

Type something...

## 01) Visualizing and Understanding

### 01\_Visualize Filters

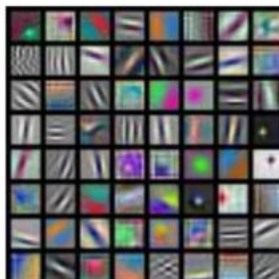
1. First Layer
2. Last Layer
3. Dimensionality Reduction
4. Maximally Activation Patches
5. Occlusion Experiments
6. Saliency Maps
7. Intermediate features via guided backprop
8. Visualizing CNN features: Gradient Ascent
9. Fooling Images / Adversarial Examples
10. DeepDream: Amplify existing features
11. Feature inversion
12. Texture Synthesis
13. Gram Matrix
14. Neural Style Transfer

## 01) Visualizing and Understanding

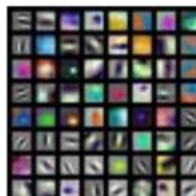
### 01\_Visualize Filters

#### 1. First Layer

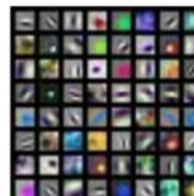
## First Layer: Visualize Filters



AlexNet:  
64 x 3 x 11 x 11



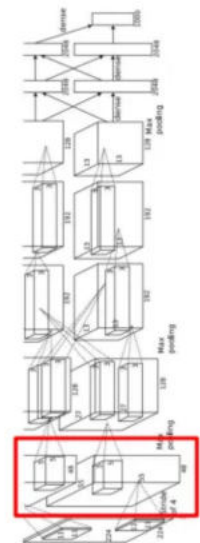
ResNet-18:  
64 x 3 x 7 x 7



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  
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016  
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

- AlexNet 첫번째 conv결과로 3 x 11 x 11 feature map이 나오게 되며, filter의 가중치와 input 이미지의 내적을 통해 구해진다.
- feature map은 보통 oriented edge나 complementary color(보색, 색상 대비를 이루는 한쌍의 색상) 등을 찾아낸다.
- 첫번째 layer는 이미지와 가장 가까운 곳에 접하기 때문에 이미지의 특성을 잡아내는데 유리하다.

## Visualize the filters/kernels (raw weights)

We can visualize filters at higher layers, but not

Weights:



Weights:



layer 1 weights

16 x 3 x 7 x 7

layer 2 weights

20 x 16 x 7 x 7



that interesting

(these are taken  
from ConvNetJS  
CIFAR-10  
demo)

Weights:

([...])

layer 3 weights

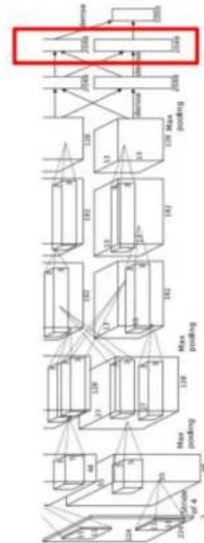
20 x 20 x 7 x 7

- $16 \times 3 \times 7 \times 7 \rightarrow 20 \times 16 \times 7 \times 7$  layer들을 통과할 수록 점점 깊어진다.
- 하지만 layer가 깊어질수록 시각화로 통해 우리가 볼 수 없다.

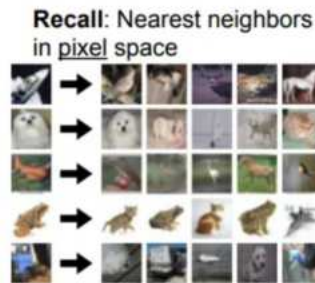
## 2. Last Layer

### Last Layer: Nearest Neighbors

4096-dim vector



Test image L2 Nearest neighbors in feature space



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

- NN을 pixel 단위로 적용시켰을 때는 정확도가 낮지만 feature space에서는 무난한 성능을 보여준다.
- 코끼리를 예시로 들면 코끼리가 왼쪽에 있든, 오른쪽에 있든 같은 객체로 인식한다는 것이다.

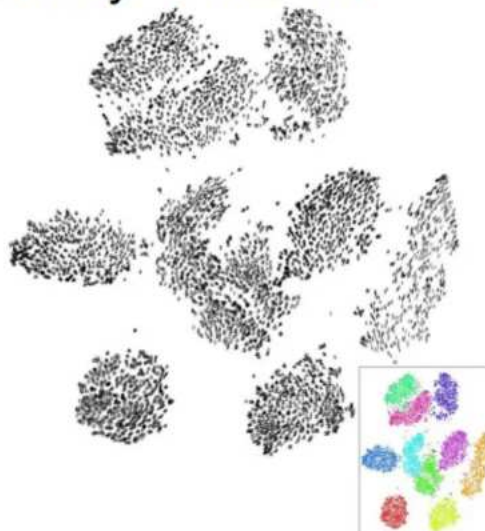
## 3. Dimensionality Reduction

### Last Layer: Dimensionality Reduction

Visualize the "space" of FC7  
feature vectors by reducing  
dimensionality of vectors from  
4096 to 2 dimensions

Simple algorithm: Principle  
Component Analysis (PCA)

More complex: **t-SNE**



Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008  
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.

- $4096 \rightarrow 2D$  차원을 축소할 때 PCA라는 기법을 많이 사용한다.
- 딥러닝에서는 **t-SNE** 라는 non-linearity를 가진 함수를 이용하여 feature를 차원 축소한다.

# Visualizing Activations

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

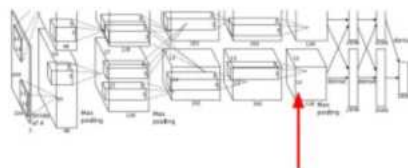


Yosinski et al. "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014. Figure copyright Jason Yosinski, 2014. Reproduced with permission.

- AlexNet의 conv5 activation map에서 사람 얼굴과 비슷한 이미지를 확인할 수 있었다.

## 4. Maximally Activation Patches

### Maximally Activating Patches



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

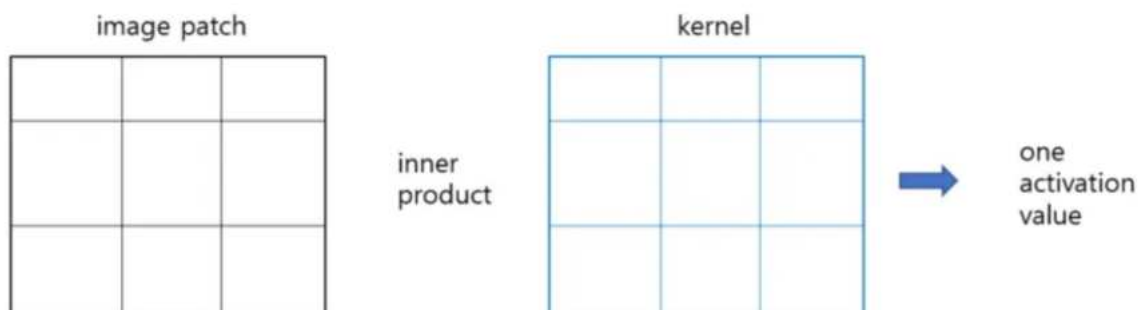
Run many images through the network, record values of chosen channel

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. Generated with DeepVis.

- Maximally Activating Patches라는 방법은 특정 neuron에서 activation이 가장 큰 patch를 시각화 하는 방법이다.



- image patch 와 kernel은 inner product했을 때 하나의 activation 값이 나오게 된다.
- activation 값이 크다는 것은 해당 filter에 대해 높게 반응했다는 것이고, activation값이 높은 부분을 캡처해보면 neural network가 중요하게 보는 부분을 확인할 수 있다.

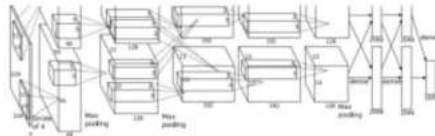
## 5. Occlusion Experiments





## Occlusion Experiments

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

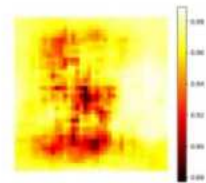


Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks" ECCV 2014

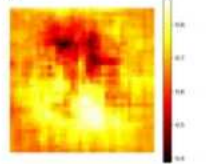
Start image = CC0 public domain  
Elephant image = CC0 public domain  
Go-Karts image = CC0 public domain

occlusion: 어떤 물체 앞에 장애물이 있어서 가려지는 현상

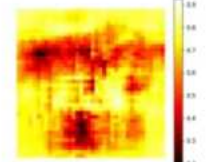
- 이미지를 가리는 부분을 옮겨가면서 어떤 변화가 있는지 측정을 하여 heatmap으로 나타내었다.
- 특정 구간을 가렸을 때 score가 극적으로 변한다면 그 부분이 classification하는데 매우 중요한 부분이라고 판단하는 것이다.
- 그래서 빨간 부분일수록 classification할 때 중요한 부분임을 나타낸다.



African elephant, *Loxodonta africana*



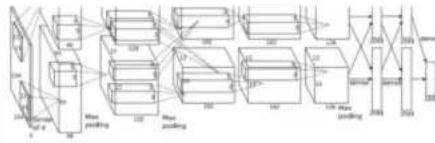
go-kart



## 6. Saliency Maps

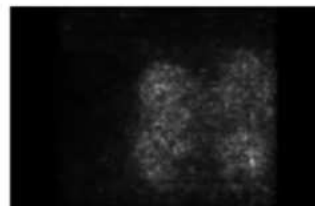
## Saliency Maps

## How to tell which pixels matter for classification?



Dog

Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



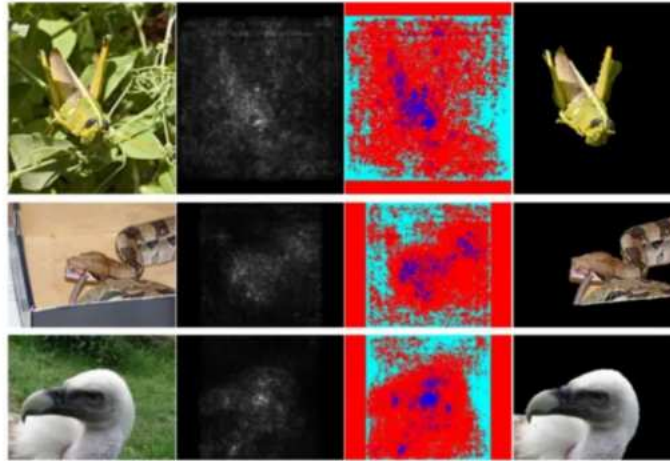
Simonyan, Vedaldi, and Zisserman, 'Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps', ICLR Workshop 2014.

Saliency map: 화면에서 눈에 띄는 영역, 다른 영역에 비해 픽셀값의 변화가 급격한 부분들을 모아서 mapping

- 즉 관심있는 물체를 관심이 없는 배경으로부터 분리시키는 것을 의미한다.
- image 각 pixel의 gradient를 map으로 시각화 한 것이다.
- activation이 보고 있는 부분에서 backward를 하게 되면 값이 살아있게 된다.
- 따라서 Neural Network가 image의 어느 부분을 보고 있는지 알 수 있다.

# Saliency Maps: Segmentation without supervision

Use GrabCut on saliency map

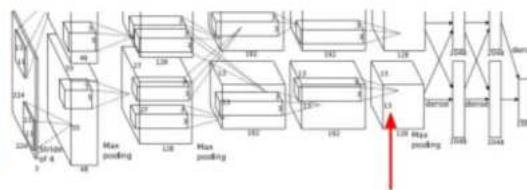


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.  
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

- Saliency map를 이용하면 label을 주지 않는 un-supervised로 segmentation 작업을 진행할 수 있다.
- 하지만 정확도는 매우 떨어진다.

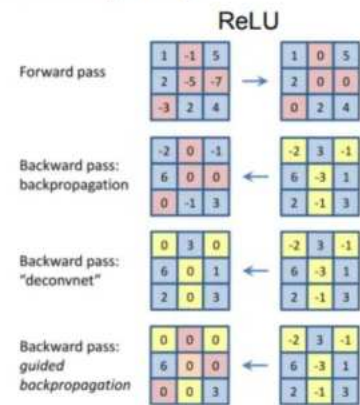
## 7. Intermediate features via guided backprop

### Intermediate features via (guided) backprop



Pick a single intermediate neuron, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of neuron value with respect to image pixels



Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Zeller and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014  
Sonnenschein et al. "Reasoning by Semantics: The 3D Probabilistic Map" ICLR Workshop 2016

- 중간의 뉴런을 골라서 이미지의 어떤 patch가 영향을 크게 줬는지 확인한다.
1. Forward Pass
    - $\text{ReLU} = \max(0, x)$  사용
  2. Backward pass: backpropagation
    - 구해진 Gradient들 중 ReLU를 활성화했던 위치에만 Gradient 전달
    - 음수였던 영역은 전달하지 않는다.
  3. Backward pass: deconvnet
    - 구해진 gradient가 음수이면 backward pass하지 않고 0 전달
    - 양수면 그대로 전달
  4. Backward Pass: Guided Back Propagation
    - 기존의 Back Propagation + deconvnet
    - ReLU가 활성화 된 동시에 gradient가 양수인 값들만 전달
    - 이렇게 되면 activation이 높은 부분만 뒤로 갈수록 살아있게 되는데 이것은 Network가 classification에 이용했다는 걸로 해석할 수 있다.

## 8. Visualizing CNN features: Gradient Ascent

### 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

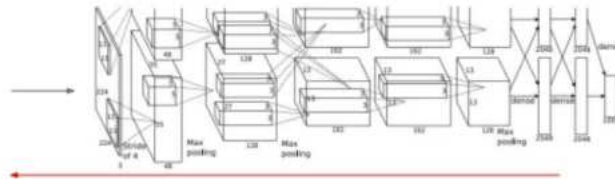
Natural image regularizer

- 어떤 neuron(weight)가 주어졌을 때 그 neuron을 활성화시키는 generalized image는 어떻게 있을까?
- Gradient Ascent는 Loss가 최대가 되는 Parameter를 찾는 방법이다.
- 고정된  $W$ 에 대해 input image의 pixel value를 gradient ascent로 바꿔가면서 score가 최대가 되도록 한다.
- Overfit를 막기 위해 Regularization을 사용한다.

$$I^* = \operatorname{argmax}_I f(I) + R(I)$$

## Visualizing CNN features: Gradient Ascent

1. Initialize image to zeros



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

score for class c (before Softmax)

Repeat:

2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

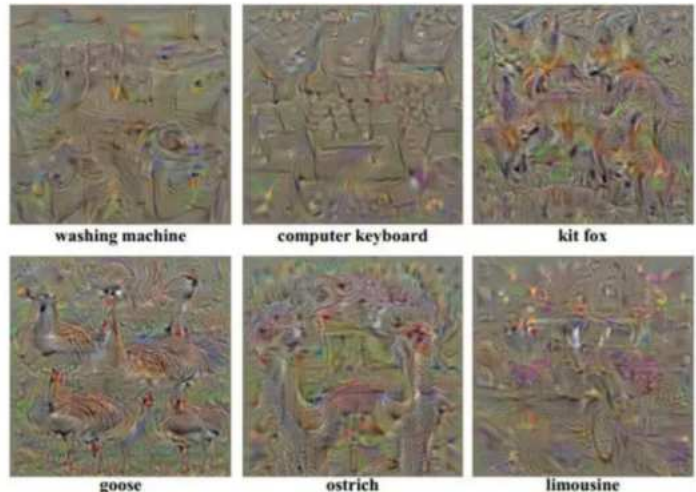
1. pixel 초기화
2. score 계산을 위해 forward
3. 뉴런값과 기울기를 얻기 위해 backprop
4. gradient ascent 진행



## Visualizing CNN features: Gradient Ascent

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

Simple regularizer: Penalize L2 norm of generated image



Yosinski et al. "Understanding Neural Networks Through Deep Visualization", ICM, DL Workshop 2014.  
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014.  
Reconstructed with permission.

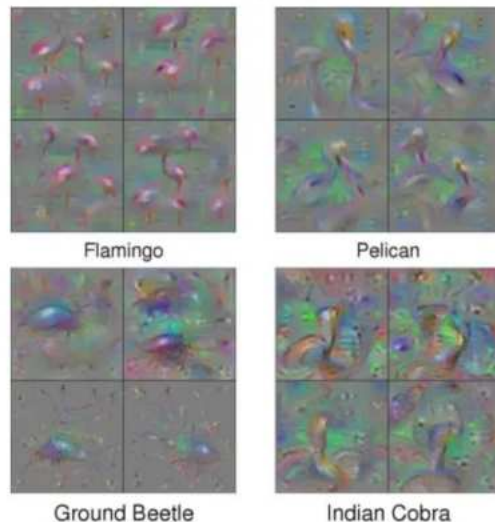
- L2 regularization을 사용시 형태를 알아보기 힘들다.
- 그래서 다른 regularizer를 사용하게 되었다.
- score를 최대화해야하기 때문에 regularization은 빼준다.

## Visualizing CNN features: Gradient Ascent

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

Better regularizer: Penalize L2 norm of image; also during optimization periodically

- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



Yosinski et al. "Understanding Neural Networks Through Deep Visualization", ICM, DL Workshop 2014.  
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reconstructed with permission.

1. Gaussian blur 이미지
2. 픽셀 중 작은 값을 가지는 것을 0으로
3. 픽셀 중 작은 gradient 가지는 것을 0으로

### 9. Fooling Images / Adversarial Examples

## Fooling Images / Adversarial Examples

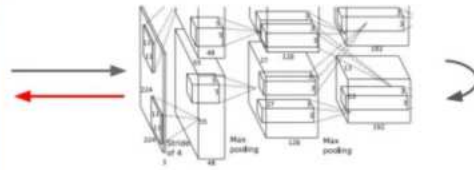
- (1) Start from an arbitrary image
- (2) Pick an arbitrary class
- (3) Modify the image to maximize the class
- (4) Repeat until network is fooled

1. 임의의 이미지에서 시작
2. 임의의 클래스를 선택
3. 클래스를 극대화하기 위해 이미지를 변경
4. network를 속일 때 까지 반복

#### 10. DeepDream: Amplify existing features

## DeepDream: 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

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

- 위에서는 특정 neuron을 maximize하는 방향으로 시각화를 했다면, Deep Dream는 neuron activations를 증가시키는 방향의 차이가 있다.
1. image와 CNN의 한 layer를 선택
  2. layer까지 forward pass하고 activation 계산
  3. layer의 gradient를 activation과 같게 설정
  4. backward pass, update image.

#### 11. Feature inversion

## Feature Inversion

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

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

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

$\Phi_0$  → Given feature vector  
 $\Phi(\mathbf{x})$  → Features of new image

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

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

Total Variation regularizer (encourages spatial smoothness)

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

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 49 May 10, 2017

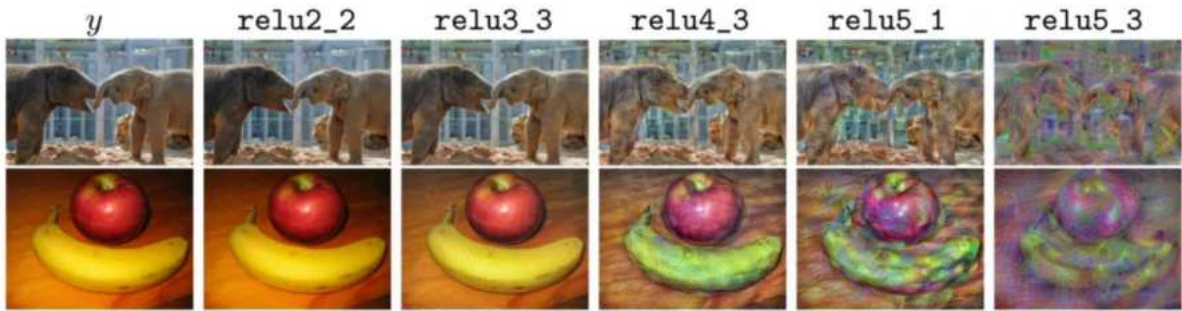
- 이미지가 네트워크를 통과시킨 후 만들어진 feature vector를 저장해준다.
- 이 feature vector를 이용해서 이미지를 재구성한다.
- 이로부터 이미지의 어떤 정보가 특정 벡터에서 포착되는지를 짐작할 수 있다.





# Feature Inversion

Reconstructing from different layers of VGG-16

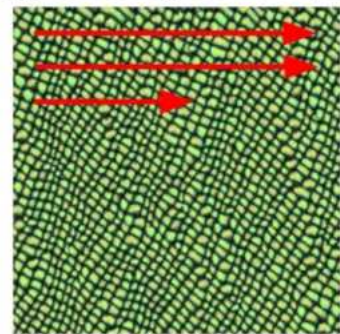
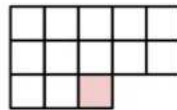


- layer가 깊어질수록 변형이 심해지지만 이미지의 기초 형태는 유지하고 있다.

## 12. Texture Synthesis

### Texture Synthesis: Nearest Neighbor

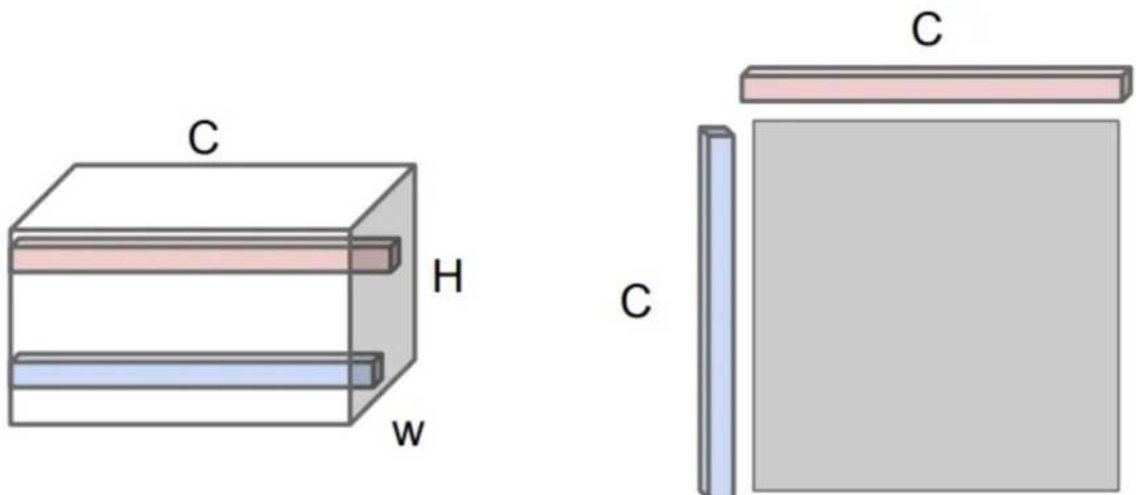
Generate pixels one at a time in scanline order; form neighborhood of already generated pixels and copy nearest neighbor from input



Wei and Levoy, "Fast Texture Synthesis using Tree-structured Vector Quantization", SIGGRAPH 2000  
 Efros and Leung, "Texture Synthesis by Non-parametric Sampling", ICCV 1999

- 이미 생성된 픽셀들을 살핀 후 입력 패치에서 가장 가까운 픽셀을 계산해서 복사하여 붙여넣는 방식이다.
- 하지만 naive한 방법으로 사용했을 때 복잡한 패턴을 키워보면 이상해지는 단점이 있었다.

## 13. Gram Matrix



1. Input texture를 가지고 CNN에 넣는다
2. 어떤 layer에서 convolution된 feature  $C \times H \times W$ 를 가져온다.

3. C vector 2개를 선택하고 외적을 진행해서 C x C matrix를 만든다. C x C vector는 특정 포인트 두점에서 공통적으로 발견되는 성질에 대한 정보를 담고 있다.
4. 모든 point에 대하여 만들고 average를 시키면 gram matrix가 생성된다.

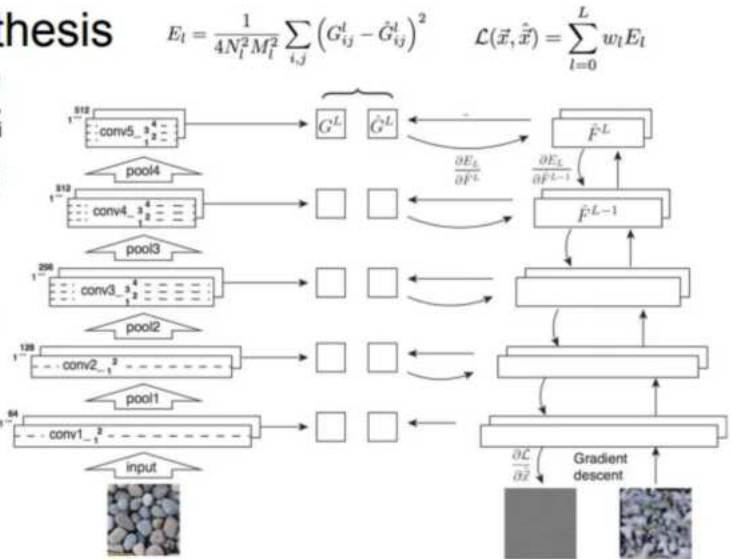
## Neural Texture Synthesis

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:

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

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

Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015  
Figure copyright Leon Gatys, Alexander S. Ecker, and Matthias Bethge, 2015. Reproduced with permission.



Fei-Fei Li & Justin Johnson & Serena Yeung

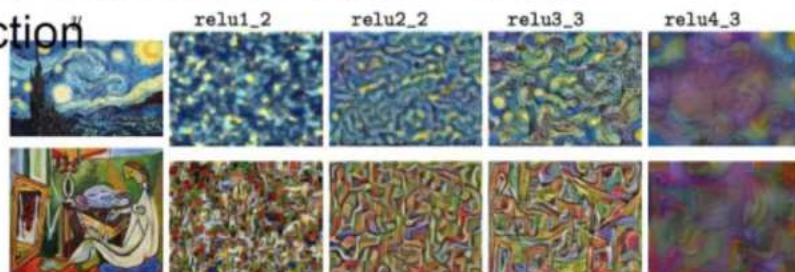
Lecture 11 - 61 May 10, 2017

1. 입력 이미지를 pretrain된 VGG network에 통과시켜 다양한 레이어에서 gram matrix를 계산한다.
2. 원본 이미지와 생성된 이미지의 gram matrix 간의 차이를 L2 norm을 이용하여 loss를 계산한다.
3. 계산된 loss를 이용하여 backpropagation을 진행
4. gradient ascent를 통해 이미지 픽셀들을 조금씩 업데이트
5. Gram matrix 계산 → loss 계산 → backprop을 반복하여 입력 텍스처와 유사한 텍스처를 만든다.

### 14. Neural Style Transfer

## Neural Style Transfer: Feature + Gram Reconstruction

Texture synthesis  
(Gram reconstruction)



Feature reconstruction

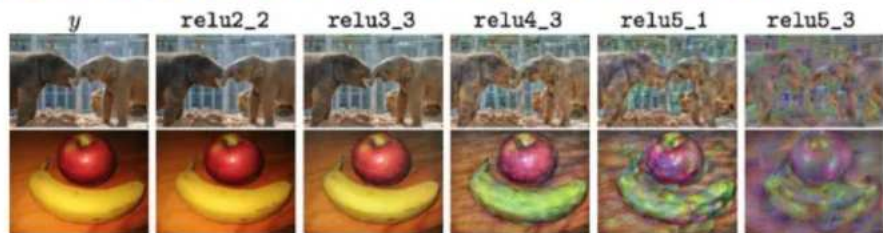


Figure from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

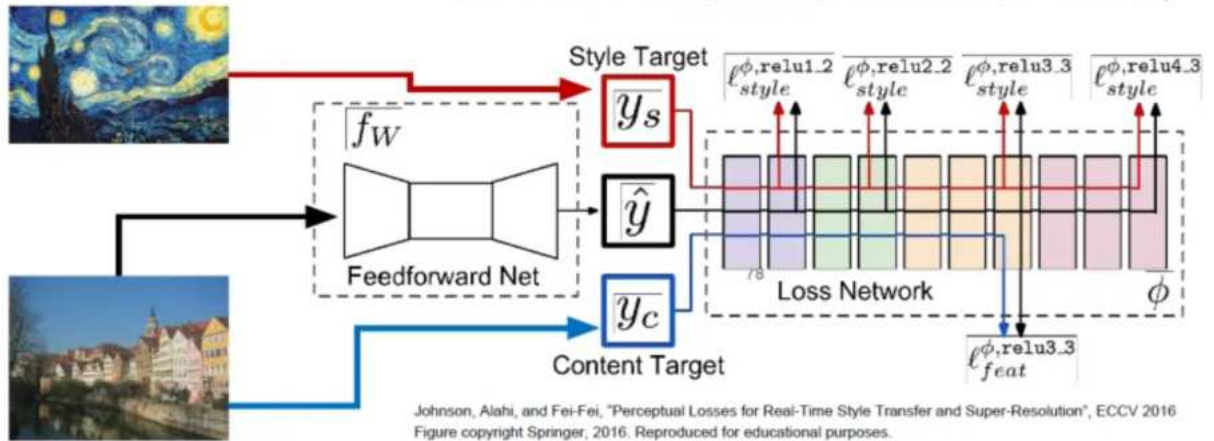
Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 64 May 10, 2017

- gram matrix를 이용한 텍스처 합성을 두 이미지에 대해 적용하고 feature inversion을 조합한다.
- 이것이 style transfer이다.

# 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



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 11 - 78 May 10, 2017

- Fast Style Transfer는 style 이미지를 고정시킨 후 content image만을 입력 받아 결과를 출력할 수 있는 단일 네트워크를 학습시키는 방법이다.
- semantics segmentation과 매우 유사하다.