

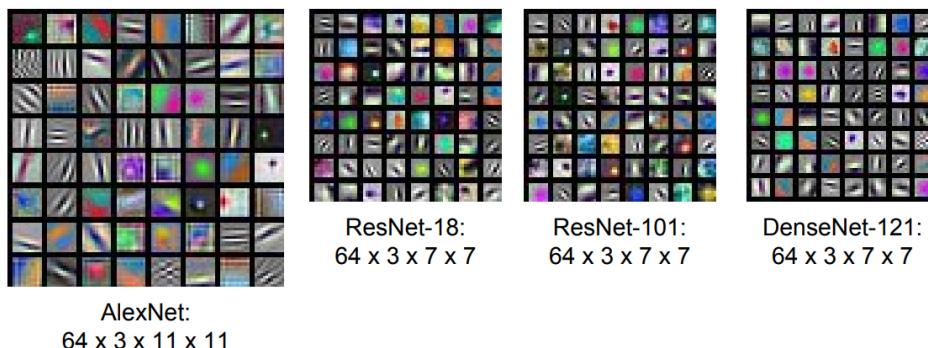
20

# Lec 12. Visualizing and Understanding

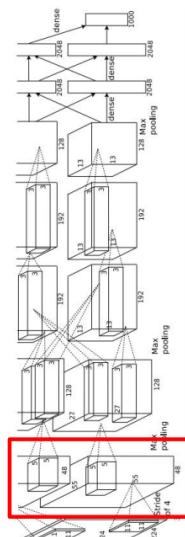
→ CNN의 내부 살펴보기

Visualize filters of first layer

## First Layer: Visualize Filters



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014  
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016  
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017



- sliding window로 이미지를 scan
- 이미지의 일부 영역과 내적 수행 → first conv layer의 output
- input image와 직접 내적 수행
- filter는 주로 edge 성분 혹은 보색을 찾음
- CNN을 어떤 model/data로 학습하더라도 첫 번째 layer는 비슷하게 생김
- first layer는 대체로 convolutional layer임

Second layer

- 첫번째 layer는 image와 직접적으로 연관되어 있어 interpret할 수 있지만, 두번째 layer부터는 쉽게 interpret할 수 없음
- 여기서 시각화한 내용은 두번째 레이어의 결과를 최대화시키는 첫번째 레이어의 output pattern이 무엇인지임
- 그래서 layer가 깊어질수록 시각화를 통해 정보를 얻어내기 어려움

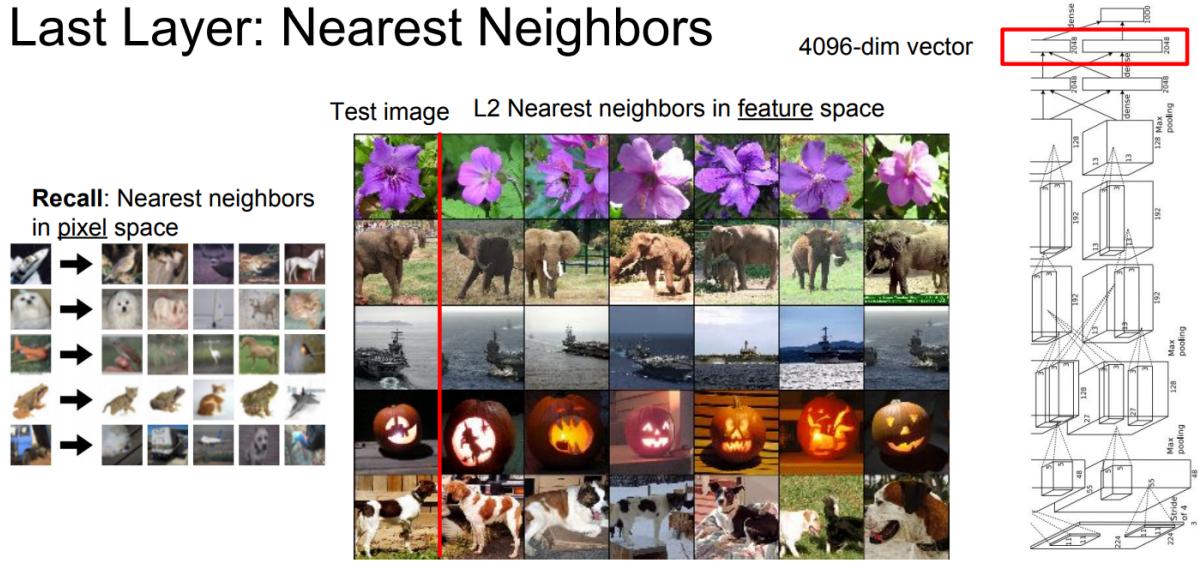
## Last layer

- predicted scores
- last layer 직전에는 Fully Connected Layer가 있음

## 시각화 방식

- Nearest Neighbor
  - 시각화를 통해 살펴본 모습에서 서로 pixel 값의 차이는 커도 feature 공간 내에서 유사한 특성을 지님

## Last Layer: Nearest Neighbors

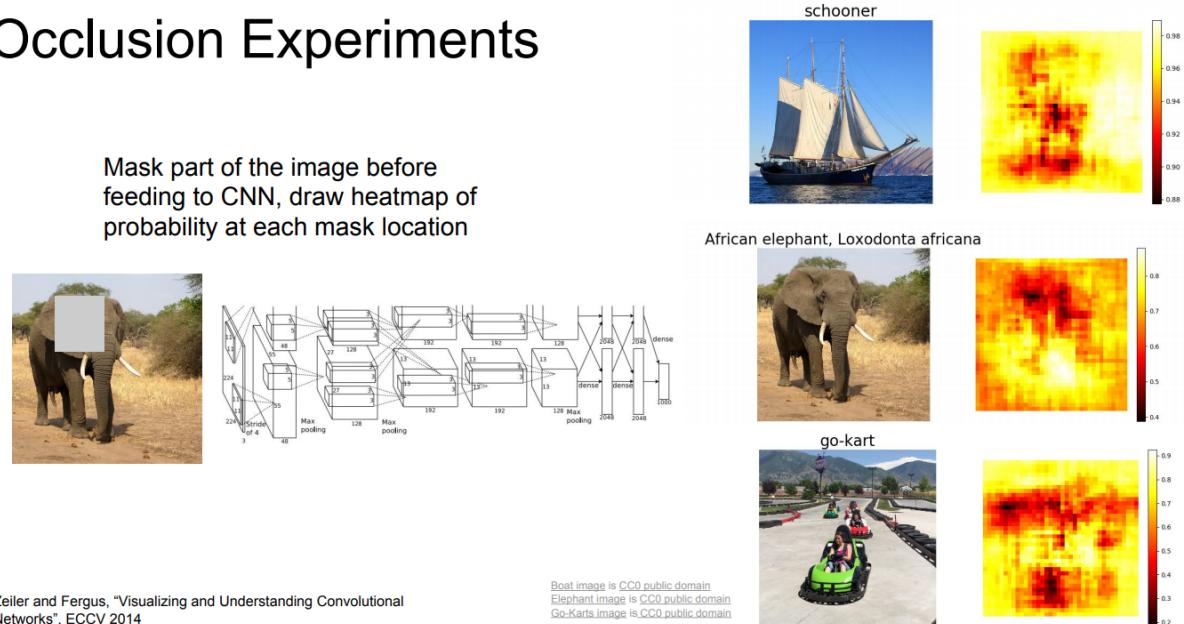


Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.  
Figures reproduced with permission.

- dimensionality reduction
  - PCA나 t-SNE 기법을 이용해 차원 축소
  - 군집화된 모습 확인 가능

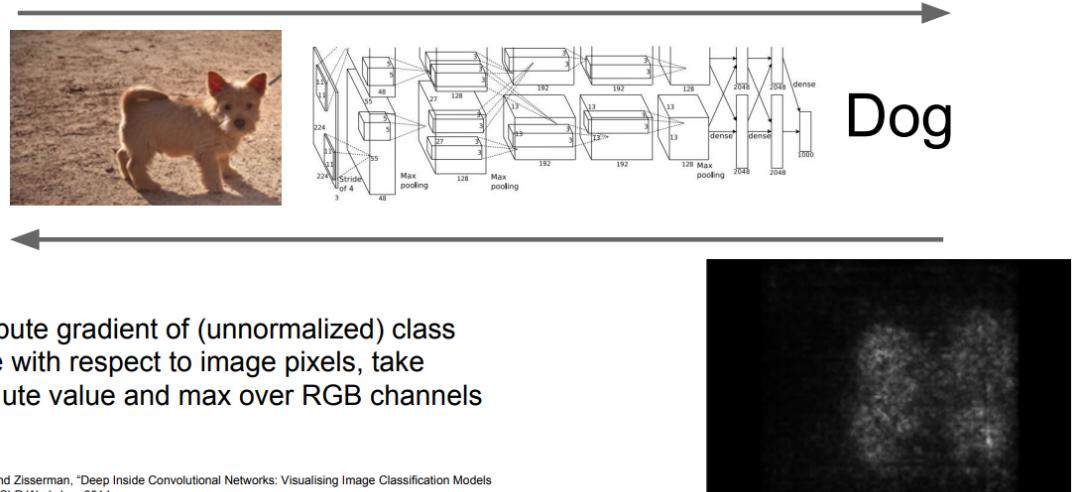
- maximally activating patches
  - 특정 뉴런에서 activation이 가장 큰 patch를 시각화
- occlusion experiments
  - 특정 구간을 block했을 때 score가 크게 변한다면 그 부분이 classification하는 데에 매우 중요한 부분이라고 판단하는 것. 특정 부분을 없앴을 때 softmax에서 probability가 적어지는 것을 시각화한 것
  - input의 어떤 부분이 classification의 근거가 되는지에 관한 실험

## Occlusion Experiments

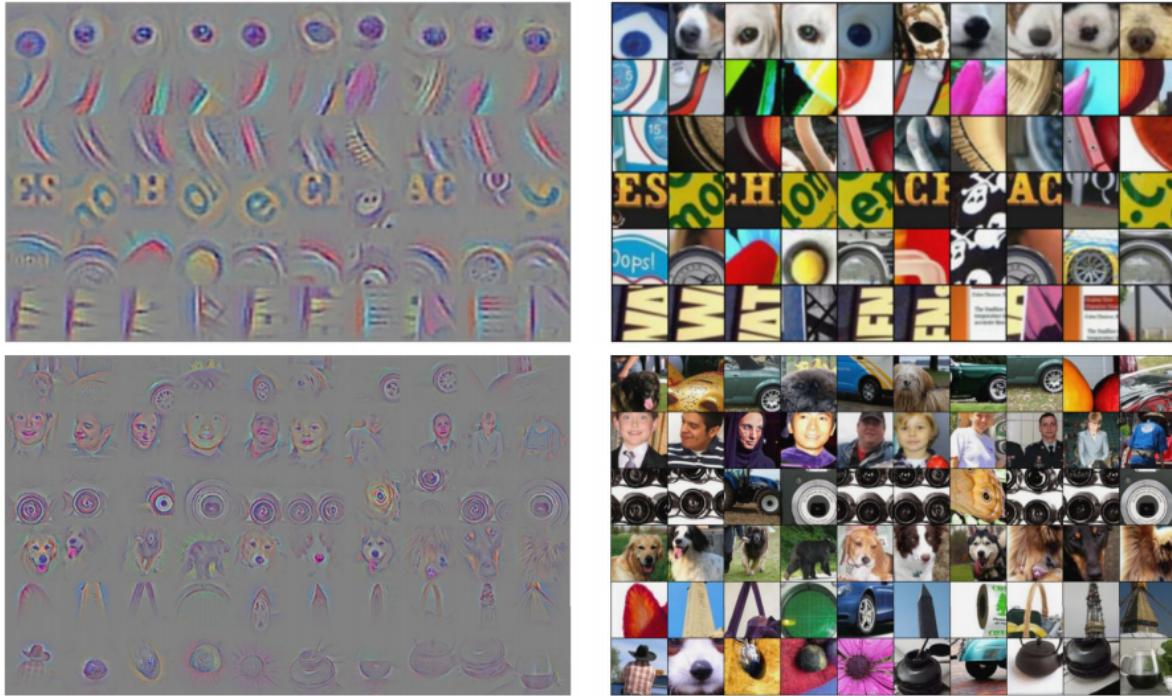


- saliency maps
  - classification score를 이미지의 각 픽셀의 gradient를 시각화한 것 → Neural Network가 image에서 어느 부분을 보고 있는지 알 수 있음
  - labeling을 하지 않고도 segmentation 작업을 학습할 수 있음
  - BUT, supervision task보다 훨씬 정확도가 떨어짐

## How to tell which pixels matter for classification?



- guided backpropagation
  - backprop 시에 ReLU의 gradient 부호가 양이면 그대로 통과, 음이면 backprop 하지 않음
  - → positive gradient만 고려하게 됨
  - 일반 backprop에 비해 더 선명하고 좋은 이미지 얻을 수 있음
  - 이런 이미지들은 어떤 픽셀들이 특정 뉴런에 영향을 미치는지 알 수 있게 해줌



- 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

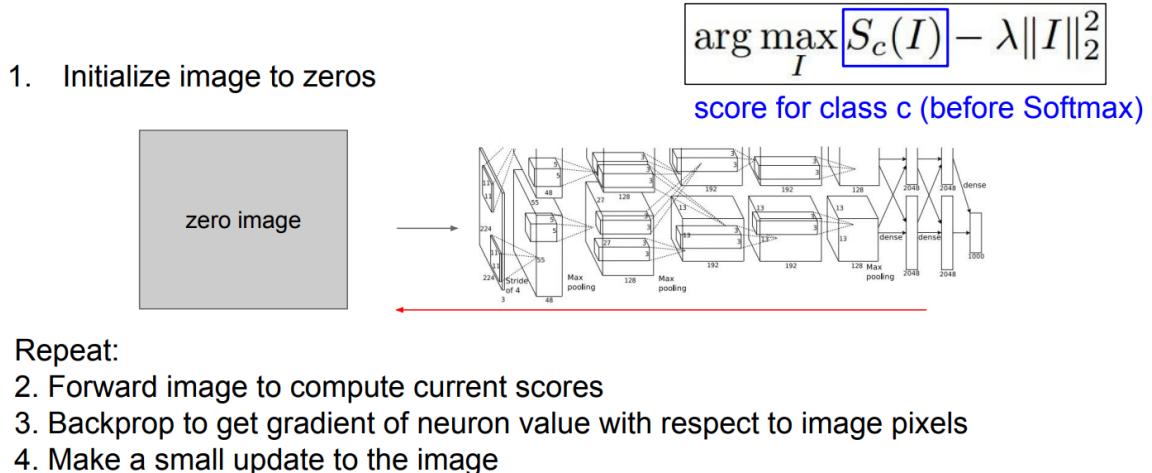
$$I^* = \arg \max_I [f(I) + R(I)]$$

Neuron value

Natural image regularizer

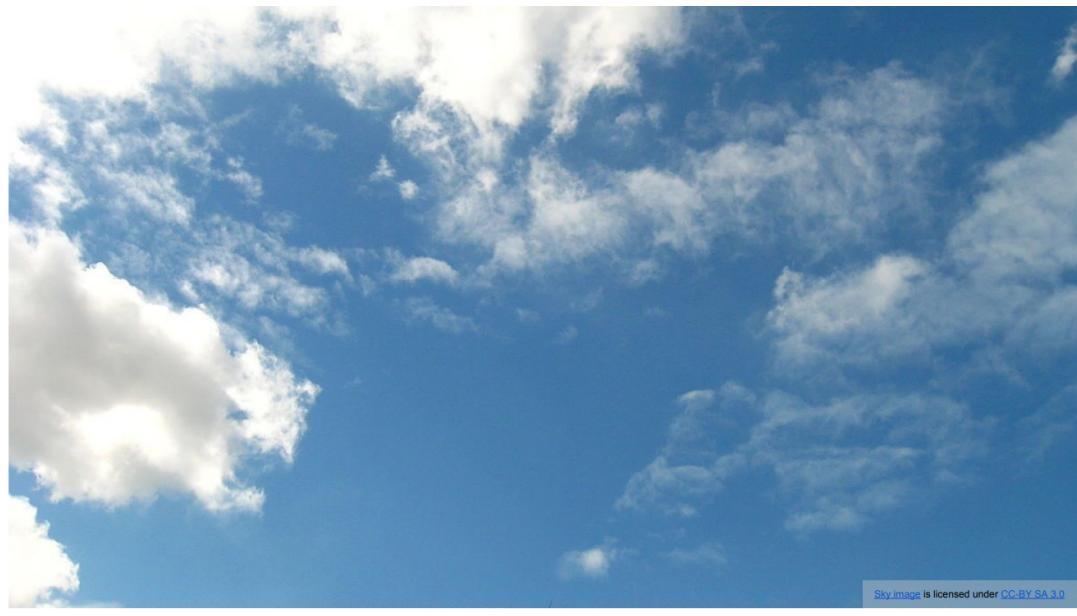
- image에 의존적이지 않은 방식
- weight을 모두 고정시킴
- 어떤 type의 이미지에서 일반적으로 어떤 뉴런을 활성화하는지 살펴보는 것
- 특정 뉴런 혹은 class에 대한 score를 높이는 방향으로 이미지를 합성하는 것

- regularization term 필요: 기존에는 weight이 overfit하는 것을 방지하는 데에 썼다면, 여기서는 생성된 이미지가 특정 네트워크의 특성에 overfit되는 것을 방지하기 위해 사용
  - 이미지가 특정 뉴런의 값을 최대화시키는 방향으로 생성되길 원함
  - 이미지가 자연스러워보여야함
- 초기화 → 이미지가 네트워크 통과 → 관심있는 뉴런 score 계산 → 이미지의 각 픽셀에 대한 해당 뉴런 score의 gradient를 계산하여 backprop 수행 (gradient ascent 이용해서 이미지 픽셀 자체를 업데이트)

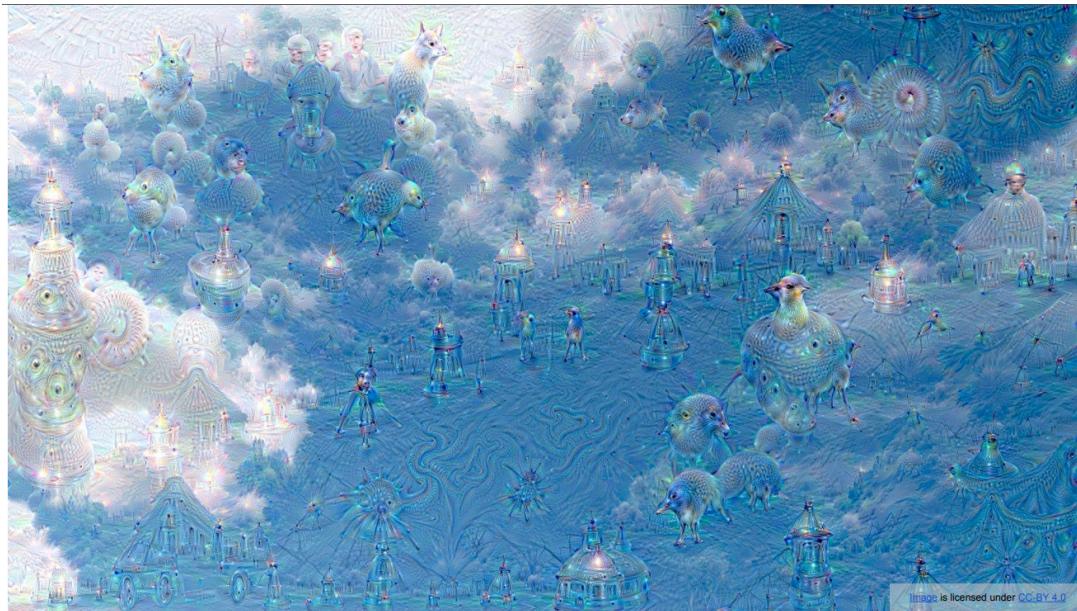


- regularizer: penalize L2 norm of generated image
  - Gaussian blur image
  - Clip pixels with small values to 0
  - Clip pixels with small gradients to 0
- multi modality
  - 각 class마다 clustering algorithm 수행: 한 class 내 서로 다른 mode끼리 다시 한 번 class가 나누고 그 나뉜 mode들과 가까운 곳으로 initialize함
  - 이미지를 생성할 때 multimodality를 명시하면 다양한 결과를 얻을 수 있음
  - input image의 pixel을 바로 optimize하는 게 아니라 FC6를 optimize하는 것
- fooling image도 할 수 있음
- DeepDream: 재미있는 이미지를 만드는 것
  - model이 이미지의 어떤 특징을 찾고 있는지 짐작할 수 있음

- gradient를 activation 값으로 두고 이미지에 바로 backwarding 시키는 것



Before



After

- 여기서 개 그림이 많아 보이는 이유는 데이터셋의 대부분이 개 카테고리였기 때문
- 다른 layer에서 혹은 다른 dataset을 이용하면 더 다양한 결과를 얻을 수 있음

## Feature Inversion

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

Given feature vector

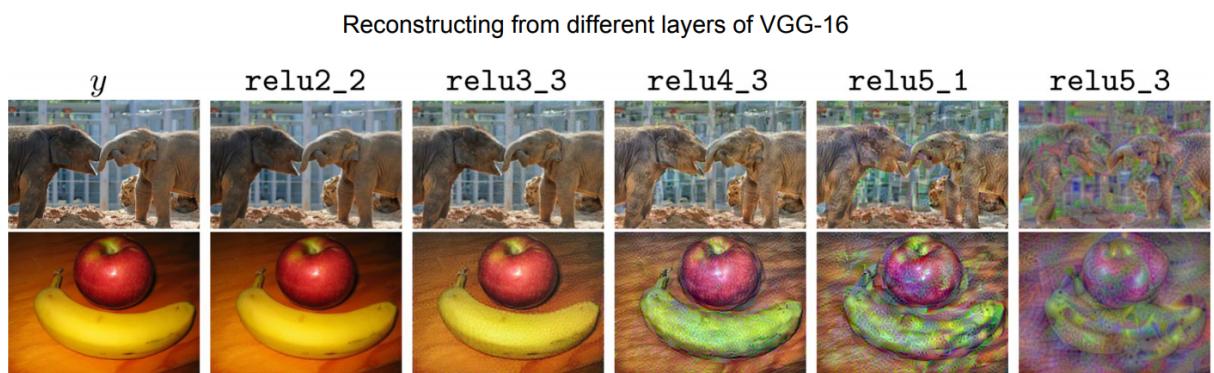
$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

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

Features of new image

Total Variation regularizer  
(encourages spatial smoothness)

- 주어진 이미지가 있을 때 forward하여 얻은 feature와 주어진 feature vector 간의 차 이를 줄이는 이미지를 찾는 것
- total variation regularizer 사용: 상하좌우 인접 pixel 간의 차이에 대한 패널티 부여, 생성된 이미지가 자연스러운 이미지가 되게 함
- 네트워크가 깊어질수록 디테일한 부분은 줄어들지만 형태는 유지하고 있는 모습을 볼 수 있음(아래 사진)



## Texture Synthesis

- 컴퓨터 그래픽스에서 아주 오래된 문제
- nearest neighbor: 좋은 편. 신경망 사용 x
  - scan line을 따라서 한 픽셀씩 이미지를 생성해나가는 것
  - 생성해야 할 픽셀 주변의 이미 생성된 픽셀을 살펴보고 가장 가까운 픽셀을 계산하여 입력 patch로부터 한 pixel을 복사해넣는 방식

## Neural Texture Synthesis

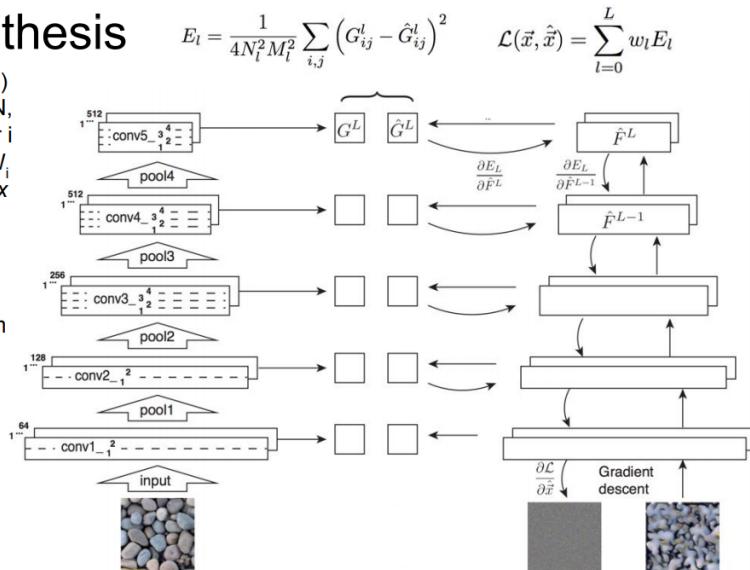
- Gram matrix
  - Neural texture synthesis를 구현하기 위해 이용
  - 공간 정보를 모두 날려버리고 features 간의 co-occurrence만 포착함
  - → 이미지의 각 지점에 해당하는 값들을 평균화시켰기 때문
  - 계산 효율적
- pretrained CNN의 매 layer에서 이미지를 forward하여 Gram matrix를 계산함 → 다른 네트워크에서 generated image에 random noise를 추가함(여기서도 Gram matrix 계산하는 과정 포함) → 이 둘 사이의 loss 계산한 후 loss를 줄이는 방향으로 gradient 만듦 → image를 바꾸는 연산 반복 → Texture Synthesis 완성

## Neural Texture Synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer  $i$  gives feature map of shape  $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

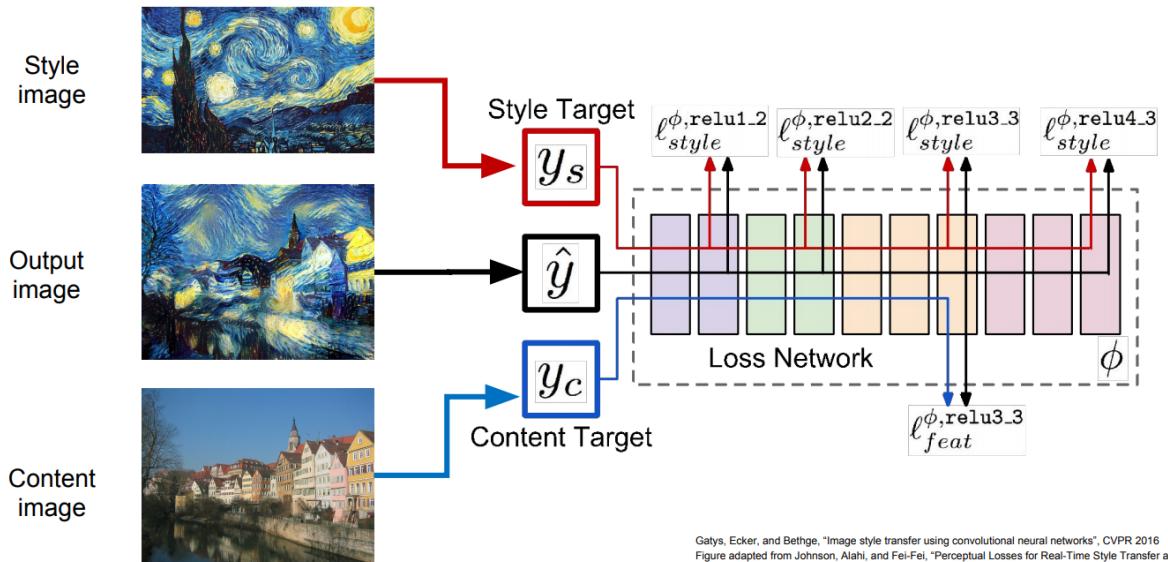
$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \text{ (shape } C_i \times C_i\text{)}$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5



## Neural Style Transfer

- Feature reconstruction + Gram reconstruction
- 네트워크에 content/style 이미지를 통과시키고 gram matrix와 feature map을 계산함
- final output image는 random noise로 초기화시킴
- forward/backward를 반복하여 계산, gradient ascent을 이용해 image update



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks". CVPR 2016  
 Figure adapted 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.

- DeepDream보다 신경써야할 부분이 많음
  - DeepDream의 경우, 네트워크에 layer와 반복 횟수만 조절하면 특이한 특징이 이미지 전체에 보임
  - Style Transfer의 경우, 원하는 결과를 만들기 위해 다양한 것들을 조절해야 함
    - hyperparameter: style/content loss 간의 weight, size of style image, style image의 수
  - 😞
    - need large memory
    - slow
  - solution: train another neural network to perform style transfer
    - content image만을 입력으로 받아서 결과를 출력할 수 있는 단일 네트워크를 학습시키는 방식
    - train 시에는 오래 걸리지만 한 번 학습시키고 나면 이미지를 network에 통과시키면 결과가 바로 나옴