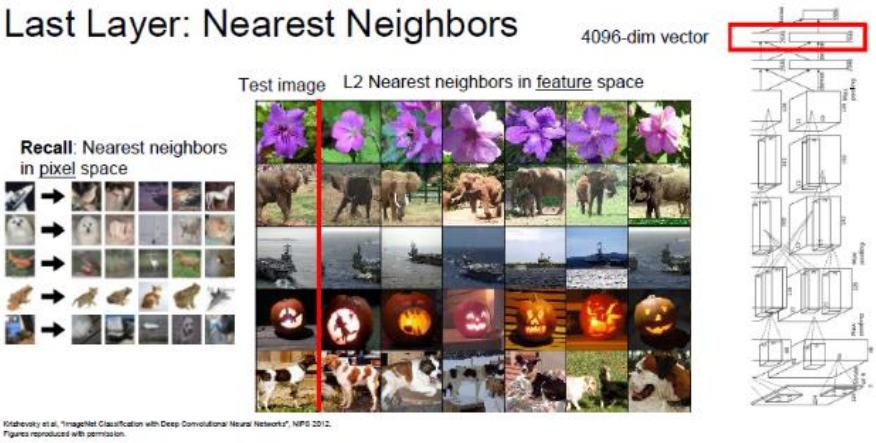
Lecture12

시각화(Visualizing)의 중요성.

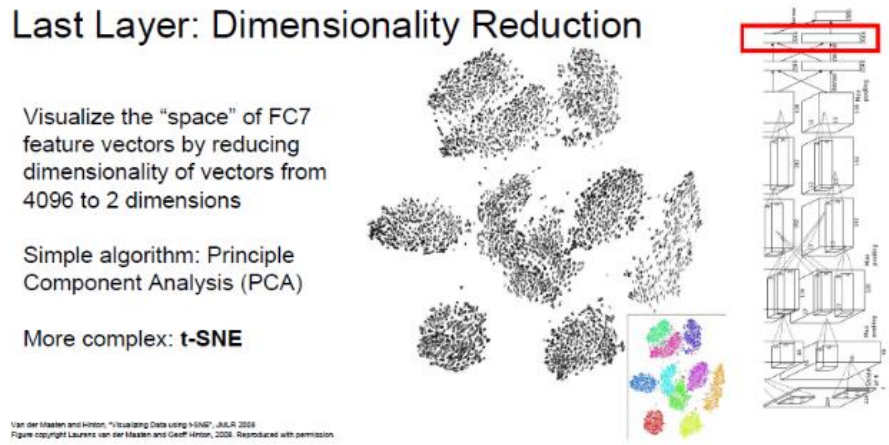
-> 딥러닝 동작하는 이유를 시각적으로 설명하기 위함

-> 딥러닝 내부에서 어떤 동작을 하는지, 설명하기 위함.

계층을 많이 거칠수록 직관적으로 해석하기 힘들다. (ex. AlexNet, ResNet-18)

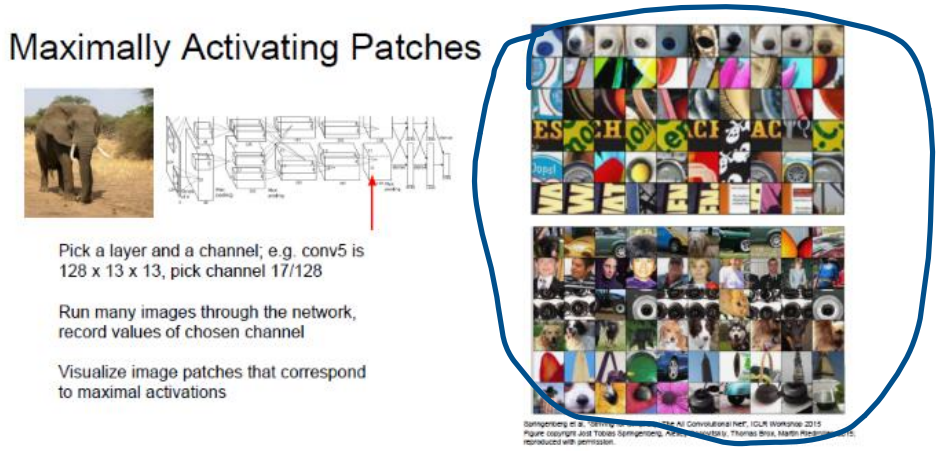


-> CNN에 넣고 돌리면, 유사한 이미지 검출. 마지막에 2차원 축소 하면,



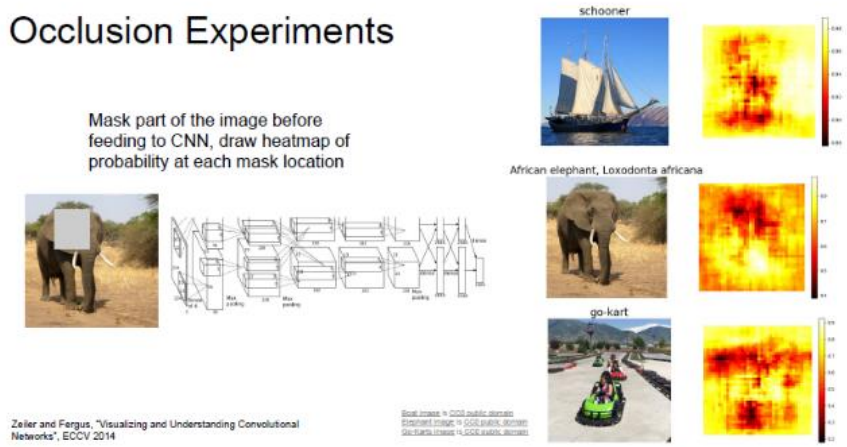
이런 식으로 군집화 되어 있음.

#그래서 모든 계층에서 무슨 일이 일어나는지 알기 어렵다는 말은 틀림. (대략적으로 알 수 있음)



->어떤 이미지가 각 뉴런들의 활성화 최대치로 만드는지 아는 방법을 'Maximally Activating Patches'

#각 channel에서 활성화 한 것. (눈을 가장 활성화 추측 가능)



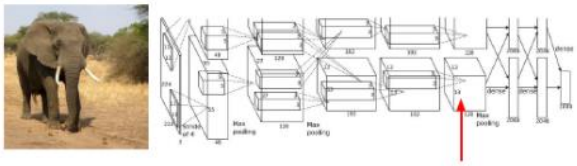
-> Occulusion Experiments방식.

사진의 일부분 가려서 점수에 변화 있는지 없는지로 확인.

-> Saliency Maps방식도 존재.(어떤 픽셀을 보고 이미지를 분류했는지 알아보는 방법)

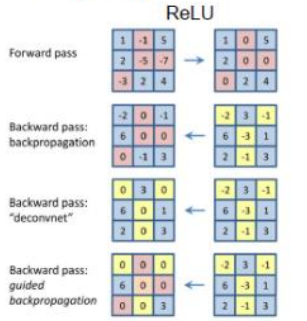
#성능이 좋지는 않음.

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)

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

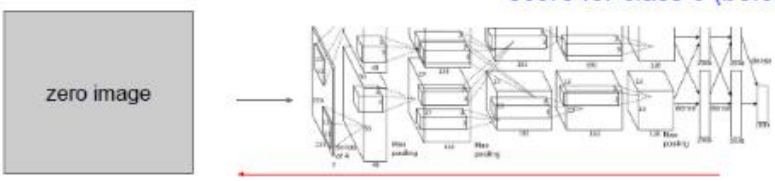
-> 활성화 함수로 ReLU 쓰는 경우 많음. 위 그림처럼 양수값만 뽑아서 기울기 구하면, 활성화 된 위치를 확인 가능. #그냥 또 다른 방식.

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

-> 기울기 구할 때, 역전파로 구했는데, 그것의 또다른 방식: 'Gradient Ascent'(Weight값 고정하고, 입력 이미지 찾는 방식)

1. 첫 이미지 0으로 초기화

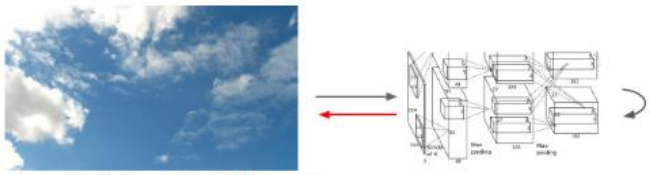
2. 이미지의 현재 스코어 계산.

3. 이미지 픽셀 단위로 쪼개고 역전파 해서 기울기 구하기

4. 업데이트 진행.(계속 반복)

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

Equivalent to:

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

-> 딥 드림 방식.

1. 입력 이미지를 CNN중간까지만 통과.

2. 기울기를 저장.

3. 역전파로 이미지 업데이트

4. 반복.

#많은 다양한 그림 뽑아내기 가능.

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)

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

Given feature vector

Features of new image

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

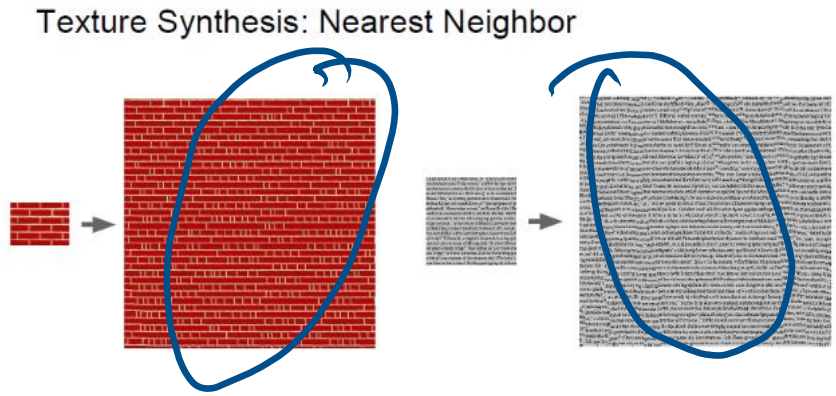
$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer (encourages spatial smoothness)

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

-> Feature Inversion방식. 각 계층에서 어떤 포착 하고 있는지 짐작 가능.

특정 계층에서 activation map추출하고, 이걸 갖고 이미지 재구성하는 것.



-> 텍스트 합성.

특정 계층에서 Activation Map가져오고, 'Gram Matrix'를 생성.

Gram Matrix: 다른 공간에 있는 Channel을 가지고, 계산해서 새로운 Matrix생성하는 것.

#계산량에서 이득보기 참 좋은 방식.

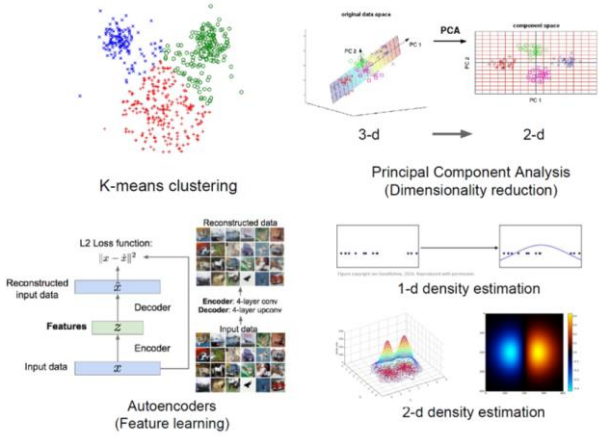
-> Neural Style Transfer방식.(예술적으로 합성한 것.)

이미지 생성에 Control해줘야 할 것 많음.

forward, backward과정 반복해야 해서 느림.

Lecture 13

비지도 학습: 데이터에 레이블이 없을 때, 학습하는 방식.



비지도 학습 예시:

1. clustering(군집화)

2. Dimensionality Reduction

3. Feature Learning

4. Density Estimation

#비지도 학습은 데이터를 많이 모을 수 있다는 장점. (동시에 분류 기준을 알 수 없다는 단점도....)

비지도 학습에 들어가는 'Generative Models'는 분포추정(Density estimation)이 중요.

Generative Models 종류

1. Pixel RNN / CNN

2. Variational Autoencoder

3. GAN

1. Pixel RNN / CNN

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

첫번째로 체인 룰로 이미지를 1차원 분포 곱의 형태로 나타내기.

그리고 pixel RNN방식은 LSTM 방식을 활용하는데, 속도가 느린 단점 존재.

-> 이걸 보완하고자 나온게 Pixel CNN.(특정 영역만 사용해서 속도가 더 빠름)

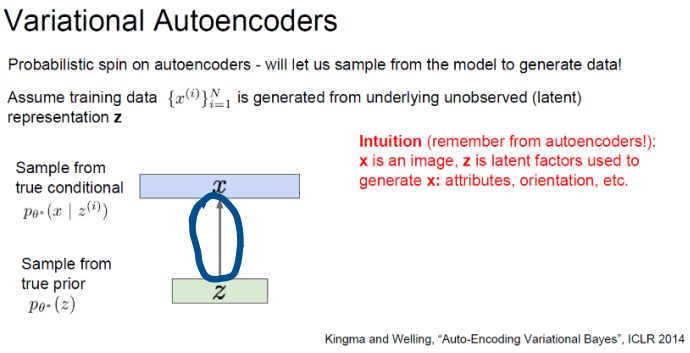
2. variational Autencoders

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

->VAE는 앞에 식처럼 나타내기 불가능해서, 하한선을 구해서 계산 가능 형태로 바꿔야 함.

Encoder: input값이 들어오면, z를 추출 하는 과정(ex. sigmoid, FC, RELU, CNN)

Decoder: Encoder의 반대과정.(ex. sigmoid, FC, ReLU, CNN)



이런식으로 z로 어떤 값을 추정함.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

이 과정을 위해서 먼저 이 식 적용.(파란색 동그라미)

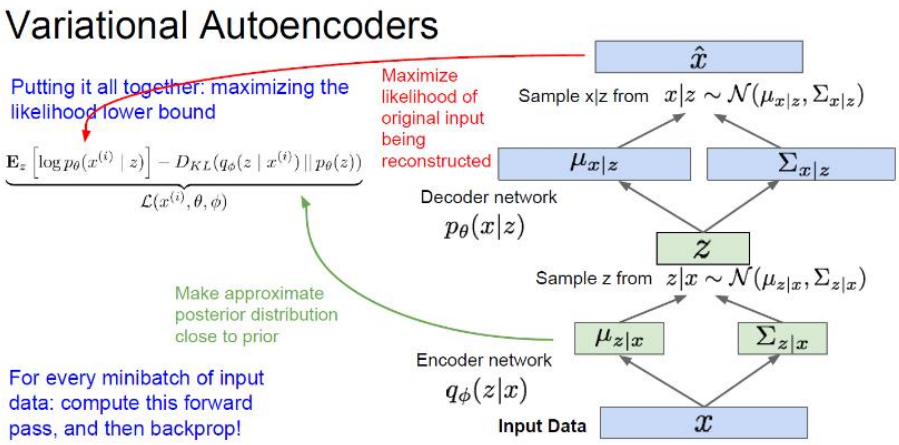
$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)}, \theta, \phi)$$

Variational lower bound ("ELBO")

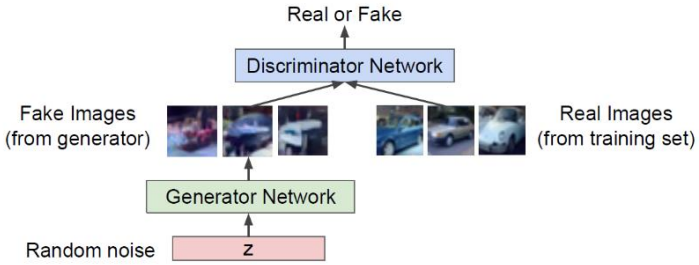
$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi)$$

Training: Maximize lower bound

-> 앞의 계산 과정 거치고, 최종 이렇게 두개 나온 식으로.



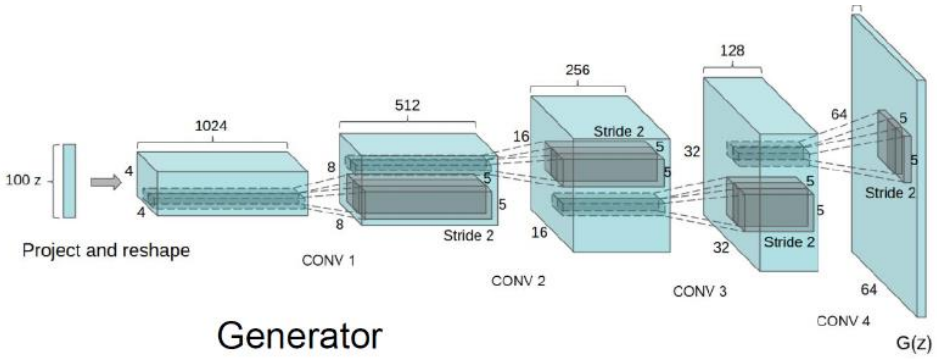
->이렇게 최종 training



->GAN방식. 위에 처럼 따로 계산 안하고 sample(결과)만 뽑아내는 것.

Generator Network(실제처럼 보이는 가짜 이미지)

Disciriminator Network는 진짜 이미지, 가짜 이미지 구별.



-> DCGAN. CNN이랑 위 GAN 섞어서 더 진짜 같은 이미지 뽑아내는 것.