Deep Learning

Prof. Jaewook Lee and Prof. Saerom Park

Department of Convergence Security Engineering
psr6275@sungshin.ac.kr

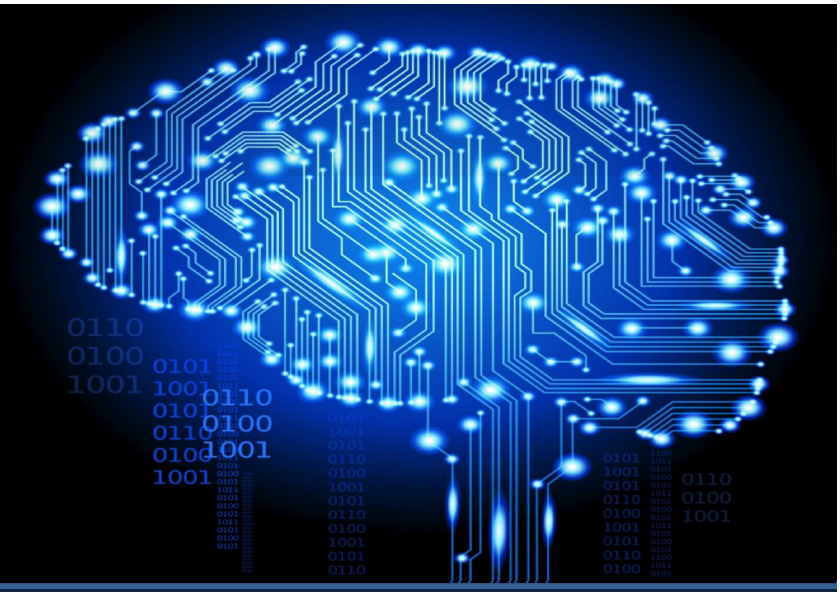
Reference

- [Goodfellow et al. (2016)] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning, The MIT Press, 2016.
- [Hinton. (2015)] G. Hinton, Online course on Neural Networks for Machine Learning.

  https://www.coursera.org/learn/neural-networks
- [Larochelle. (2014)] H. Larochelle, Online course on Neural Networks.

  http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html
- [Gavves. (2017)] E. Gavves, UvA Deep Learning Course.

  http://uvadlc.github.io/
- Borrows slides (some modified) from Larochelle & Hinton & Gavves



Prof. Saerom Park

Department of Convergence Security Engineering
psr6275@sungshin.ac.kr

Lecture 4: Convolutional Neural Networks for Computer Vision

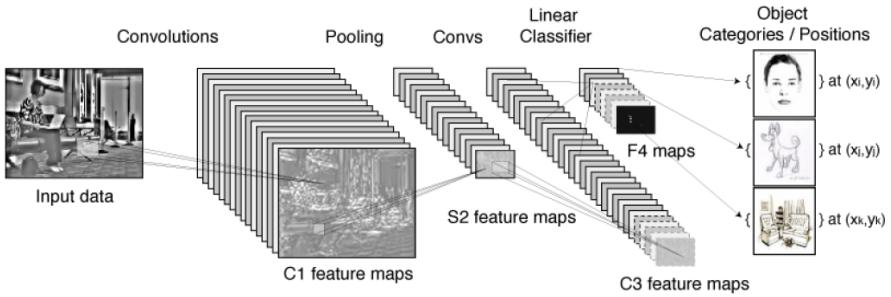
INTRODUCTION TO DEEP LEARNING

Lecture Overview

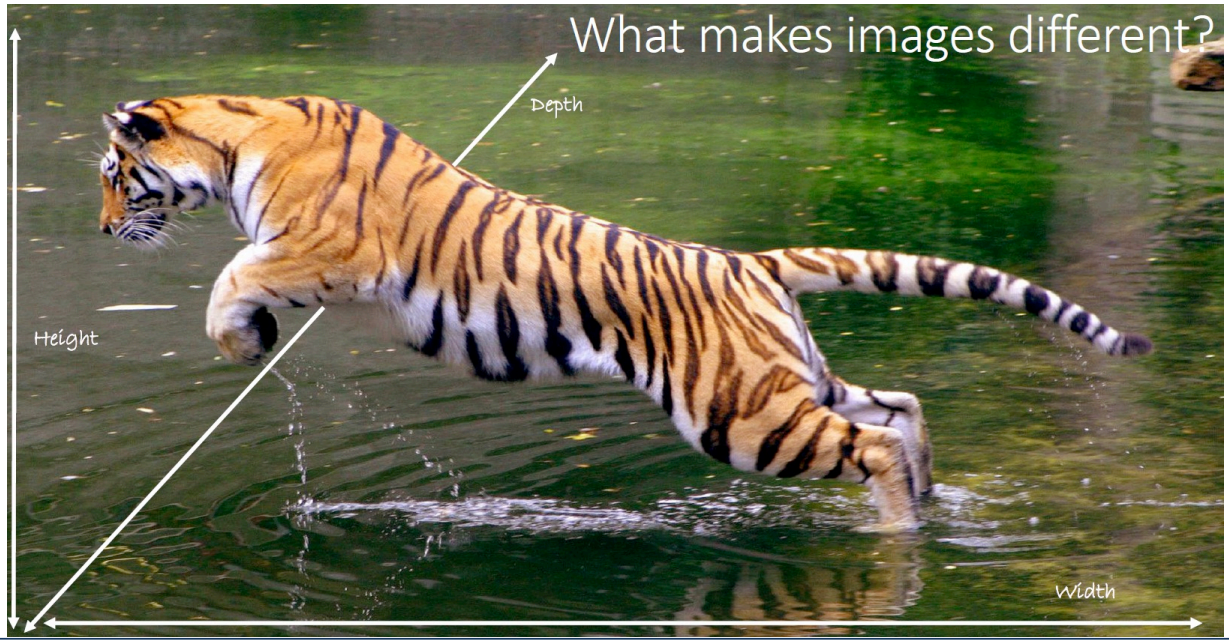
- What are the Convolutional Neural Networks?
- Why are they important in Computer Vision?
- Differences from standard Neural Networks
- How to train a Convolutional Neural Network?

Convolutional Neural Networks

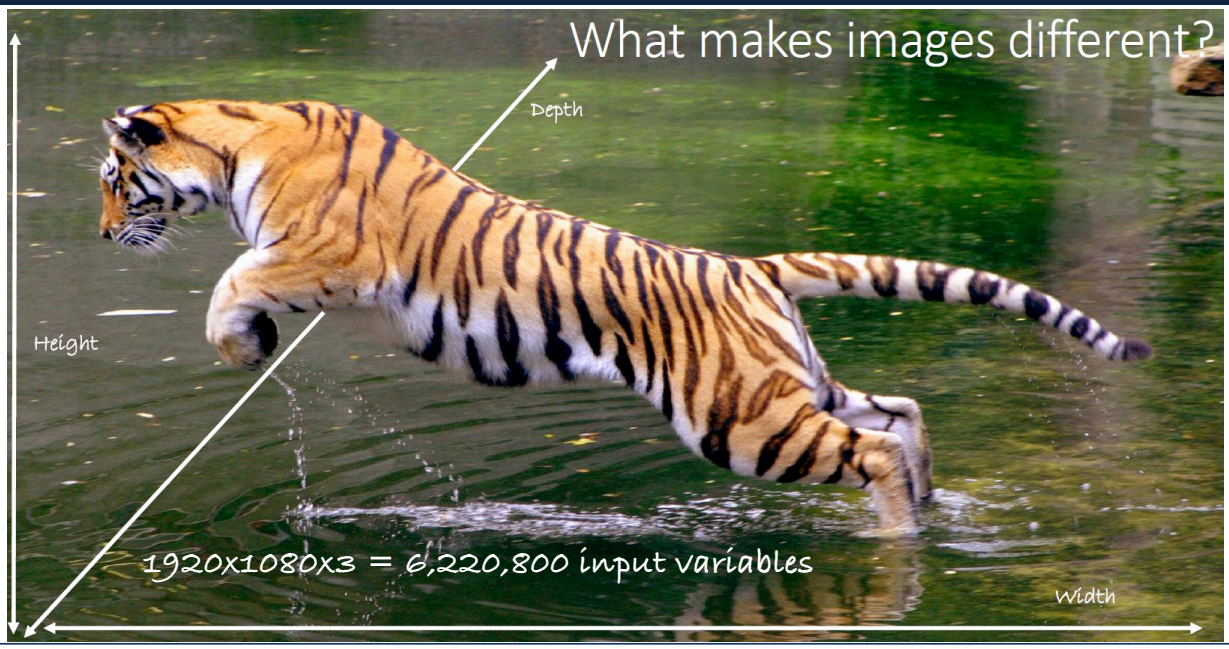
Prof. Saerom Park



What makes images different?



What makes images different?



What makes images different?

What makes images different

- An image has spatial structure
- Huge dimensionality
  - A 256x256 RGB images amounts to ~200K input variables
  - 1-layered NN with 1,000 neurons → 200 million parameters
- Images are stationary signal → they share features
  - After variances images are still meaningful
  - Small visual changes (often invisible to naked eye) → big changes to input vector
  - Still, semantics remain
  - Basic natural image statistics are the same

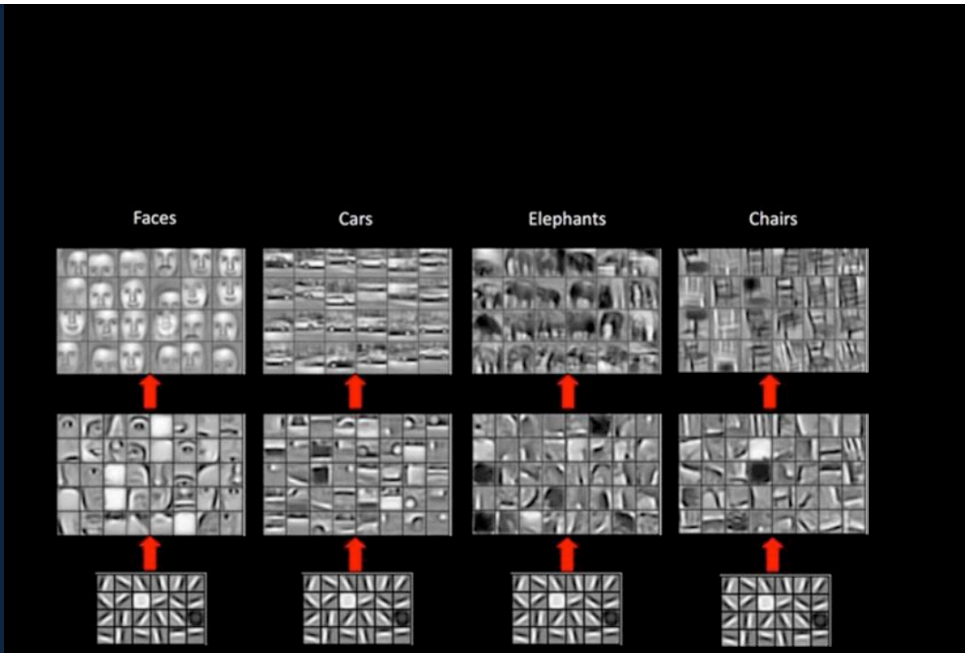
Problem!

- Clearly, too many parameters
- With a only 30x30 pixels image and a single hidden layer of depth 5 we would need 85K parameters
  - With a 256x256 image we would need $46x10^6$ parameters
- Problems 1: Fitting a model with that many parameters is not easy
- Problems 2: Finding the data for such a model is not easy
- Problems 3: Are all these weights necessary?

Convolutional Neural Networks

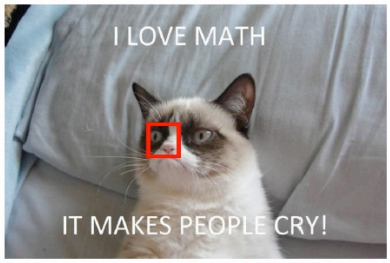
- Question: Spatial structure?
  - Answer: Convolutional filters
- Question: Huge input dimensionalities?
  - Answer: Parameters are shared between filters
- Question: Local variances?
  - Answer: Pooling

Preserving Spatial structure

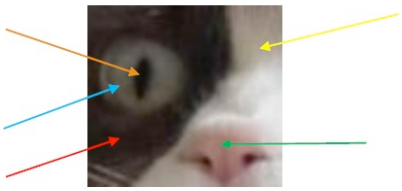


Why spatial?

- Images are 2-D
  - k-D if you also count the extra channels
  - RGB, hyperspectral, etc.

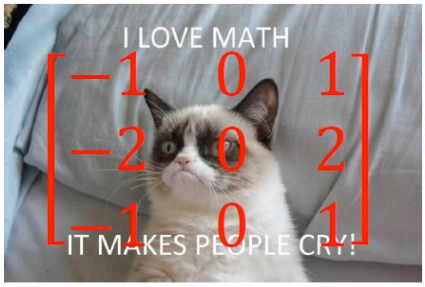


- What does a 2-D input really mean?
  - Neighboring variables are locally correlated

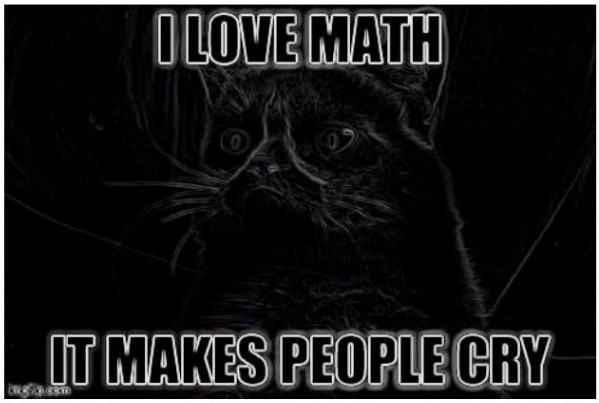


Example filter when k=1

e.g. Sobel 2-D filter



$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



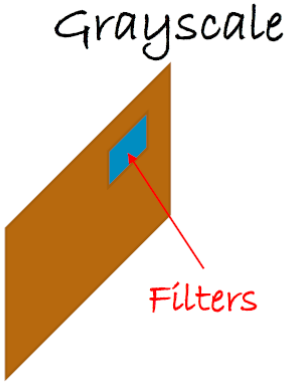
Learnable filters

- Several, handcrafted filters in computer vision
  - Canny, Sobel, Gaussian blur, smoothing, low-level segmentation, morphological filters, Gabor filters
- Are they optimal for recognition?
- Can we learn them from our data?
- Are they going resemble the handcrafted filters?

$$\begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ \theta_7 & \theta_8 & \theta_9 \end{bmatrix}$$

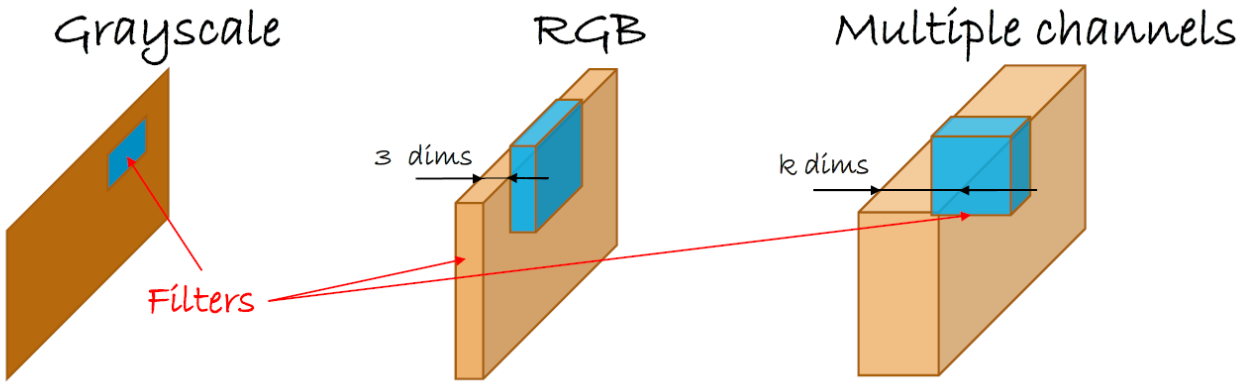
2-D filters (parameters)

- If images are 2-D, parameters should also be organized in 2-D
  - That way they can learn the local correlations between input variables
  - That way they can "exploit" the spatial nature of images

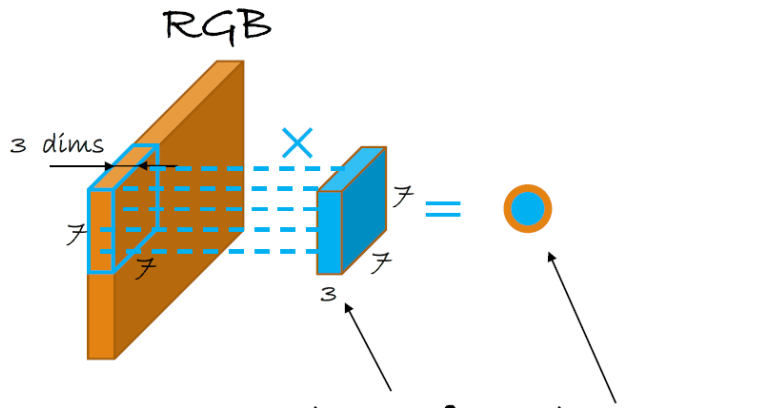


k-D filters (parameters)

- Similarly, if images are k-D, parameters should also be k-D



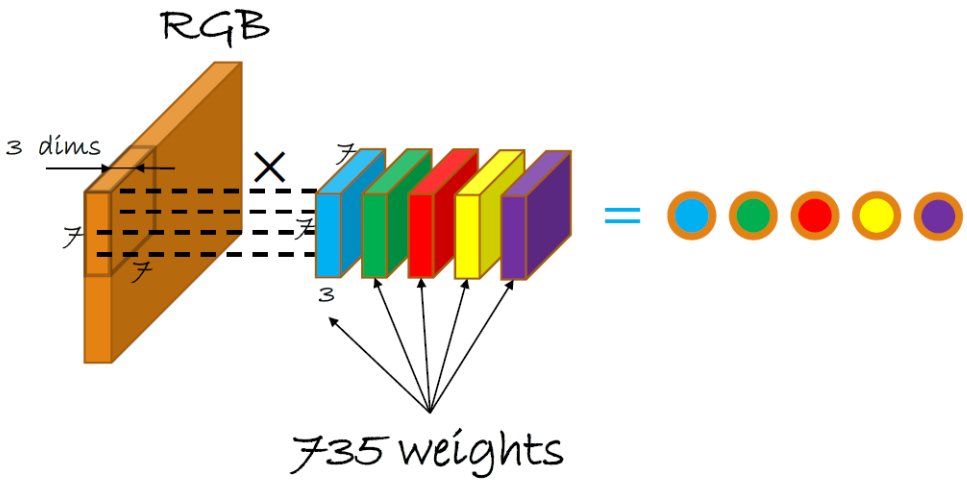
Think in space



How many weights for this neuron?

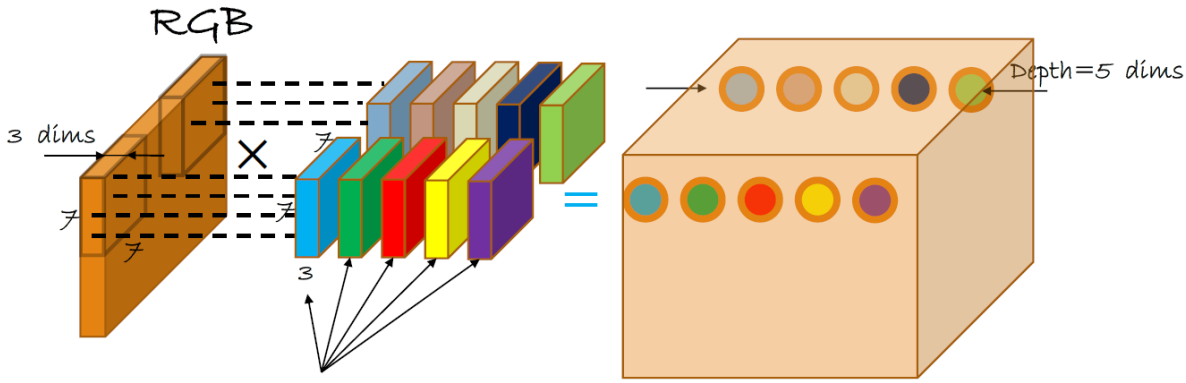
$$7 \cdot 7 \cdot 3 = 147$$

Again, think in space



Think in space

The activations of a hidden layer form a volume of neurons, not a 1-d "chain"
This volume has a depth 5, as we have 5 filters

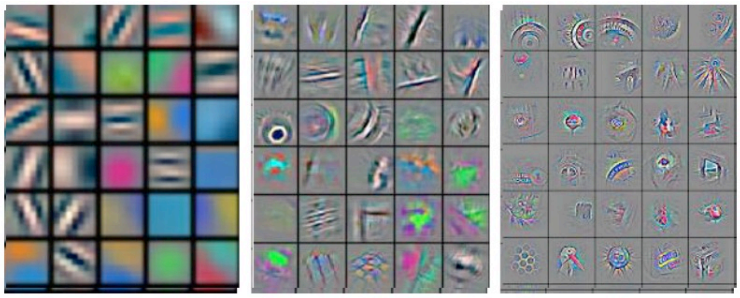


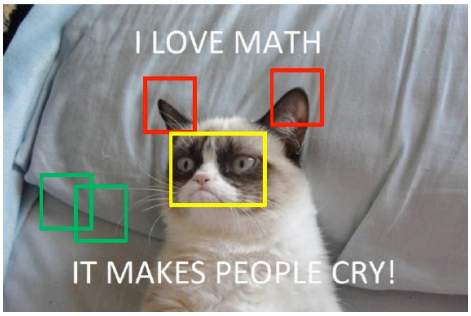
How many weights for these 5 neurons?

$5 \cdot 7 \cdot 7 \cdot 3 = 735$

Hypothesis

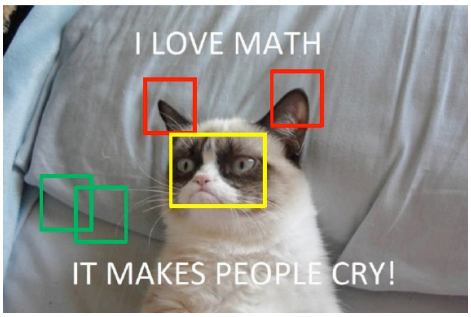
- Imagine
  - With the right amount of data …
  - … and if we connect all inputs of layer l with all outputs of layer $l+1$, …
  - … and if we would visualize the (2d) filters (local connectivity → 2d) …
  - … we would see very similar filters no matter their location





Hypothesis

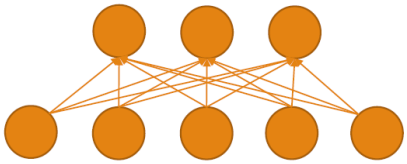
- Imagine
  - With the right amount of data …
  - … and if we connect all inputs of layer l with all outputs of layer $l+1$, …
  - … and if we would visualize the (2d) filters (local connectivity → 2d) …
  - … we would see very similar filters no matter their location

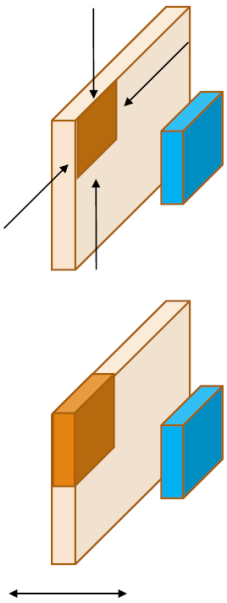


- Why?
  - Natural images are stationary
  - Visual features are common for different parts of one or multiple image

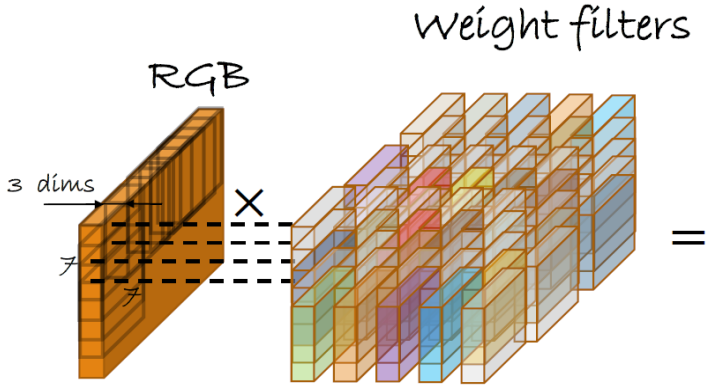
Local connectivity

- Share weights spatially (after translation)
- The weight connections are surface-wise local!
- The weights connections are depth-wise global
- For standard neurons no local connectivity
  - Everything is connected to everything

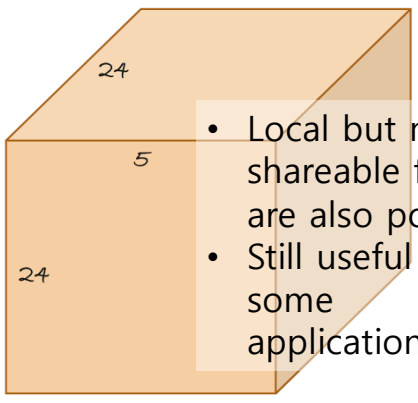




Local connectivity ≠ Convolutional filters



=



- Local but non-shareable filters are also possible
- Still useful for some applications

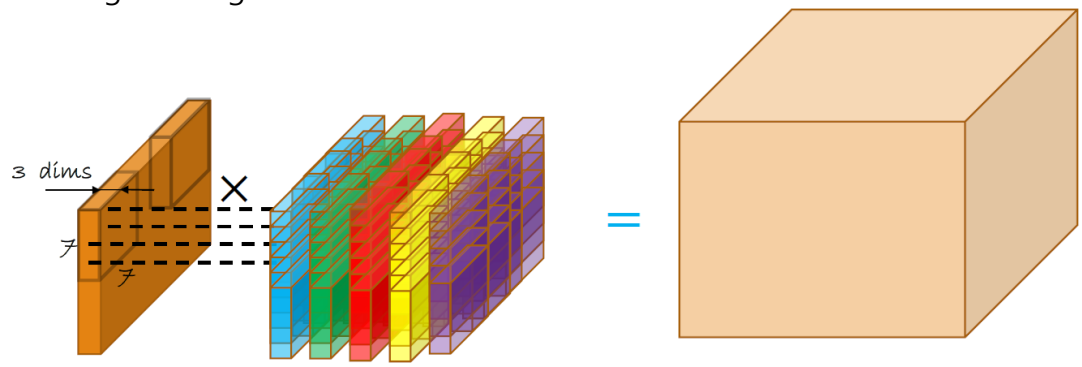
Assume the image is 30x30x3.
1 filter every pixel (stride =1)
How many parameters in total?

24 filters along the x axis
24 filters along the y axis
Depth of 5
× $7 * 7 * 3$ parameters per filter

$423K$ parameters in total

Solution? Share!

- So, if we are anyways going to compute the same filters, why not share?
  - Sharing is caring



Assume the image is 30x30x3.
1 column of filters common across the image.
How many parameters in total?

Depth of 5
× $7 * 7 * 3$ parameters per filter
735 parameters in total

Shared 2-D filters → Convolutions

Original image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image

1 x1	1 x0	1 x1	0	0
0 x0	1 x1	1 x0	1	0
0 x1	0 x0	1 x1	1	1
0	0	1	1	0
0	1	1	0	0

Result

4		

Inner product

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$$

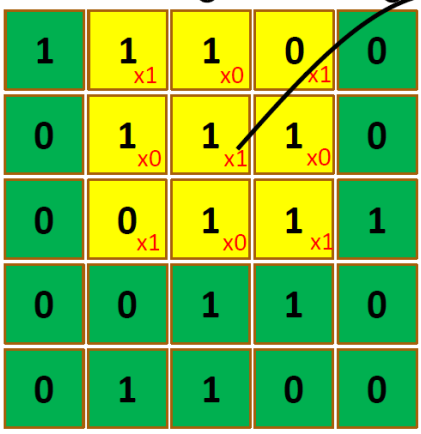
Shared 2-D filters → Convolutions

Original image

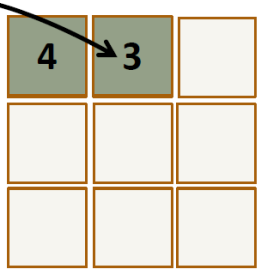
Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image



Result

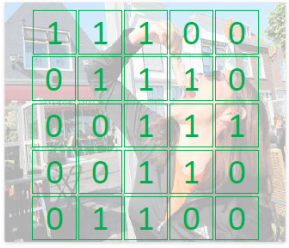


Inner product

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$$

Shared 2-D filters → Convolutions

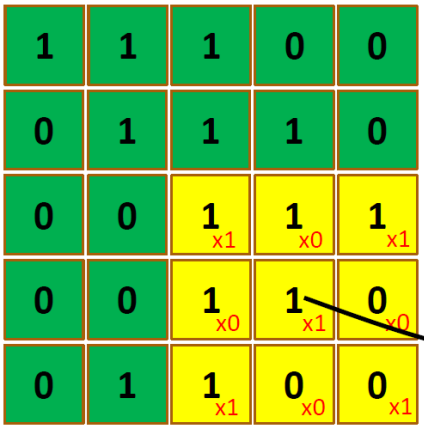
Original image



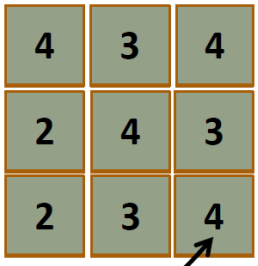
Convolutional filter 1

1	0	1
0	1	0
1	0	1

Convolving the image



Result



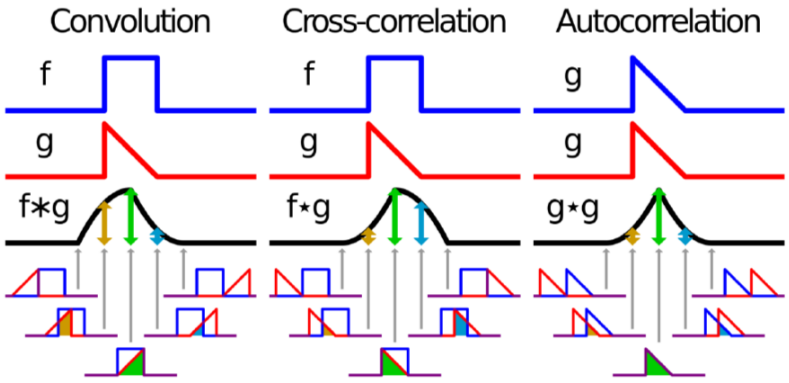
Inner product

$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$$

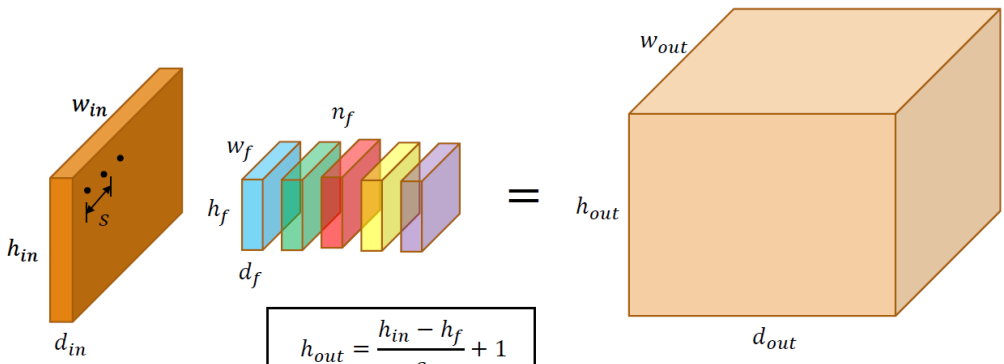
Why call them convolutions?

- Definition: The convolution of two functions f and g is denoted by * as the integral of the product of the two functions after one is reversed and shifted

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau$$



Output dimensions?



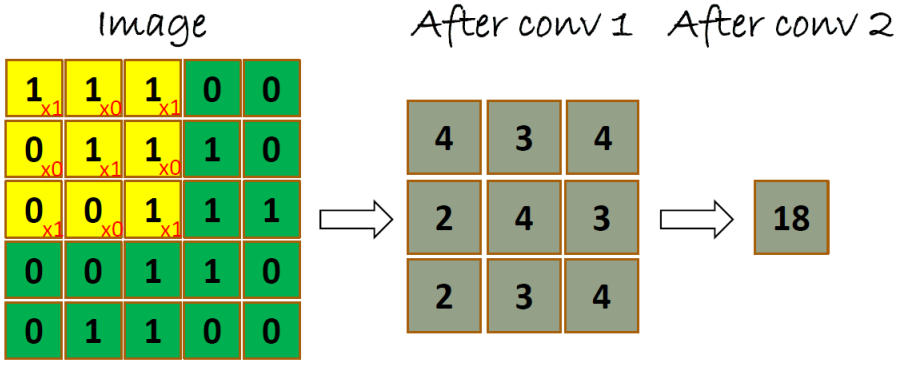
$$h_{out} = \frac{h_{in} - h_f}{s} + 1$$
$$w_{out} = \frac{w_{in} - w_f}{s} + 1$$
$$d_{out} = n_f$$
$$d_f = d_{in}$$

Convolutional Module(New module!!!)

- Activation function
  - $a_{rc} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} x_{r-i,c-j} \theta_{ij}$
- Essentially a dot product, similar to linear layer
  - $a_{rc} \sim x_{region}^{T} \theta$
- Gradient wrt the parameters
  - $\frac{\partial a_{rc}}{\partial \theta_{ij}} = x_{r-i,c-j}$

Problem, again :S

- Our images get smaller and smaller
- We run out of "latent pixels" → Not too deep architectures
- Details are lost → recognition accuracy drops



Solution? Zero-padding!

- Zero-padding to maintain input dimensionality
- For s=1, surround the image with $(h_f - 1)/2$ and $(w_f - 1)/2$ layers of 0

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

$*$

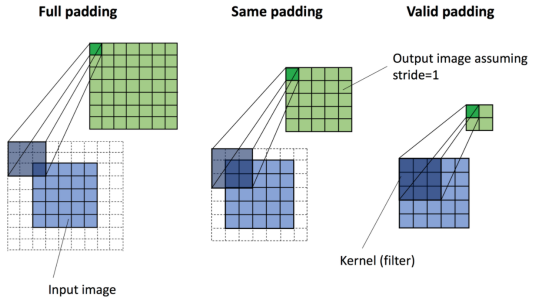
0	0	1
0	1	1
1	1	1

$=$

1	1	2	0	0
0	1	1	1	0
0	0	1	2	1
1	0	2	1	0
0	1	1	3	0

The effect of zero-padding in a convolution

- Performing a discrete convolution in one dimension
  - 앞선 예제와 같이 패딩이 없는 경우, 중간에 있는 점에 대해 집중적으로 계산이 몰리는 경향이 있어서 양 옆에 패딩을 해준다.
  - 패딩의 종류는 3가지가 있다 (feature map 크기:m)
    - Full mode : 패딩의 개수 = m-1
    - Same padding : 아웃풋의 크기가 x와 같게 해주는 padding (제일 많이 이용한다)
    - Valid padding : no padding



Good practice

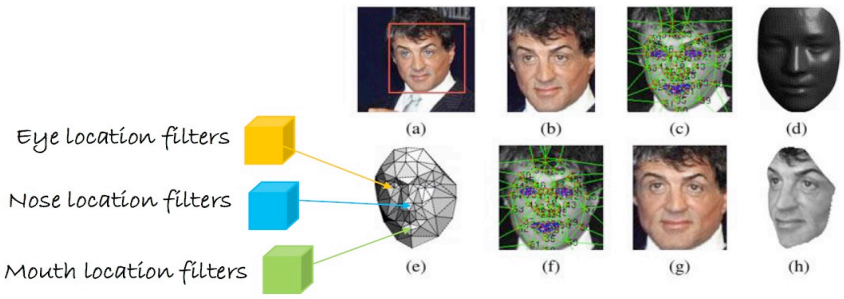
- Resize the image to have a size in the power of 2
- Use stride s=1
- A filter of (h_f, w_f) = [3x3] works quite alright with deep architectures
- Convolution Module for NNs
  - 앞에서, 함수의 convolution 연산과 동일하게 CNN의 convolution module 연산을 정의하였으나, 실제 학습에 있어서는 이러한 복잡성이 불필요함
  - 따라서, 함수의 convolution과는 별개로 간단하게 conlvolution 연산을 수행함

No dropout in convolutional layers

- Convolutional filters are much sparser weight arrays
  - No room for co-dependencies and overfitting
  - Dropping out 'single pixels' → too little influence
  - Likely dropout will not contribute
- Also, convolutions by definition capture local correlations
  - If use dropout within convolutions then local correlations get disturbed
  - Convolutions then become ineffective

P.S. Sometimes convolutional filters are not preferred

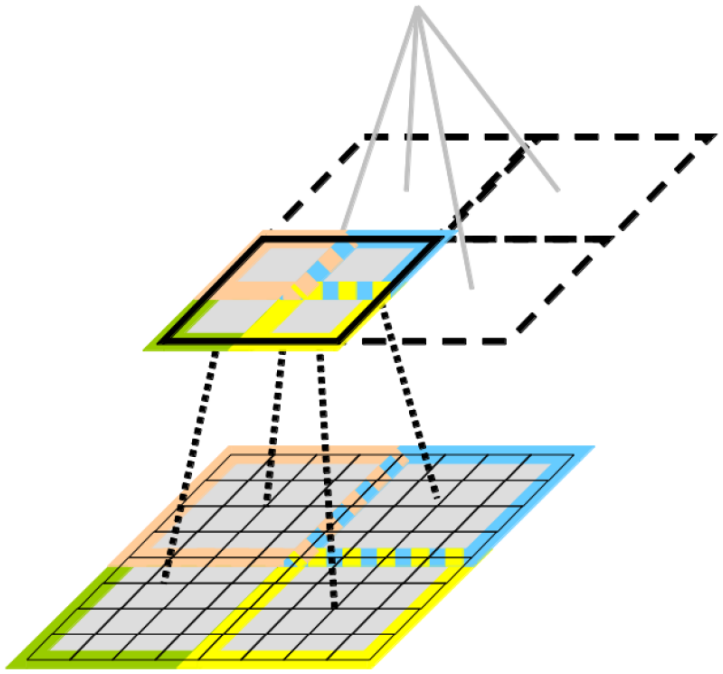
- When images are registered and each pixel has a particular significance
  - E.g. after face alignment specific pixels hold specific types of inputs, like eyes, nose, etc.
  - 코나 눈 이런 것들은 특정 위치에 있는게 맞음 → Spatial invariant 하지가 않음
- In these cases maybe better every spatial filter to have different parameters
  - Network learns particular weights for particular image locations [Taigman 2014]



Pooling

Prof. Saerom Park

INTRODUCTION TO DEEP LEARNING

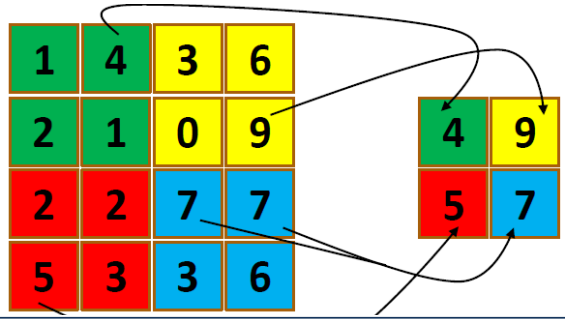


Pooling

- Aggregate multiple values into a single value
  - Invariance to small transformations
  - Reduces the size of the layer output/input to next layer → Faster computations
  - Keeps most important information for the next layer
- Max pooling
- Average pooling

Max pooling (New module!)

- Run a sliding window of size $[h_f, w_f]$
- At each location keep the maximum value
- Activation function: $i_{\max}, j_{\max} = argmax_{i,j \in \Omega(r,c)} x_{ij}$
- Gradient wrt input $\frac{\partial a_{rc}}{\partial x_{ij}} = \begin{cases} 1, if\ i = i_{max}, j = j_{max} \\ 0, \quad otherwise \end{cases}$
- The preferred choice of pooling



Average pooling (New module!)

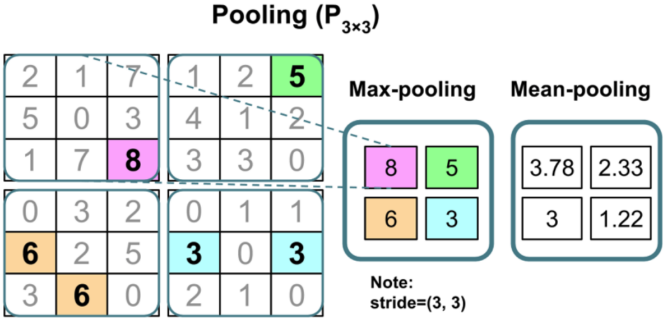
- Run a sliding window of size $[h_f, w_f]$
- At each location keep the maximum value
- Activation function: $a_{rc} = \frac{1}{rc}\sum_{i,j\in\Omega(r,c)} x_{ij}$
- Gradient wrt input $\frac{\partial a_{rc}}{\partial x_{ij}} = \frac{1}{rc}$

1	4	1	6
2	3	0	9
1	2	7	1
4	1	0	2

5	8
4	5

Subsampling

- Pooling
  - 국소적인 변화에 대해 덜 민감한 모델을 구성 할 수 있다.
  - Feature의 크기를 줄여주기 때문에 computational efficiency가 증가된다.
  - Max pooling – 강한 자극만 기억
  - Mean Pooling – 평균적인 자극만 기억

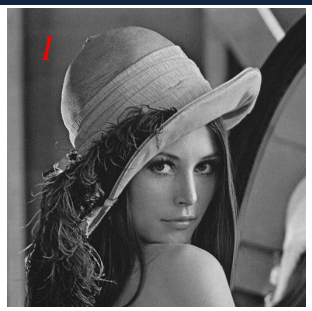


Understanding convnets

Dr. Saerom Park

Convolution activations are “images”

- $h_I(x,y) = I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot h(i,j)$
- For every image pixel we compute a convolved image pixel
- After each convolution we end up with a new image
- 하지만 filter를 거친후의 output (activation)을 보는 것과 filter 자체가 어떠한 성질을 띄고 있는 지를 보는 것은 구분되어야 한다.



Several interesting questions

- What do the image **activations** in different layers look like?
- What types of images create the strongest activations?
- What are the activations for the class "ostrich"?
- Do the activations occur in meaningful positions?

Feature maps

- The convolution activations are also called feature maps
- A deep network has several hierarchical layers
  - hence several feature maps going from less to more abstract

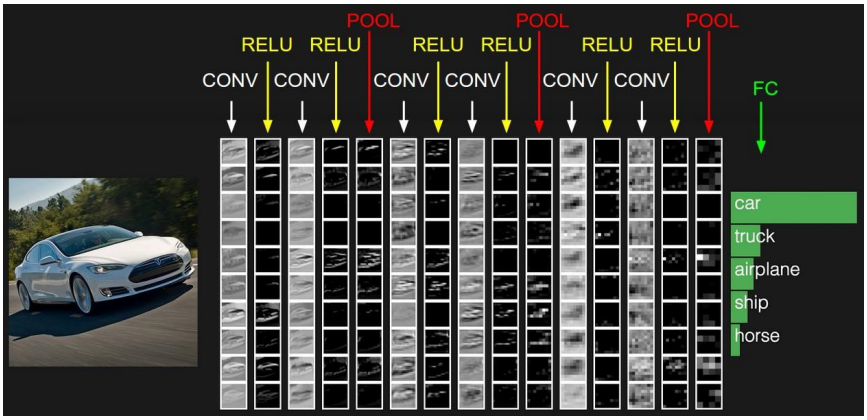


Image borrowed by A. Karpathy

Feature maps strength

- Naturally, given an image some feature maps will be "stronger"
  - **Contribute higher amount to the final classification score**
- Why? What pixel structure excited these particular feature maps?
  - **Generally, what part of the picture excites a certain feature map?**

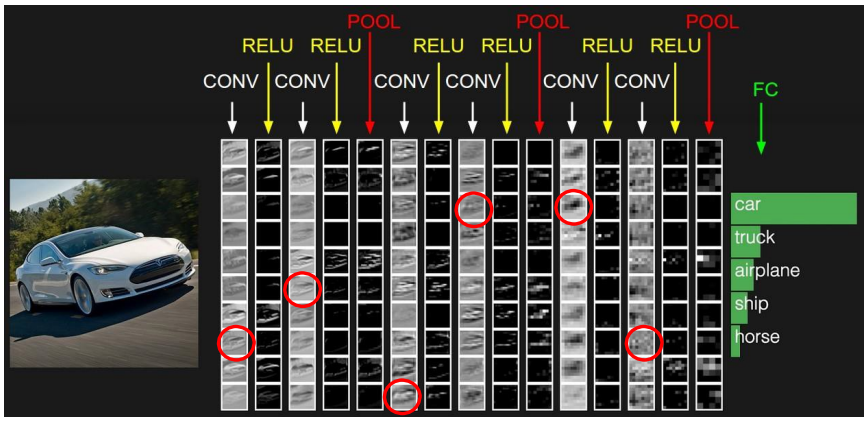


Image borrowed by A. Karpathy

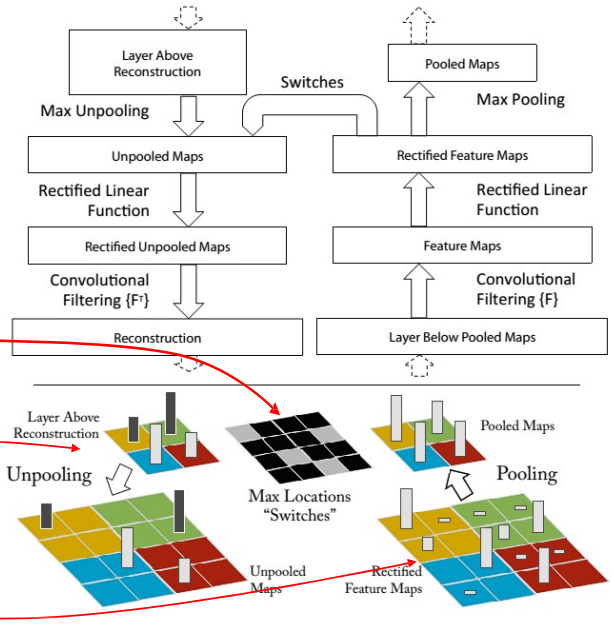
Visualize Filters

- Filter들을 시각화 하는 것은 특정 input을 넣었을 때에 특정 filter 에 의해 어떠한 output이 나오는 지와 다름
- Filter 시각화 방법
  - **Filter에 가장 많이 영향을 주는 input을 찾음**
  - **Deconv 를 이용하여 특정 filter의 activation으로부터 image 찾음**

Visualization with a Deconvnet [Zeiler2014]

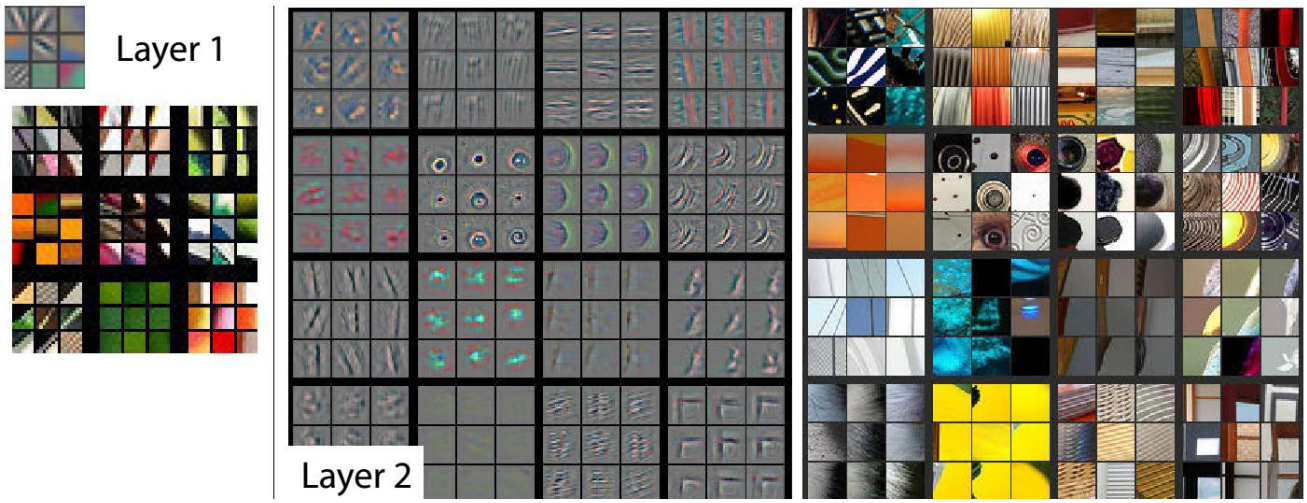
- Same as convent, but in reverse
  - Instead of mapping pixels to activations, mapping activations to pixels
- Attach a deconvnet layer on every convnet layer
- Unpooling via switches that remember where max pooling was activated
- Deconv filtering equals to "reverse conv filtering"

$$\mathrm{F} = \begin{bmatrix} 1 & 5 \\ 6 & 3 \end{bmatrix} \quad F^T = \begin{bmatrix} 3 & 6 \\ 5 & 1 \end{bmatrix}$$



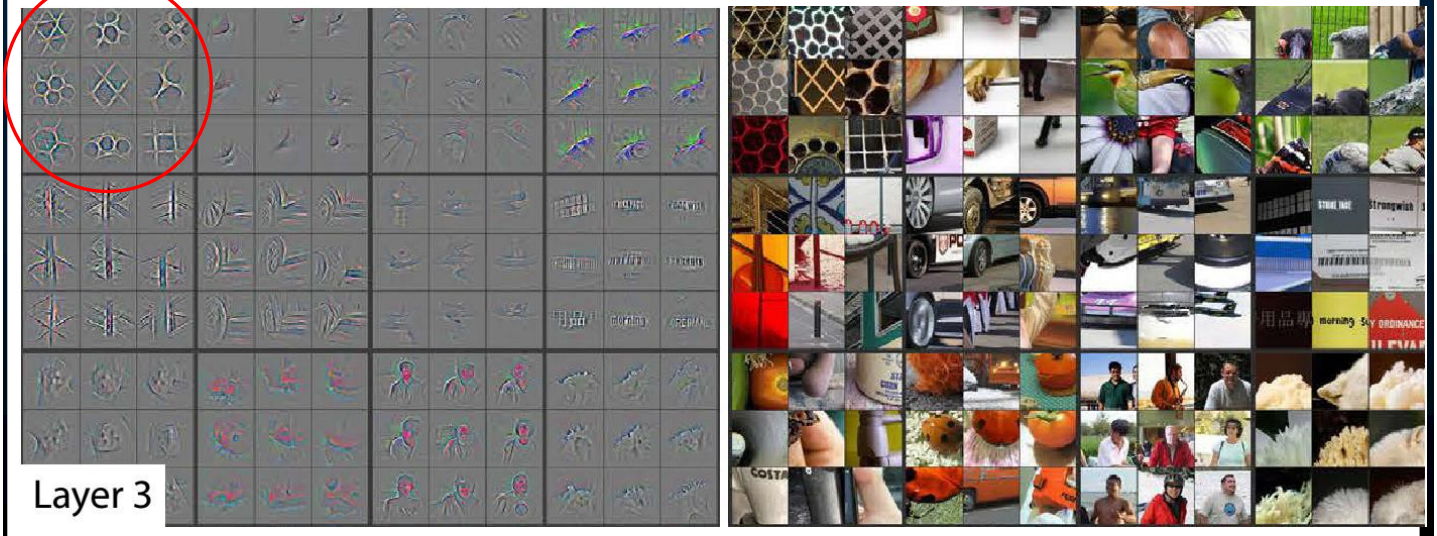
What excites feature maps?

- "Given a random feature map what are the top 9 activations across validation data?"



What excites feature maps?

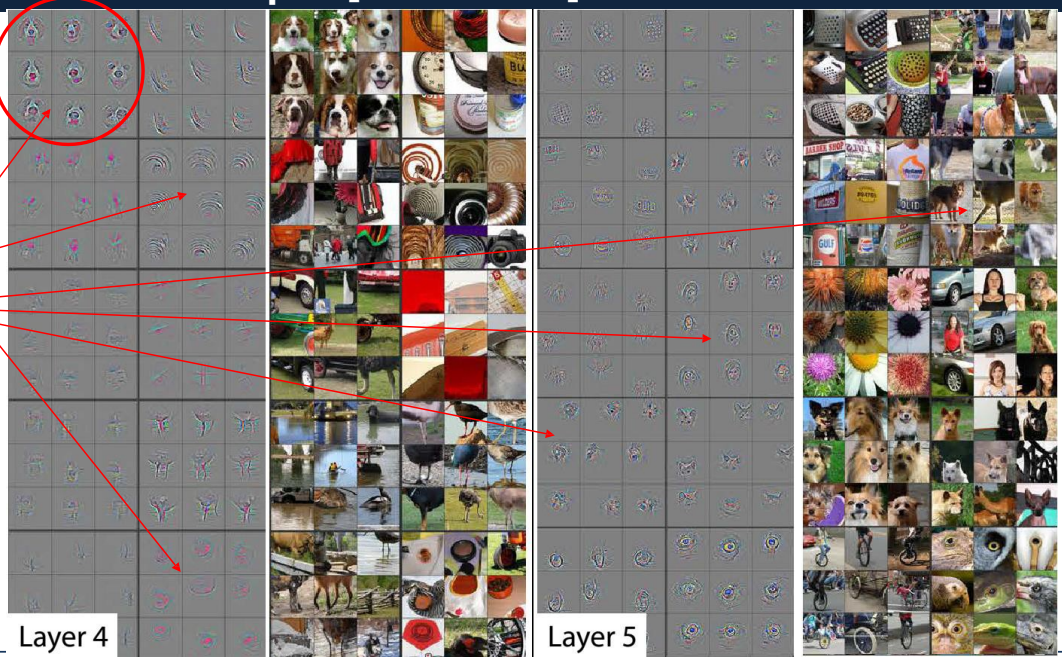
Similar activations from lower level visual patterns



What excites feature maps? [Zeiler2014]

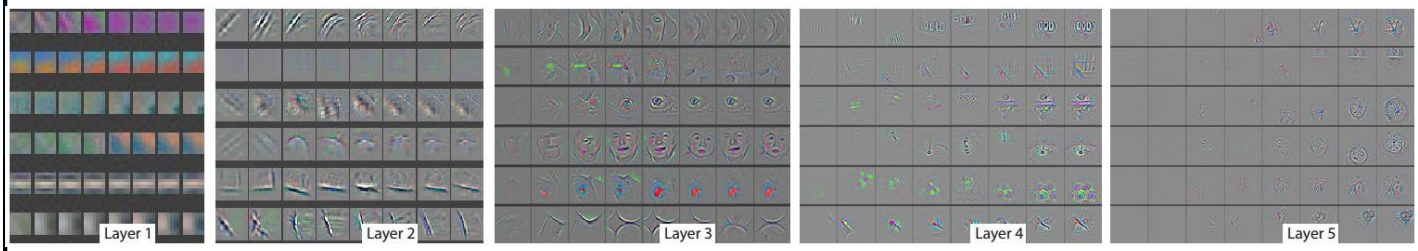
Similar activations from semantically similar pictures

Visual patterns become more and more intricate and specific (greater invariance)



Feature evolution over training

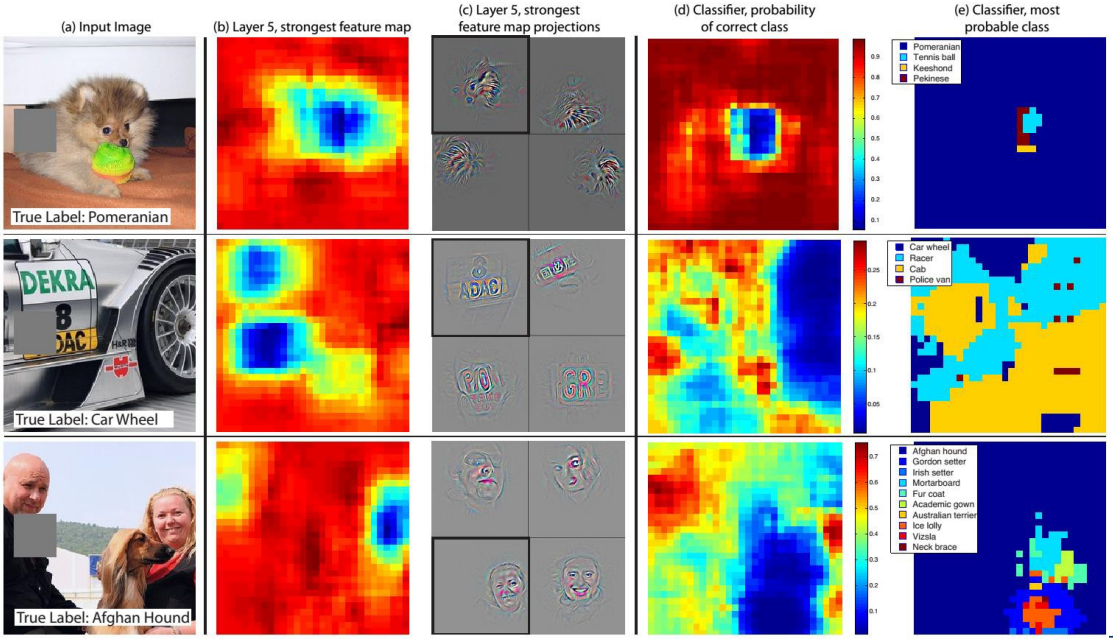
- For a particular neuron (that generates a feature map)
- Pick the strongest activation (across all training examples) within a given feature map during training
- For epochs 1, 2, 5, 10, 20, 30, 40, 64
- The lower layes of the model can be seen to converge within a few epochs, but the upper layers only develop after a considerable number of epochs



Class activation map visualization

- 특정 class 로 분류하게 되는데 영향을 끼치는 정도를 시각화
  - Heatmap 이용
- 시각화 방법
  - 최종 output에서 예측하게 되는 class의 예측 값에 대한 activation의 gradient를 구함
  - Activation의 gradient 값이 클수록 더 큰 영향을 끼치는 것으로 생각
  - 이를 곱한 후에 heatmap으로 나타냄

But does a Convnet really learn the object?



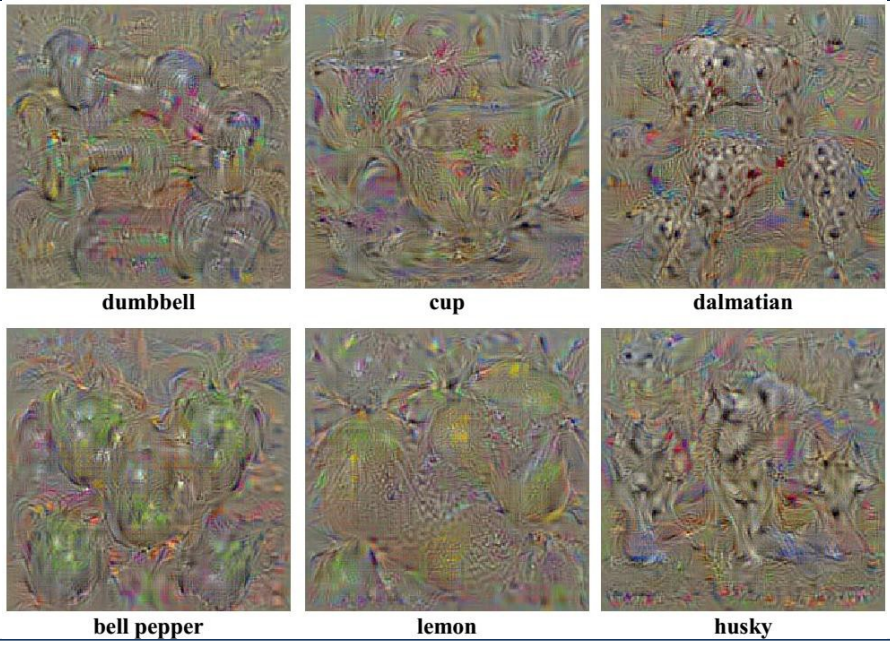
What is a "Convnet dog", however? [Simonyan2014]

- What is the image with the highest "dogness score"

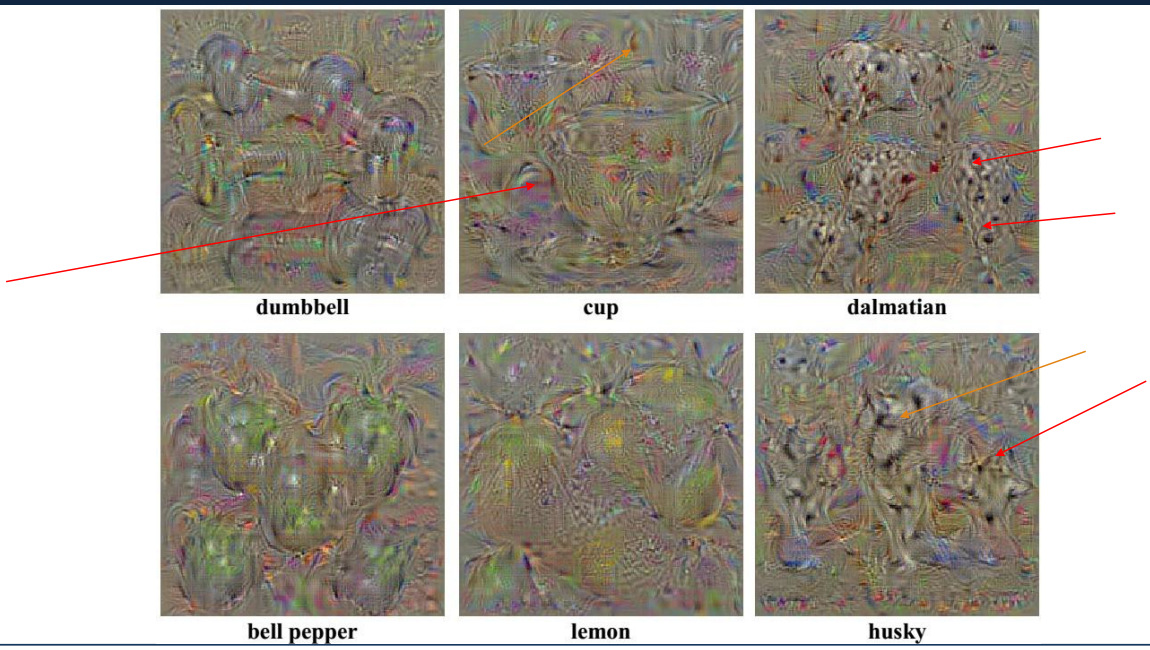
$$\text{argmax}_I \, S_c(I;\theta) - \lambda |I|^2$$

- The parameters θ are fixed during the training
- Optimization is done with respect to the image I
- Initialized with the "average training image"

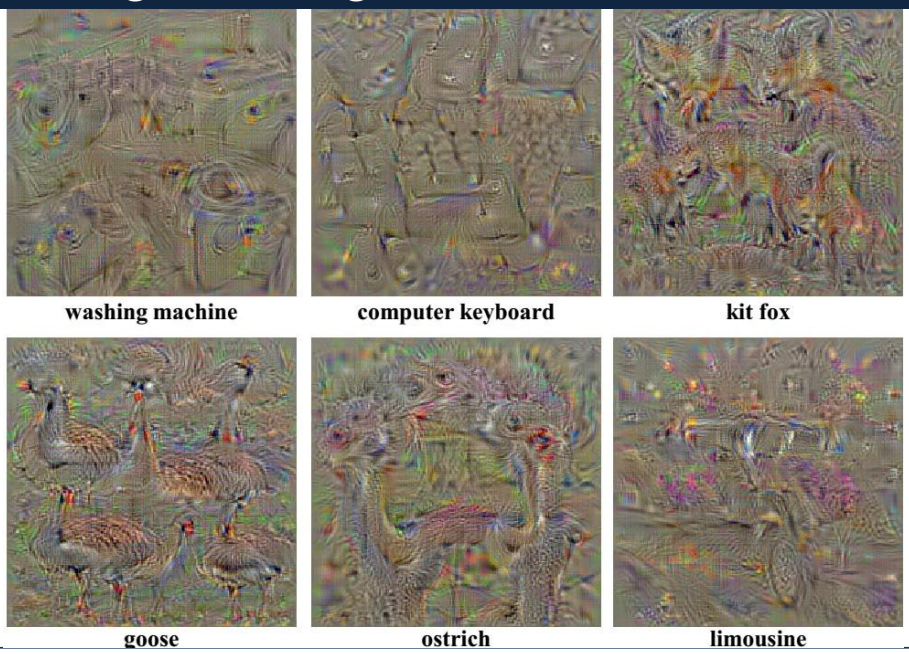
Maximum-scoring class image



Maximum-scoring class image



Maximum-scoring class image



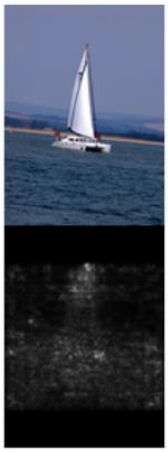
Class-specific image saliency

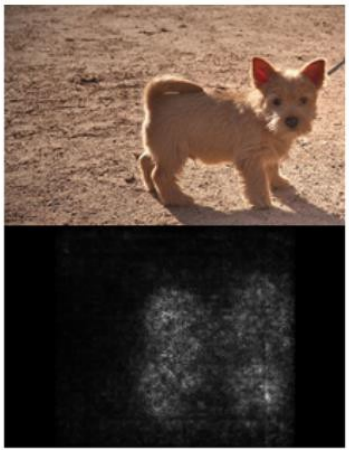
- Given the "monkey" class, what are the most "monkey-ish" parts in my image?
- Approximate S_c around an initial point I_0 with the first order Taylor expansion

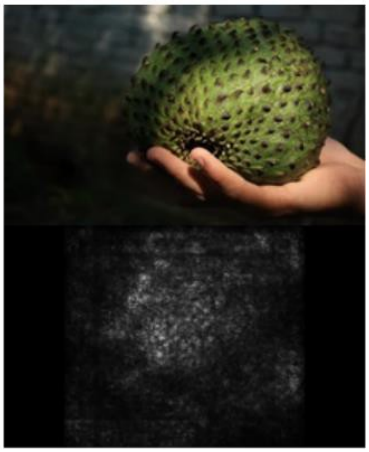
$$S_c(I)|_{I_0} \approx w^T I + b, where\ w = \frac{\partial s_c}{\partial I}|_{I_0} from\ backpropagation$$

- Solution is locally optimal

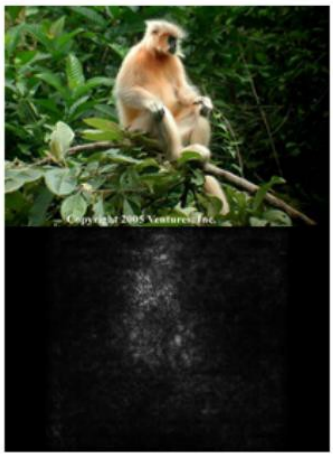
Examples



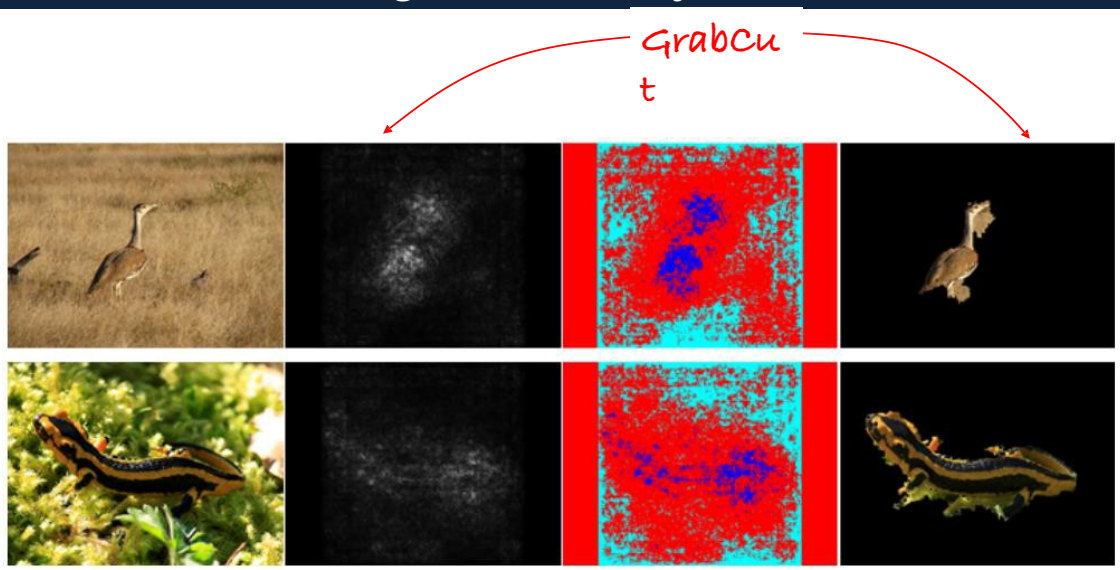




Examples



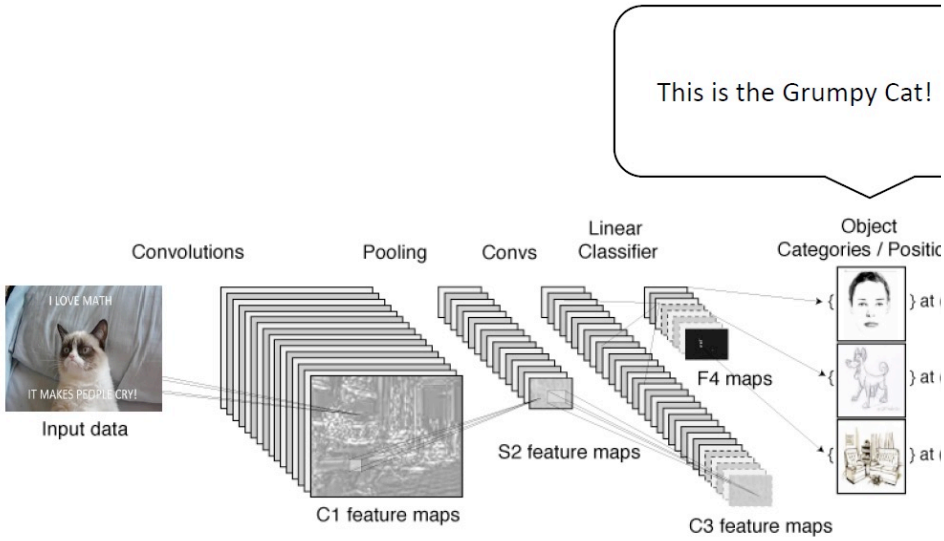
Object localization using class saliency



Convnets for Object Recognition

Prof. Saerom Park

INTRODUCTION TO DEEP LEARNING



Performance of CNN

- ImageNet Large Scale Visual Recognition Challenge
  - Ran from 2010 to 2017
    - 현재는 Kaggle competition
  - Main task: image classification
    - Automatically label 1.4M images with 1K objects
    - 2015년 부터는 인간보다 이미지 분류과제를 더 빠르고 정확하게 수행

Constructing ImageNet

Step 1:
Collect candidate images via the Internet

Step 2:
Clean up the candidate Images by humans

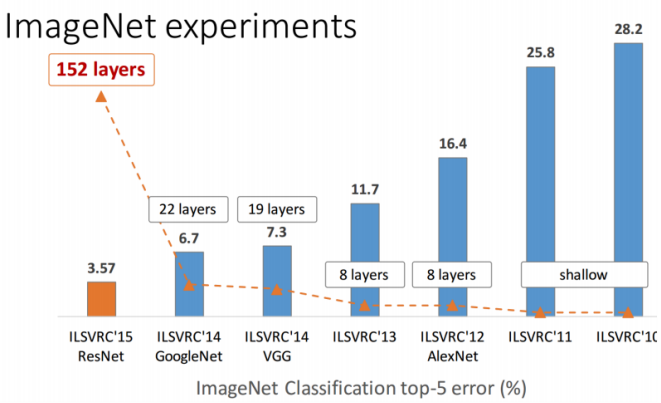
amazon mechanical turk
Artificial Artificial Intelligence

Some statistics

- July 2008: 0 images
- Dec 2008: 3 million images, 6K+synsets
- April 2010: 11 million images, 15K + synsets
- Currently: 14 million images, 21K synsets indexed

Performance of CNN

- ImageNet



CNN based, non-CNN based

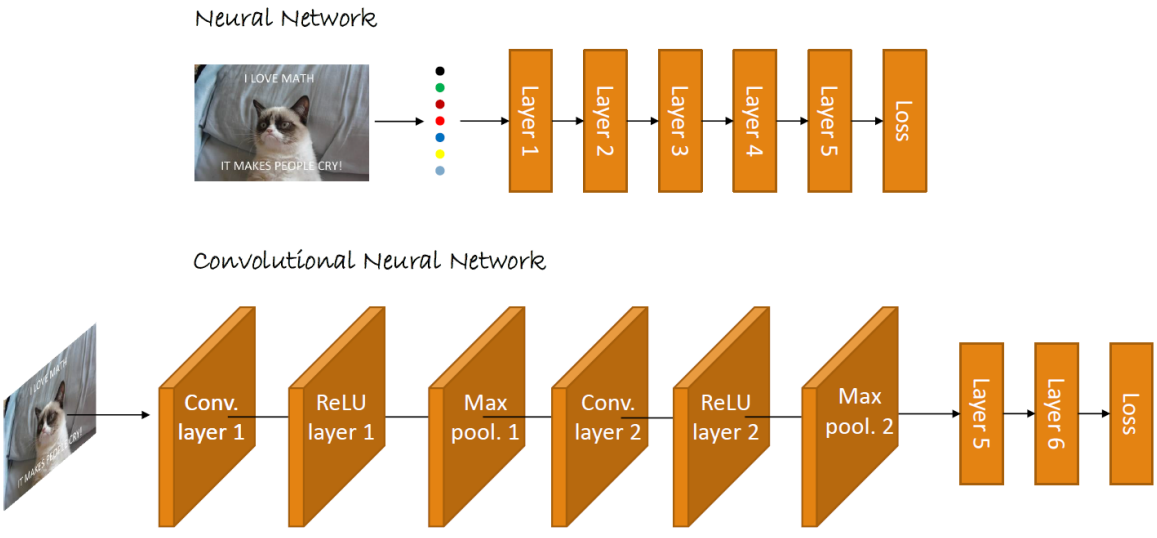
2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

Figures from Y. LeCun's CVPR 2015 plenary talk

Standard Neural Network vs Convnets



Convnets in practice

- Several convolutional layers
  - 5 or more
- After the convolutional layers non-linearities are added
  - The most popular one is the ReLU
- After the ReLU usually some pooling
  - Most often max pooling
- After 5 rounds of cascading, vectorize last convolutional layer and connect it to a fully connected layer
- Then proceed as in a usual neural network

CNN Case Study 1: Alexnet

Prof. Saerom Park

INTRODUCTION TO DEEP LEARNING

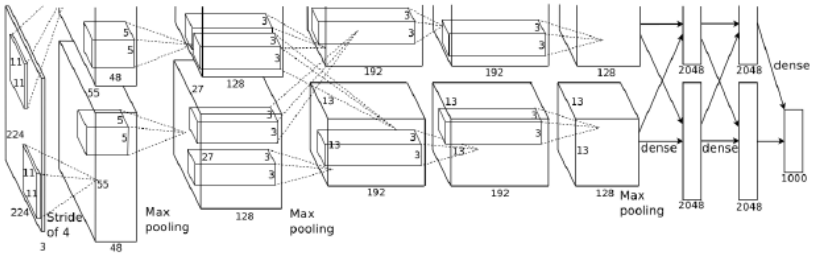
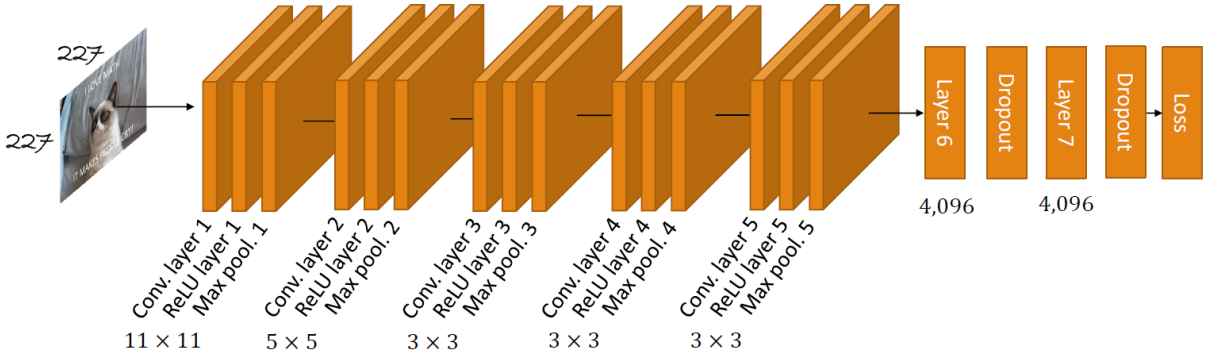


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

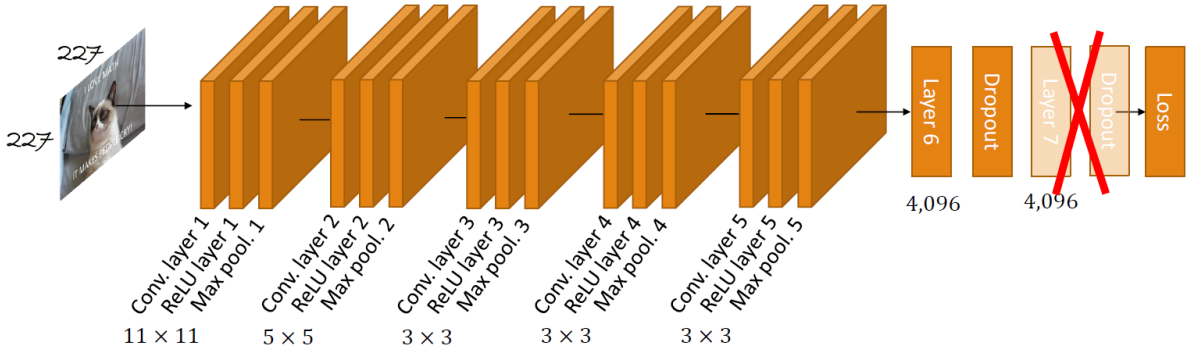
Architetural detials

18.2% error in Imagenet



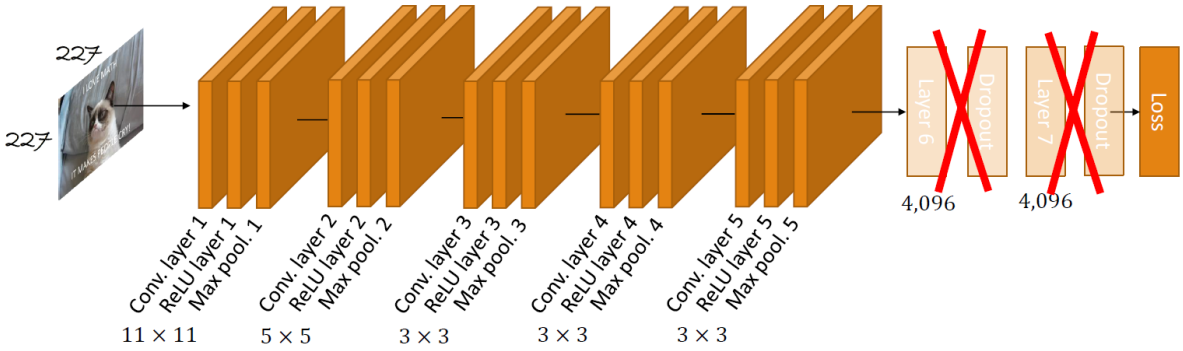
Removing layer 7

1.1% drop in performance, 16 million less parameters



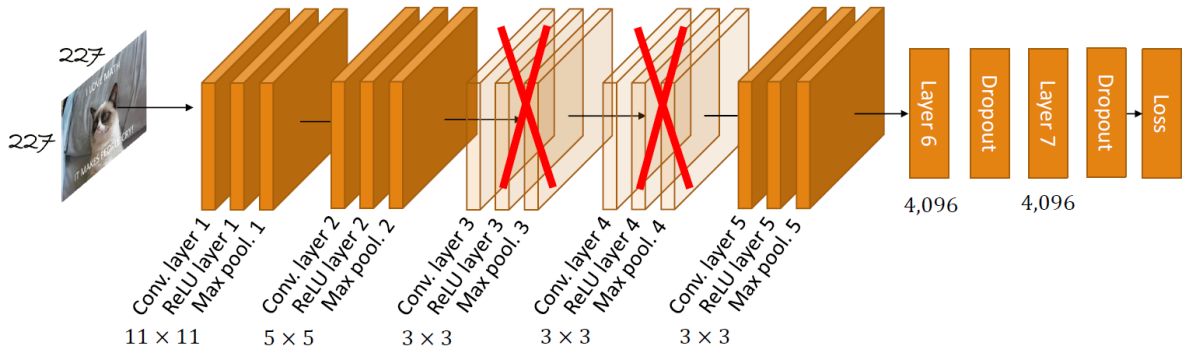
Removing layer 6,7

5.7% drop in performance, 50 million less parameters



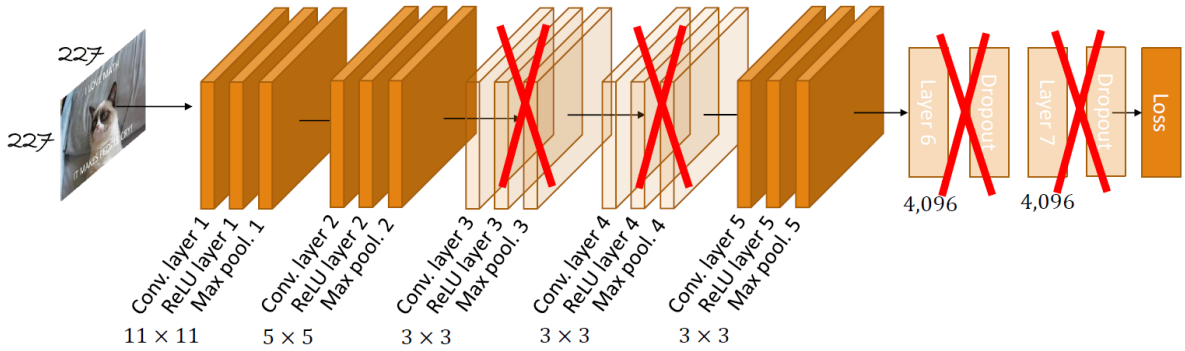
Removing layer 3,4

3.0% drop in performance, 1 million less parameters. Why?

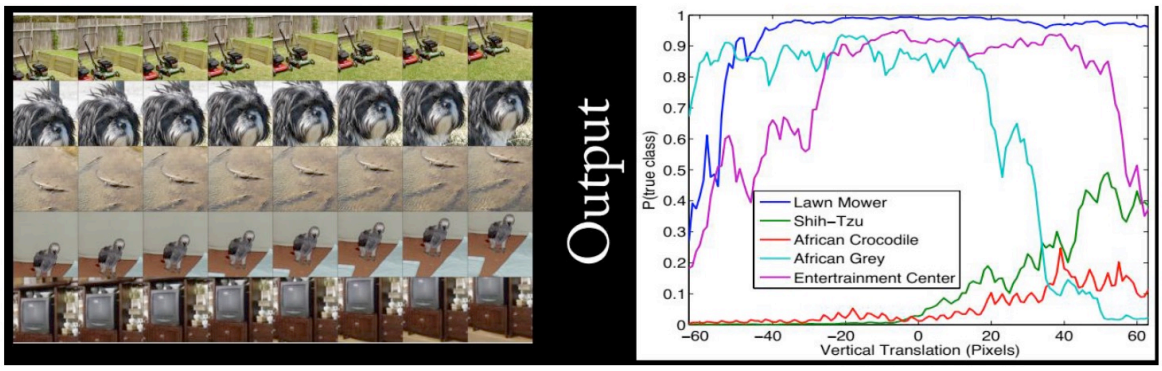


Removing layer 3,4,6,7

33.5% drop in performance. Conclusion? Depth!

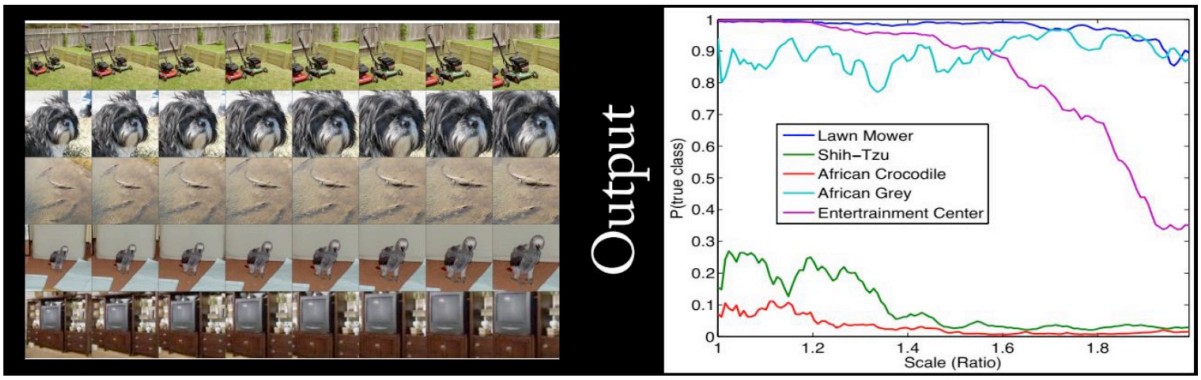


Translation invariance



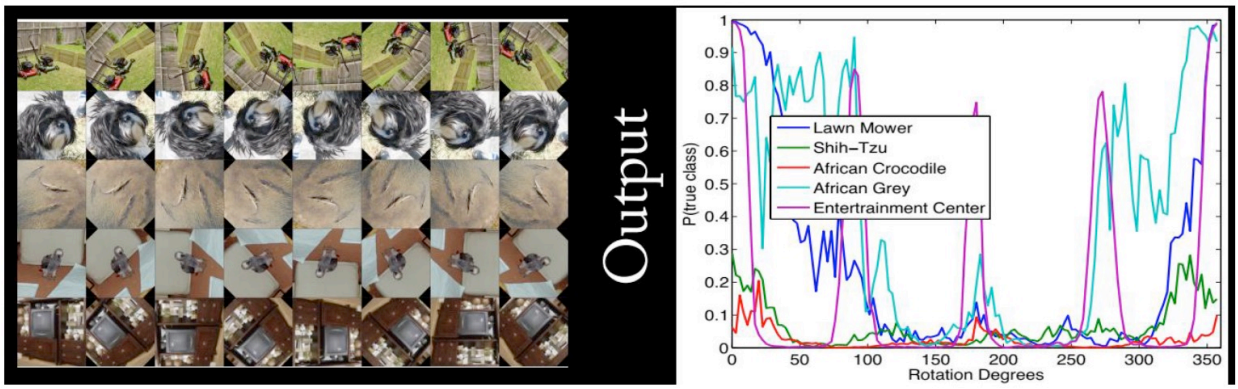
Credit: R. Fergus slides in Deep Learning Summer School 2016

Scale invariance



Credit: R. Fergus slides in Deep Learning Summer School 2016

Rotation invariance



Credit: R. Fergus slides in Deep Learning Summer School 2016

CNN Case Study 2: VGGNet

Prof. Saerom Park

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

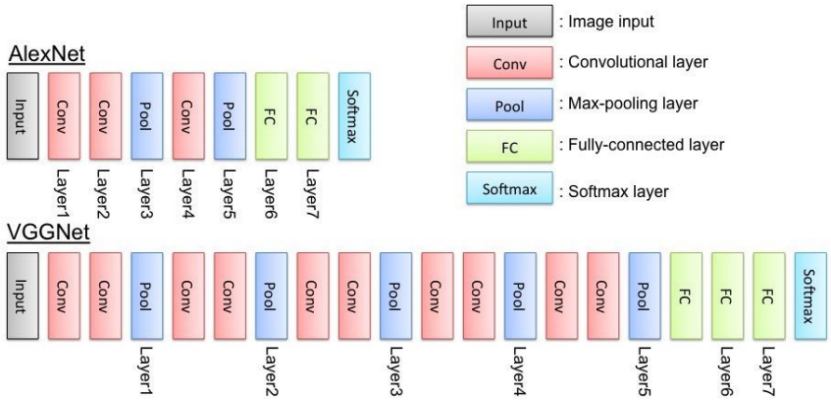
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Differences from Alexnet

- Much more accurate
  - 6.8% vs 18.2% top-5 error
- About twice as many layers
  - 16 vs 7 layers
- Filters are much smaller
  - 3x3 vs 7x7 filters
- Harder/slower to train

Performance of CNN

- VGG Network
  - 19층으로 구성됨
  - 단순하게 AlexNet을 깊이 늘린 형태
  - 2014년 준우승

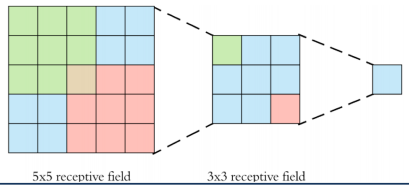


VGGNet Characteristics

- Input size: 224 x 224
- Filter sizes: 3 x 3
- Convolution stride: 1
  - Spatial resolution preserved
- Padding: 1
- Max pooling: 2 x 2 with a stride of 2
- ReLU activations
- No fancy input normalizations
- Although deeper, number of weights is not exploding

Effective receptive field

- The number of pixels contributing at the activation in l-th layer
  - Not just the ones from the previous layers, but the others before that too
- A large filter can be replaced by a deeper stack of successive smaller filters
  - Two 3 x 3 filters have the receptive field of one 5 x 5
  - Three 3 x 3 filters have the receptive field of one 7 x 7
- Depth increases effective receptive field
- Deeper stacks of smaller filters likely more powerful than a single large filter
  - Three more nonlinearities for the same "size" of patter learning
  - Fewer parameters and regularization



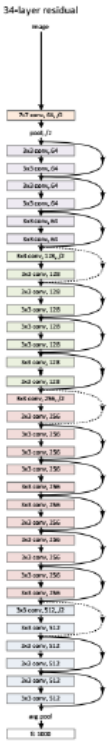
Picture credit: Arden Dertat

Training

- Batch size: 256
- SGD with momentum ($\gamma = 0.9$)
- Weight decay $\lambda = 5 \cdot 10^{-4}$
- Dropout on first two fully connected layers
- Starting learning rate $\eta_0 = 10\text{^} - 2$
  - Divided by 10 when validation accuracy stops improving
  - 3X decreasing learning rate
- Smaller filters → Faster training

CNN Case Study 3: Inception

Prof. Saerom Park

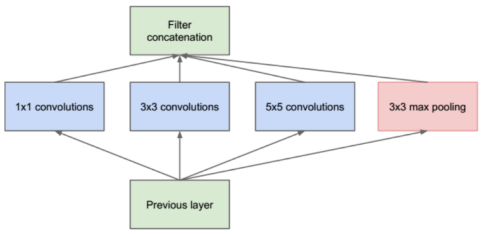


Challenges in object recognition

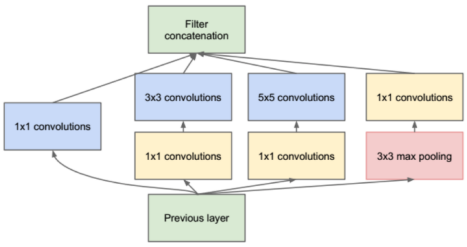
- Salient parts have great variation in sizes
  - Hence, the receptive fields should vary in size accordingly
- Intuitively, deeper models are preferred
  - But very deep nets are prone to overfitting

Inception Module

- Multiple kernel filters of different sizes (1 x 1, 3 x 3, 5 x 5)
  - Naïve version
  - Very expensive!
- Add intermediate 1 x 1 convolutions for compression



(a) Inception module, naïve version

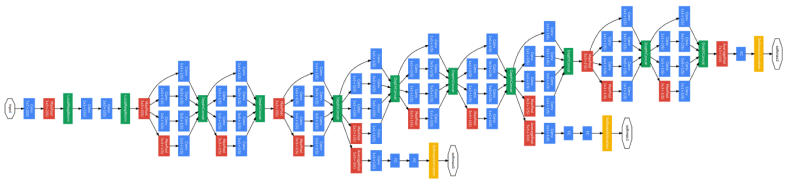


(b) Inception module with dimension reductions

Picture credit: Bharath Raj

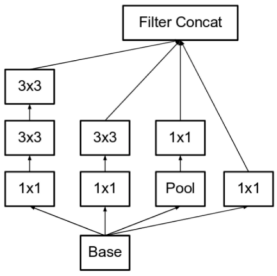
Architecture

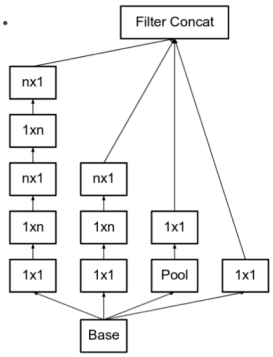
- 9 Inception modules
  - 22 layers deep (27 with the pooling layers)
  - Global average pooling at the end of last Inception Module
- Because of the increased depth → Vanishing gradents
  - Inception solution to vanishing gradients: intermediate classifiers
  - Intermediate classifiers removed after training
- 6.67% Imagenet error (Alexnet: 18.2%)

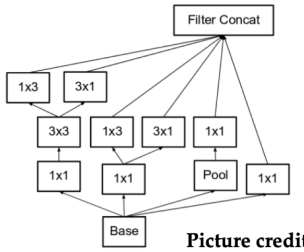


Inceptions v2, v3, v4

- Factorize 5 x 5 in two 3 x 3 filters
- Factorize n x n in two n x 1 and 1 x n filters (quite a lot cheaper)
- Make nets wider
- RMSprop, BatchNorms, …



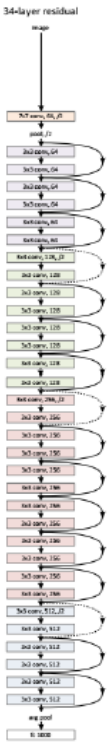




Picture credit: Bharath Raj

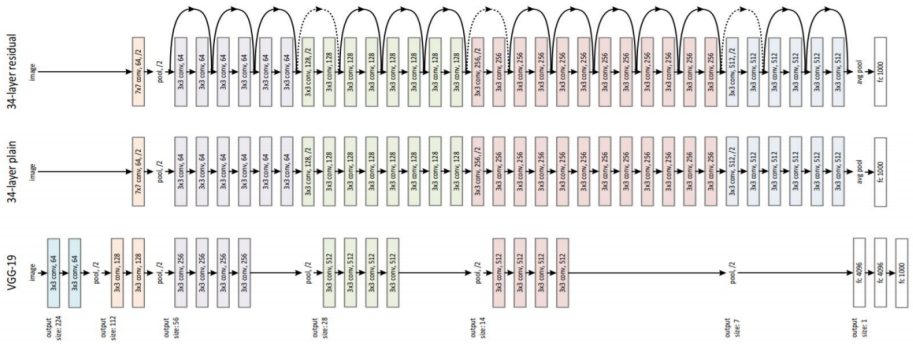
CNN Case Study 4: ResNet [He2015]

Prof. Saerom Park



Performance of ResNet

- ResNet
  - 이전 Layer의 값을 더해주는 방식의 residual connection을 사용함.
  - 152층의 Layer 구조
  - 2015년 1등 (인간보다 뛰어난 성능, ~3% error with ensembles)
  - Won all object classification, detection, segmentation, et . challenges



Problem

- Hypothesis: Can we have a very deep network at least as accurate as averagely deep networks?

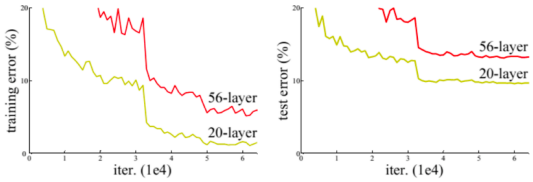


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Testing the hypothesis
  - Training a shallow and a deeper architecture
  - The deeper model does worse in training error!
  - Performance degradation not by overfitting → just harder optimization
  - Assuming optimizers are doing their job fine
    - Not all networks are the same as easy to optimize

How does ResNet work?

- Add to you module output F(x) the input x
  - H(x) = F(x)=x
  - If dimensions don't match zero padding or a projection layer
- Adding identity layers should lead to larger networks that have at least lower training error
- Forward Path
  - $x_L = x_l + \sum_{i=l}^{L-1} F(x_l)$
- Backward Path
  - $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_l)\right)$

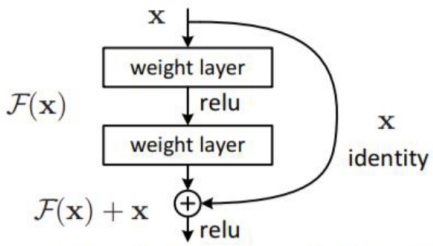


Figure 2. Residual learning: a building block.

- Without residual connections deeper networks attain worse scores
- Ridiculously low error in ImageNet
- Up to 1000 layers ResNets trained
  - Previous deepest network ~30-40 layers on simple datasets

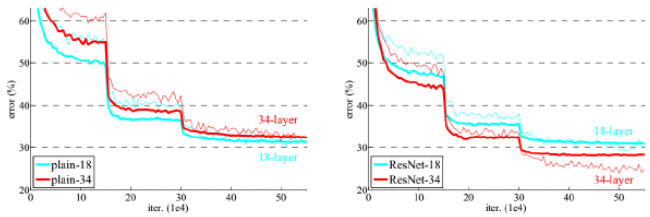
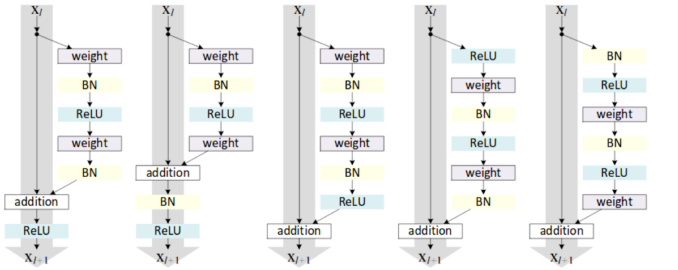


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

ResNet architecture and ResNeXt



case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

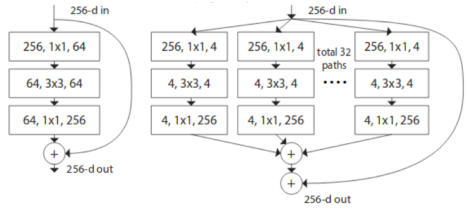


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100d**	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

Aggregated Residual Transformations for Deep Neural Networks, Xie et al., 2016

CNNs and residual connections: insights

- BatchNorms absolutely necessary because of vanishing gradients
- Identity shortcuts cheaper and almost equal to project shortcuts
- Networks with skip connections converge faster
  - Compare to the same network without skip connections
- Generally, skip/residual connections are an asset for deeper architectures

DenseNet

- Add skip connections to multiple forward layers
  - $y = h(x_l, x_{l-1}, \cdots, x_{l-n})$
- Assume layer 1 captures edges, while layer 5 captures faces (and other stuff)
  - Why not have a layer that combines both faces and edges (e.g. to model a scarred face)
- Standard ConvNets do not allow for this
  - Layer 6 combines only layer 5 patterns, not lower

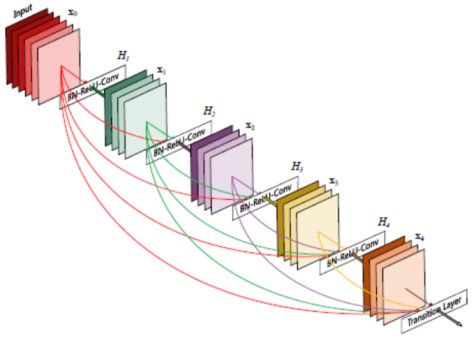
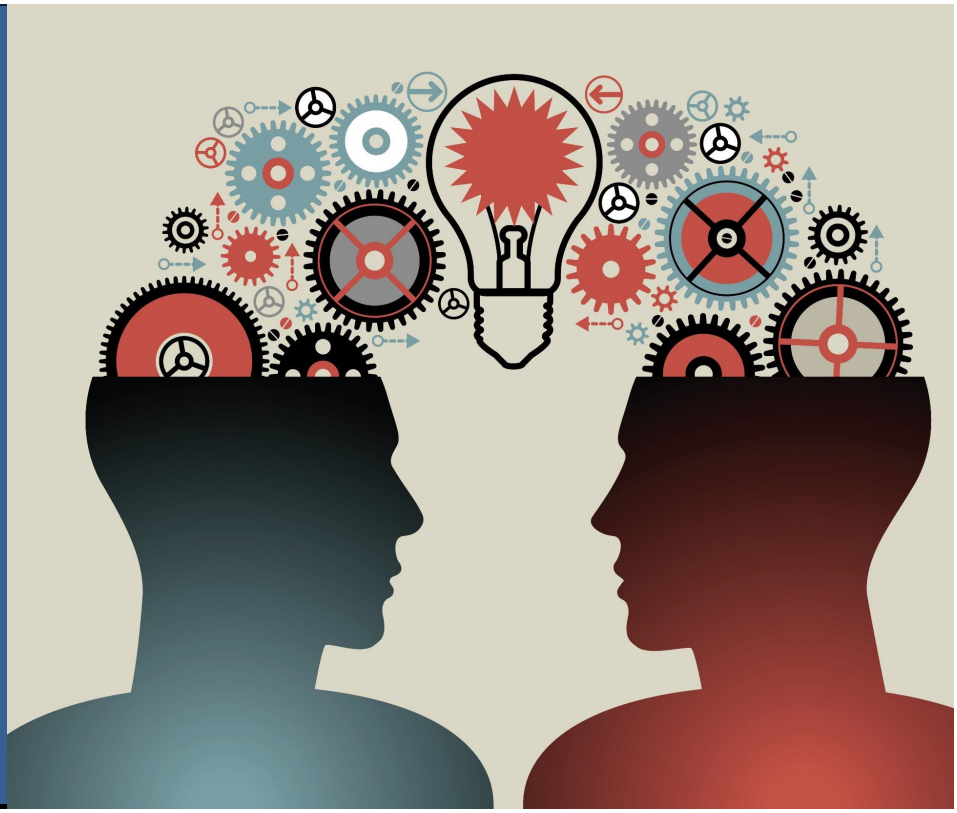


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Knowledge transfer



Prof. Saerom Park

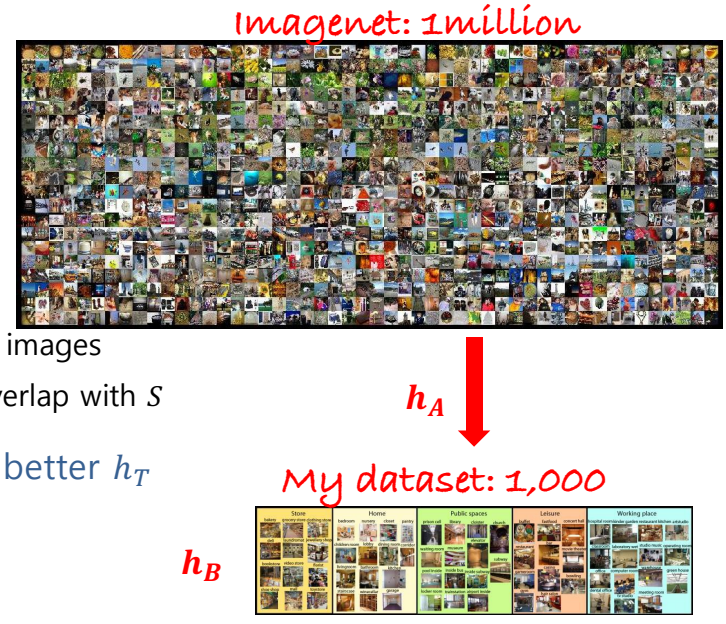
INTRODUCTION TO DEEP LEARNING

CNNs and dataset size

- A CNN can have millions of parameters
- What about the dataset size?
- Could we still train a CNN without overfitting problems?

Transfer learning

- Assume two datasets, T and S
- Dataset S is
  - fully annotated, plenty of images
  - We can build a model h_s
- Dataset T is
  - Not as much annotated, or much fewer images
  - The annotations of T do not need to overlap with S
- We can use the model h_s to learn a better h_T
- This is called transfer learning



Convnets are good in transfer learning

- Even if our dataset T is not large, we can train a CNN for it
- Pre-train a network on the dataset S
- Then, there are two solutions
  - Fine-tuning
  - CNN as feature extractor

Solution I: Fine-tune using h_T as h_S initialization

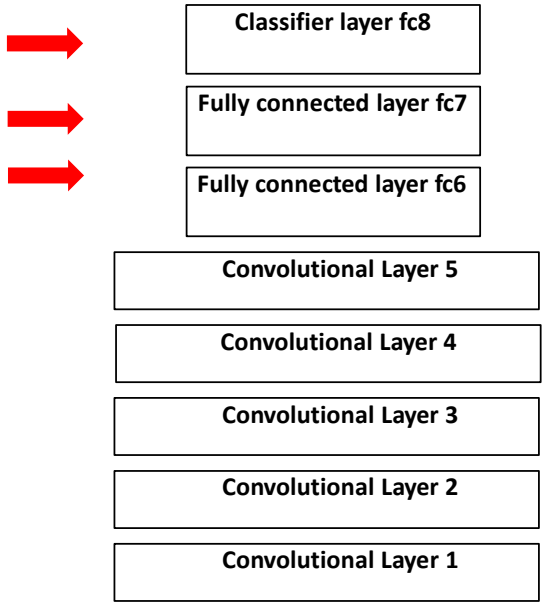
- Assume the parameters of S are already a good start near our final local optimum
- Use them as the initial parameters for our new CNN for the target dataset

$$\theta_{T,l}^{(t=0)} = \theta_{S,l} \; for \; some \; layer \; l = 1,2 \ldots$$

- This is a good solution when the dataset T is relatively big
  - **E.g. for Imagenet S with approximately 1 million images**
  - **For a dataset T with more than a few thousand images should be ok**
- What layers to initialize and how?

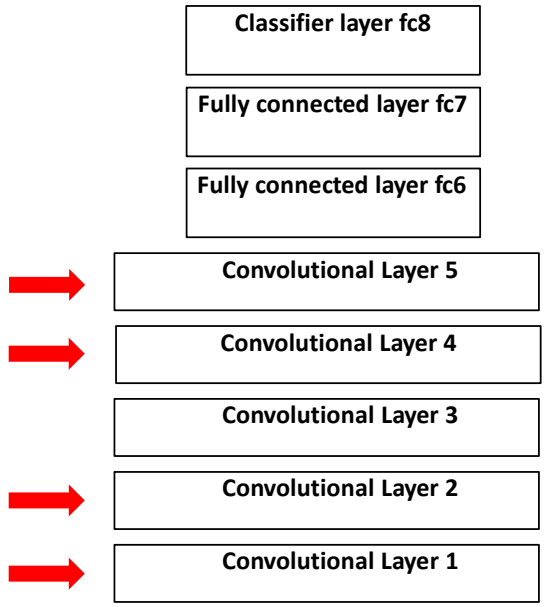
Initializing h_T with h_s

- Classifier layer to loss
  - The loss layer essentially is the "classifier"
  - Same labels -> keep the weights from h_s
  - Different labels -> delete the layer and start over
  - When too few data, fine - tune only this layer
- Fully connected layers
  - Very important for fine-tuning
  - Sometimes you need to completely delete the last before the classification layer if datasets are very different
  - Capture more semantic, "specific" information
  - Always try first when fine-tuning
  - If you have more data, fine-tune also these layers



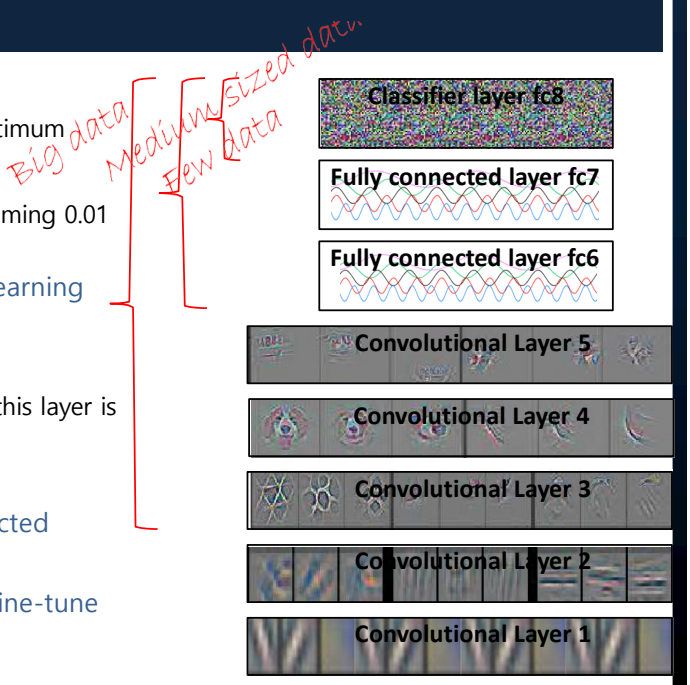
Initializing h_T with h_s

- Upper convolutional layers (conv4, conv5)
  - Mid-level spatial features (face, wheel detectors ...)
  - Can be different from dataset to dataset
  - Capture more generic information
  - Fine-tuning pays off
  - Fine-tune if dataset is big enough
- Lower convolutional layers (conv1,conv2)
  - They capture low level information
  - This information does not change usually
  - Probably, no need to fine-tune but no harm trying



How to fine-tune?

- For layers initialized from h_s use a mild learning rate
  - Remember: your network is already close to a near optimum
  - If too aggressive, learning might diverge
  - A learning rate of 0.001 is a good starting choice (assuming 0.01 was the original learning rate)
- For completely new layers (e.g. loss) use aggressive learning rate
  - If too small, the training will converge very slowly
  - Remember: the rest of the network is near a solution, this layer is very far from one
  - A learning rate of 0.01 is a good starting choice
- If datasets are very similar, fine-tune only fully connected layers
- If datasets are different and you have enough data, fine-tune all layers



Solution II: Use h_s as a feature extractor for h_T

- Essentially similar to a case of solution I
  - but train only the loss layer
- Essentially use the network as a pretrained feature extractor
- This is a good solution if the dataset T is small
  - Any fine-tuning of layer might cause overfitting
- Or when we don't have the resources to train a deep net
- Or when we don't care for the best possible accuracy

Deep features from different layers

Convolutional Layer 1

Convolutional Layer 2

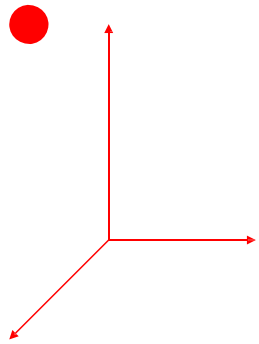
Convolutional Layer 3

Convolutional Layer 4

Convolutional Layer 5

Fully connected layer fc6

Fully connected layer fc7



fc7 feature space

Deep features from different layers

Convolutional Layer 1

Convolutional Layer 2

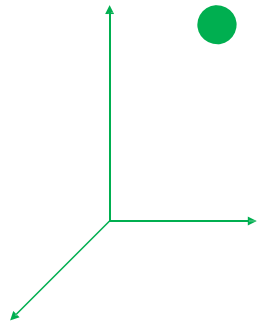
Convolutional Layer 3

Convolutional Layer 4

Convolutional Layer 5

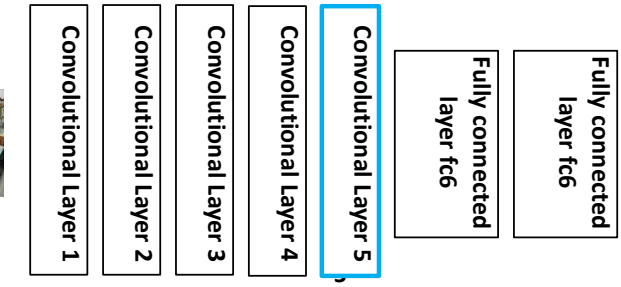
Fully connected layer fc6

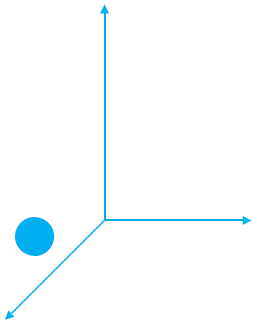
Fully connected layer fc6



fc6 feature space

Deep features from different layers





Conv5 feature space

Which layer?

Table 6. Analysis of the discriminative information contained in each layer of feature maps within our ImageNet-pretrained convnet. We train either a linear SVM or softmax on features from different layers (as indicated in brackets) from the convnet. Higher layers generally produce more discriminative features.

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

Higher layer features are capture more semantic information. Good for higher level classification

Lower layer features capture more basic information (texture, etc). Good for image-to-image comparisons, image retrieval

Summary

- What are the Convolutional Neural Networks?
- Why are they so important for Computer Vision?
- How do they differ from standard Neural Networks?
- How can we train a Convolutional Neural Network?

Reading material & references

- http://www.deeplearningbook.org/
  - Part II: Chapter 9

[He2016] He, Zhang, Ren, Sun. Deep Residual Learning for Image Recognition, CVPR, 2016[Simonyan2014] Simonyan, Zisserman/ Very Deep Convolutional, Networks for Large-Scale Image Recognition, arXiv, 2014 [Taigman2014] Taigman, Yang, Ranzato, Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR, 2014[Zeiler2014] Zeiler, Fergus. Visualizing and Understanding Convolutional Networks, ECCV, 2014[Krizhevsky2012] Krizhevsky, Hinton. ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012[LeCun1998] LeCun, Bottou, Bengio, Haffner. Gradient-Based Learning Applied to Document Recognition, IEEE, 1998