Convolutional Neural Network (CNN, R-CNN)

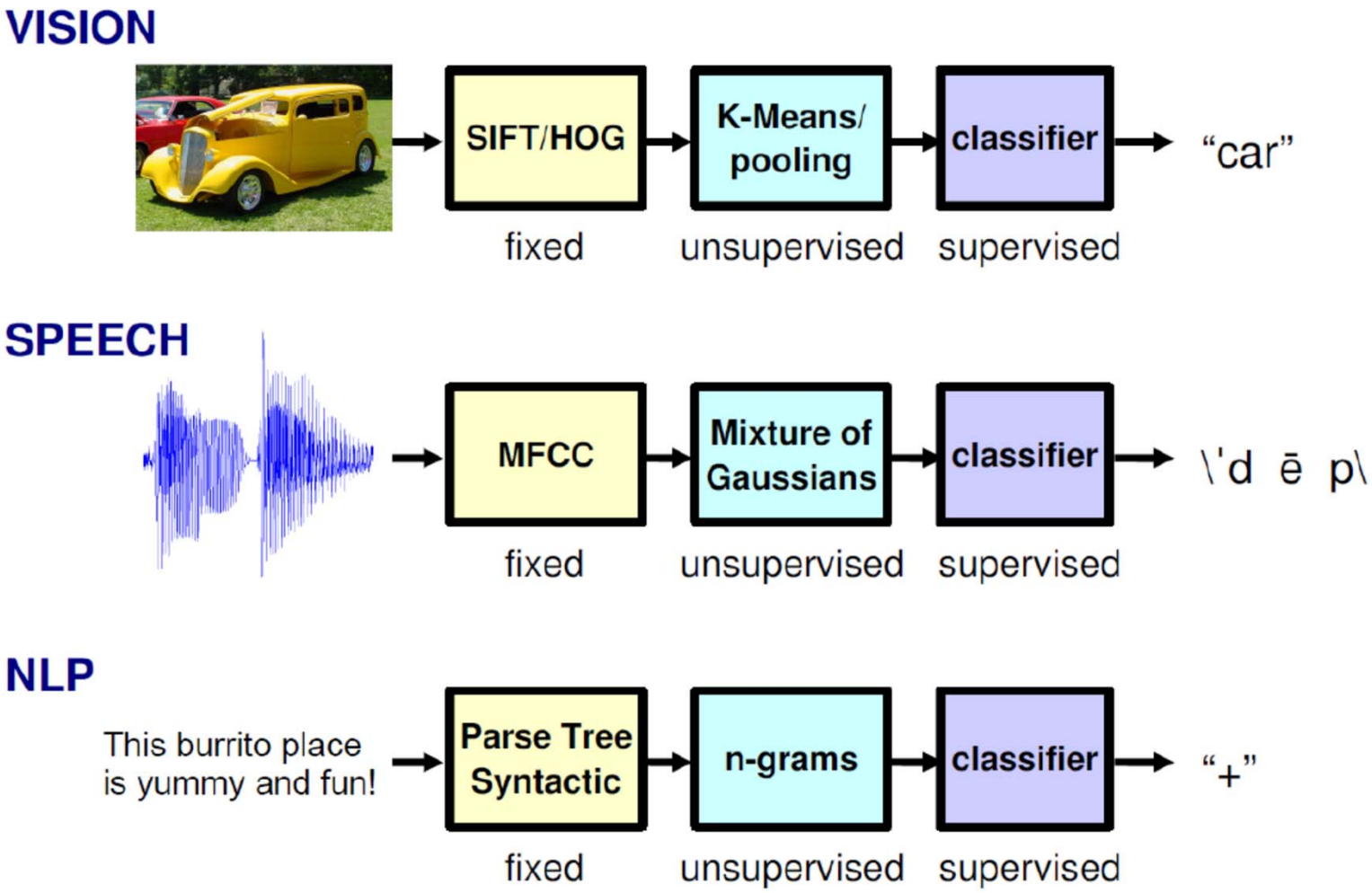
머신러닝 기반 빅데이터 엔지니어링 과정

빅데이터 X Campus (단국대학교)

2018.08

컴퓨터공학과 최상일 교수

Traditional Pattern Recognition



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Hierarchical Compositionality (DEEP)

VISION

pixels → edge → texton → motif → part → object

SPEECH

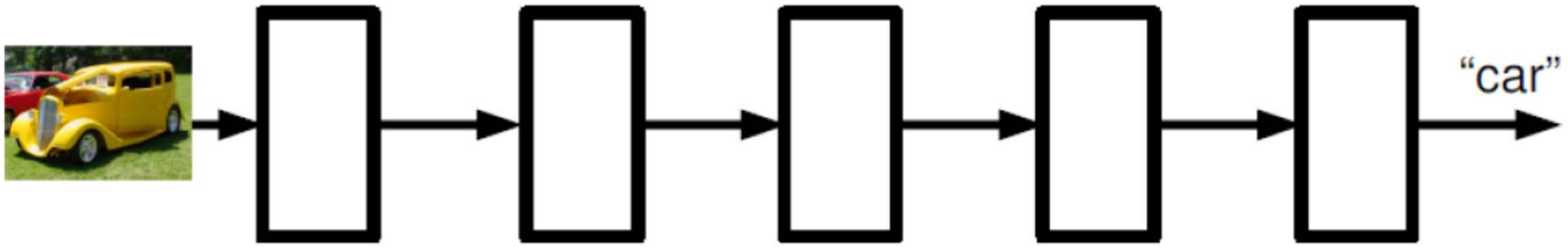
sample → spectral band → formant → motif → phone → word

NLP

character → word → NP/VP/.. → clause → sentence → story

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Deep Learning

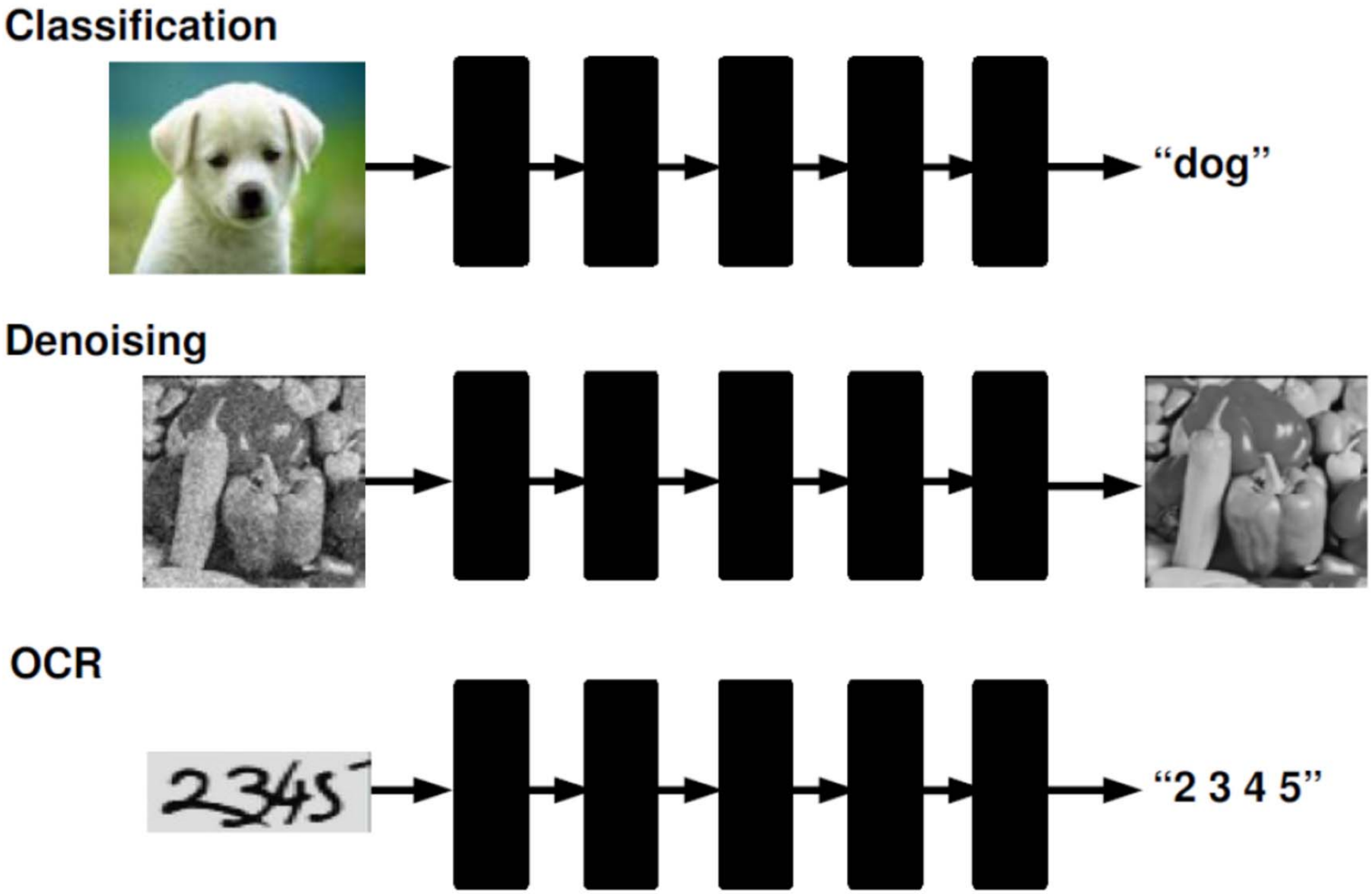


What is deep learning?

- Nothing new!
- (Many) cascades of nonlinear transformations
- End-to-end learning (no human intervention / no fixed features)

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

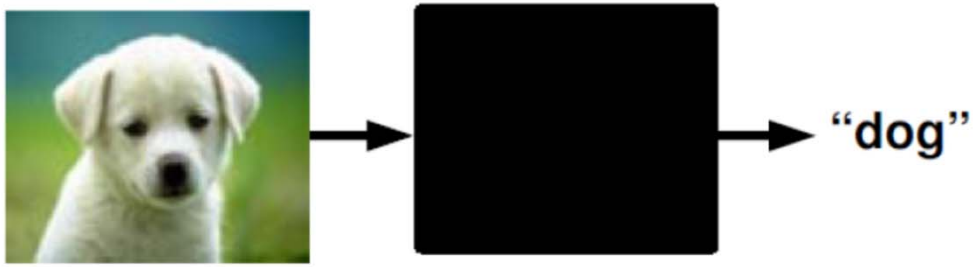
Deep vs. Shallow(?) Learning Examples



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Deep vs. Shallow(?) Learning Examples

Classification



classification

Denoising

regression

OCR

structured prediction

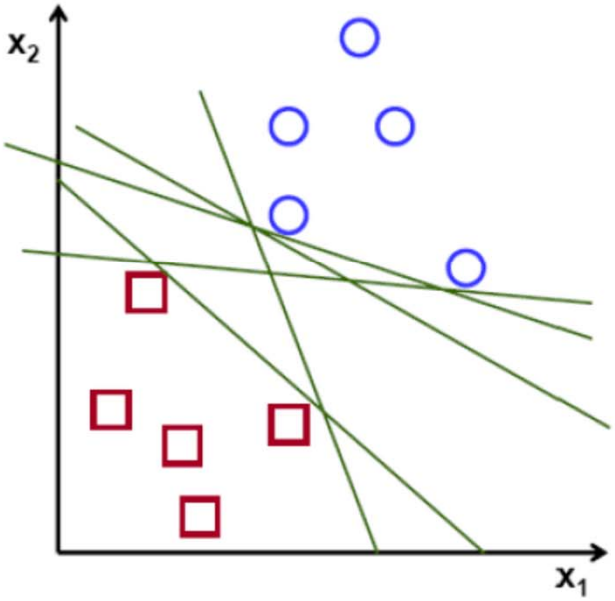
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

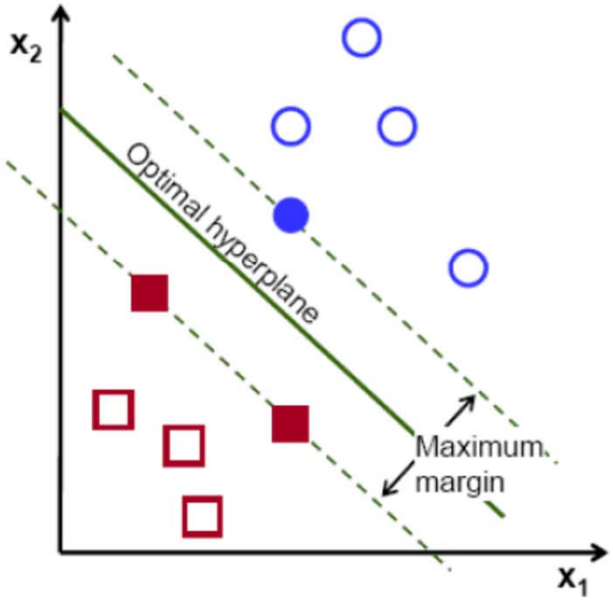
Classification Problems

- Given inputs $\boldsymbol{x}$, and outputs $t \in \{-1, 1\}$
- Find a hyperplane that divides the space into half (binary classification)

$$y_* = \text{sign}(\boldsymbol{w}_*^T \boldsymbol{x} + \boldsymbol{w}_0)$$

$\Rightarrow$ SVM tries to maximize the margin.





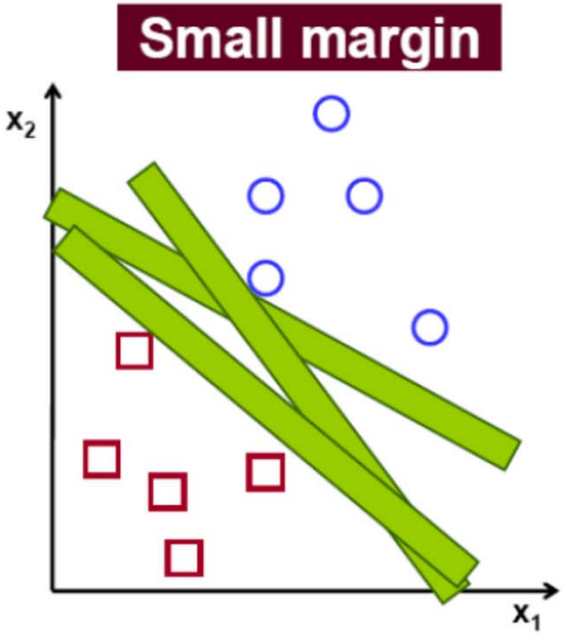
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

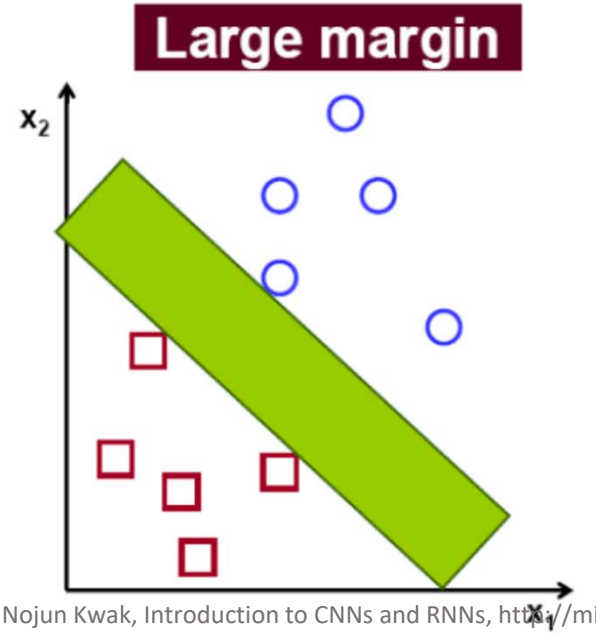
Classification Problems

- Given inputs $\boldsymbol{x}$, and outputs $t \in \{-1, 1\}$
- Find a hyperplane that divides the space into half (binary classification)

$$y = \text{sign}(\boldsymbol{w}^T \boldsymbol{x} + b)$$

$\Rightarrow$ SVM tries to maximize the margin.





출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Nonlinear Predictors

How can we make our classifier more powerful?

- Compute nonlinear functions of the input

$$y = F(\boldsymbol{x}, \boldsymbol{w})$$

Two types of widely used approaches

- Kernel Trick: Fixed functions and optimize linear parameters on nonlinear mappings $\phi(\boldsymbol{x})$

$$y = \text{sign}(\boldsymbol{w}^T \phi(\boldsymbol{x}) + b)$$

- Deep Learning: Learn parametric nonlinear functions

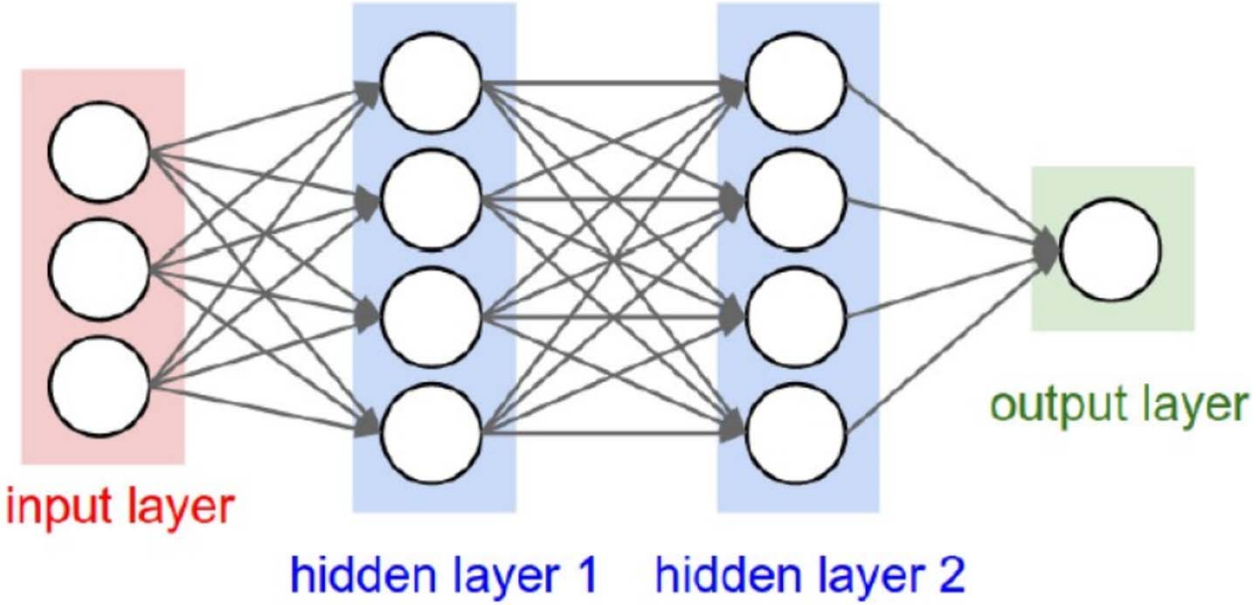
$$y = F(\boldsymbol{x}, \boldsymbol{w}) = \cdots (\boldsymbol{h}_2(\boldsymbol{w}_2^T \boldsymbol{h}_1(\boldsymbol{w}_1^T \boldsymbol{x} + b_1) + b_2) \cdots$$

$\boldsymbol{h}_{1,2}$: activation function at layer 1 or 2

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

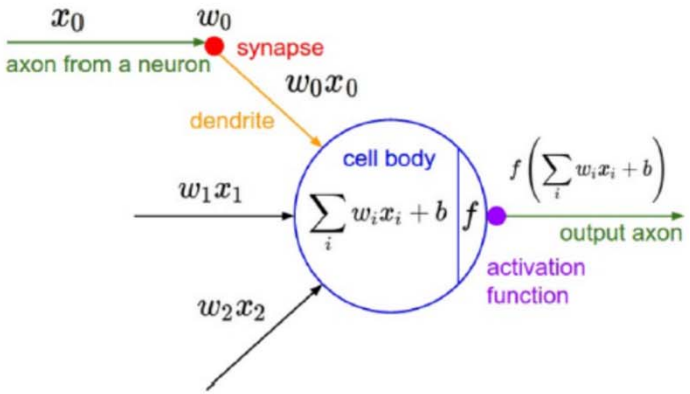
Neural Networks

- Deep learning uses composite of simpler functions, e.g., ReLU, sigmoid, tanh, max
- Note: a composite of linear functions is linear!
- Example: 2 layer NNet (Convention: input and output layers are not taken as a layer)



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

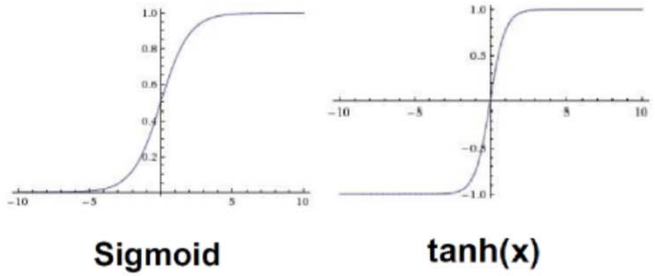
Nonlinearity – Activation function



- A singlue neuron can be used as a binary linear classifier
- Regularization has the interpretation of gradual forgetting

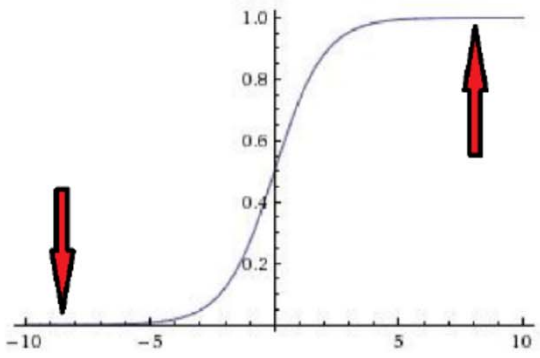
Classical NNs used sigmoid or tanh function as an activation function.

- sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- tanh: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Nonlinearity – Sigmoid function



Sigmoid

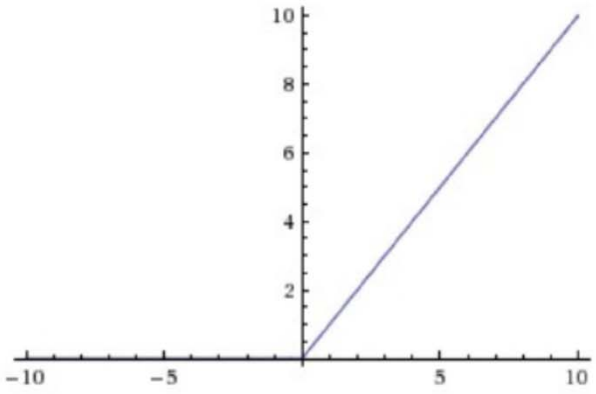
- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron

2 BIG problems:

1. Saturated neurons kill the gradients (cannot backprop further)
   ⇒ Major bottleneck for the conventional NNs: not able to train more than 2 or 3 layers
2. Sigmoid outputs are not zero-centered
   ⇒ Restriction on the gradient directions

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Nonlinearity – ReLu function

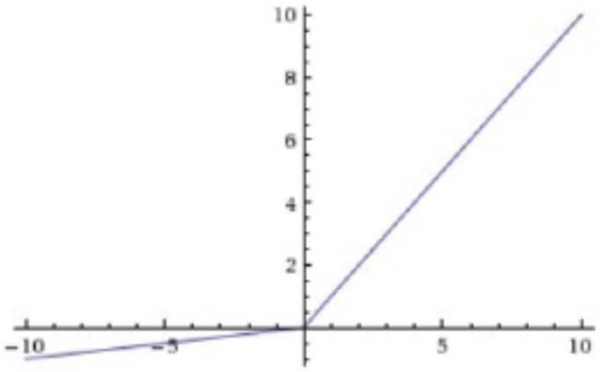


ReLU

- $f(x) = \max(0, x)$
- Does not saturate
- Computationally very efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Nonlinearity – ReLu function



- $f(x) = \max(0, x)$
- Does not saturate
- Computationally very efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

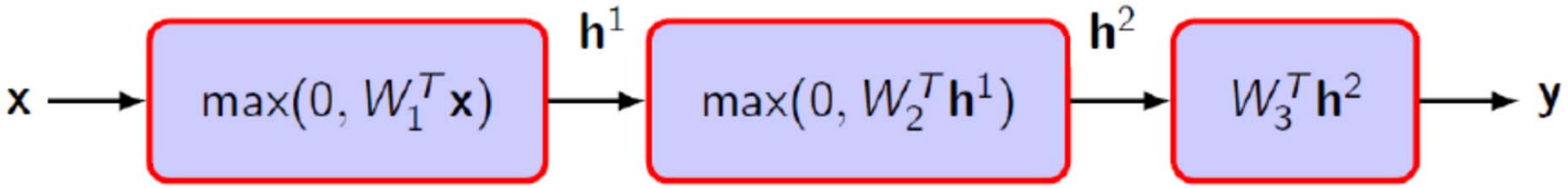
Leaky ReLU

- One annoying problem ⇒ Dead neurons
- Solution: leaky ReLU (small slope for negative input)
  - Never dies.
  - However, almost the same performance in practice.

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Forward Pass: Evaluating the function

- Forward propagation: compute the output $\boldsymbol{y}$ given the input $\boldsymbol{x}$

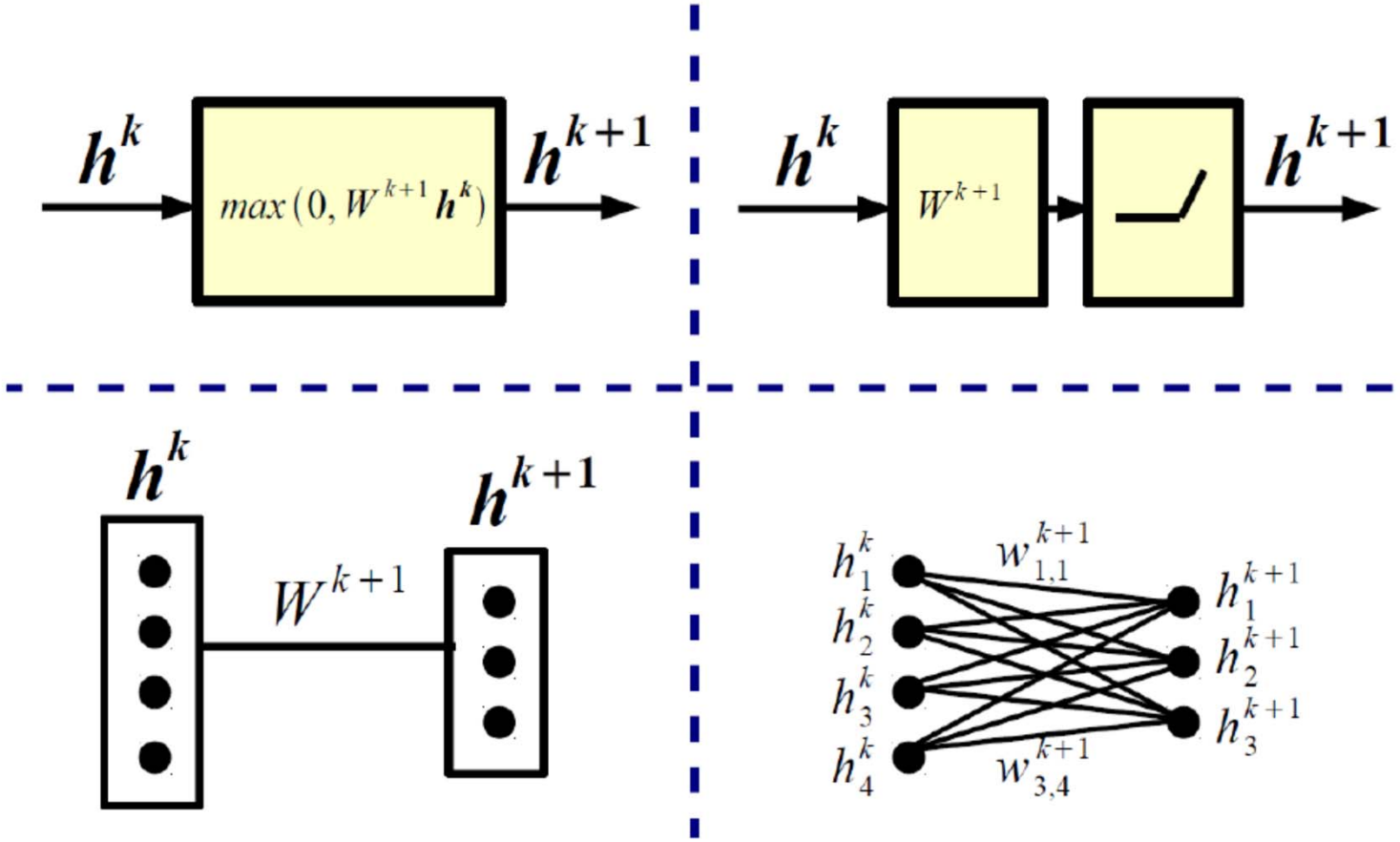


- Fully connected layer
- Nonlinearity comes from ReLU
- Do it in a compositional way

$$\boldsymbol{x} \Rightarrow \boldsymbol{h}^1 \Rightarrow \boldsymbol{h}^2 \Rightarrow \boldsymbol{y}$$

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Alternative graphical representation



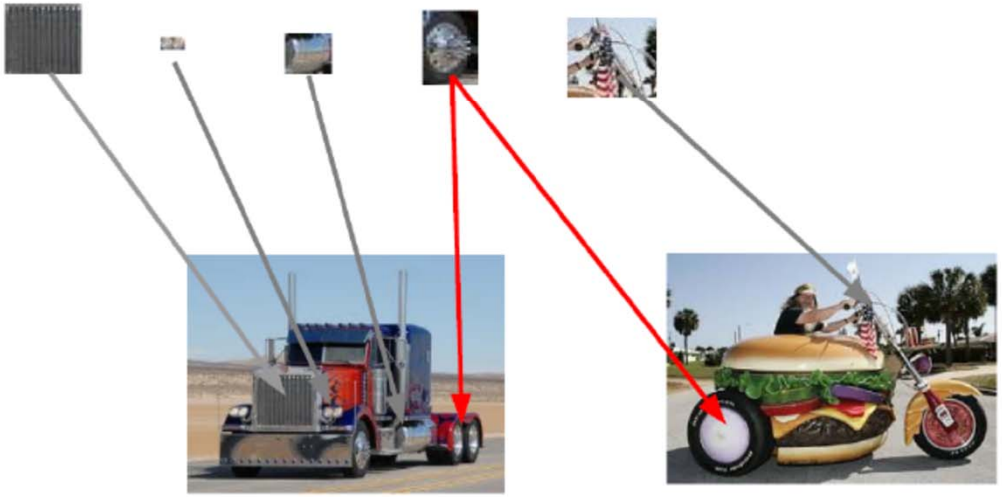
Slide from M. Ranzato

Why Many Layers?

Hierarchically distributed representations

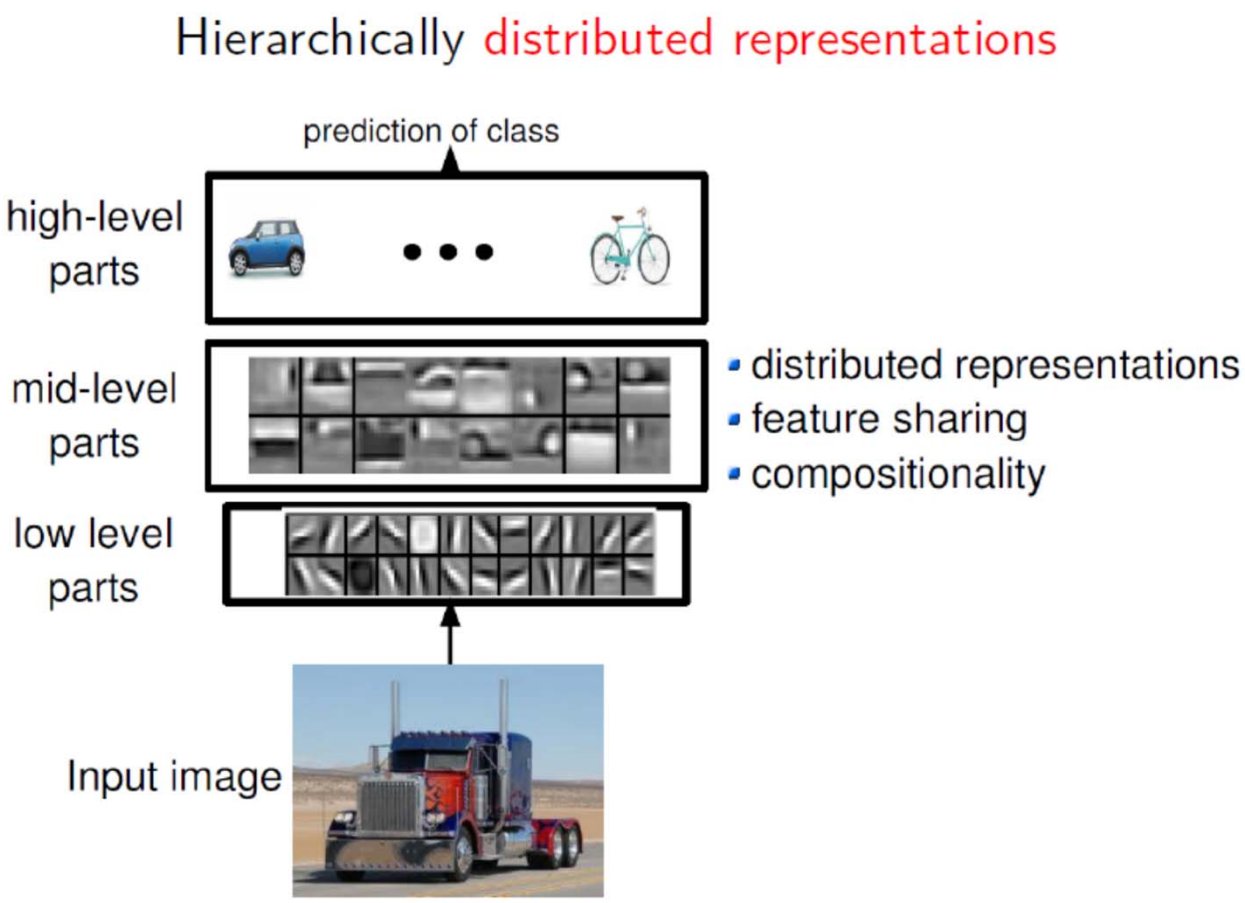
[1 1 0 0 0 1 0 **1** 0 0 0 0 1 1 0 1 . . .] motorbike

[0 0 1 0 0 0 0 **1** 0 0 1 1 0 0 1 0 . . .] truck



Lee et al. "Convolutional DBN's · · · " ICML 2009

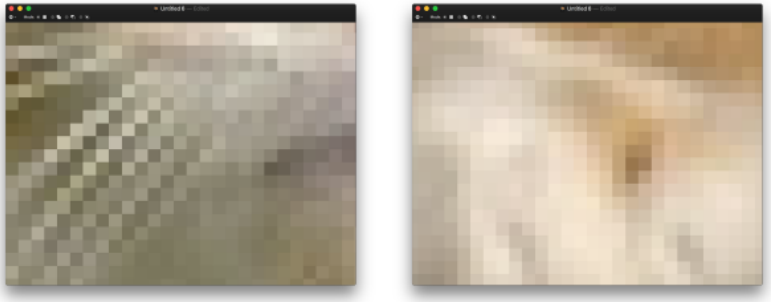
Why Many Layers?



Lee et al. "Convolutional DBN's · · · " ICML 2009

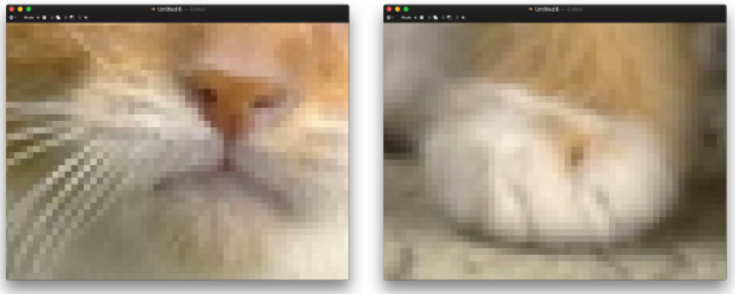
Convolutional Neural Network (CNN)

그림을 눈앞 1cm거리에서 본다고 생각해보자.



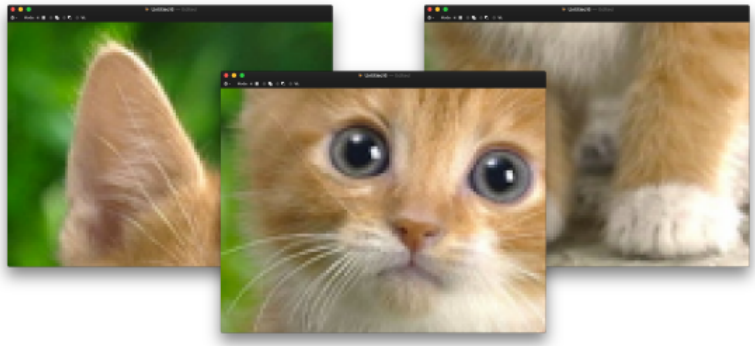
처음에는 점과 선, 이상한 질감 몇개 밖에 모르겠다.

점과 선, 질감을 충분히 배우고, 조금 떨어져서 보자.



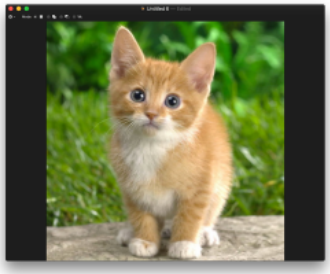
점과 선이 질감이 합쳐져 삼각형, 동그라미, 북실함이 보인다.

삼각형, 원, 사각형, 북실함등을 조합해서 보니



뾰족귀와 땡그란눈과 복실한 발을 배웠다.

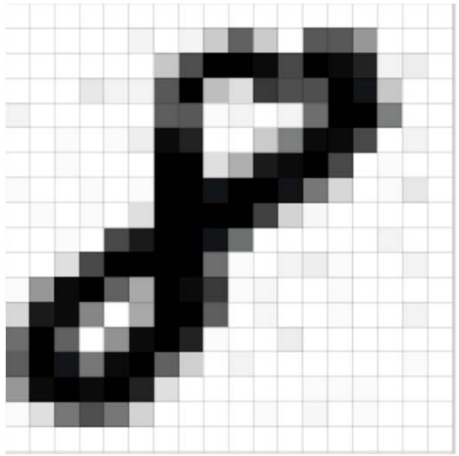
더 멀리서 보니, 그것들이 모아져있다. 이것은?



고양이!!

Convolutional Neural Network (CNN)

- CNN(Convolutional Neural Network)구조 설계에 동기
  - 신경생물학적 관찰
  - 사람의 뇌는 물체의 방향과 장소가 바뀌어도 별 문제없이 인식 가능
    - CNN은 물체의 위치와 방향에 관계없이 물체의 고유한 특징을 학습
- 이미지의 픽셀 값으로부터 직접 시각 패턴을 학습
- 컴퓨터에서 모든 이미지는 픽셀(Pixel)값으로 전환 가능
  - 0~255 범위의 픽셀 값
    - 픽셀 값 0 : 흰색
    - 픽셀 값 255 : 검정색
- CNN을 비롯한 인공신경망 알고리즘들에서 이미지를 입력으로 사용 → 픽셀 값들을 이용해서 계산해 나가겠다는 의미



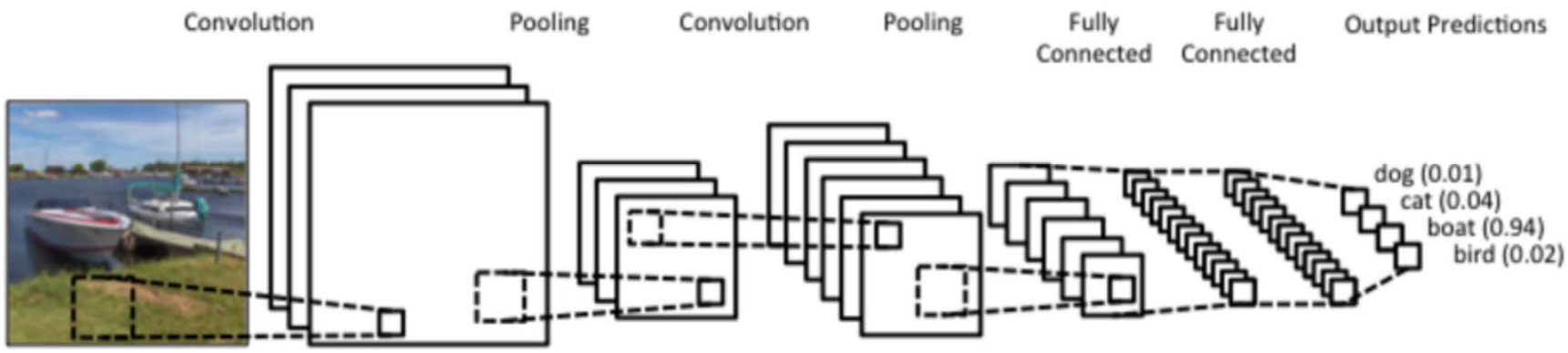
Convolutional Neural Network (CNN)

- 조각을 보고, 패턴을 익히고, 점점 멀리서 조합을 보는 이 방식을 흉내내면 컴퓨터도 그림을 잘 이해할 수 있지 않을까?

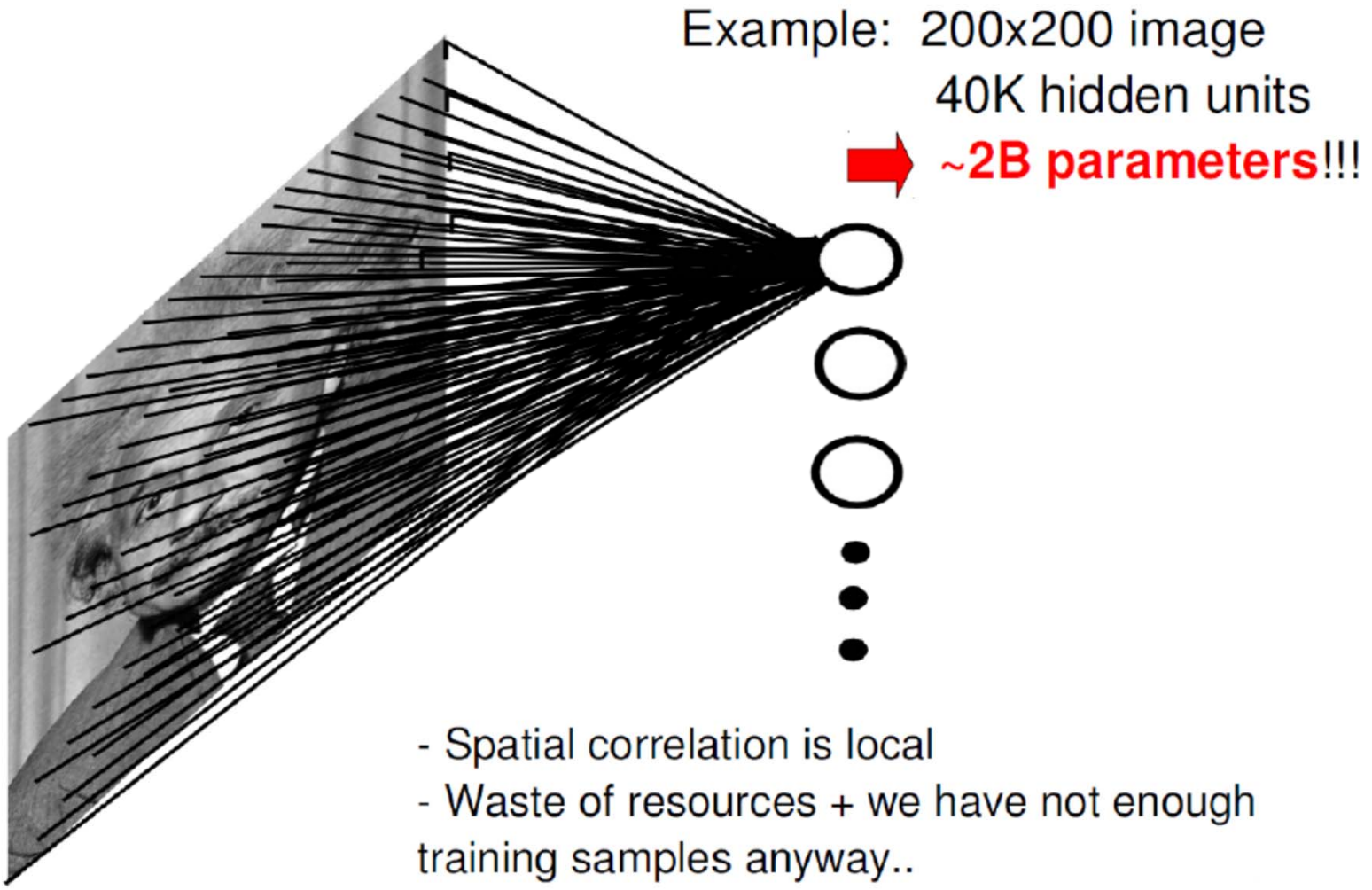
- Convolution 연산
  - 완전 연결(fully connected) 인공신경망을 사용시 너무 많은 파라미터들을 학습해야 함 → 계산량, 시간
  - 이미지에서는 인접한 픽셀들간에는 높은 상관관계를 보이고, 멀리 떨어져 있는 픽셀들 간에는 상관관계를 보이지 않음
    - 인접한 픽셀들간에 묶어서 계산 → 의미 추출 효과, 계산적 이점

CNN 네트워크 구조

- 다음의 4가지 핵심 단계가 CNN 네트워크를 형성
  - Convolution Layer
  - Non-linearity Step
  - Pooling Layer
  - Fully-connected Layer

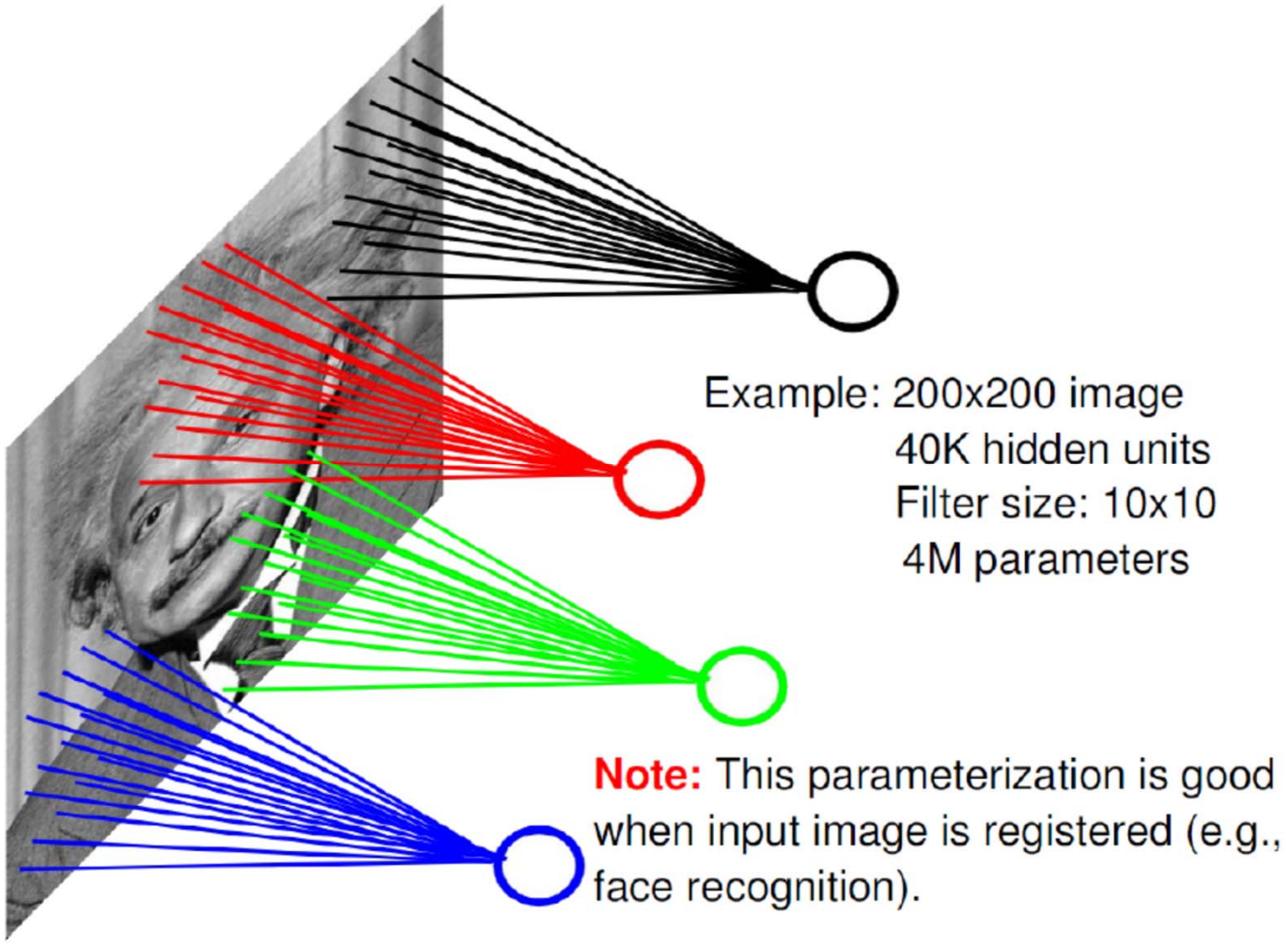


Idea background for CNN



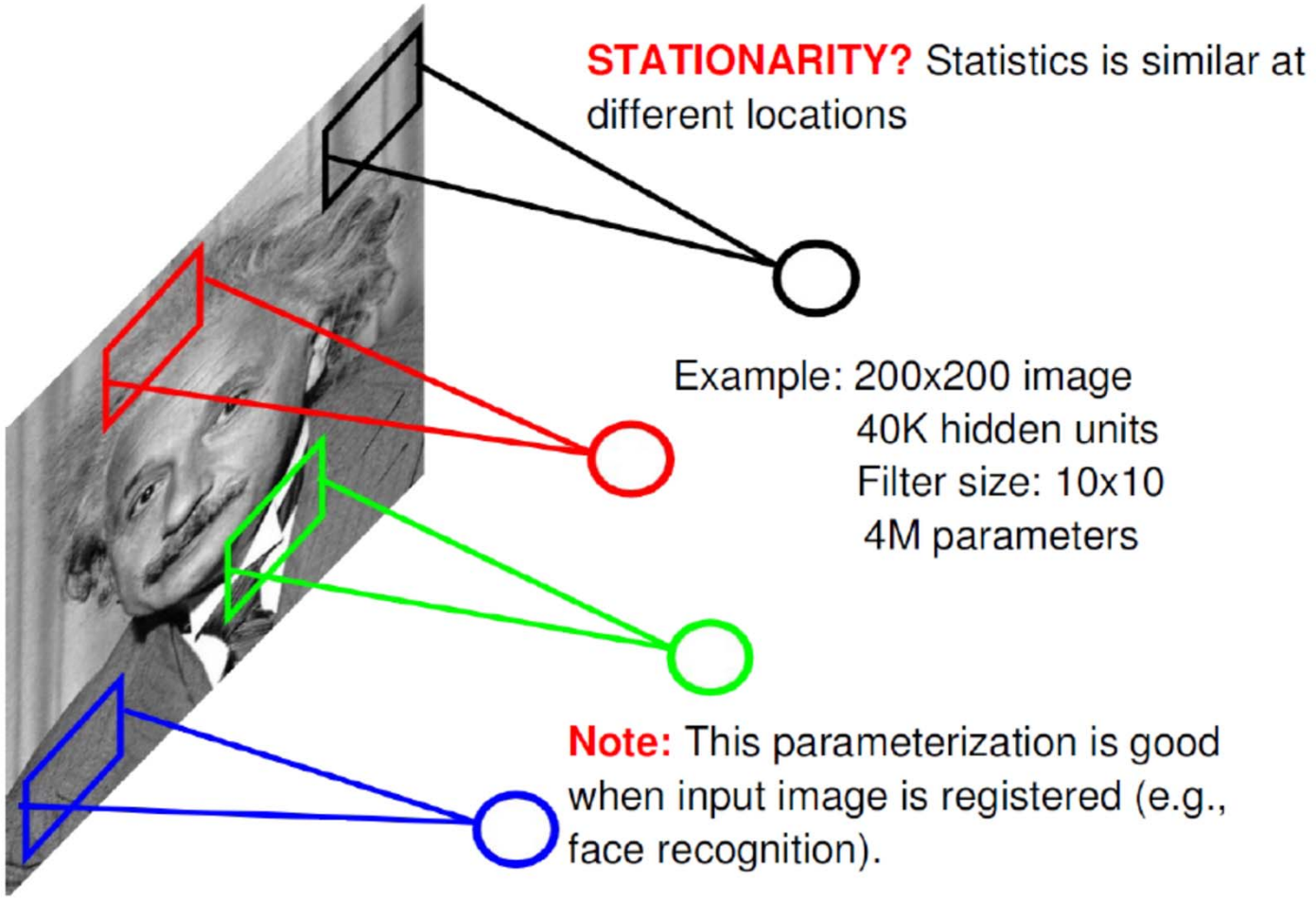
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Idea background for CNN



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

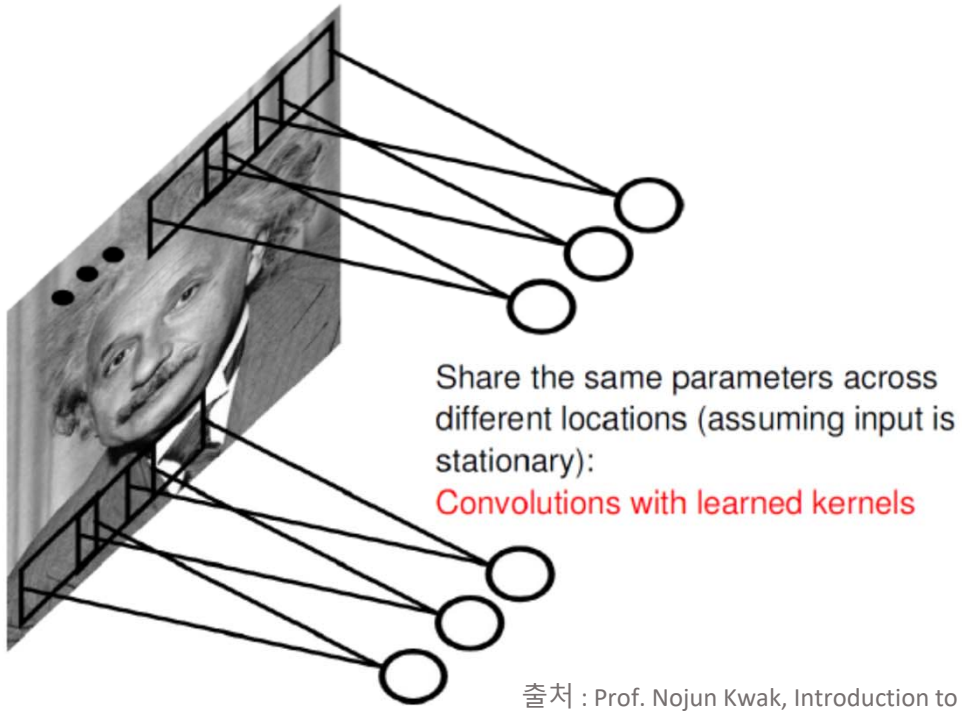
Idea background for CNN



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Idea background for CNN

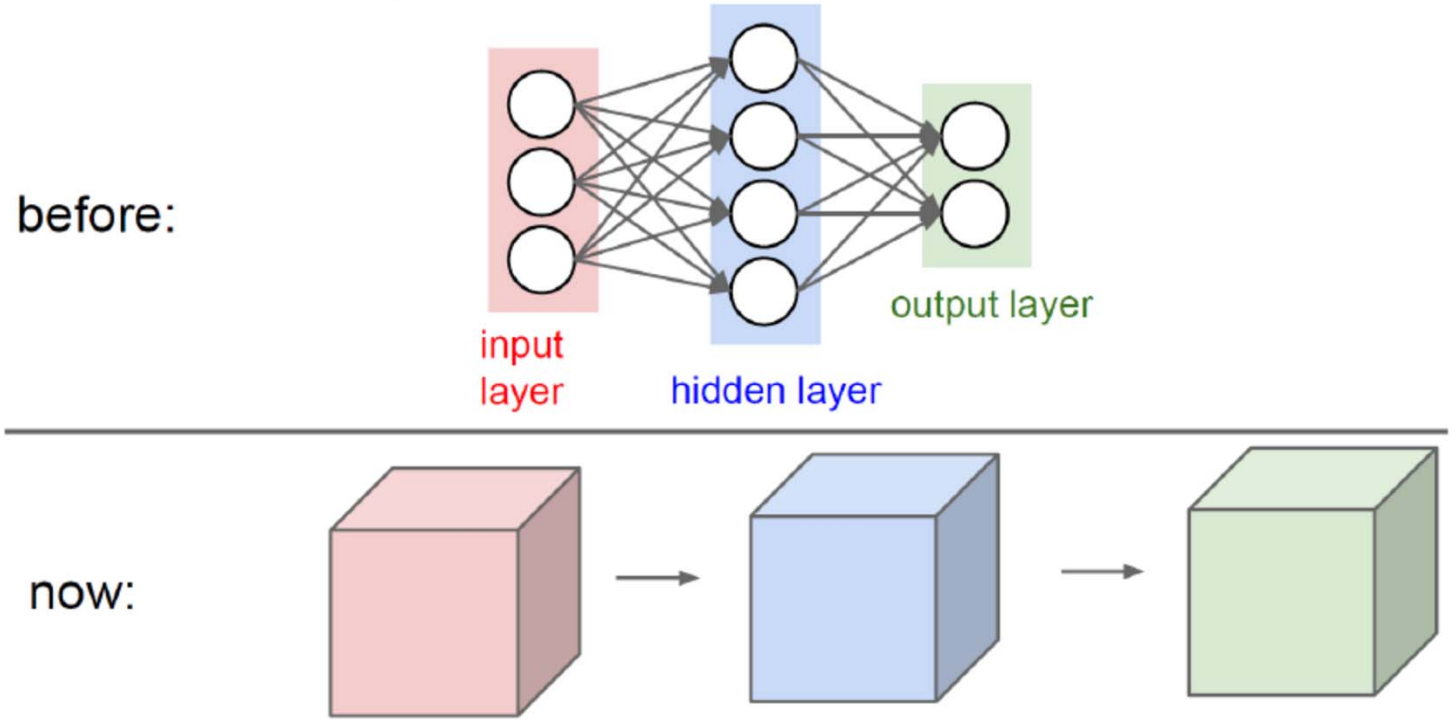
- Idea: statistics are similar at different locations (Lecun 1998)
- Connect each hidden unit to a small input patch and share the weight across space
- This is called convolution layer and the network is a convolutional neural network



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer

- Number of filters (neurons) is considered as a new dimension (depth)
  ⇒ Volumetric representation

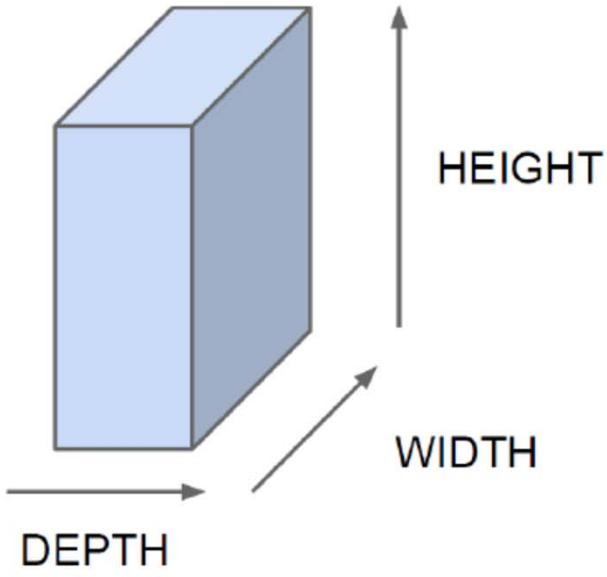


출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer

- Number of filters (neurons) is considered as a new dimension (depth)
  ⇒ Volumetric representation

All Neural Net activations arranged in **3 dimensions:**



For example, a CIFAR-10 image is a 32x32x3 volume
32 width, 32 height, 3 depth (RGB channels)

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer

CNNs are just neural nets BUT:

1. Local connectivity

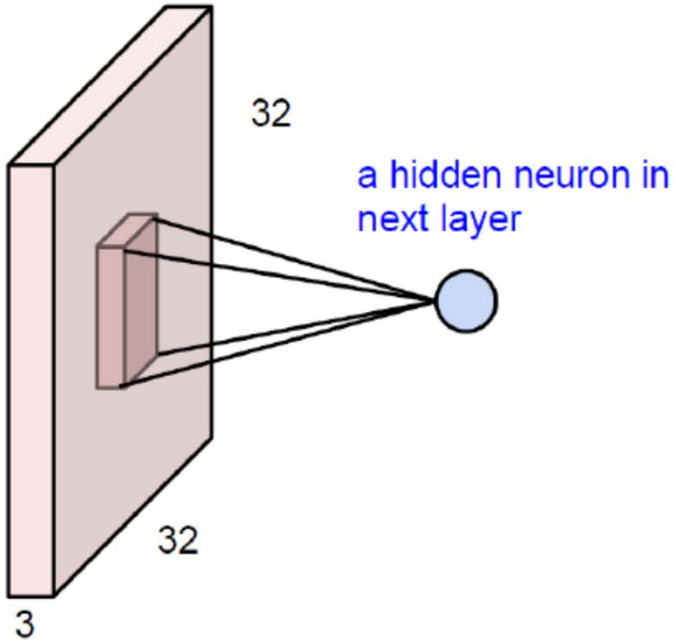


image: 32x32x3 volume

before: fully connected: 32x32x3 weights

now: one neuron will connect to, e.g., 5x5x3 chunk (receptive field) and only have 5x5x3 weights

connectivity is:

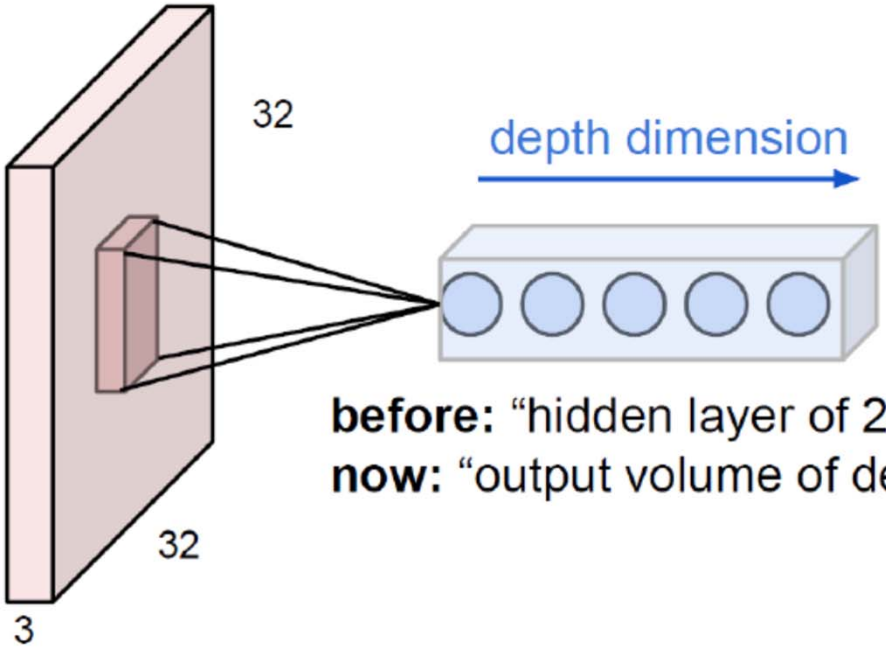
- local in space (5x5 instead of 32x32)
- but full in depth (all 3 depth channels)

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer

CNNs are just neural nets BUT:

1. Local connectivity



Multiple neurons all looking at the same region of the input volume, stacked along depth.

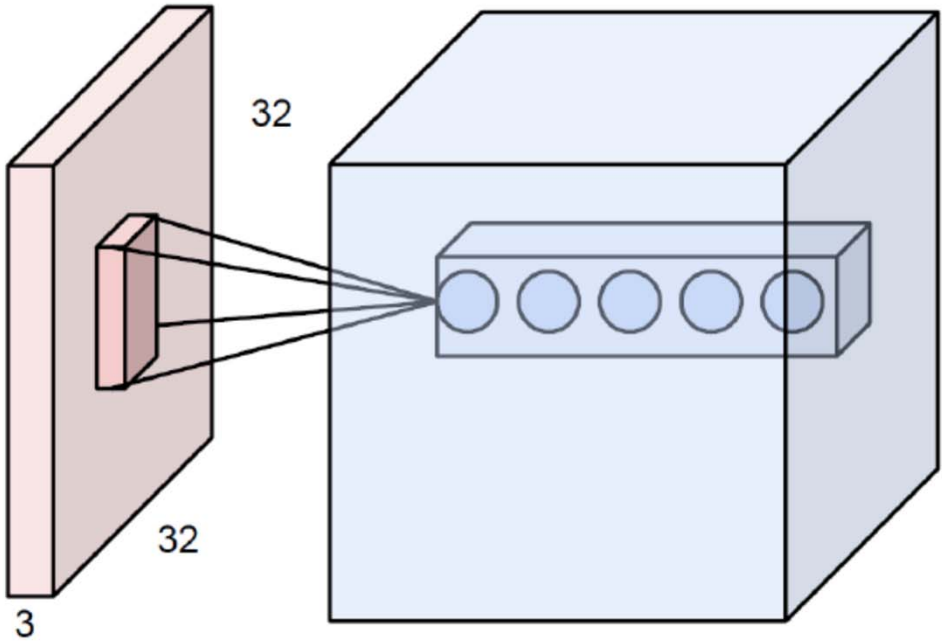
before: "hidden layer of 200 neurons"
now: "output volume of depth 200"

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer

CNNs are just neural nets BUT:

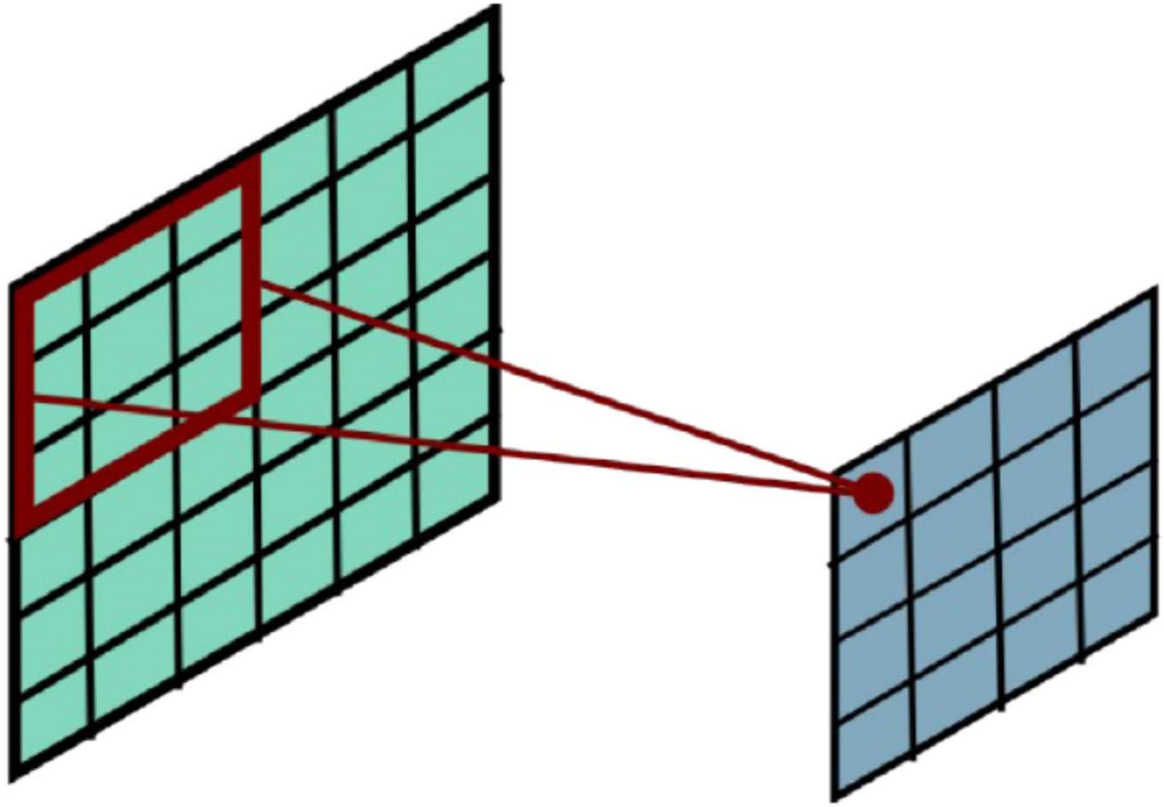
2. Weight sharing



- Weights are shared across different locations
- Each depth slice is called one feature map

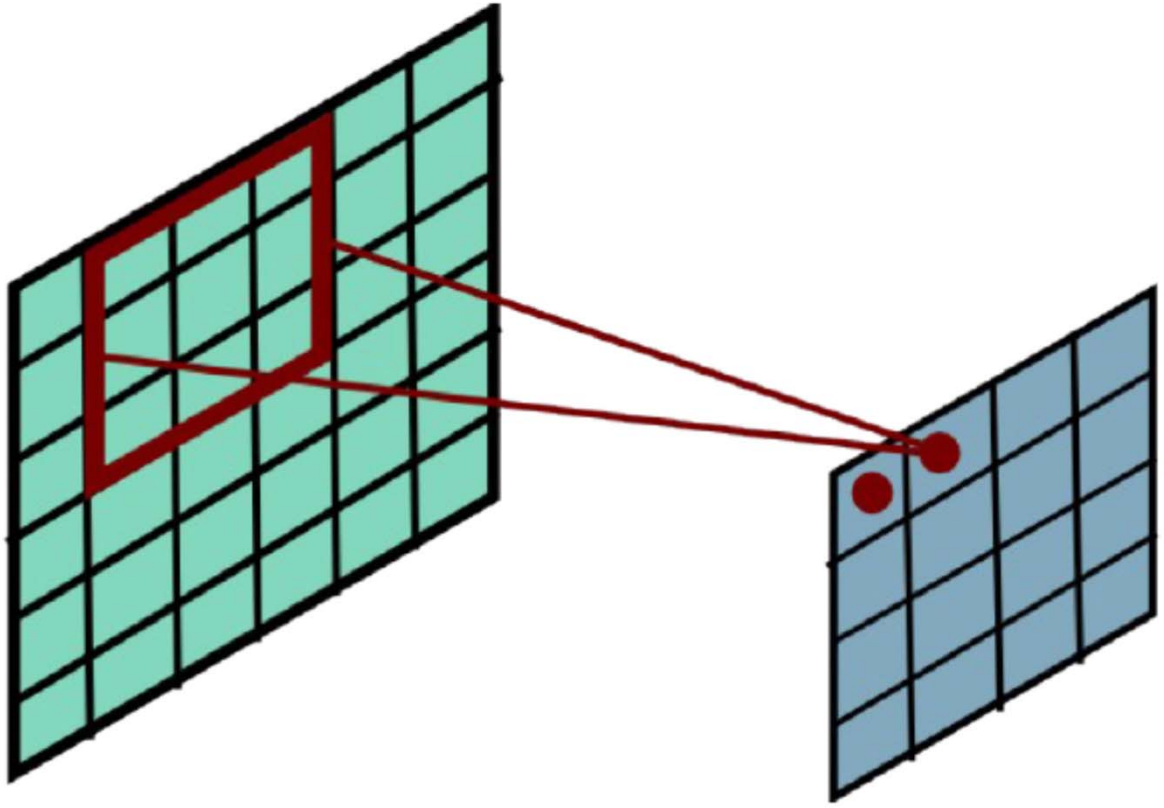
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer



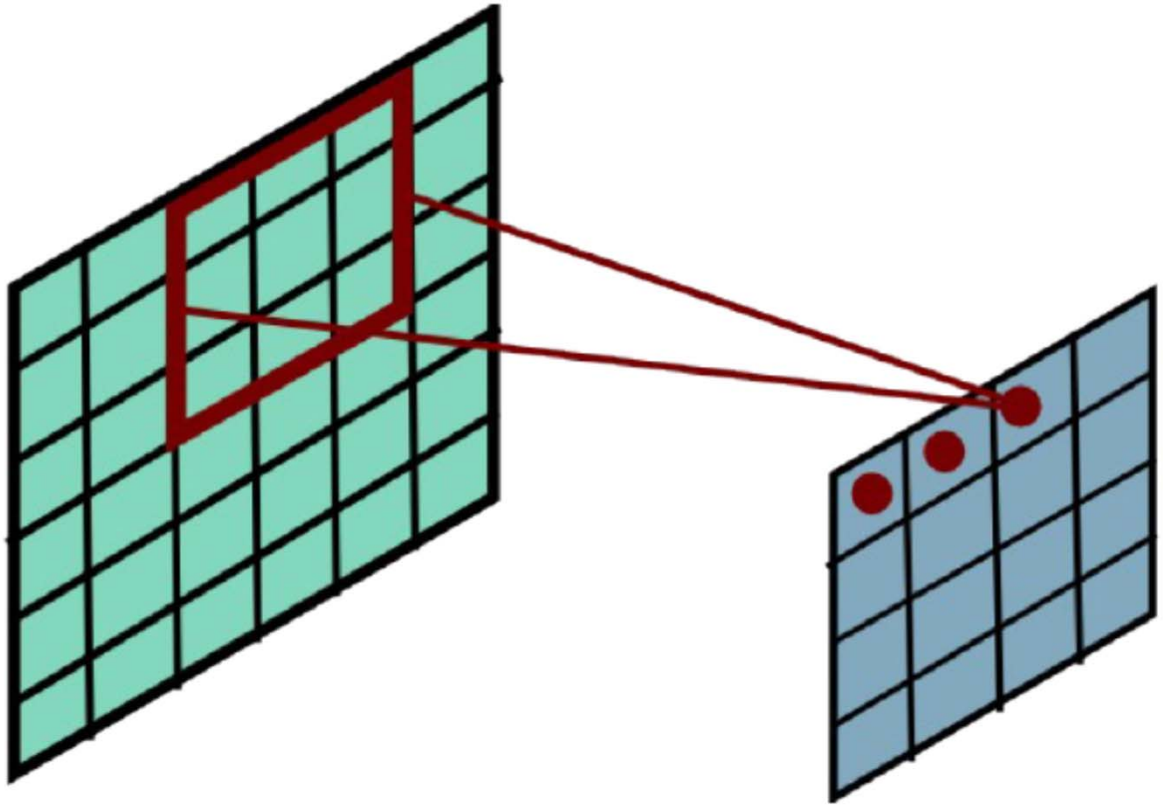
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer



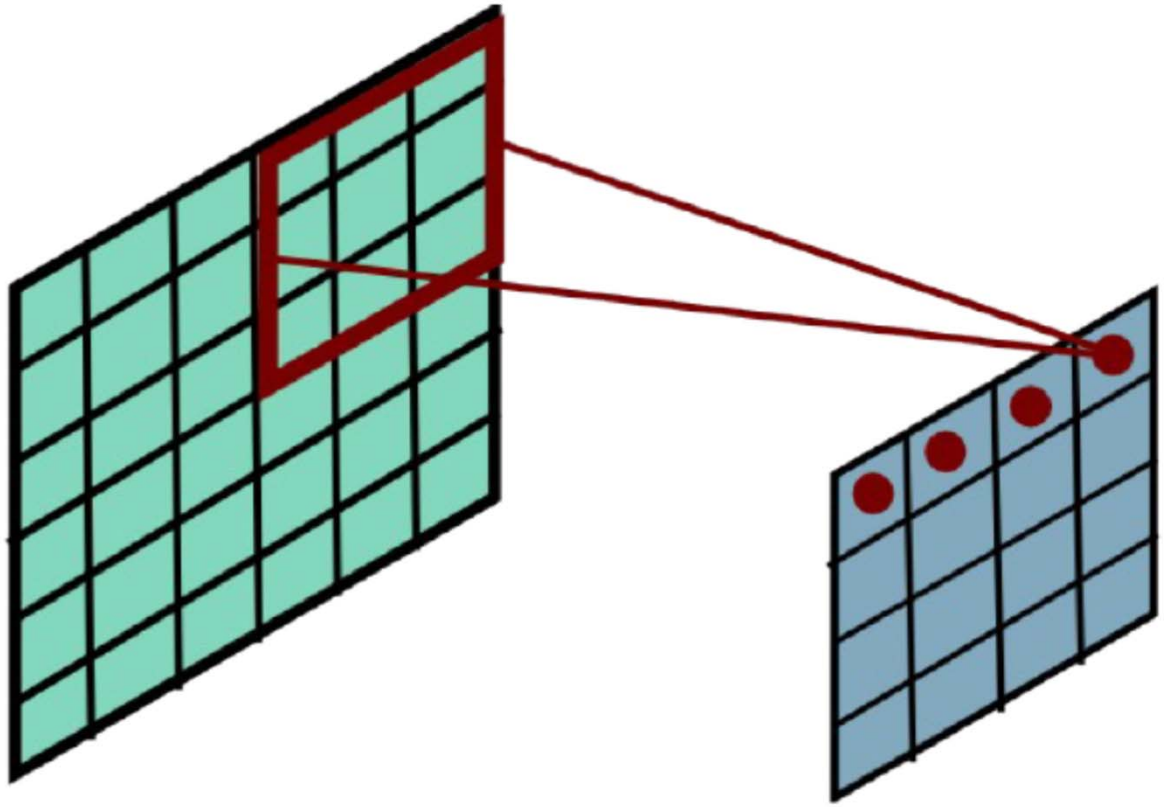
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer



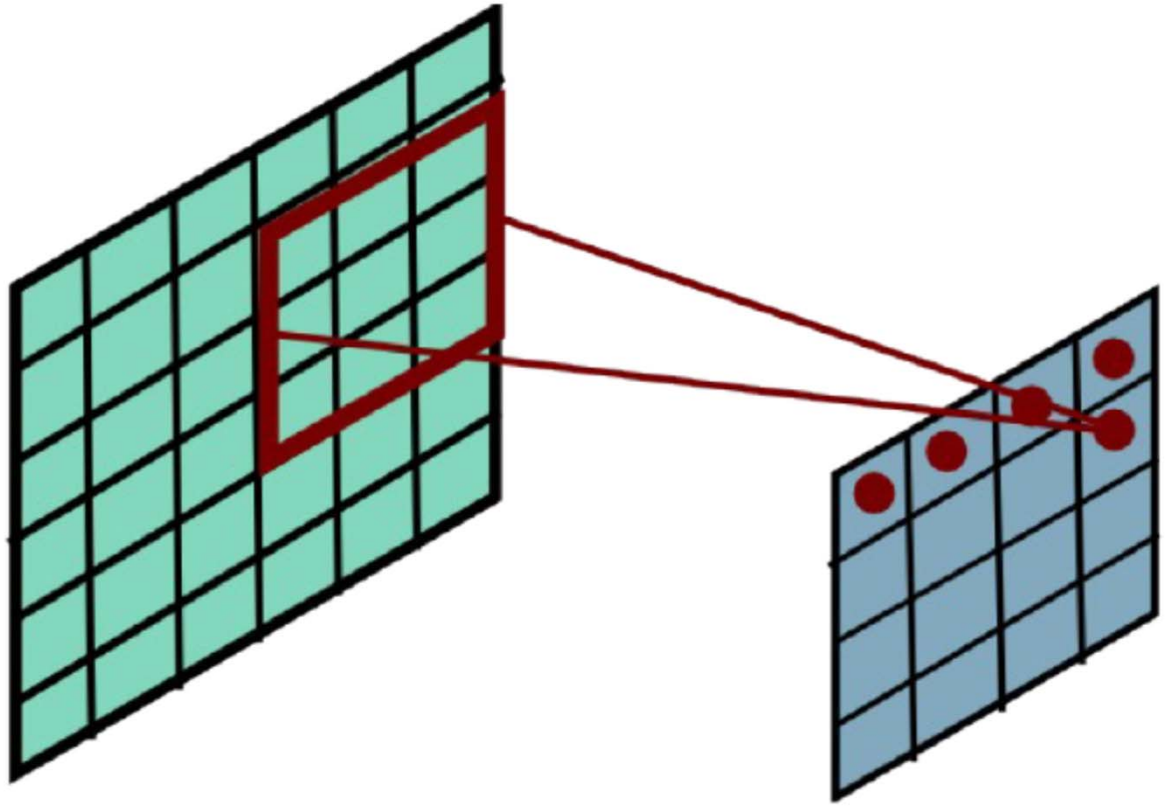
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer



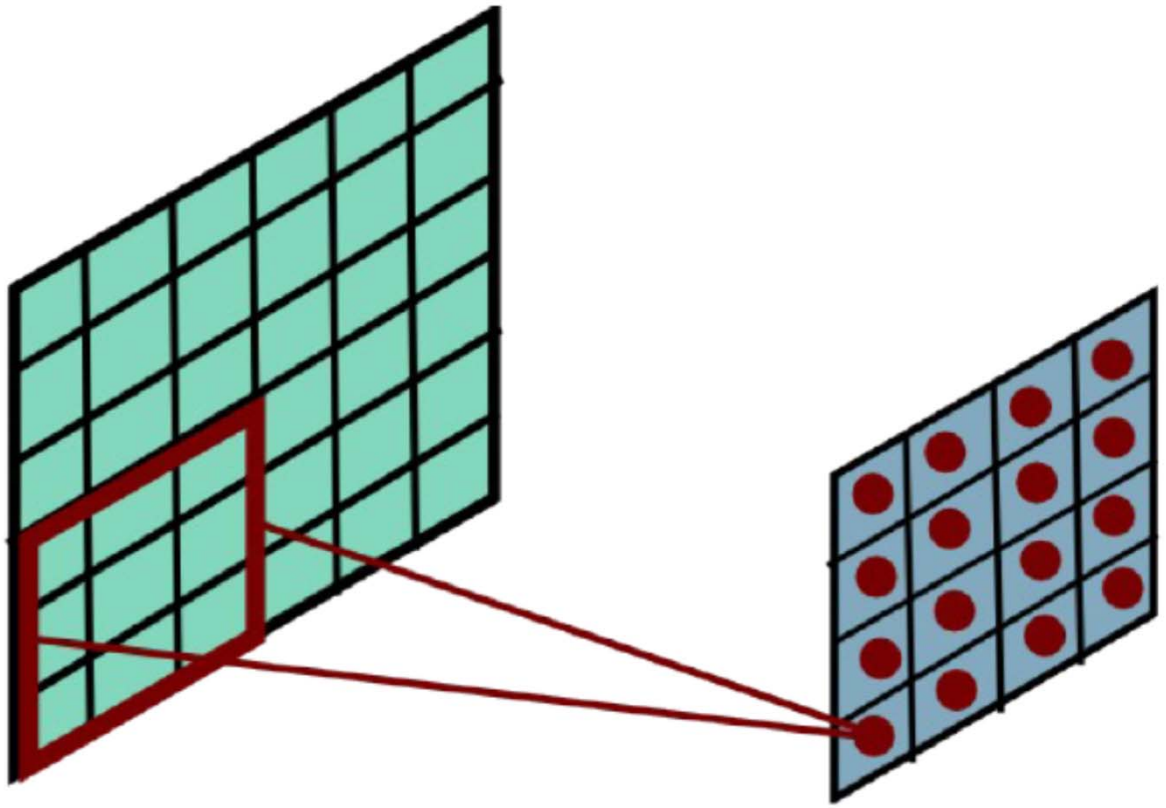
출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer



출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

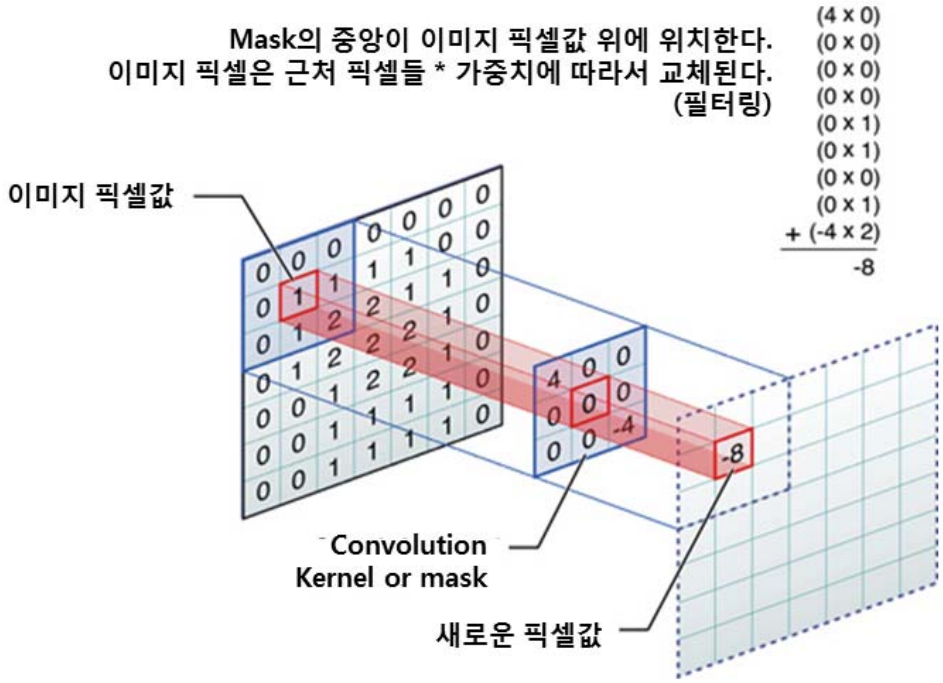
Convolution Layer

- Input volume of size [W1 x H1 x D1]
- using K neurons with receptive fields F x F
- and applying them at strides of S gives
- Output volume: [W2, H2, D2]
  W2 = (W1-F)/S+1,
  H2 = (H1-F)/S+1,
  D2 = K

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Convolution Layer

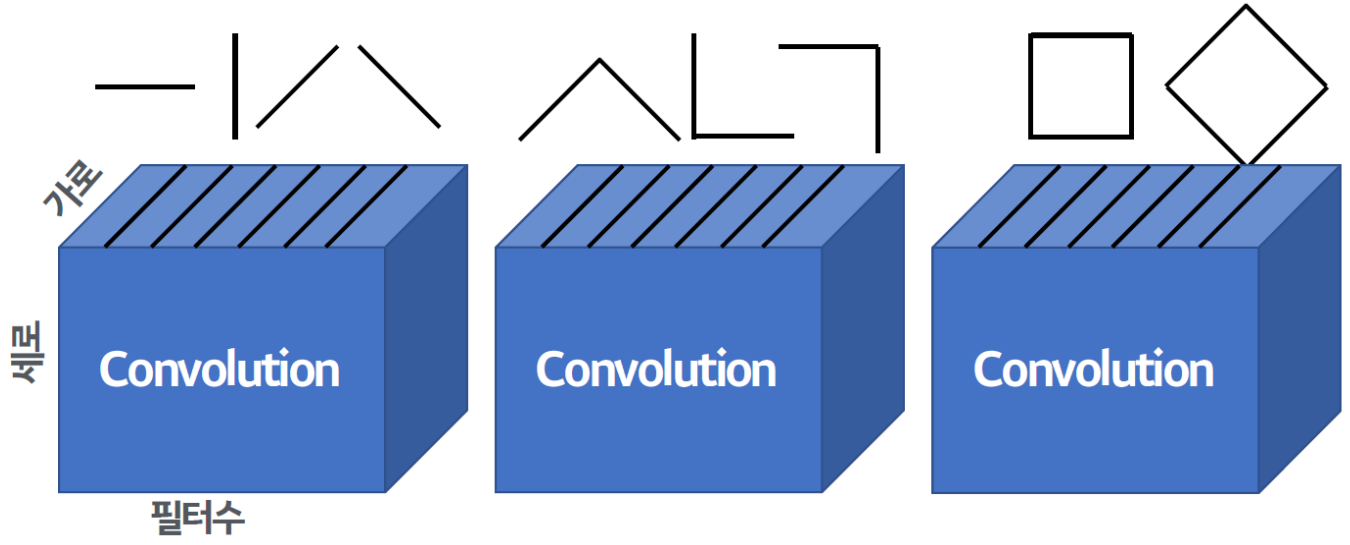
- 입력 이미지로부터 **필터**를 사용하여 **특징을 추출**
  - 특징을 추출하는 **전처리 단계가** 포함
    → CNN은 기존의 분류학습과의 차이



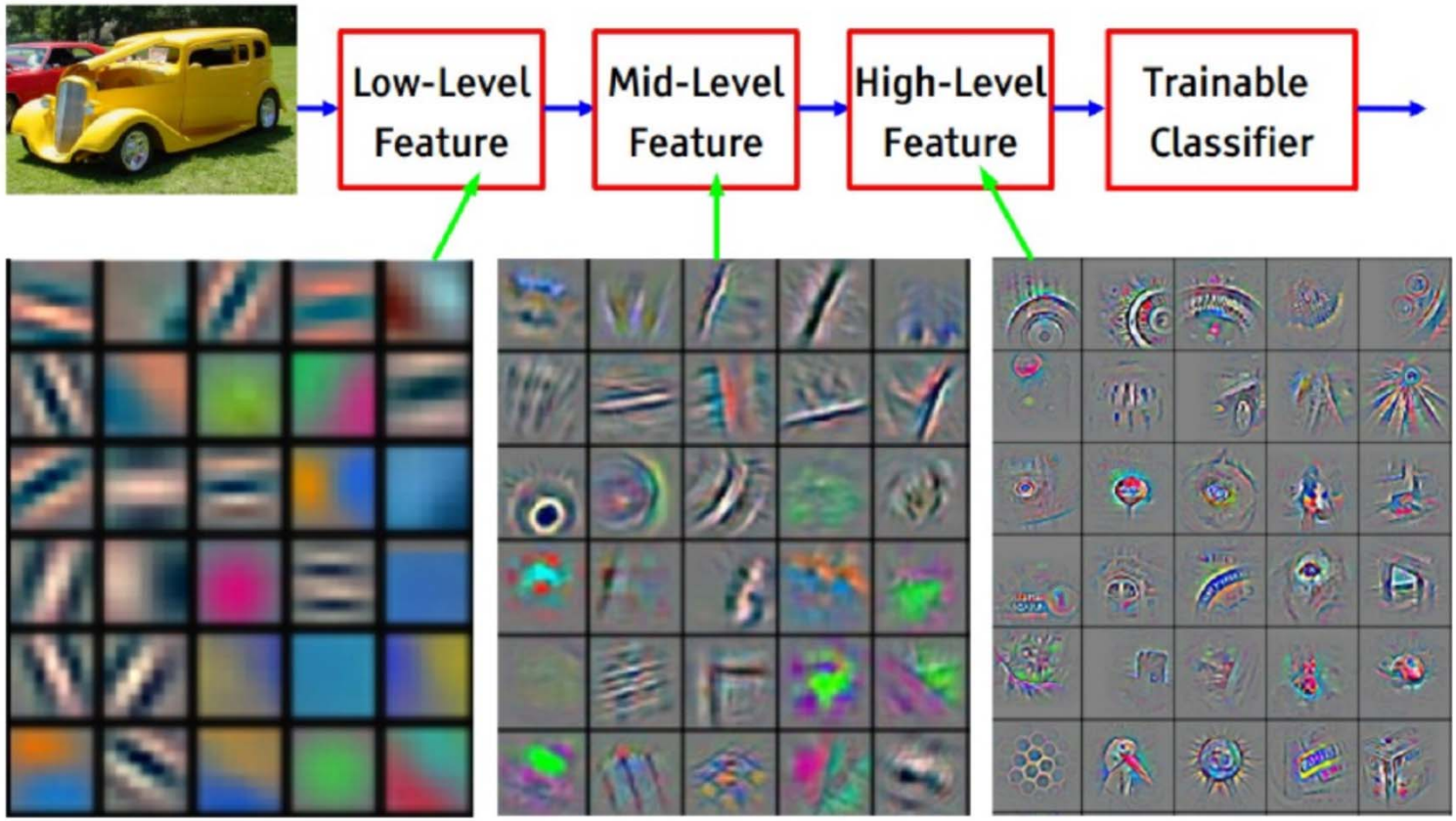
https://medium.com/qandastudy/mathpresso-%EB%A8%B8%EC%8B%A0-%EB%9F%AC%EB%8B%9D-%EC%8A%A4%ED%84%B0%EB%94%94-10-cnn-convolution-neural-nerwork-1-3e42fbb62a0c

Convolution Layer

- CNN은 간단한 필터들이 여러 layer에 거쳐서 쌓여가면서 복잡한 필터들을 만들게 되고, 이런 필터들은 인공신경망이 알아서 찾아준다.

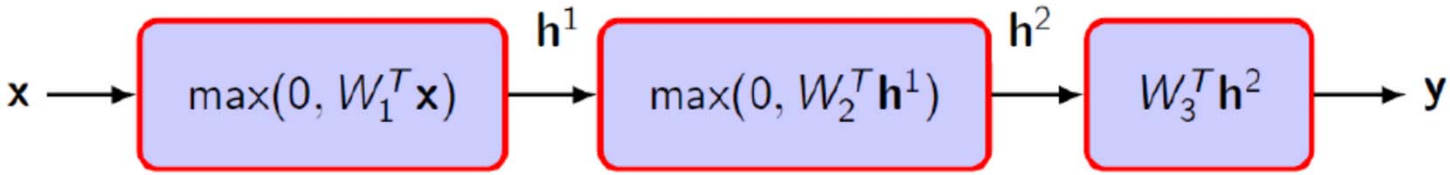


Feature (Filter) Visualization



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Training



- We want to estimate the parameters, biases and hyper-parameters (e.g., number of layers, number of neurons) for good predictions.
- Collect a training set of input-output pairs $\{\boldsymbol{x}_i, t_i\}_{i=1}^N$.
- Encode the output with 1-K encoding $t = [0, \cdots, 1, \cdots, 0]$.
- Define a loss per training example and minimize the empirical loss

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} l(\boldsymbol{w}, \boldsymbol{x}_i, t_i) + \mathcal{R}(\boldsymbol{w})$$

N: number of training examples
$\mathcal{R}$: regularizer
$\boldsymbol{w}$: set of all parameters

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Loss functions

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} l(\boldsymbol{w}, \boldsymbol{x}_i, t_i) + \mathcal{R}(\boldsymbol{w})$$

- Softmax (Probability of class k given input):

$$p(c_k = 1|\boldsymbol{x}) = \frac{\exp(y_k)}{\sum_{j=1}^{C}\exp(y_j)}$$

- Cross entropy (most popular loss function for classification):

$$l(\boldsymbol{w}, \boldsymbol{x}, t) = -\sum_{k=1}^{C} t^{(k)} \log p(c_k|\boldsymbol{x})$$

- Gradient descent to train the network

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{w})$$

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Learning with Gradient Descent

- Gradient descent to train the network

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} l(\boldsymbol{w}, \boldsymbol{x}_i, t_i) + \mathcal{R}(\boldsymbol{w})$$

- At each iteration, we need to compute

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \gamma_n \nabla \mathcal{L}(\boldsymbol{w}_n)$$

- Use the backward pass to compute $\nabla \mathcal{L}(\boldsymbol{w}_n)$ efficiently
- Recall that the backward pass requires the forward pass first

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Dealing with Big Data

- At each iteration, we need to compute

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \gamma_n \nabla \mathcal{L}(\boldsymbol{w}_n)$$

with

$$\nabla \mathcal{L}(\boldsymbol{w}_n) = \frac{1}{N} \sum_{i=1}^{N} \nabla l(\boldsymbol{w}_n, \boldsymbol{x}_i, t_i) + \nabla \mathcal{R}(\boldsymbol{w}_n)$$

- Too expensive when having millions of training examples
- Instead, approximate the gradient with a mini-batch (subset of examples: 100 $\sim$ 1,000) - called stochastic gradient descent

$$\frac{1}{N} \sum_{i=1}^{N} \nabla l(\boldsymbol{w}_n, \boldsymbol{x}_i, t_i) \approx \frac{1}{|S|} \sum_{i \in S} \nabla l(\boldsymbol{w}_n, \boldsymbol{x}_i, t_i)$$

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Stochastic Gradient Descent with momentum

- Stochastic Gradient Descent update

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \gamma_n \nabla \mathcal{L}(\boldsymbol{w}_n)$$

with

$$\nabla \mathcal{L}(\boldsymbol{w}_n) = \frac{1}{|S|} \sum_{i \in S} \nabla l(\boldsymbol{w}_n, \boldsymbol{x}_i, t_i)$$

- We can use momentum

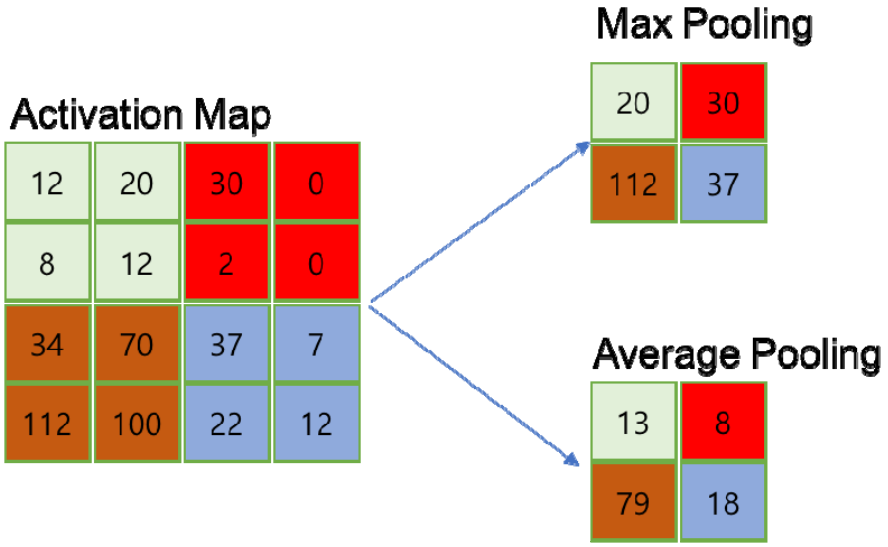
$$\boldsymbol{w} \longleftarrow \boldsymbol{w} - \gamma \Delta$$
$$\Delta \longleftarrow \kappa \Delta + \nabla \mathcal{L}$$

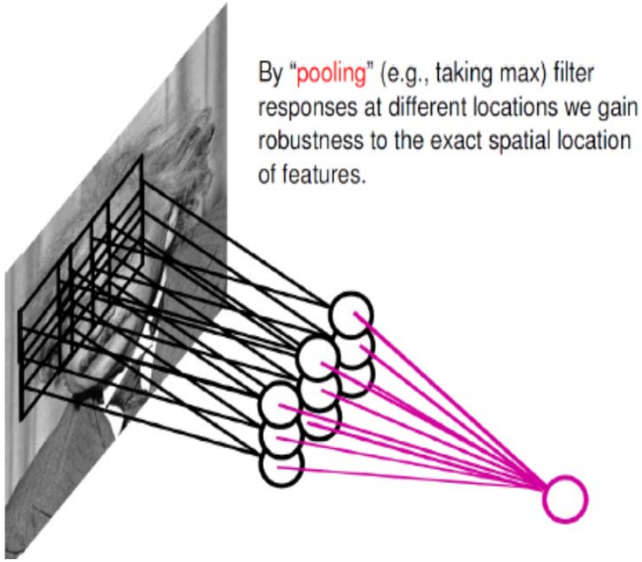
- We can also decay learning rate γ as iterations goes on

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

Pooling Layer

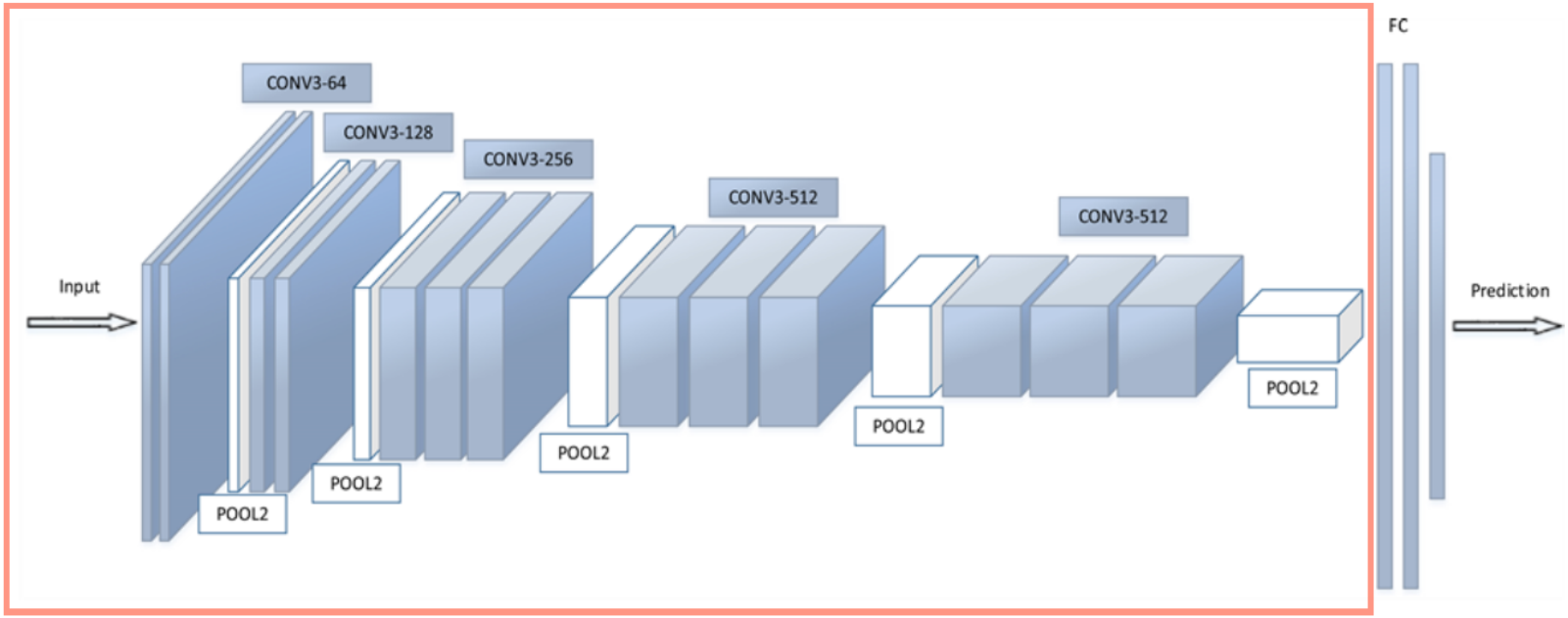
- Pooling Layer는 Convolution layer의 출력(Activation Map)을 입력으로 받아 activation map의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용
  - Max Pooling, Average Pooling, Min Pooling





Feature Extraction

패턴들을 쌓아가며 점차 복잡한 패턴을 인식(Conv)

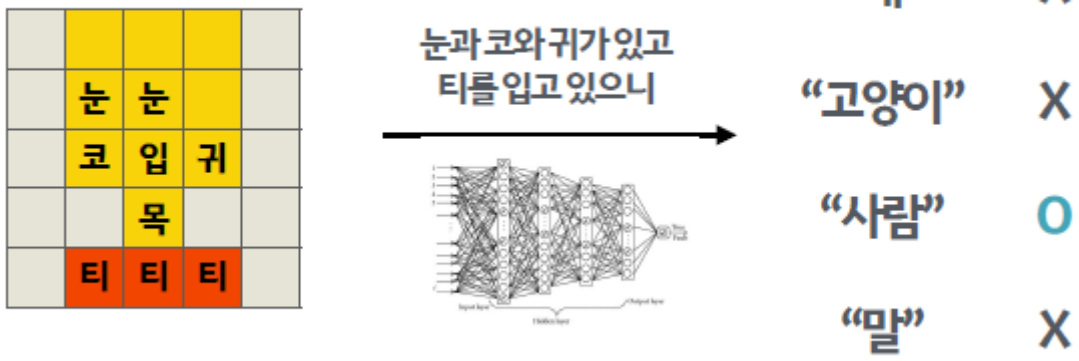


사이즈를 줄어가며, 더욱 추상화(Pooling)

해당 과정을 거치면서 입력 데이터로부터 Feature들이 추출

Fully Connected Layer

- Convolutional layer에서 추출된 특징을 기존의 인공 신경망에 넣어서 분류
  - ‘Fully Connected’ : ‘Vectorization’ → 입력으로 받은 데이터를 1열로 쭉 나열
- 보통 Fully Connected layer이후에 Softmax를 activation으로 사용
  - 입력 데이터가 각 클래스에 속할 확률 계산



Improving Generalization

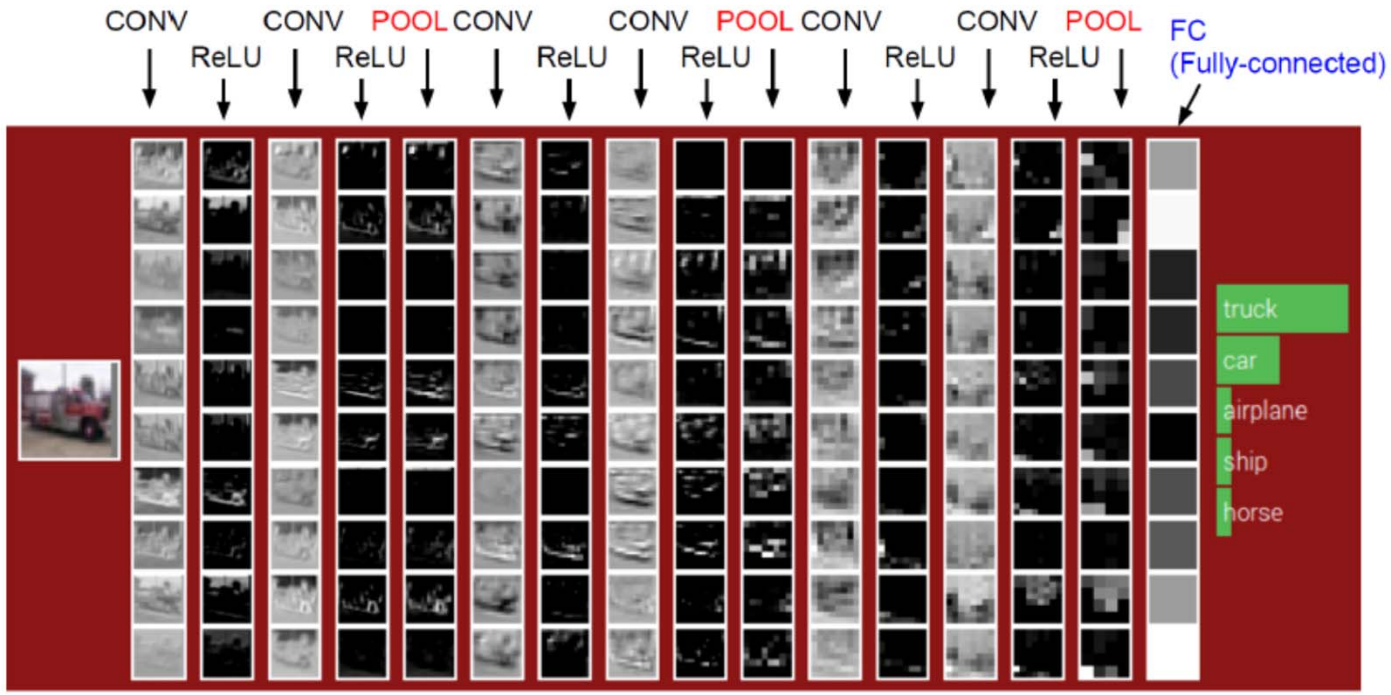
- Weight sharing (Reduce the number of parameters)
- Data augmentation (e.g., jittering, noise injection, tranformations)
- Dropout [Hinton et al.]: randomly drop units (along with their connections) from the neural network during training. Use for the fully connected layers only
- Regularization: Weight decay (L2, L1)
- Sparsity in the hidden units
- Multi-task learning
- Transfer learning

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

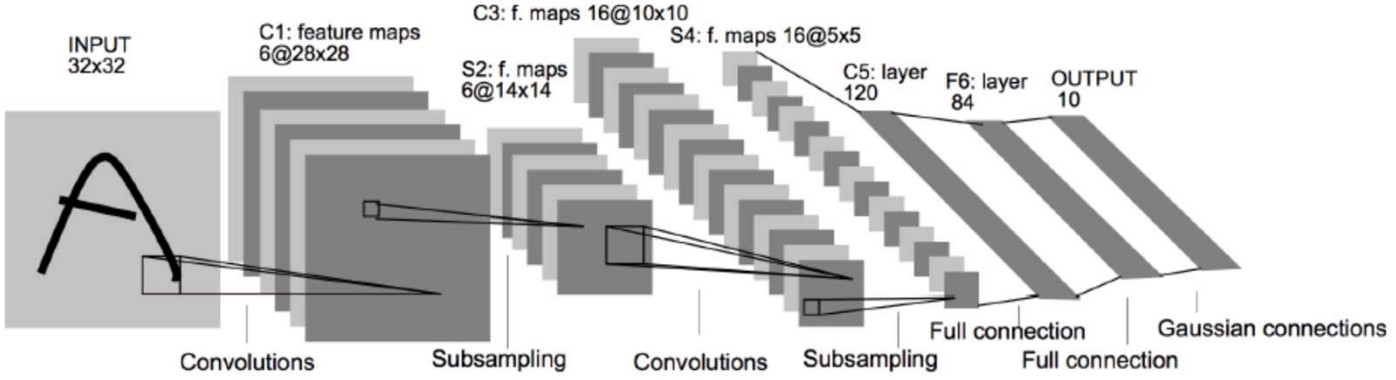
ConvNets

Typical ConvNet:

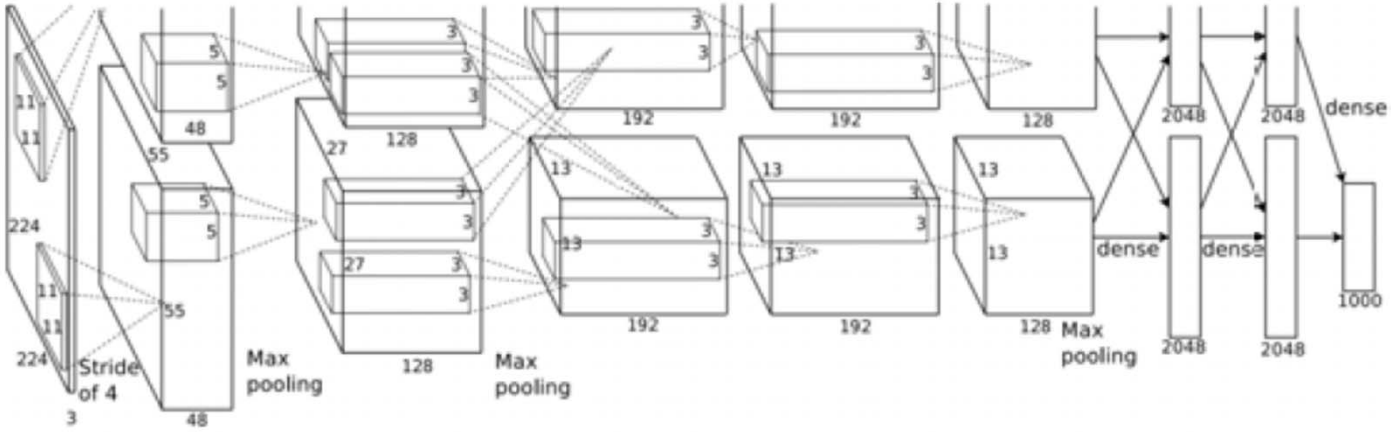
Image → [Conv - ReLU] → (Pool) → [Conv - ReLU] → (Pool) →
FC (fully-connected) → Softmax



ConvNets - Flavours

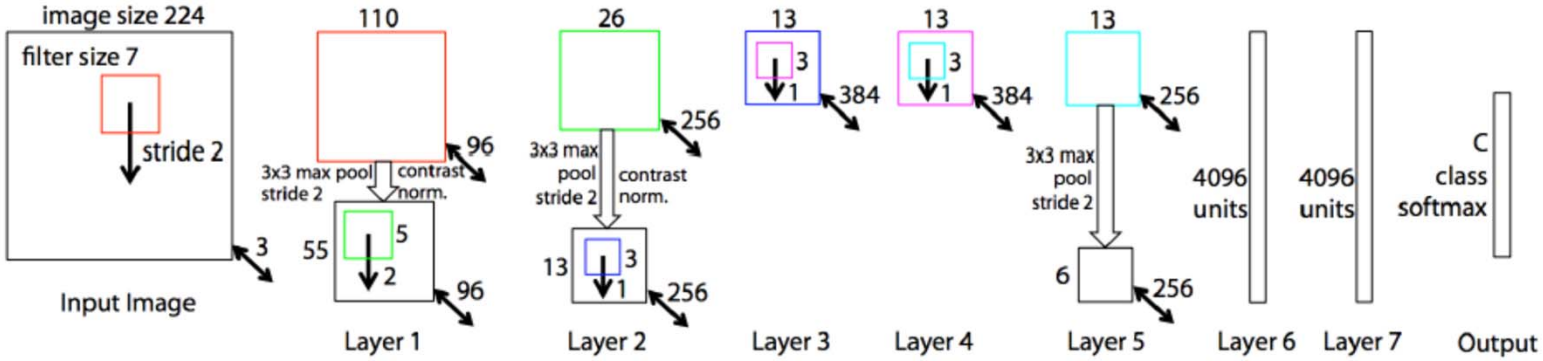


Lenet5 (Yann Lecun 1998)

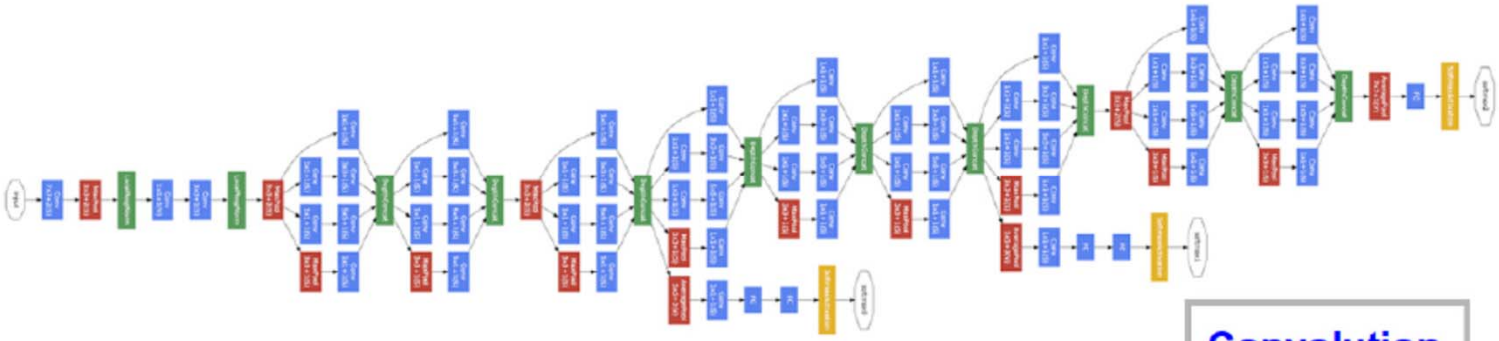


Alexnet (Alex Krizhevsky et. al., 2012)

ConvNets - Flavours



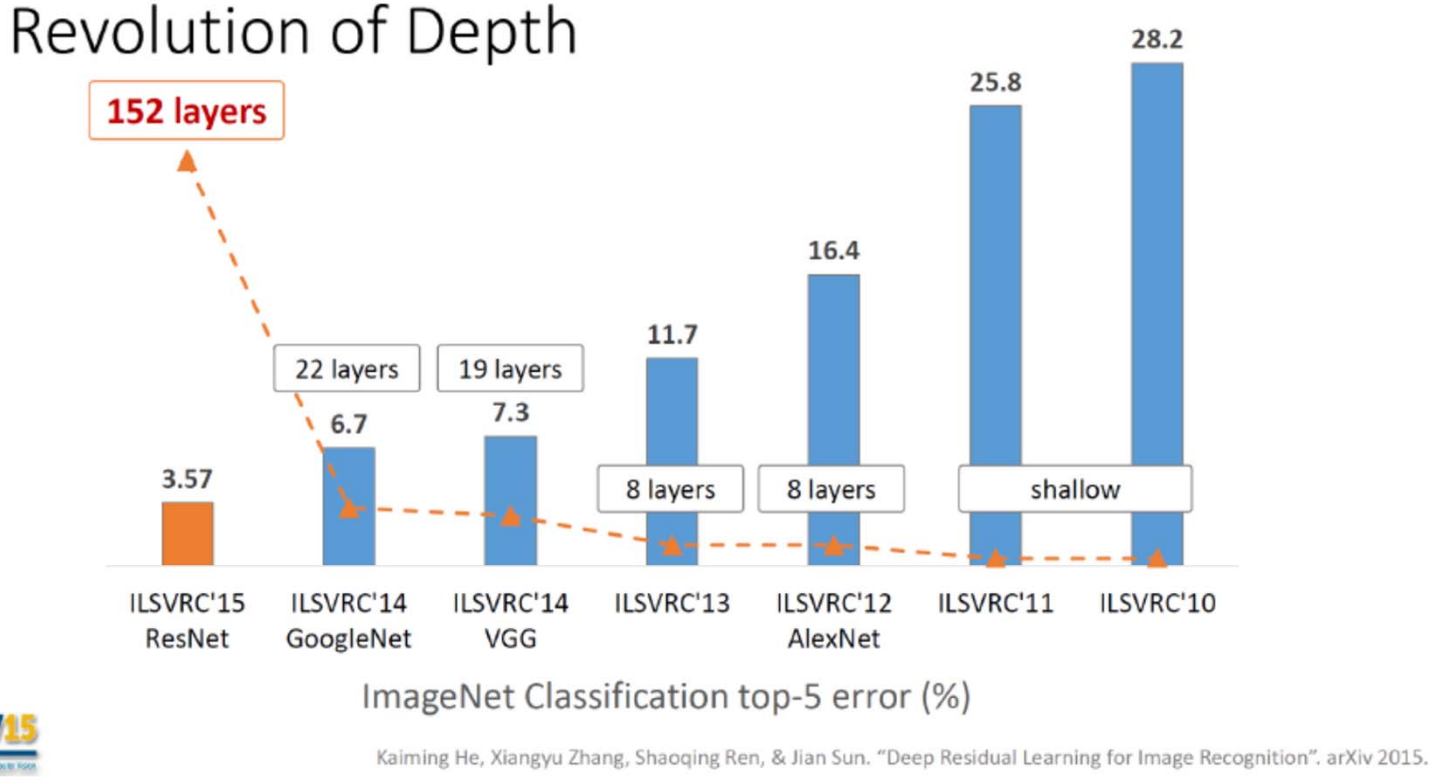
ZFnet: Clarifai (Matt Zeiler and Rob Fergus, 2013)



Googlenet (Google, 2014)

Convolution
Pooling
Softmax
Other

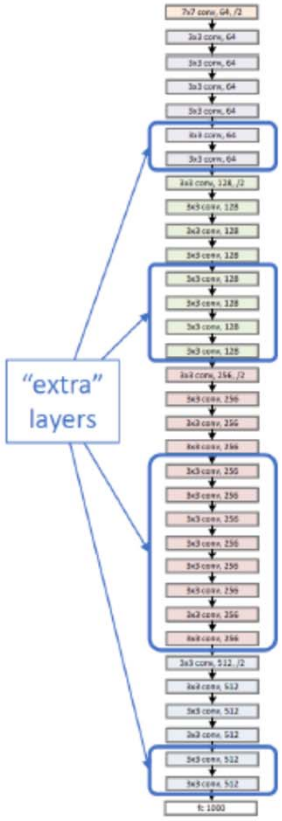
Recent Trend



- Human: 5.1% (Karpathy), Baidu cheating (2015.05) - 4.58%

Recent Trend

a shallower model (18 layers)



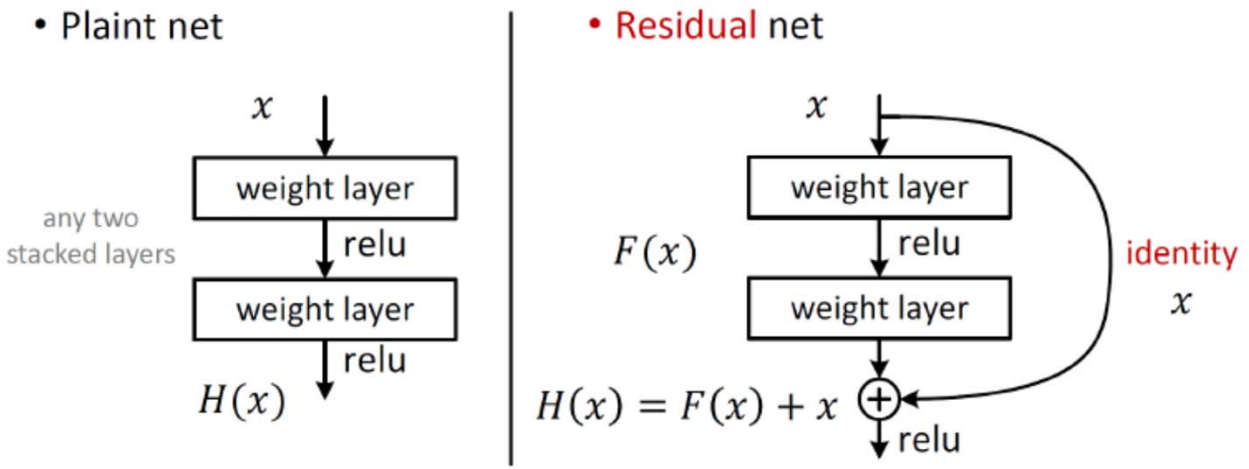
a deeper counterpart (34 layers)

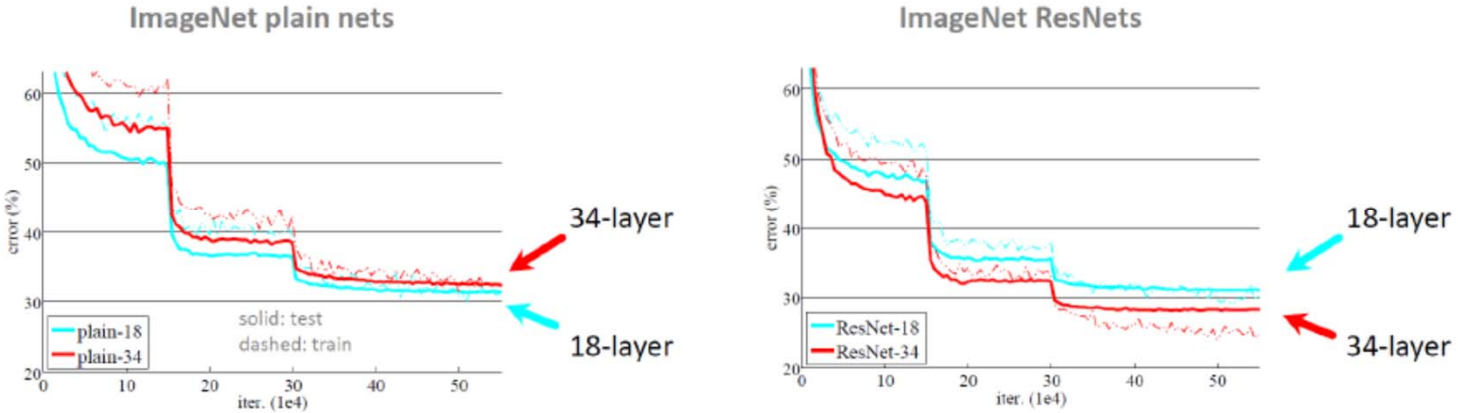
Microsoft Research

- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Recent Trend





- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

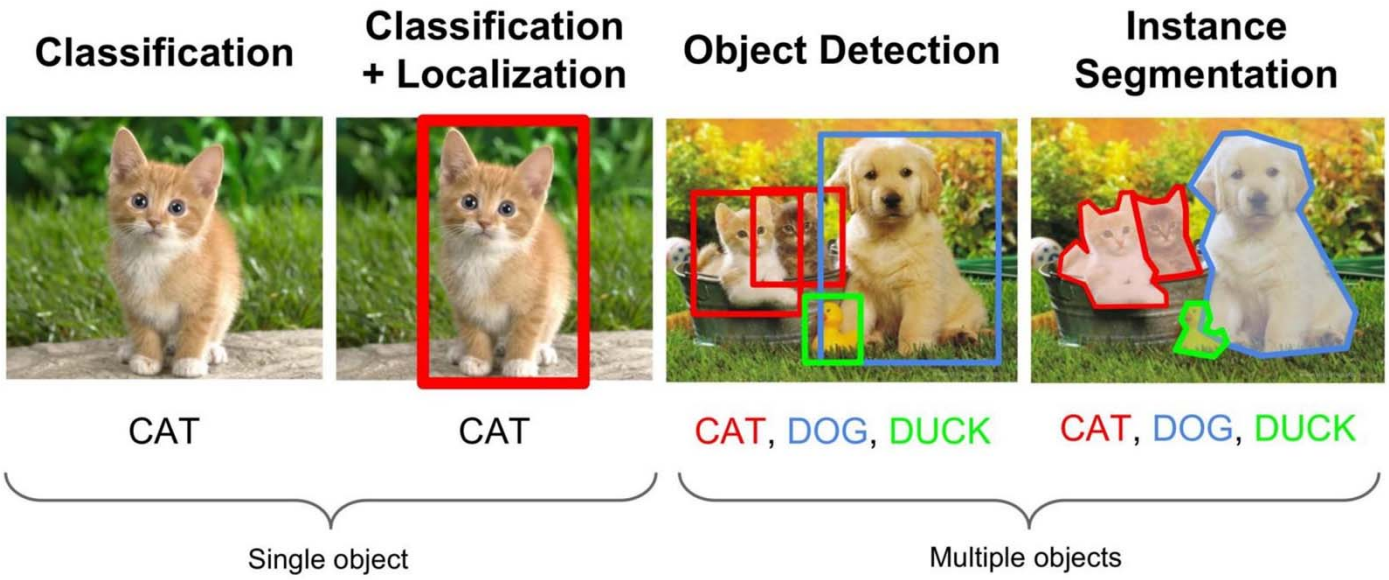
Conclusions - CNNs

- Deep Learning = learning hierarhical models.
- ConvNets are the most successful example. Leverage large labeled datasets.
- Optimization
  - Don't we get stuck in local minima? No, they are all the same!
  - In large scale applications, local minima are even less of an issue.
- Scaling
  - GPUs
  - Distributed framework (Google)
  - Better optimization techniques
- Generalization on small datasets (curse of dimensionality):
  - data augmentation
  - weight decay
  - dropout
  - unsupervised learning
  - multi-task learning

출처 : Prof. Nojun Kwak, Introduction to CNNs and RNNs, http://mipal.snu.ac.kr

What is Multi Object Detection?

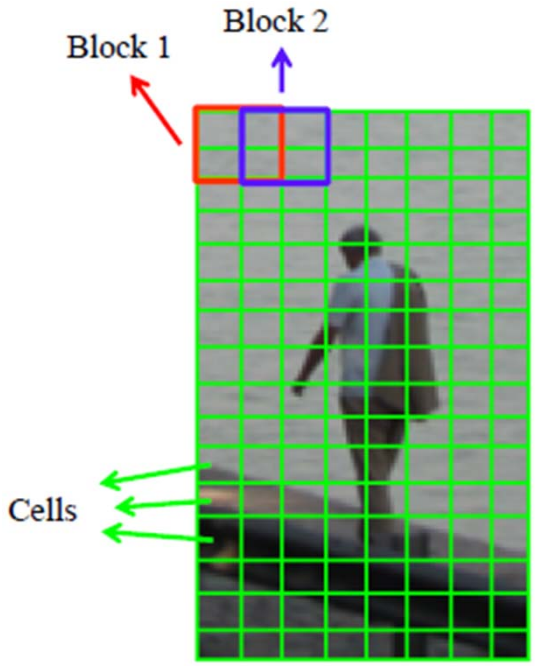
- Multi Object Detection



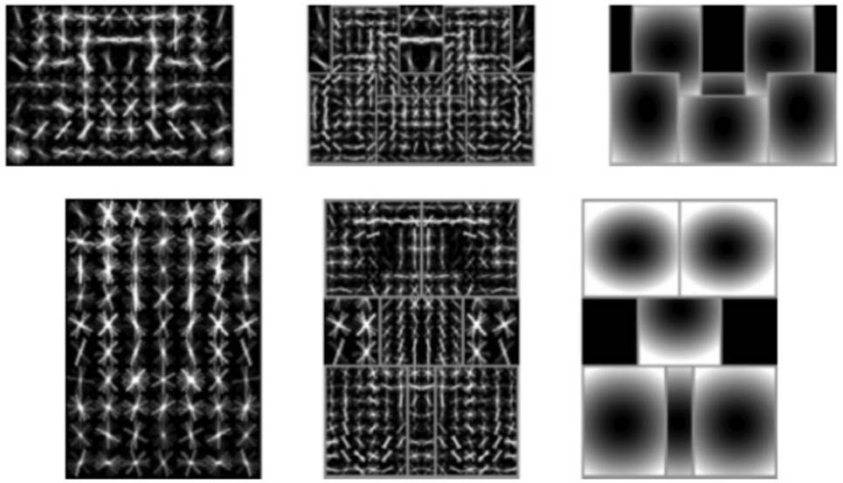
What is Multi Object Detection?

- Before Deep Learning

sliding Windows



Deformable part models(DPM)



root filters
coarse resolution

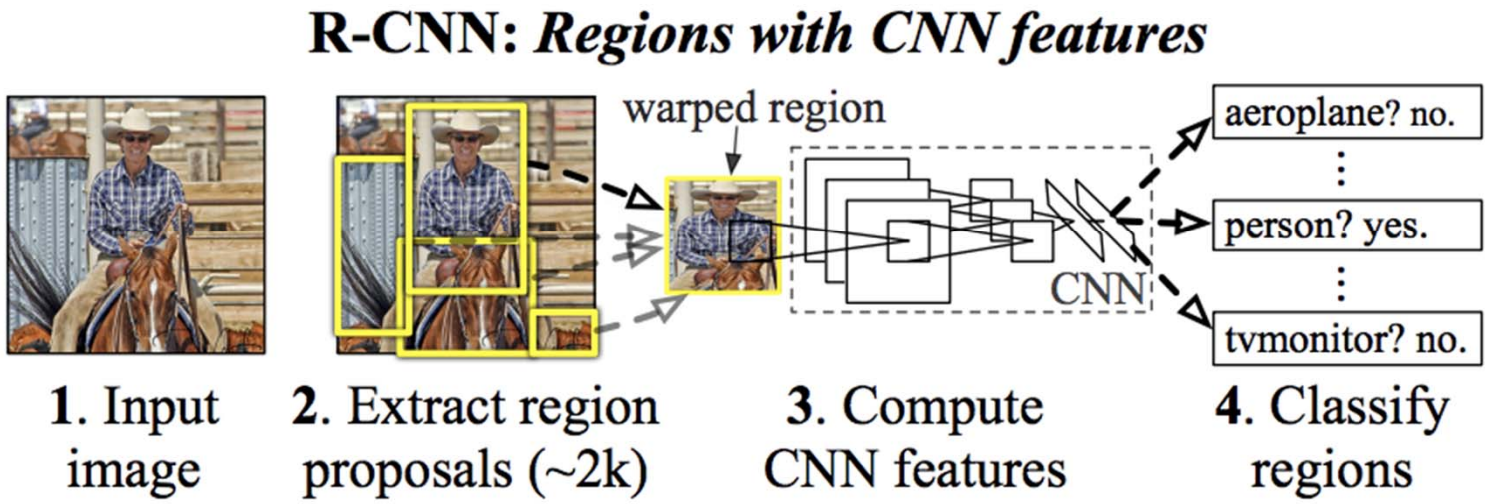
part filters
finer resolution

deformation
models

R-CNN

- Regions with CNN features(R-CNN)
- Region proposal + CNN
- Use selective search to come up with regional proposal
- First object detection method using CNN

R-CNN

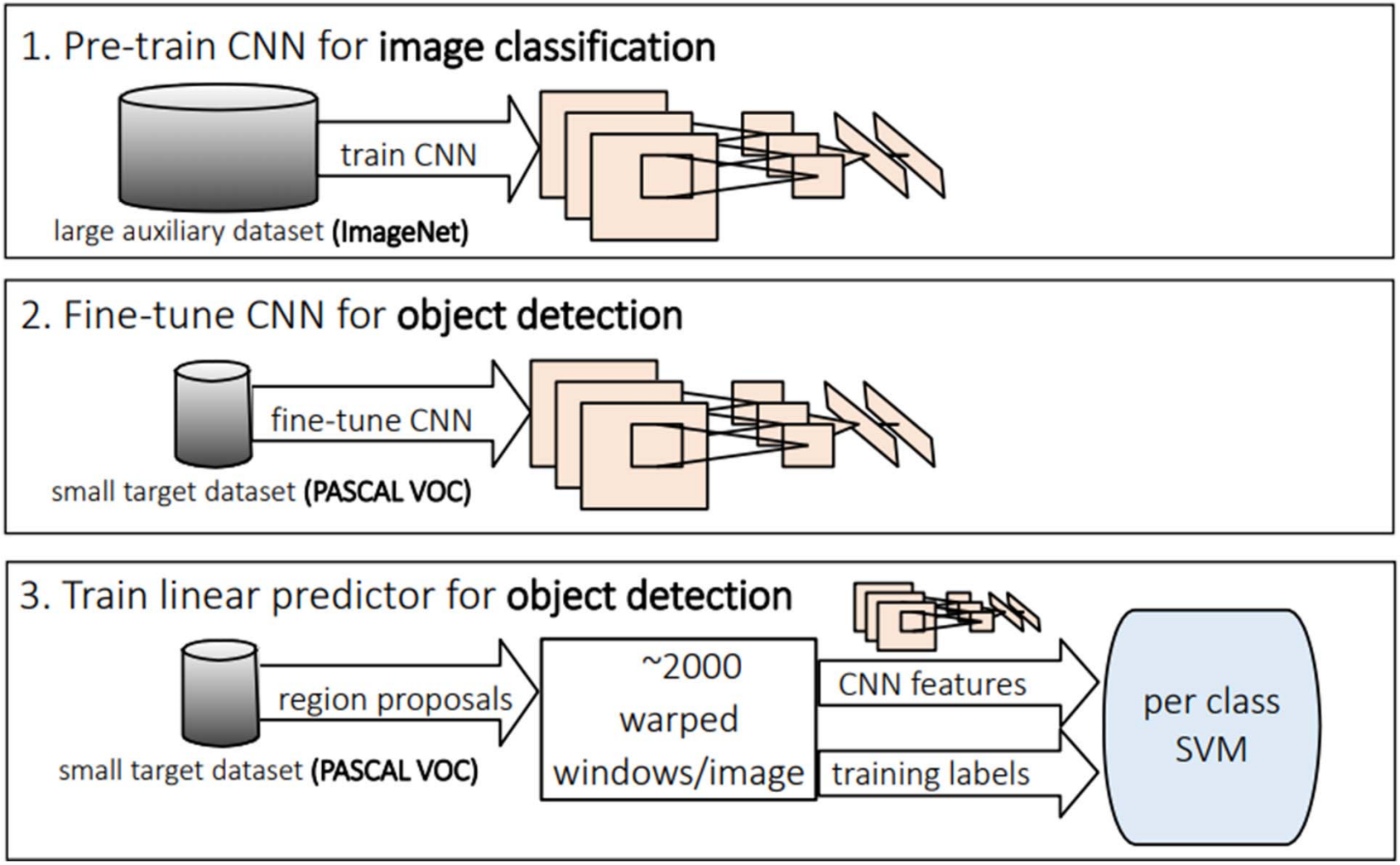


- (데이터 부족으로 인해) 우선 Imagenet dataset을 이용해 모델을 classification 문제에 대하여 pretraining

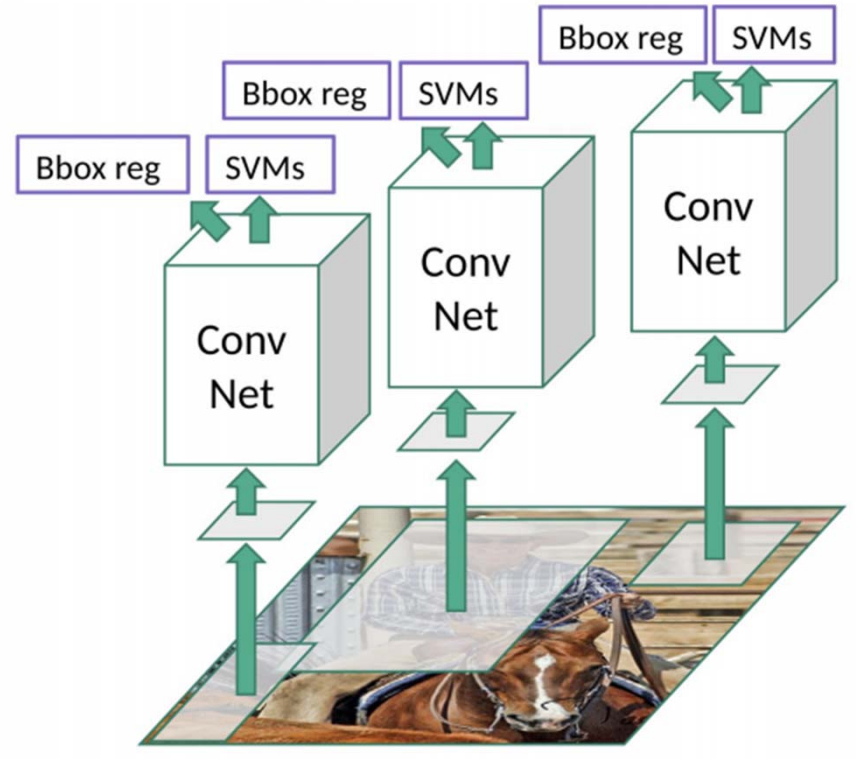
- Object detection 목적으로 만들어진 (상대적으로 작은 dataset인) PASCAL VOC 를 이용해 모델을 fine-tune

- Region proposa로 얻은 작은 이미지 패치들을 모델에 넣어 얻은 CNN feature과 training label을 이용하여 SVM을 학습

R-CNN



R-CNN



Fast R-CNN

- Problems of R-CNN

1. Training is a multi-stage pipeline.

- R-CNN first finetunes a ConvNet on object proposals using log loss
- Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning.
- In the third training stage, bounding-box regressors are learned.

2. Training is expensive in space and time.

- For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk.
  : With very deep networks, such as VGG16, takes 2.5 GPU-days for the 5k images of the VOC07 train/val set
  : requires hundreds of gigabytes of storage

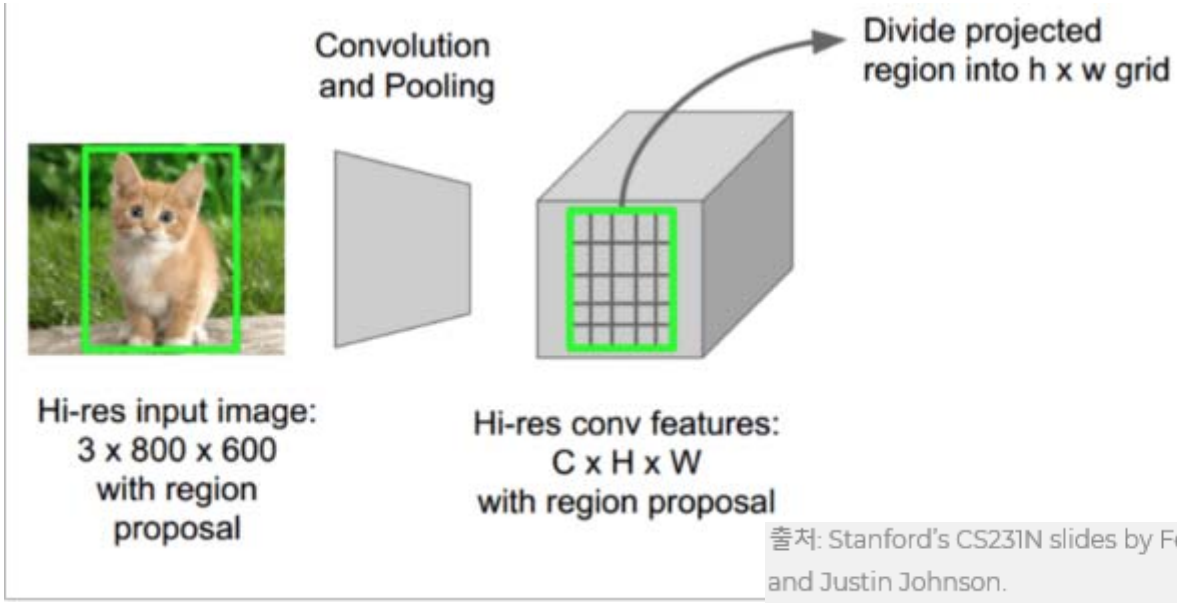
3. Object detection is slow.

- At test-time, features are extracted from each object proposal in each test image → Detection with VGG16 takes 47s / image (on a GPU)

Fast R-CNN

- Key Idea 1
  - CNN을 이미지에 한 번만 수행
  - 중복되는 영역(proposal)에 대해, 계산된 값들을 공유

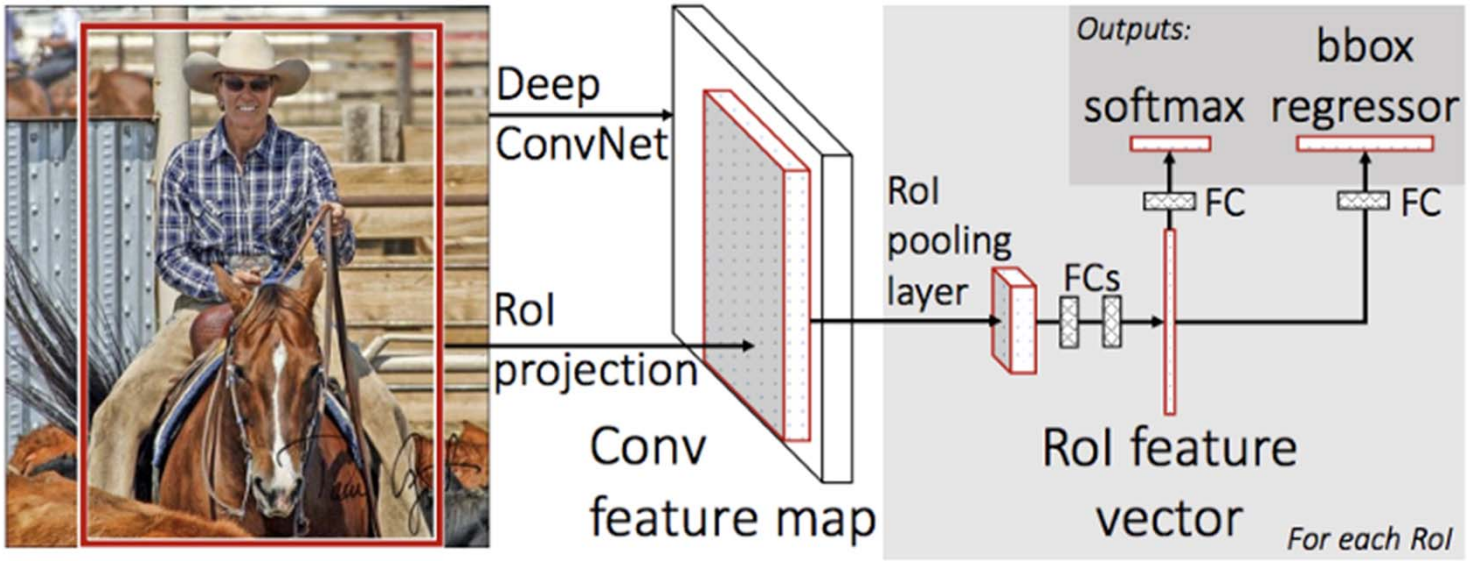
- RoIPool (Region of Interest Pooling)
  - 한 이미지의 subregion에 대한 forward pass값을 서로 공유
  - Selective search 에서 찾은 bounding box 정보들을 CNN을 통과하면서 유지시키고 최종 CNN 특징 맵으로부터 해당 영역을 추출하여 풀링



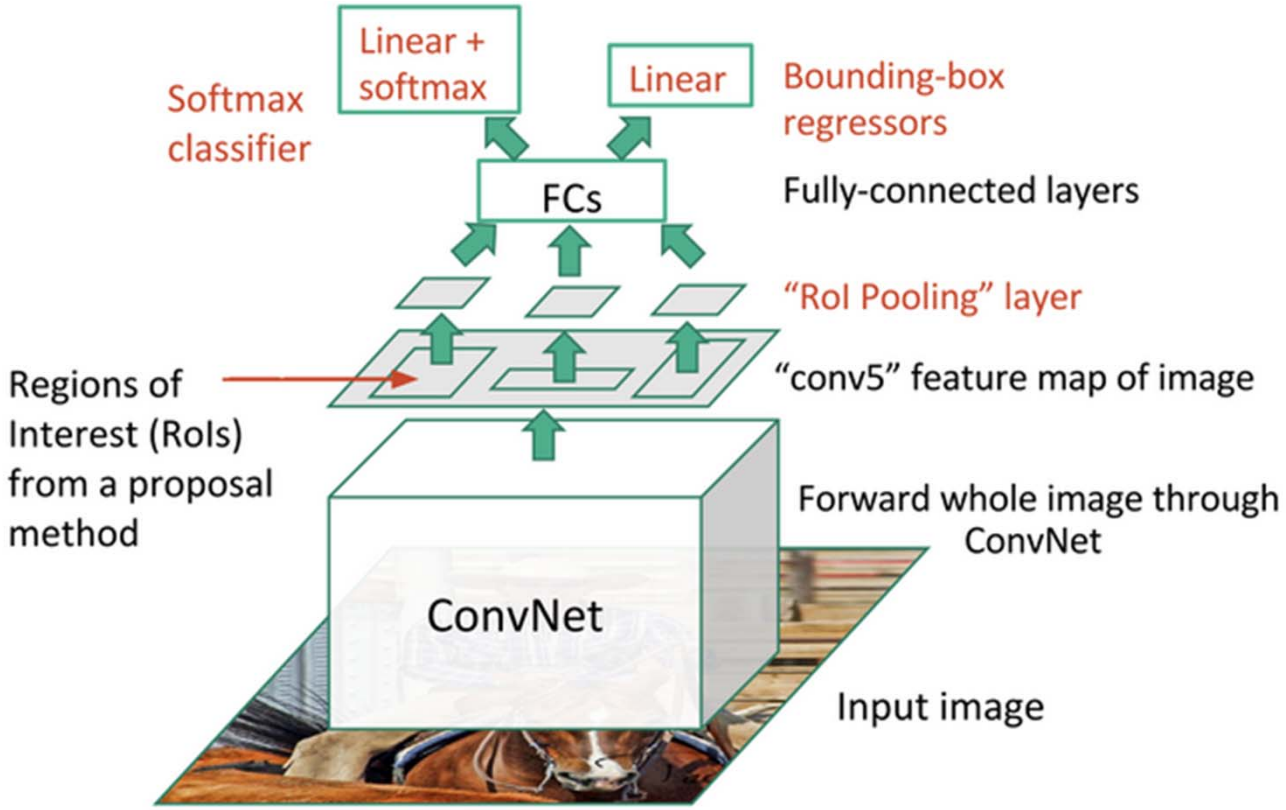
출처: Stanford's CS231N slides by Fei Fei Li, Andrei Karpathy, and Justin Johnson.

Fast R-CNN

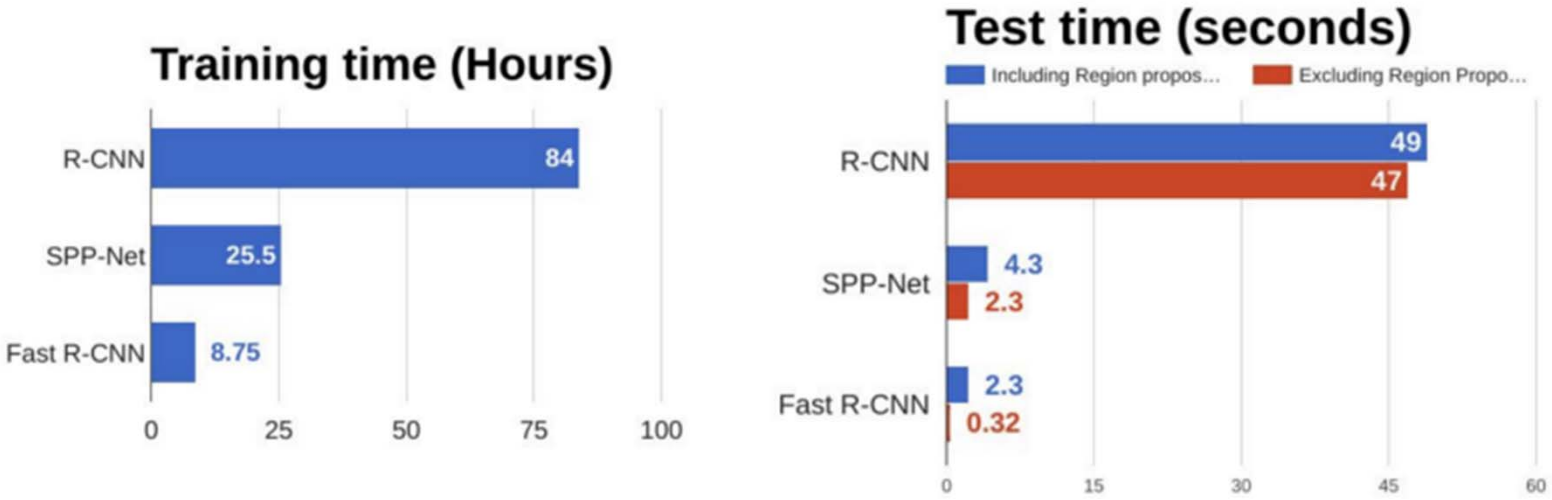
- Key Idea 2
  - 세 개의 모델(CNN+SVN+Regressor)을 하나의 네트워크로 구성
    - SVM의 classifier대신 top layer에 softmax layer로 배치 → CNN의 결과를 class로 출력
    - Box regression layer를 softmax layer와 평행하게 배치 → bounding box 출력



Fast R-CNN

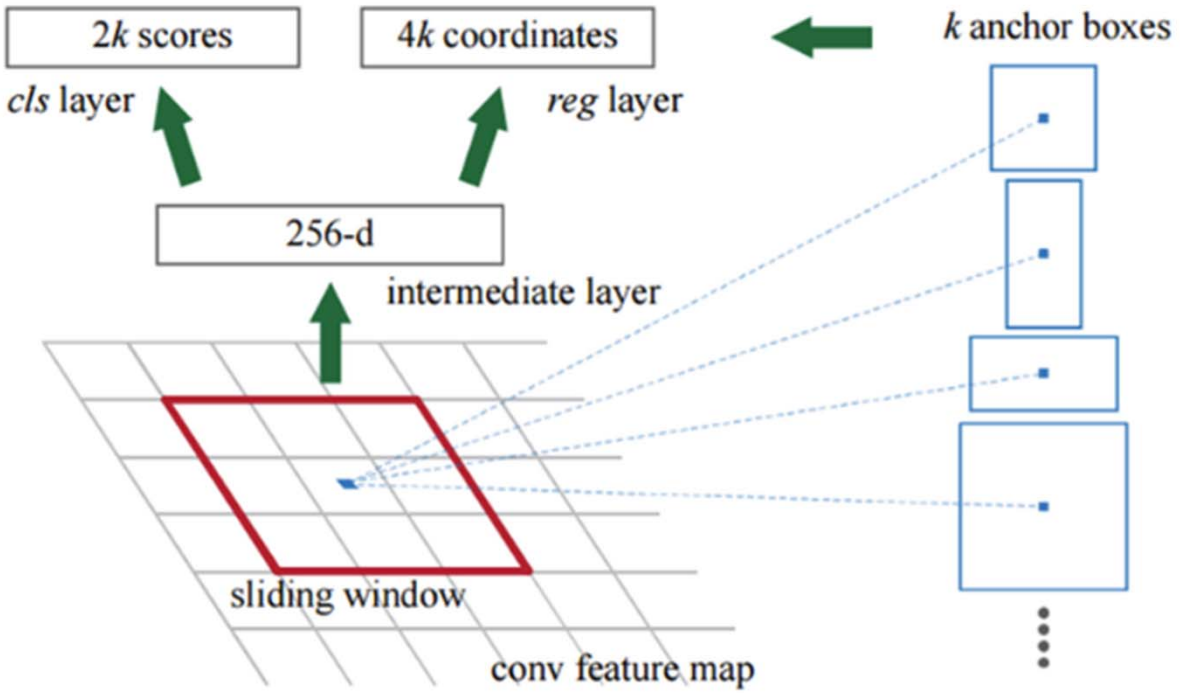


Fast R-CNN

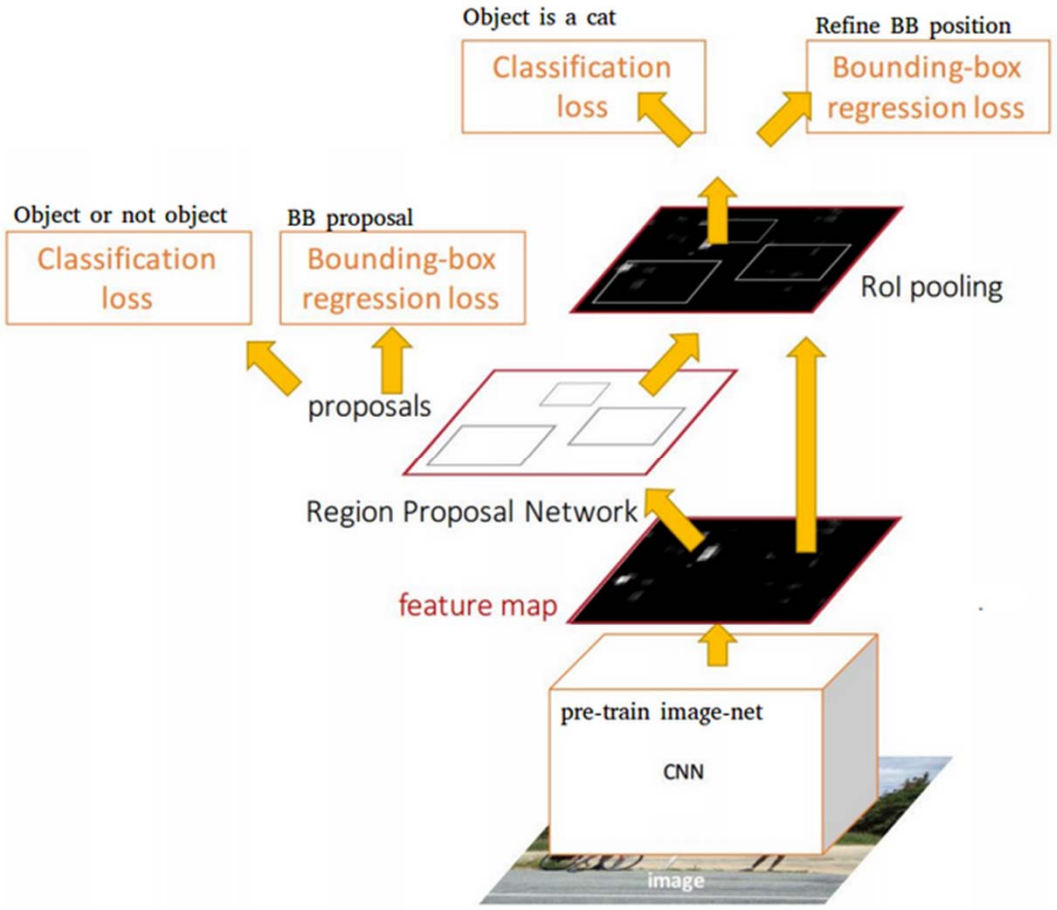


Faster R-CNN

- Faster R-CNN
- Don’t need to have external regional proposals
- Using RPN - Regional Proposal Network



- Faster R-CNN



- Test-Time Speed