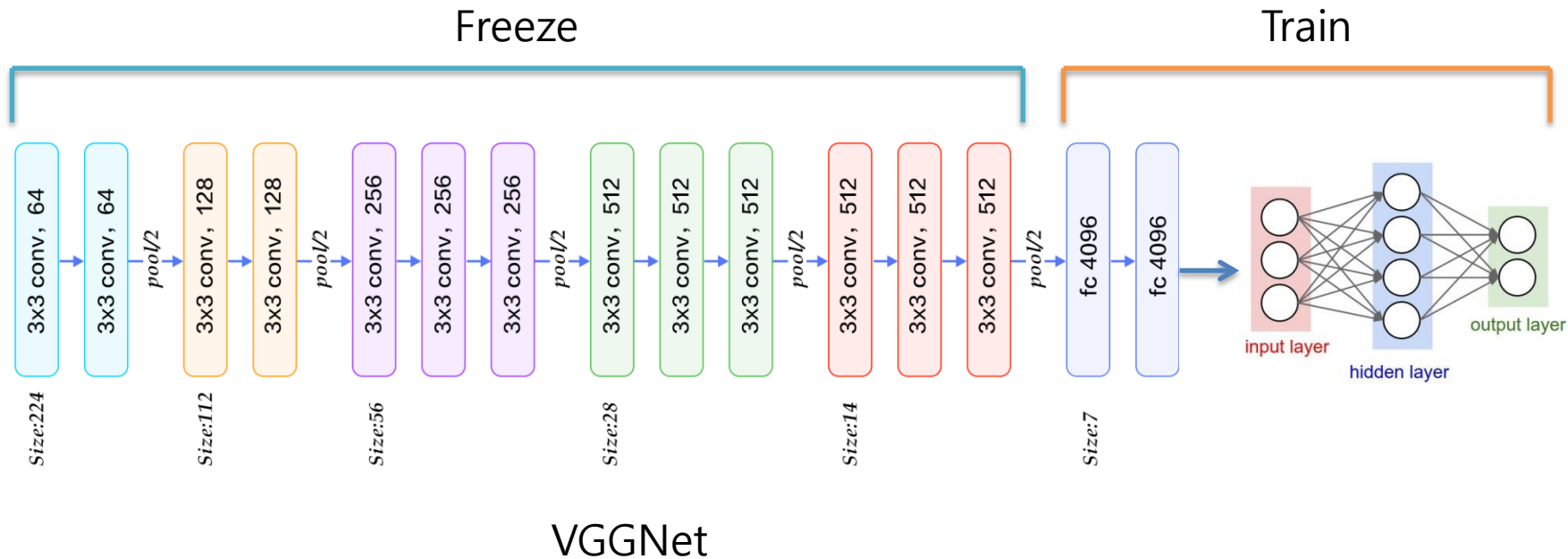
Train



Transfer Learning

데이터가 더 많을 때

1. 초기화 시켜서 한번에 학습
2. 뒷부분을 어느 정도 학습한 후에 fc 부분 학습



Transfer Learning

ResNet 예시

Pretrained Resnet-50
named_children() 함수는 해당
모듈 child의 이름과 파라미터를
순차적으로 리턴 해줌



Resnet 모듈의 child를 보면
어디가 모델 중 어느 부분인지
알 수 있음.

그 중에 쓸 부분만 새로운 Resnet
클래스로 불러오면 됨

```
resnet = models.resnet50(pretrained=True)
for name,module in resnet.named_children():
    print(name)
```

resnet without fully connected layers

```
class Resnet(nn.Module):
    def __init__(self):
        super(Resnet,self).__init__()
        self.layer0 = nn.Sequential(*list(resnet.children())[0:1])
        self.layer1 = nn.Sequential(*list(resnet.children())[1:4])
        self.layer2 = nn.Sequential(*list(resnet.children())[4:5])
        self.layer3 = nn.Sequential(*list(resnet.children())[5:6])
        self.layer4 = nn.Sequential(*list(resnet.children())[6:7])
        self.layer5 = nn.Sequential(*list(resnet.children())[7:8])

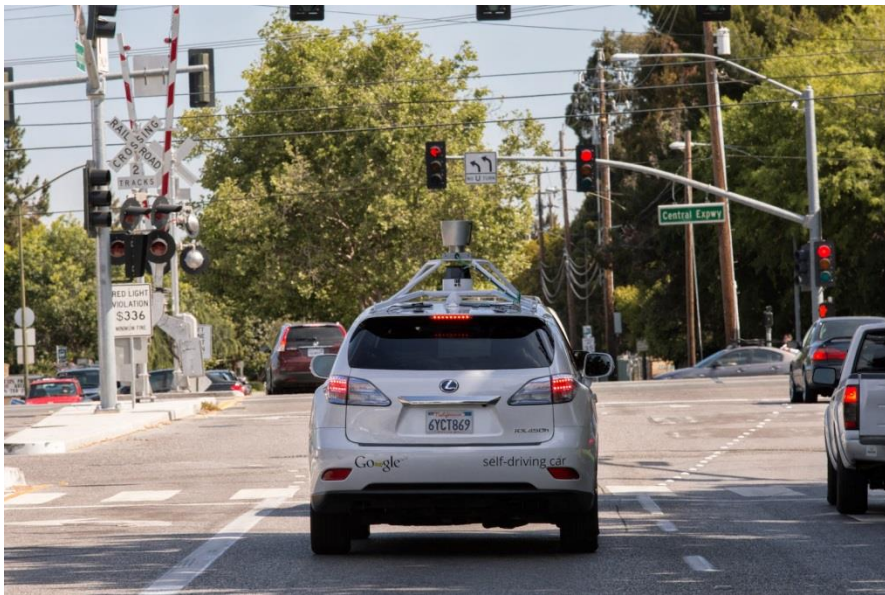
    def forward(self,x):
        out_0 = self.layer0(x)
        out_1 = self.layer1(out_0)
        out_2 = self.layer2(out_1)
        out_3 = self.layer3(out_2)
        out_4 = self.layer4(out_3)
        out_5 = self.layer5(out_4)

        return out_5
```

Transfer Learning

자율주행차

Real World



학습이 안된 상태로 돌아다니다가
큰일남.

데이터를 모으기 힘들고 학습도 느림

다양한 상황을 학습하기 힘들



어떻게 해결할 수 있을까?

(출처: <https://googleblog.blogspot.kr/2014/04/the-latest-chapter-for-self-driving-car.html>)

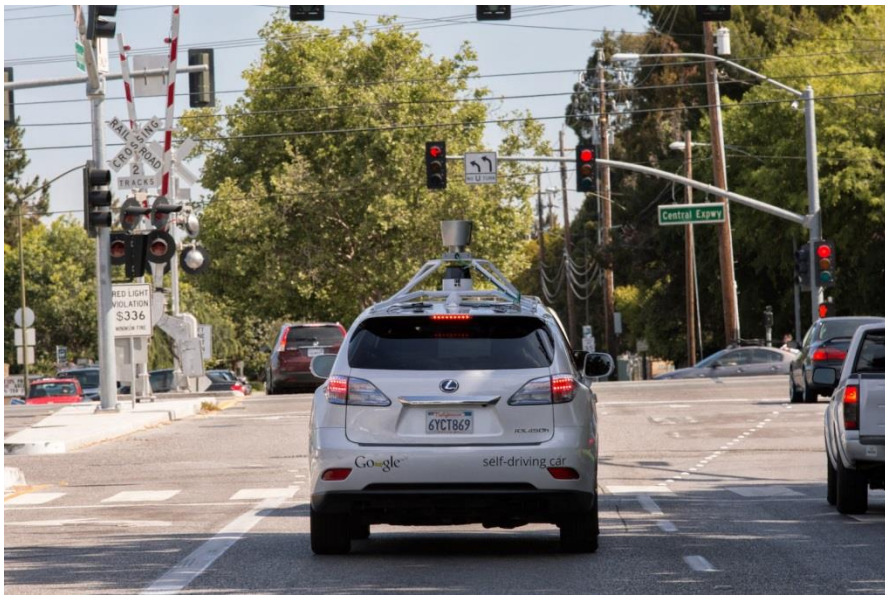
Transfer Learning

자율주행차

Transfer Learning

Real World

Simulation

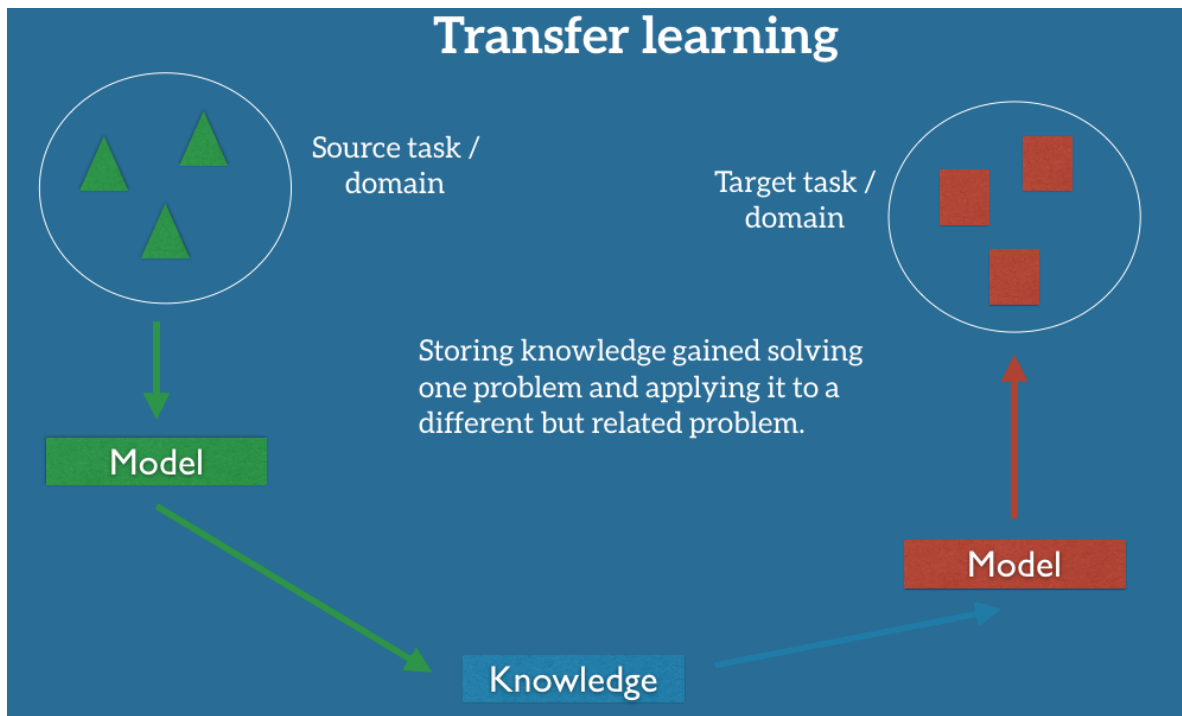


(출처: <https://googleblog.blogspot.kr/2014/04/the-latest-chapter-for-self-driving-car.html>)



(출처: <https://techcrunch.com/2017/02/08/udacity-open-sources-its-self-driving-car-simulator-for-anyone-to-use/>)

Transfer Learning



특정 도메인에서 습득한 지식을 다른 도메인에 적용시키는 것

Transfer Learning

Transfer Learning에 대한
숙련된 이우진 조교님의 발표

Style Transfer



"A neural algorithm of artistic style", Gatys et al(2015)

Style Transfer



Content Image

+

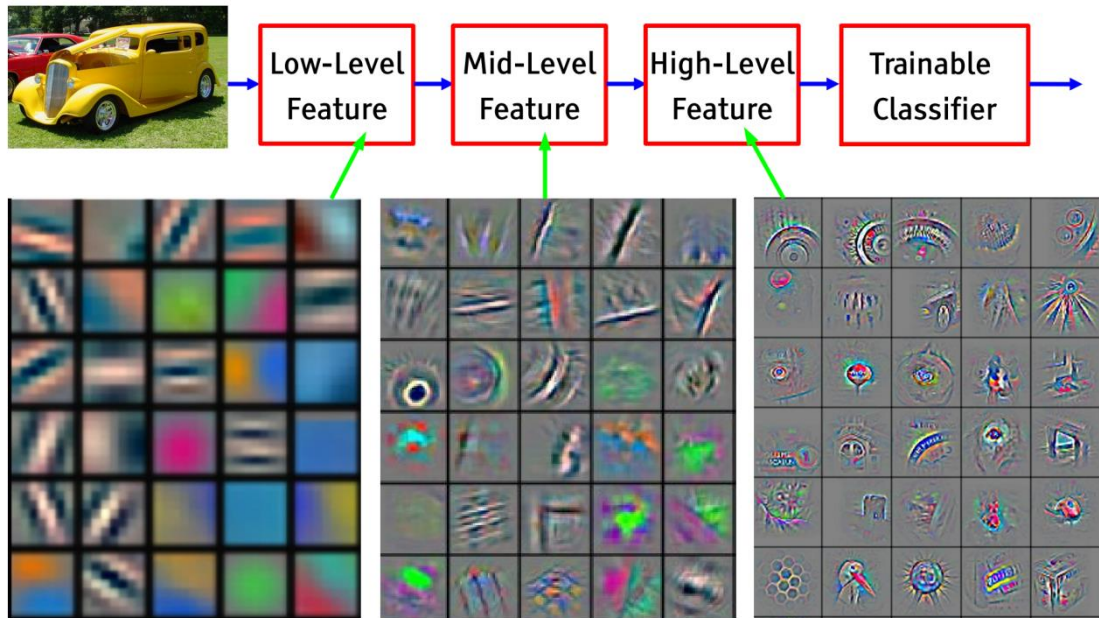
Style Image

=

Style Transferred Image

Style Transfer

스타일을 어떻게 뽑지?



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Pretrained CNN 모델은 모양
및 색을 구분할 수 있게 필터가
학습된 상태



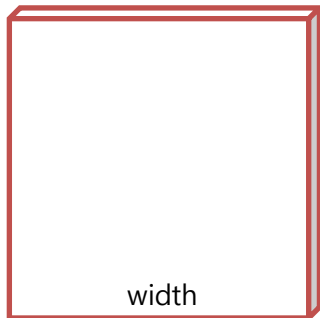
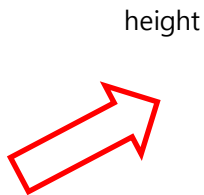
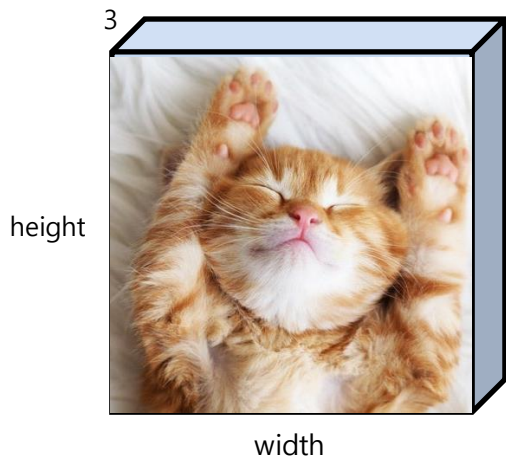
Convolution 연산의 결과값
-으로 나온 filtered image들은
각 필터에 대한 activation 정도



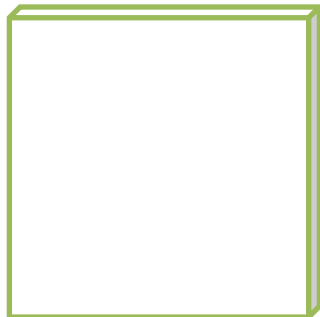
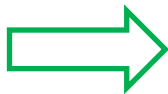
필터 별 activation간의 관계를
구하면 Style을 뽑아낼 수 있다.

Style Transfer

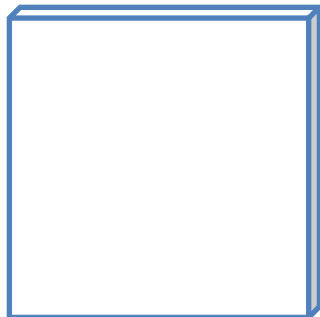
ex) 만약 필터가 Red,
Green, Blue만 있다면



=



=



=



Style Transfer

필터 activation간의 관계

1. 
2. 
3. 

$$\begin{matrix} & 1 & 2 & 3 \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 1 * 1 & 1 * 2 & 1 * 3 \\ 2 * 1 & 2 * 2 & 3 * 3 \\ 3 * 1 & 3 * 2 & 3 * 3 \end{bmatrix} \end{matrix}$$

Gram Matrix

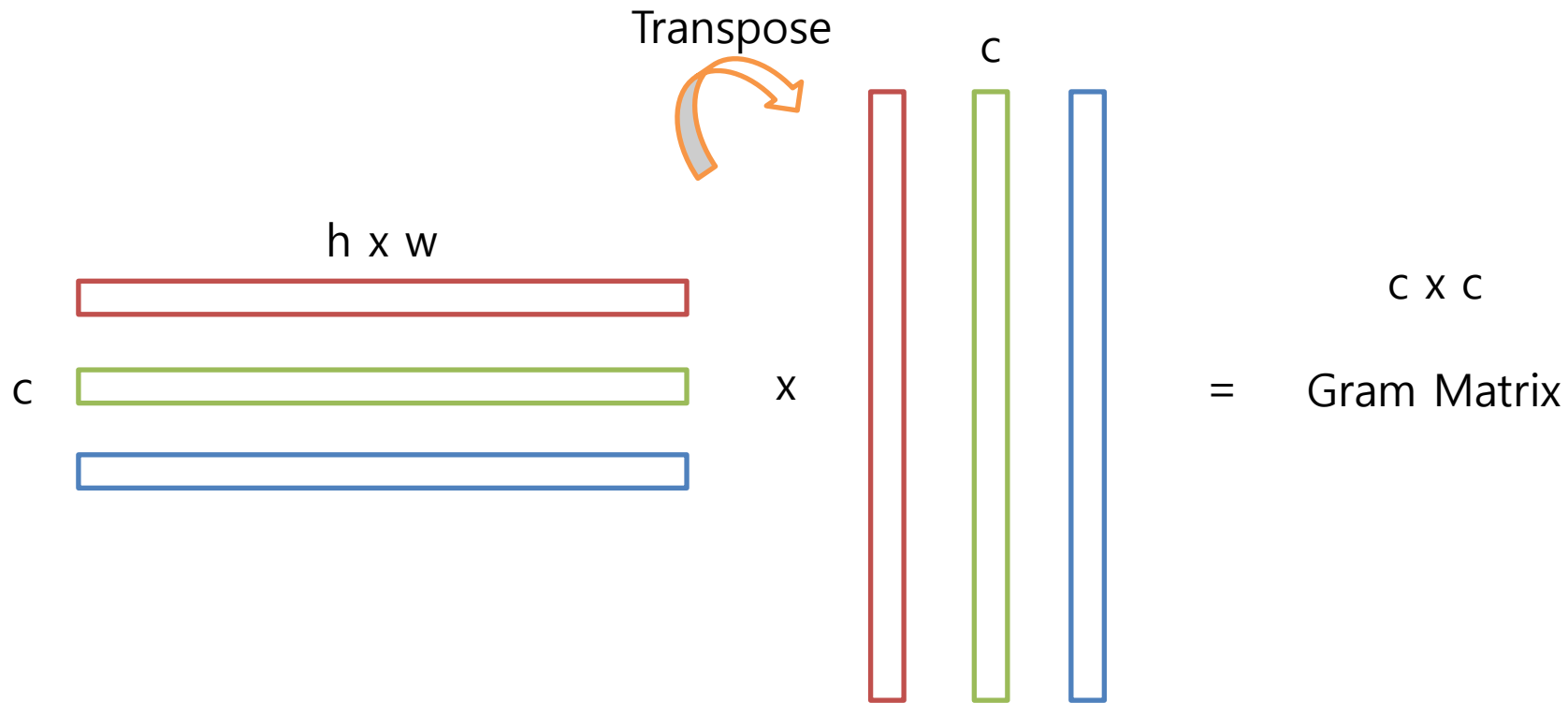
ex)

$$1 * 2 = \sum$$

$$\left(\begin{array}{c} \text{red box} \\ * \\ \text{green box} \end{array} \right)$$

element-wise mult & sum

Style Transfer

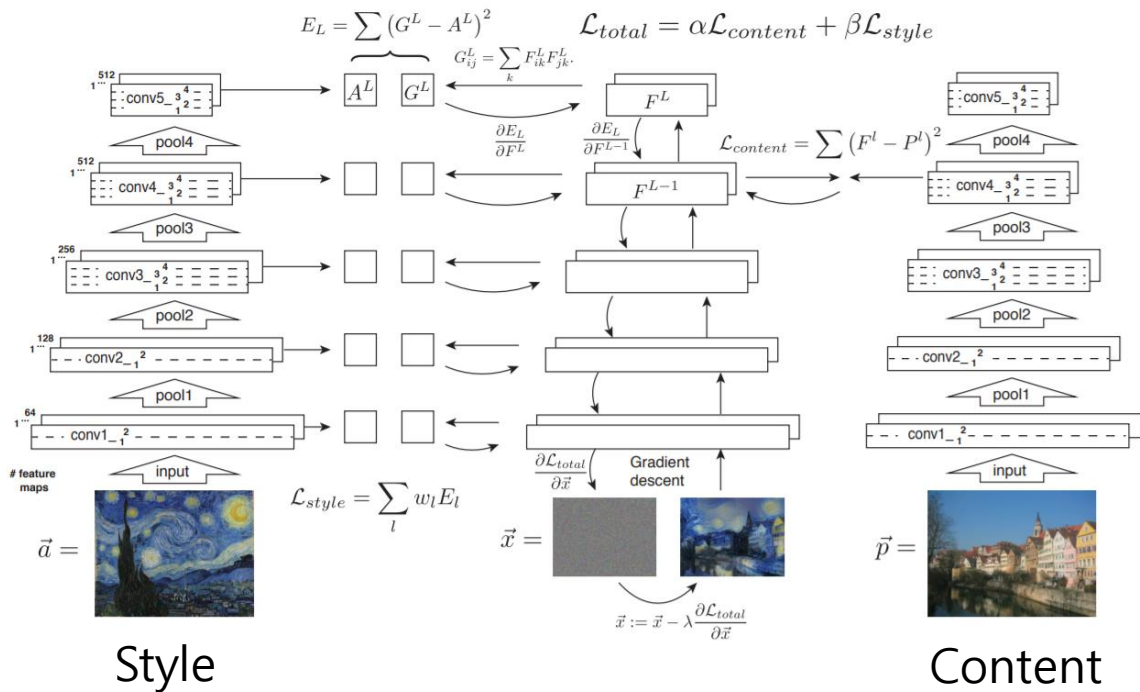


Style Transfer

weight가 아닌 이미지를
업데이트 하는 것



Output Image



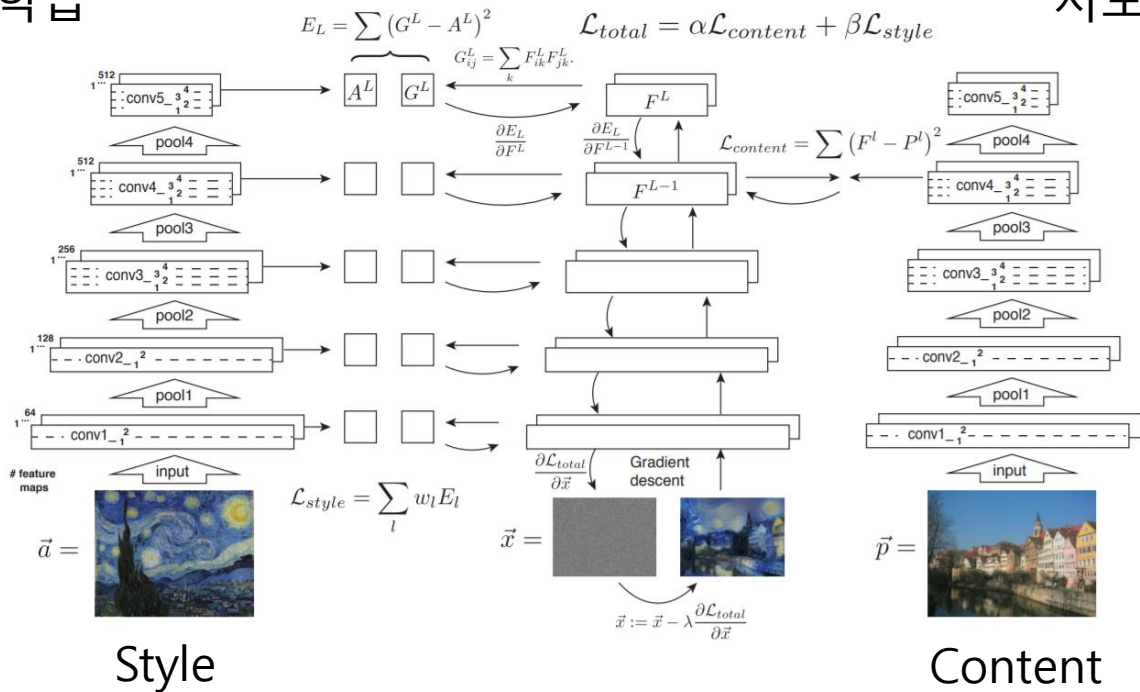
Style Transfer

비율은?

각 layer별 stlye
(Gram matrix)이
같아지도록 학습

Output Image

특정 layer의 content
(feature map)가 같아
지도록 학습

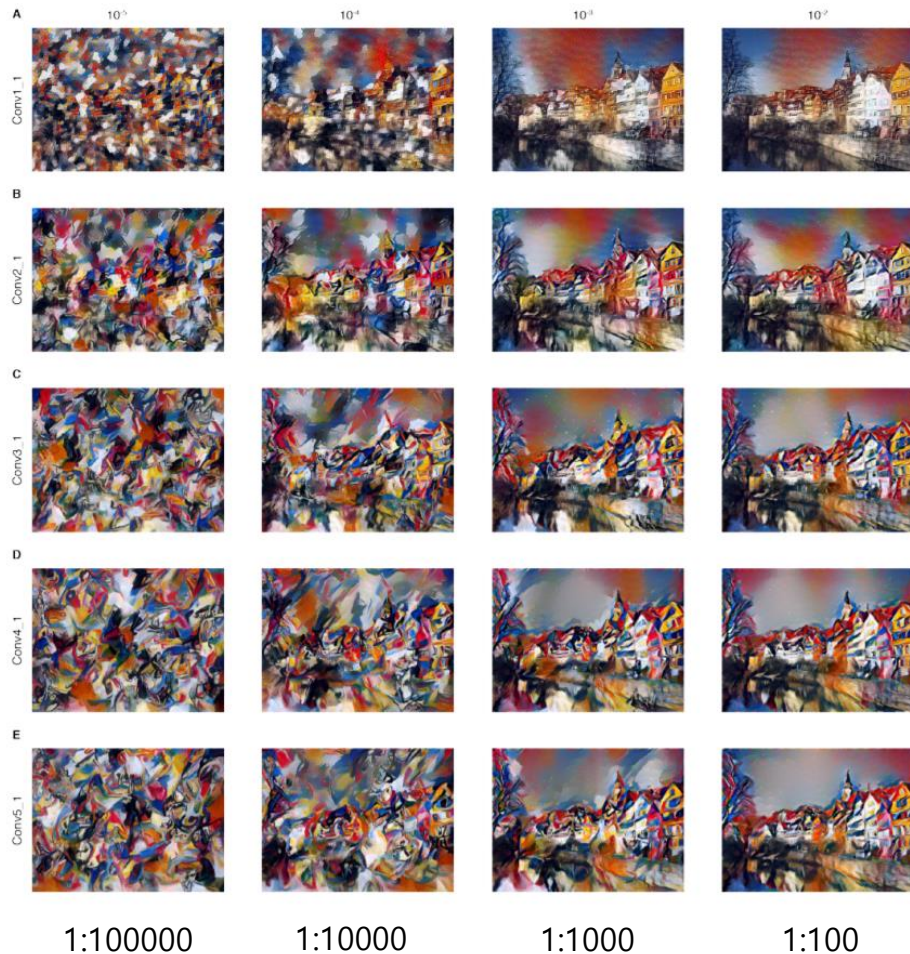


Style Transfer

Total Loss를 아래와 같이 정의할 때

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x})$$

α 와 β 의 비율에 따른 결과의 분포



Style Transfer

그러면 이대로 적절한 비율을 정해서 학습하면 되나?



일반적인 Optimizer를 써도 되긴 하지만 변수가 적기 때문에 Second Order Optimization Method를 써도 됨



L-BFGS

변수가 이미지이므로
256x256x3 이라고 하면
786,432 byte = 768 KB = 0.75MB

Style Transfer

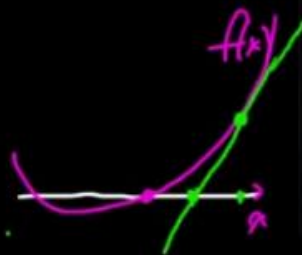
Newton's Method (for optimization) - 2nd order

(Grad. descent $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$ - 1st order)

Analogy (1D):

zero-finding: $x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$

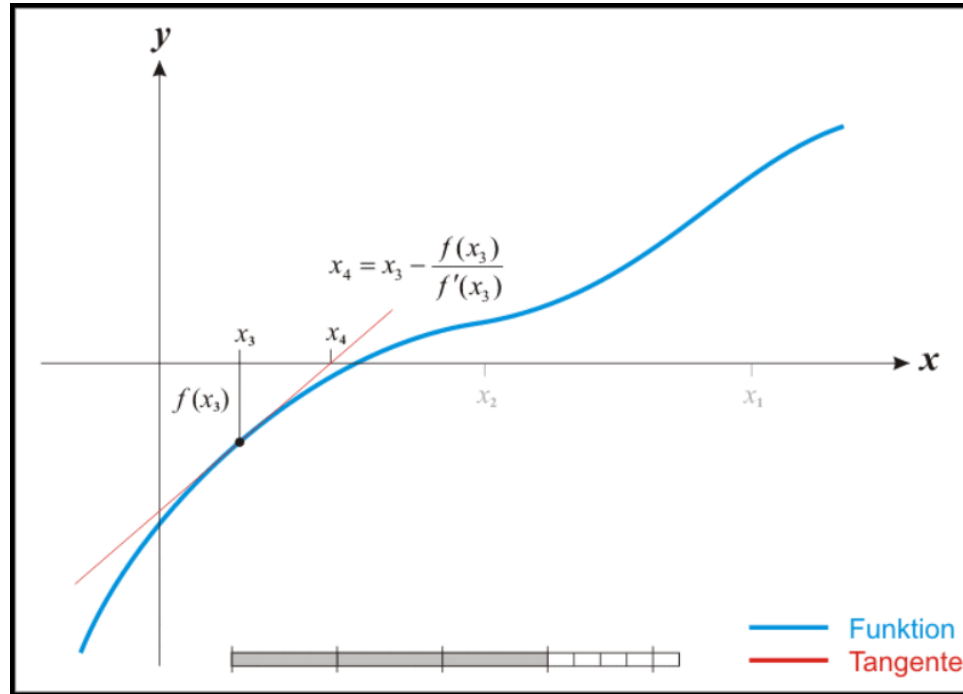
minimizing/
max.: $x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$



Newton

https://www.youtube.com/watch?v=28BMpgxn_Ec&t=327s

Style Transfer

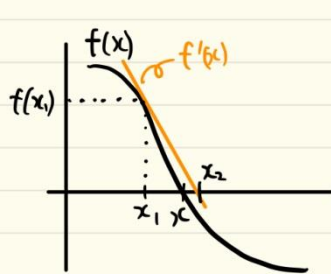


https://en.wikipedia.org/wiki/Newton%27s_method#/media/File:NewtonIteration_Ani.gif

Style Transfer

6. Newton's Method

• Zero finding



$$f'(x_1) = \frac{f(x_1) - 0}{x_1 - x_2}$$

$$x_1 - x_2 = \frac{f(x_1)}{f'(x_1)}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \Rightarrow x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

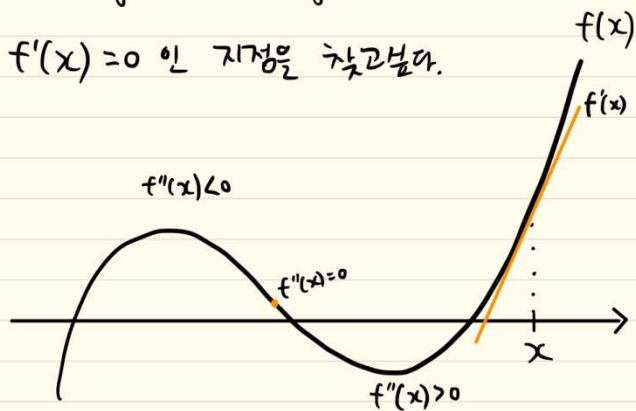
\Rightarrow 계속 하면 x 에 근접하게 됨.

기울기가 0인 지점을 찾으려면 $f''(x)$ 를 계산해야 하는데, 이를 이용하여 min/max를 찾는 방법이라고 이해하면 쉬움

Style Transfer

• Minimizing & Maximizing

$f'(x) = 0$ 인 지점을 찾고볼다.



$$x_{t+1} = x_t - \frac{f'(x)}{f''(x)}$$

이 식은 어떻게 나올지?

$f'(x) = 0$ 인 지점을 찾고 싶으니까 위의 zero-finding에서처럼 식을 만들지 마예 $f'(x)$ 의 그래프를 기점으로 하면 위와같은 식이 나옴!

기울기가 0인 지점을 찾으려면 $f''(x)$ 를 계산해야 하는데, 이를 이용하여 min/max를 찾는 방법이라고 이해하면 쉬움



Second Order Optimization은 $f''(x)$ 를 알고 있는 경우에 더 빠르게 수렴하는데 보통 모델 같은 경우에는 $f''(x)$ 의 계산이 힘들기 때문에 안 씀

Style Transfer

Pretrained resnet50
모델을 불러옴

```
resnet = models.resnet50(pretrained=True)
for name,module in resnet.named_children():
    print(name)
```

```
# resnet without fully connected layers
# return activations in each layers
```

원하는 위치마다
feature map을 뽑아내
서 쓸 수 있도록 재구성

```
class Resnet(nn.Module):
    def __init__(self):
        super(Resnet,self).__init__()
        self.layer0 = nn.Sequential(*List(resnet.children())[0:1])
        self.layer1 = nn.Sequential(*List(resnet.children())[1:4])
        self.layer2 = nn.Sequential(*List(resnet.children())[4:5])
        self.layer3 = nn.Sequential(*List(resnet.children())[5:6])
        self.layer4 = nn.Sequential(*List(resnet.children())[6:7])
        self.layer5 = nn.Sequential(*List(resnet.children())[7:8])

    def forward(self,x):
        out_0 = self.layer0(x)
        out_1 = self.layer1(out_0)
        out_2 = self.layer2(out_1)
        out_3 = self.layer3(out_2)
        out_4 = self.layer4(out_3)
        out_5 = self.layer5(out_4)

        return out_0, out_1, out_2, out_3, out_4, out_5
```

Style Transfer

feature map이 input으로
들어오면 Gram Matrix를
계산해주는 함수

```
class GramMatrix(nn.Module):  
    def forward(self, input):  
        b,c,h,w = input.size()  
        F = input.view(b, c, h*w)  
        G = torch.bmm(F, F.transpose(1,2))  
        return G
```

Gram Matrix간의 차이를 계
산하여 loss를 리턴 해주는
함수

```
class GramMSELoss(nn.Module):  
    def forward(self, input, target):  
        out = nn.MSELoss()(GramMatrix()(input), target)  
        return(out)
```

```
# initialize resnet and put on gpu  
# model is not updated so .requires_grad = False
```

```
resnet = Resnet().cuda()
```

모델은 학습되지 않도록 함

```
for param in resnet.parameters():  
    param.requires_grad = False
```

Style Transfer

content image, style image 준비.

output image는 시작점을 content image로 설정

```
content = Variable(image_preprocess(content_dir), requires_grad=False).cuda()  
style = Variable(image_preprocess(style_dir), requires_grad=False).cuda()  
generated = Variable(content.data.clone(), requires_grad=True)
```

```
# set targets and style weights
```

```
style_target = list(GramMatrix().cuda()(i) for i in resnet(style))  
content_target = resnet(content)[content_layer_num]  
style_weight = [1/n**2 for n in [64,64,256,512,1024,2048]]
```

target 값들 세팅.

style weight는 각 layer별 feature map의 크기가 다르기 때문에 균형을 맞춰주는 것

Style Transfer

LBFGS 설정

```
optimizer = optim.LBFGS([generated])
```

```
iteration = [0]  
while iteration[0] < epoch:
```

```
    def closure():
```

```
        optimizer.zero_grad()
```

```
        out = resnet(generated)
```

```
        style_loss = [GramMSELoss().cuda()(out[i], style_target[i]) * style_weight[i] for i in range(len(style_target))]
```

```
        content_loss = nn.MSELoss().cuda()(out[content_layer_num], content_target)
```

```
        total_loss = 1000 * sum(style_loss) + sum(content_loss)
```

```
        total_loss.backward()
```

```
        iteration[0] += 1
```

```
    return total_loss
```

```
optimizer.step(closure)
```

LBFGS는 연산 특성상 보통 optimizer와 다르게 closure 함수를 필요로 하는데 사실 closure 함수 내부는 기존의 loss 및 계산과 거의 같다.

Style Transfer

```
optimizer.step(closure)
```

Some optimization algorithms such as Conjugate Gradient and LBFGS need to reevaluate the function multiple times, so you have to pass in a closure that allows them to recompute your model. The closure should clear the gradients, compute the loss, and return it.

Example:

```
for input, target in dataset:
    def closure():
        optimizer.zero_grad()
        output = model(input)
        loss = loss_fn(output, target)
        loss.backward()
        return loss
    optimizer.step(closure)
```

LBFGS 같은 경우는 second order optimization method이기 때문에 closure를 사용함

그런데 과연 이렇게 뽑은 style이란게 사람들이 감각을 통해 느끼는 것과 비슷할까?



결과를 시각화해서 볼 수 있으면 직관적으로 와 닿을 것 같은데 현재는 데이터의 차원이 너무 높다

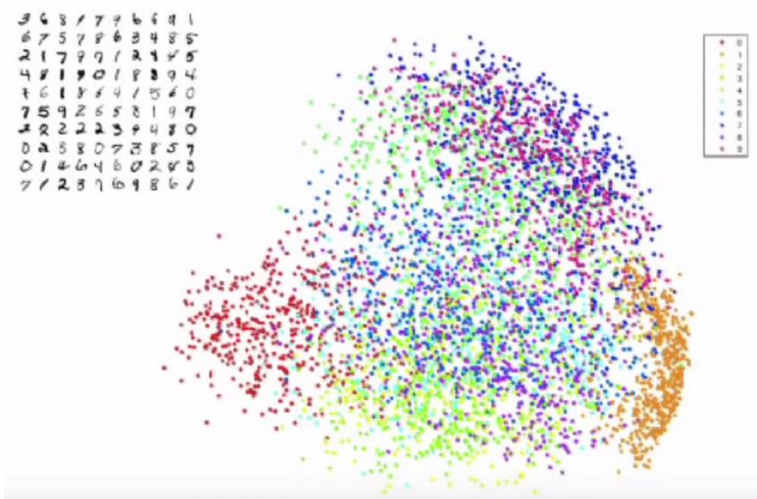


고차원 데이터를 그 구조를 유지하면서 3차원 이하로 내리면 가능할 것 같다.

Dimension Reduction

데이터 분포를 확인하려면 고차원의 데이터를
3차원 이하로 압축할 필요가 있는데 실제로
차원축소에는 다양한 방법들이 있음

Principal Components Analysis

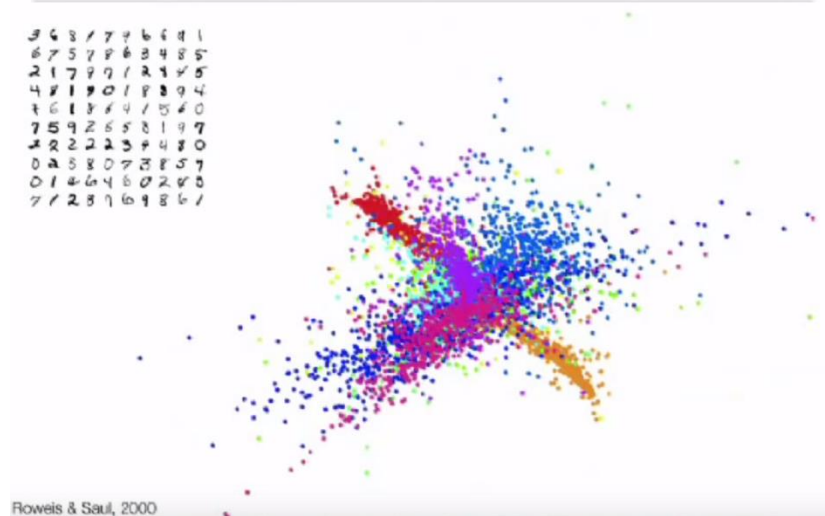


Isomap

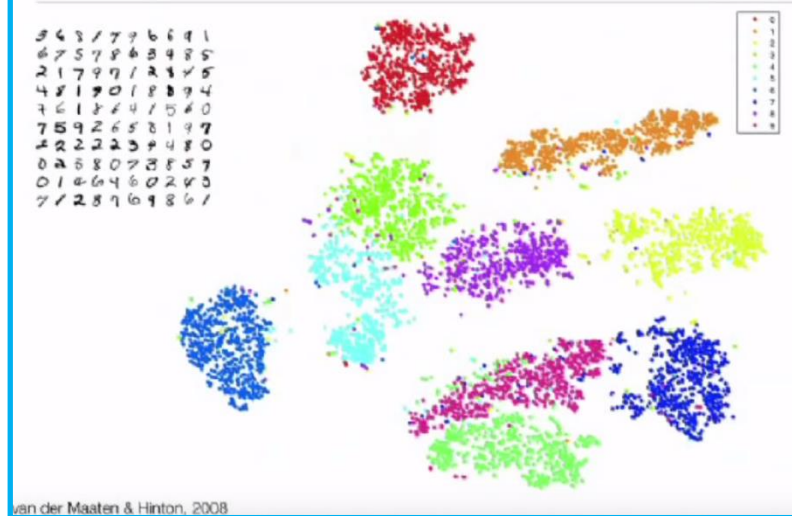


t-SNE

Locally Linear Embedding



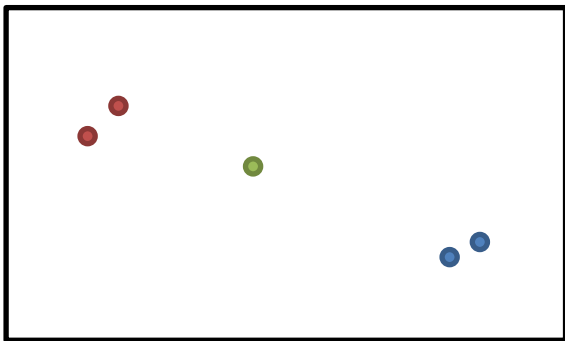
t-Distributed Stochastic Neighbor Embedding



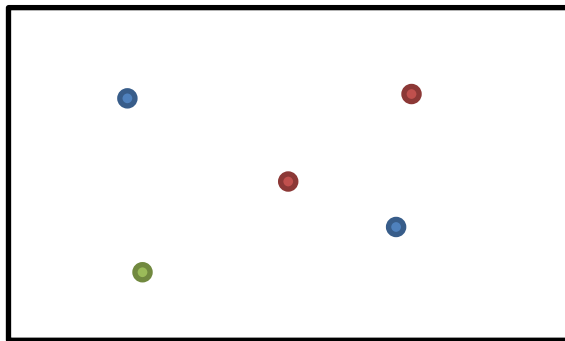
t-SNE

t-SNE (t-distributed stochastic neighbor embedding)

저차원에서의 점 간의 거리가 원래의 고차원 분포에서의 거리와 비슷해지도록 학습시켜 저차원에 embedding하는 방법



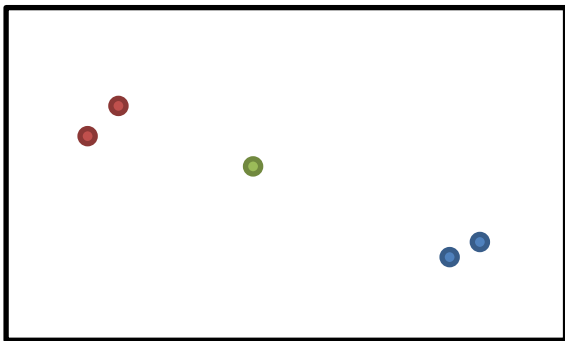
원래의 고차원 분포



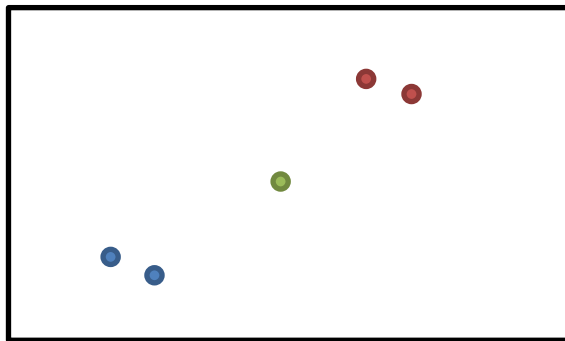
원래의 저차원 분포

t-SNE (t-distributed stochastic neighbor embedding)

저차원에서의 점 간의 거리가 원래의 고차원 분포에서의 거리와 비슷해지도록 학습시켜 저차원에 embedding하는 방법



원래의 고차원 분포



원래의 저차원 분포


원래 차원에서의 점간의 거리

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)},$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

축소된 차원에서의 점간의 거리

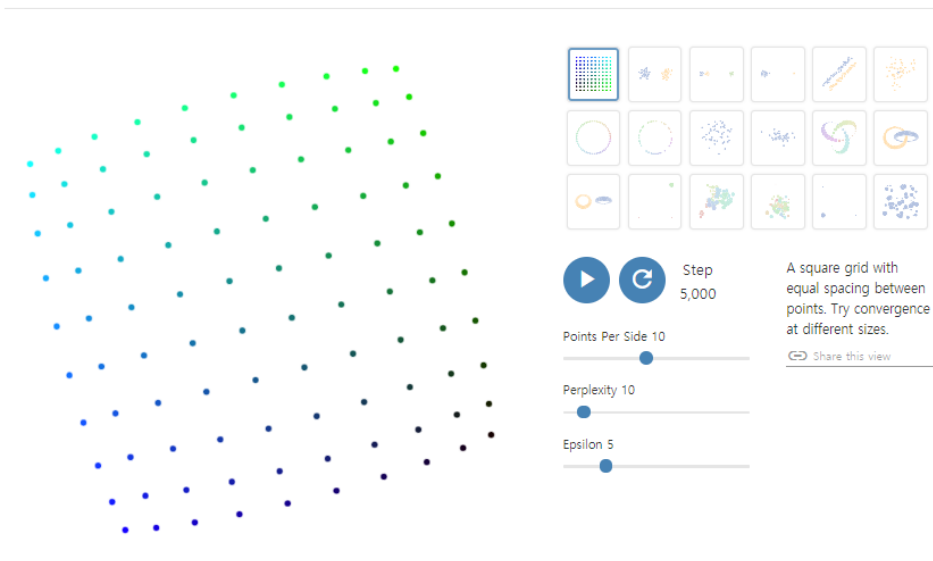
$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq m} (1 + \|\mathbf{y}_k - \mathbf{y}_m\|^2)^{-1}}$$


$$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

두 분포간의 KL divergence를
minimize 하도록 학습!

How to Use t-SNE Effectively

Although extremely useful for visualizing high-dimensional data, t-SNE plots can sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.



MARTIN WATTENBERG Google Brain FERNANDA VIÉGAS Google Brain IAN JOHNSON Google Cloud Oct. 13 2016 Citation: Wattenberg, et al., 2016

<http://distill.pub/2016/misread-tsne/>

```
total_arr = []
label_arr = []
for idx, (image, label) in enumerate(data):
    i = Variable(image).cuda()
    i = i.view(-1, i.size()[0], i.size()[1], i.size()[2])

    style_target = list(GramMatrix().cuda()(i) for i in resnet(i))
    arr = torch.cat([style_target[0].view(-1), style_target[1].view(-1), style_target[2].view(-1), style_target[3].view(-1)], 0)
    gram = arr.cpu().data.numpy().reshape(1, -1)
    total_arr.append(gram.reshape(-1))
    label_arr.append(label)

print(idx)

print(label_arr)
```

앞부분들은 Style Transfer와 동일.

Style을 나타내는 gram matrix들을 쭉 펴서 한 이미지당 한 줄로 저장

몇 차원으로 임베딩할 것인가

시작 state를 무엇으로 할 것인가

sklearn.manifold.TSNE

```
class sklearn.manifold.TSNE(n_components=2, perplexity=30.0, early_exaggeration=4.0, learning_rate=1000.0,
n_iter=1000, n_iter_without_progress=30, min_grad_norm=1e-07, metric='euclidean', init='random', verbose=0,
random_state=None, method='barnes_hut', angle=0.5)
```

[\[source\]](#)

정확도를 약간 포기하고 속도를 올리는 방법

거리 측정 메트릭

Examples

```
>>> import numpy as np
>>> from sklearn.manifold import TSNE
>>> X = np.array([[0, 0, 0], [0, 1, 1], [1, 0, 1], [1, 1, 1]])
>>> model = TSNE(n_components=2, random_state=0)
>>> np.set_printoptions(suppress=True)
>>> model.fit_transform(X)
array([[ 0.00017599,  0.00003993],
       [ 0.00009891,  0.00021913],
       [ 0.00018554, -0.00009357],
       [ 0.00009528, -0.00001407]])
```

```
from sklearn.manifold import TSNE

# Apply TSNE

print("\n-----Starting TSNE-----\n")

model = TSNE(n_components=2, init='pca', random_state=0)
result = model.fit_transform(total_arr)

print("\n-----TSNE Done-----\n")
```

t-SNE

offsetbox 는 하나의 기준점을
중심으로 여러 값을 한 화면에
나타낼 때 사용됨

plt.gca()는 창을 생성하는 함수

plt.imread()를 통해 이미지 읽어옴

OffsetImage를 통해 읽어온 이미지를
offsetbox instance로 바꿔줌

AnnotationBbox()를 통해 이미지와
좌표를 전달하고

update_datalim()을 통해 plot update

모든 이미지에 대해 업데이트 후 plt.show()

```
import matplotlib.pyplot as plt
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from matplotlib.cbook import get_sample_data

def imscatter(x, y, image, ax=None, zoom=1):
    if ax is None:
        ax = plt.gca()
    try:
        image = plt.imread(image)
    except TypeError:
        # Likely already an array...
        pass
    im = OffsetImage(image, zoom=zoom)
    x, y = np.atleast_1d(x, y)
    artists = []
    for x0, y0 in zip(x, y):
        ab = AnnotationBbox(im, (x0, y0), xycoords='data', frameon=False)
        artists.append(ax.add_artist(ab))
    ax.update_datalim(np.column_stack([x, y]))
    ax.autoscale()
    return artists

# scatter plot images
print("\n-----Starting to plot-----\n")

for i in range(len(result)):
    print("{} / {}".format(i, len(result)))
    img_path = img_list[i]
    imscatter(result[i, 0], result[i, 1], image=img_path, zoom=0.2)

plt.show()
```

Q&A
