

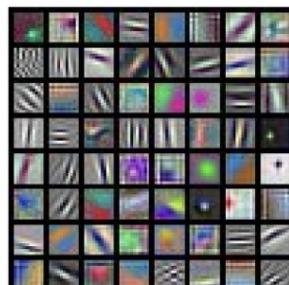
Week14: Visualizing and Understanding

④ 예습	
④ 복습	
복습과제 날짜	
복습과제 날짜	@2022년 11월 29일
☰ 내용	

CNN 레이어를 통과하면서 어떤 일이 일어나는지

첫번째 레이어의 시각화

First Layer: Visualize Filters



AlexNet:
64 x 3 x 11 x 11



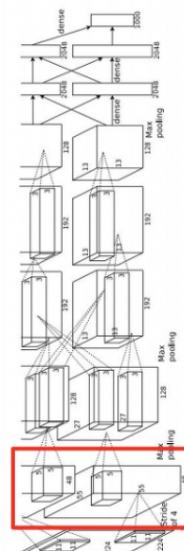
ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014.

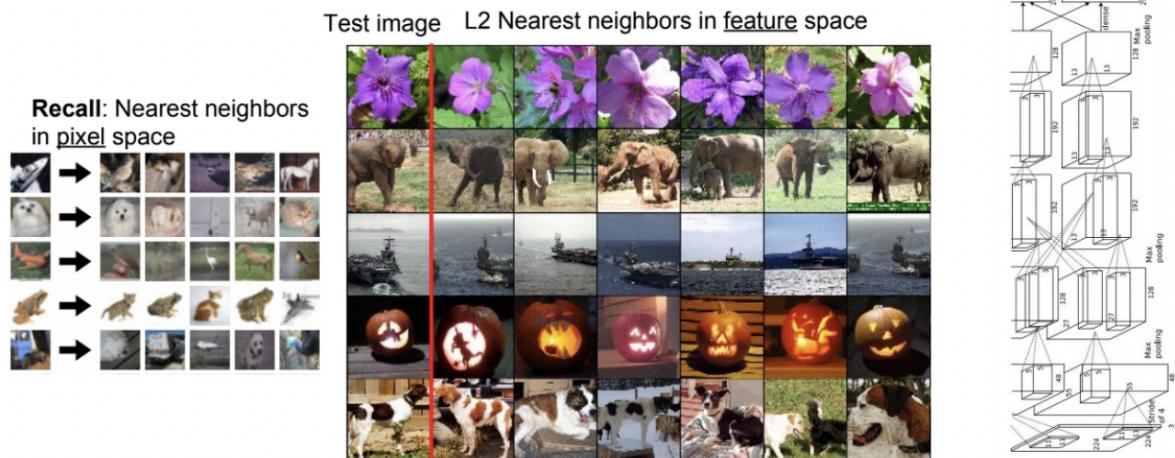
알렉스넷: edge 같은 특징이 보인다.

레이어가 깊어질수록 이해하기 어려운 필터들이 생기기도 한다.

마지막 레이어의 시각화

Last Layer: Nearest Neighbors

4096-dim vector



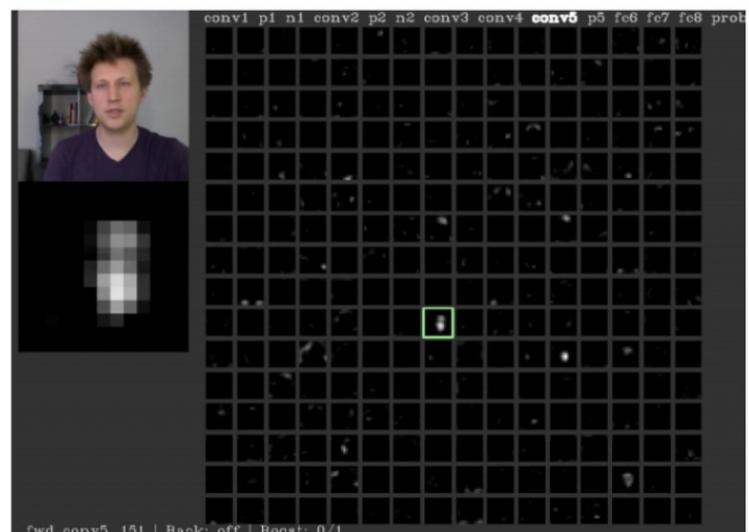
FC layer에서 4096차원으로 데이터를 flatten하게 만들었다. 벡터의 시각화는 NN으로 이루어진다.

4096차원의 벡터를 어떤 기법으로 2차원으로 축소시켜서 시각화한다. t-SNE과 PCA로 2차원으로 군집화한다.

feature map을 시각화한 것이다.

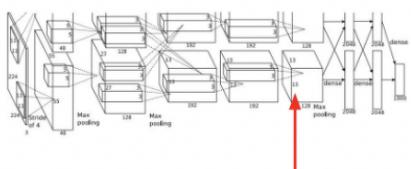
Visualizing Activations

conv5 feature map is
128x13x13; visualize
as 128 13x13
grayscale images



어떤 부분이 이미지 내에서 중요한 역할을 하는지 알 수 있다.

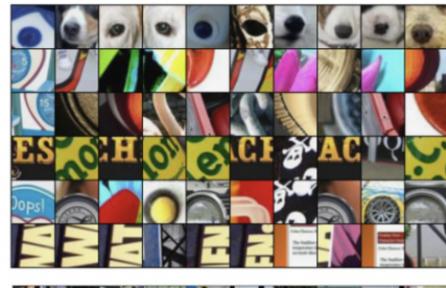
Maximally Activating Patches



Pick a layer and a channel; e.g. conv5 is $128 \times 13 \times 13$, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations



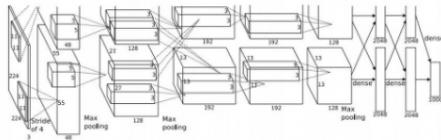
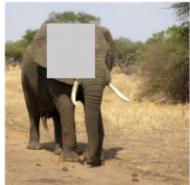
Springenberg et al., "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015;
reproduced with permission.

conv 5 layer에서 17번째 채널의 일부분을 시각화하였다. 가장 위를 보면 동그라미에 집중해서 레이어가 깊어질 수록 더욱 그부분에 집중한다.

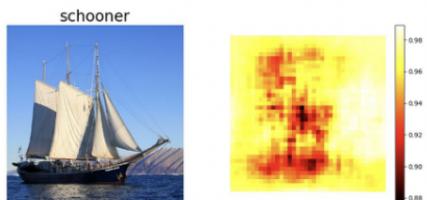
occlusion experiment는 이미지의 일부를 가린 후 분류가 잘 되는지 볼 수 있다. 이미지에서 어느 부분이 중요한지 알 수 있다.

Occlusion Experiments

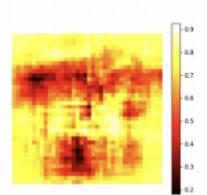
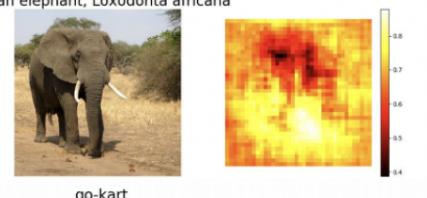
Mask part of the image before feeding to CNN, draw heatmap of probability at each mask location



Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain



African elephant, Loxodonta africana



Visualizing CNN features : Gradient Ascent

(Guided) backprop:

Find the part of an image that a neuron responds to

Gradient ascent:

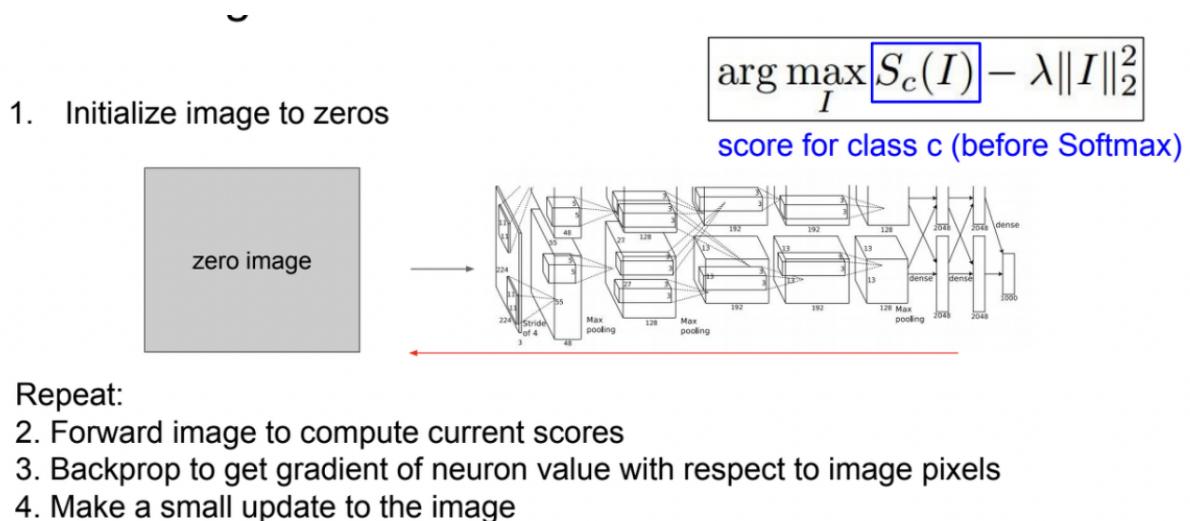
Generate a synthetic image that maximally activates a neuron

$$I^* = \arg \max_I [f(I) + R(I)]$$

Neuron value Natural image regularizer

input image에서 학습된 일부를 살펴본다. gradient descent는 로스를 최소화하는 방향으로 네트워크를 학습한다면, gradient ascent는 이미 학습된 네트워크의 가중치를 모두 고정하고 점수가 최대가 되는 방향으로 백지를 넣어 synthetic 이미지를 만들어낸다.

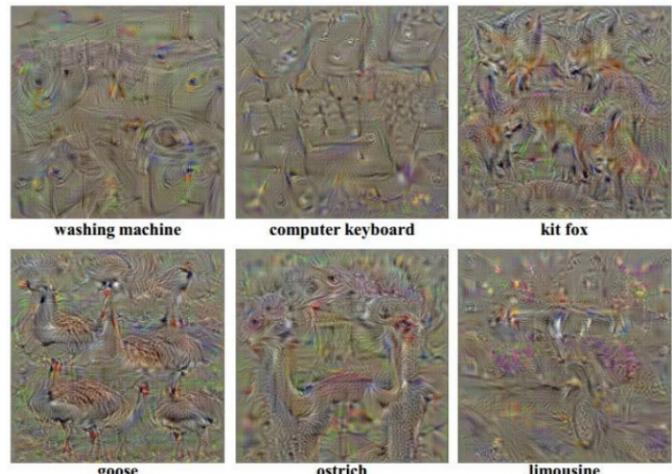
regularization function은 특정 뉴런에 오버피팅되는것을 방지한다.



1. 이미지 픽셀을 0으로 초기화
2. forward를 통해 이미 가중치를 고정한 네트워크에 넣어 현재 점수 계산
3. image pixel의 뉴런값의 gradient를 packprop으로 구현
4. 이미지를 특정 뉴런들의 최대화를 위해 픽셀단위로 update

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Simple regularizer: Penalize L2 norm of generated image

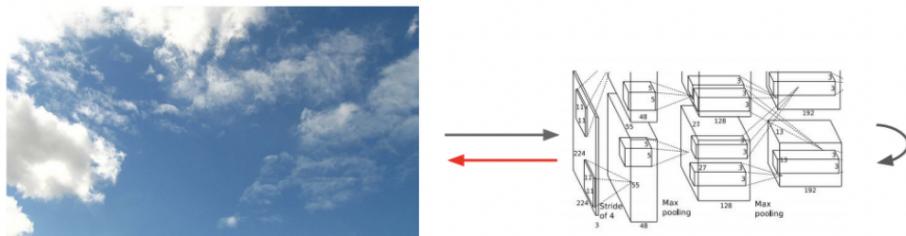


I et al., "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014, copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014.

위는 각 카테고리별 score를 max로 생성된 synthetic image들이다.

Deep Dream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network



Choose an image and a layer in a CNN; repeat:

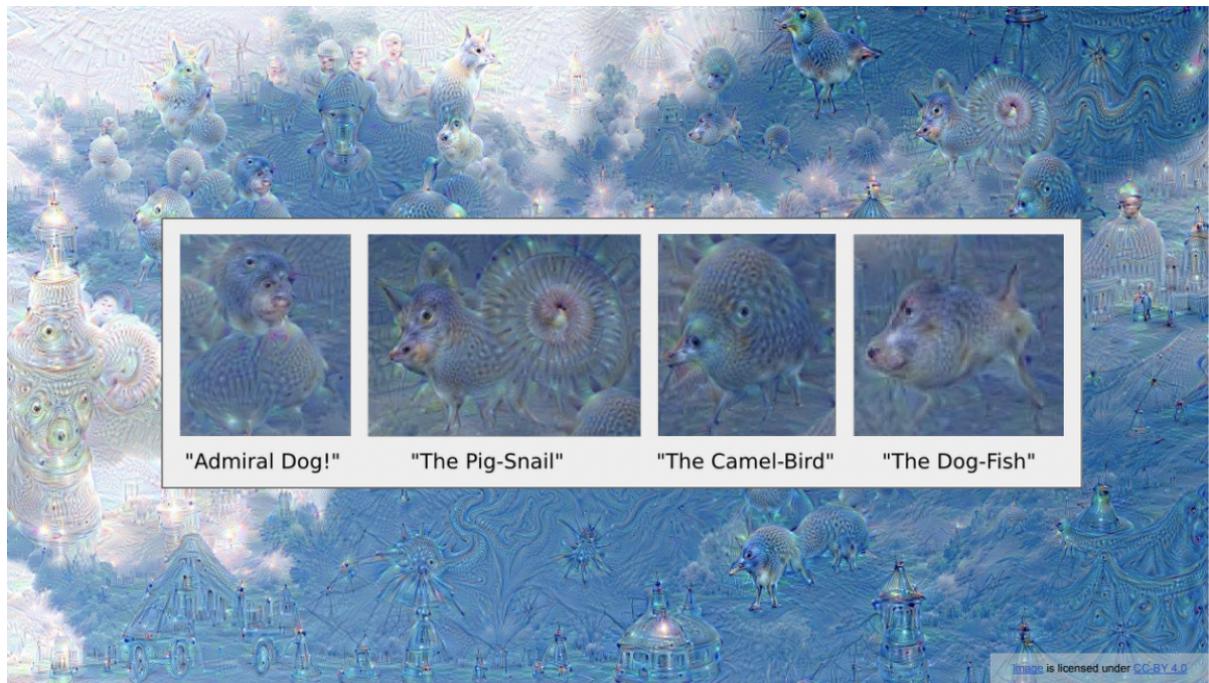
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

Equivalent to:
 $I^* = \arg \max_I \sum_i f_i(I)^2$

Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", Google Research Blog. Images are licensed under CC-BY

위에서 synthetic image를 만들 때의 목적은 중요했던 양수의 기울기를 가진 특정 뉴런을 최대화 했다면, deepdream은 중간의 레이어에서 모든 뉴런의 값을 확대한다.

1. forward 과정에서 선택했던 layer의 activation 값을 계산한다.
2. 그 activation 값을 gradient 값들로 set한다.
3. backward를 통해 그대로 학습시킨다.
4. 반복하여 이미지를 뽑아낸다.



해당 이미지는 어떤 레이어의 모든 뉴런 값을 확대한 채로 학습시켜 이미지를 뽑은 것이다. 이번에는 특징보다 합성된 특이한 그림처럼 보인다. neuron activation에 집중하였기 때문이다.

낮은층에서 시각화를 해보면 단순한 라인이나 엣지만 확인 할 수 있다.

Freature Inversion

feature에서 원본이미지를 다시 생성해보고 네트워크 각각의 레이어에서 어떤식으로 이미지가 포착되는지 파악할 수 있다.

Given a CNN feature vector for an image, find a new image that:

- Matches the given feature vector
- “looks natural” (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

Given feature vector
Features of new image

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

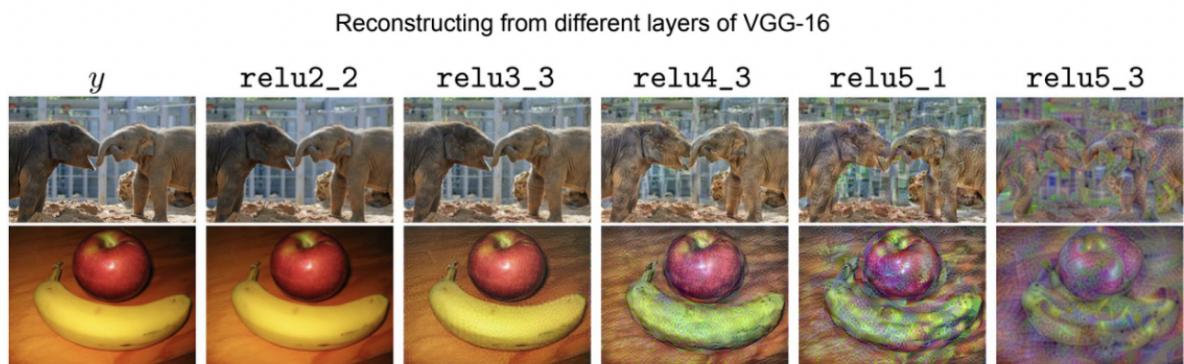
Total Variation regularizer
(encourages spatial smoothness)

endran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

이미지는 화이트노이즈로부터 시작된다.

생성된 feature map의 벡터와 원래 이미지의 feature vector 간의 distance값을 최소로 한다. 최고로 할 때 gradient ascent를 실행한다.

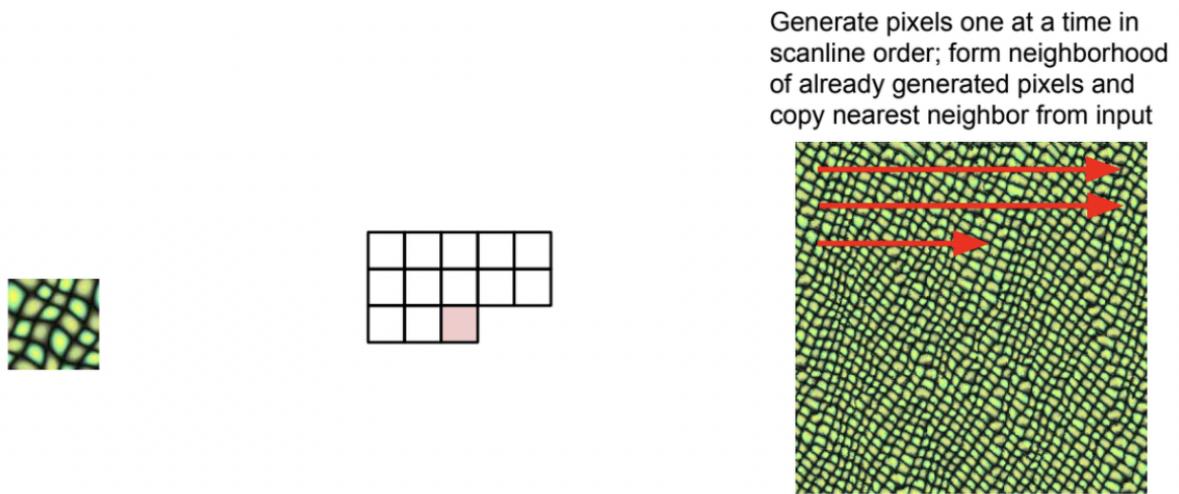
regularizaiton은 인접한 feature들의 패널티를 부과하여 이미지에서 매끄러움을 준다.



Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015
Figure from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016.
Reproduced for educational purposes.

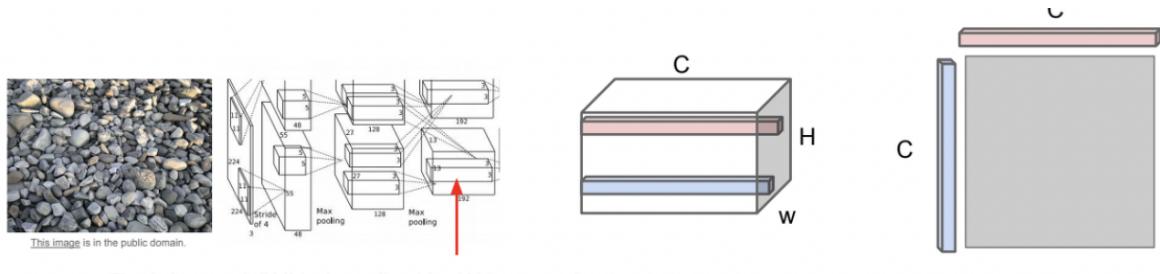
레이어가 깊어질수록 전체적인 구조와 같은 좀 더 의미있는 정보를 남기는 것일 수도 있다.

Texture Synthesis



texture synthesis는 텍스쳐의 input patch를 얻고 싶은 아이디어였다. 고전적으로 nearest neighbor 방법이 있다. 픽셀을 라인으로 훑으며 주위에 생성된 픽셀값을 계산하여 그 자리에 한 픽셀을 복사하는 방식이다. 단순한 텍스쳐에서만 잘 되는 방법이라 neural 네트워크를 이용하여 방법을 시도해보았다.

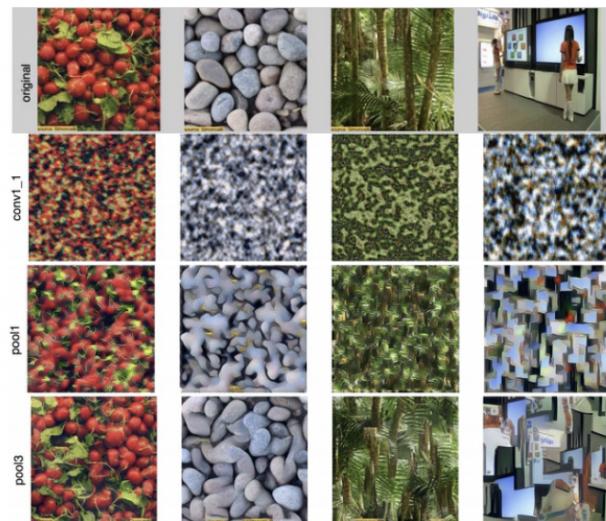
Gram Matrix



1. Input data를 뉴럴넷에 입력
2. 몇몇 레이어에서 feature map을 추출
3. 해당 feature map에서 슬라이드처럼 서로다른 픽셀의 feature vector의 차원으로 추출
4. 3으로 뽑은 feature vector를 서로 외적하여 새로운 행렬을 생성
5. 이 모든 과정을 $H \times W$ 차원에 대해 수행하여 평균을 내면 $C \times C$ 크기의 gram matrix를 생성할 수 있다.

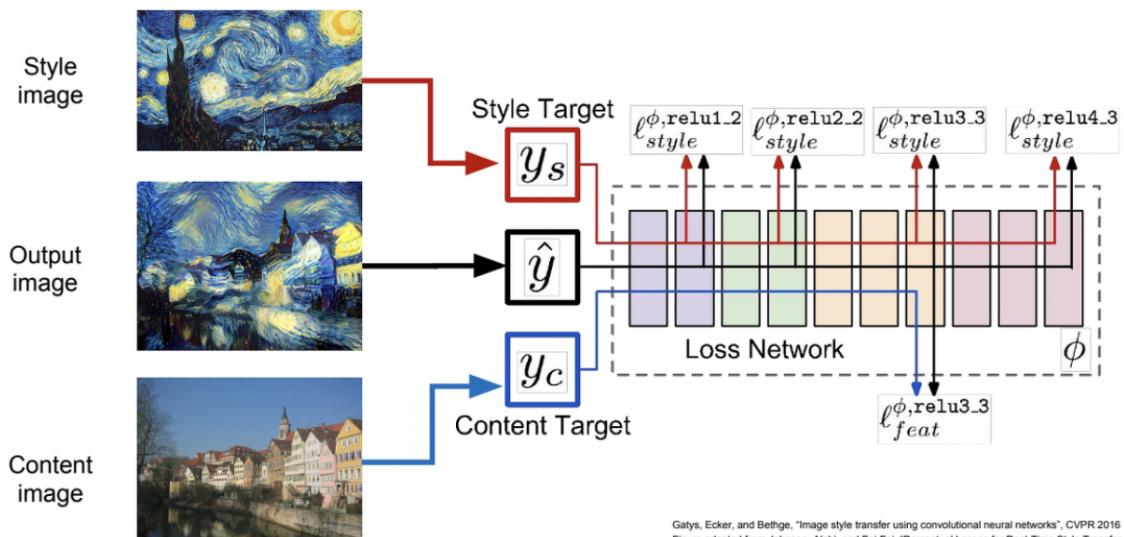
Neural Texture Synthesis

Reconstructing texture from higher layers recovers larger features from the input texture



특정 레이어에서 뽑은 neural texture synthesis는 레이어가 깊을 수록 더 잘 표현되는 것을 볼 수 있다.

Neural Style transfer



Neural Texture Synthesis를 아트워크에 적용. texture synthesis와 feature inversion을 합친 개념이다.

입력데이터는 2가지 :

- style image: gram matrix를 이용하여 texture를 뽑는다.

- content image: feature inversion을 이용하여 네트워크를 거쳐 나온 feature map으로 디테일이 부족한 image 생성

2가지 로스 :

- style image에서 나온 gram matrix loss
- content image에서 나온 feature reconstruction loss

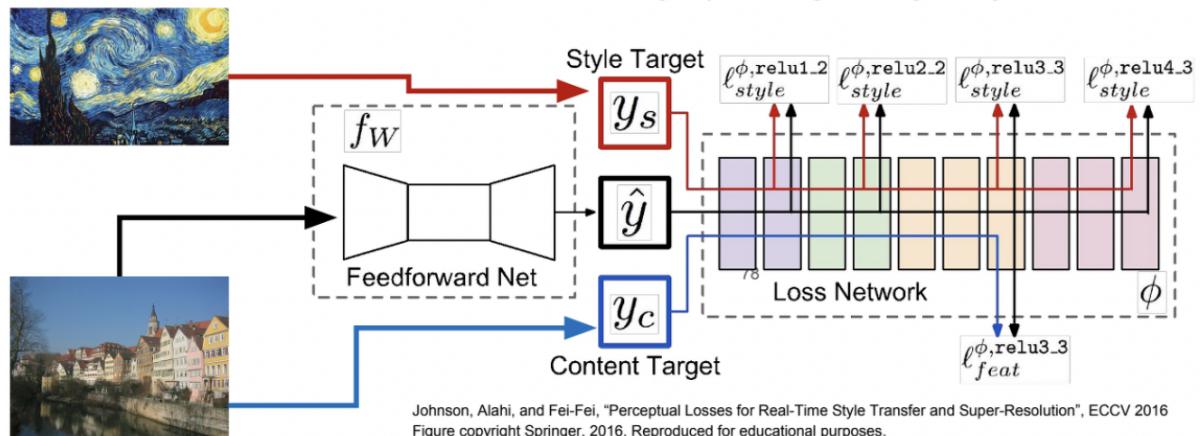
Resizing style image before running style transfer algorithm can transfer different types of features



자유도가 높아 output은 어느 loss에 치중하느냐에 따라 결과물이 위처럼 달라질 수 있다. 그러나 style transfer은 두 이미지를 함께 사용하여 학습을 진행하기 때문에 content image가 바뀔 때마다 학습을 다시해야해서 많은 연산을 필요로 한다.

Fast Style Transfer

- (1) Train a feedforward network for each style
- (2) Use pretrained CNN to compute same losses as before
- (3) After training, stylize images using a single forward pass



몇몇 원하는 스타일의 이미지를 학습시켜 고정하고, input으로 content image만 넣어 단일 네트워크학습을 하는 것이다. 학습시에 style과 content의 로스를 모두 같이 계산하는데 단지 빨라진 것이다.