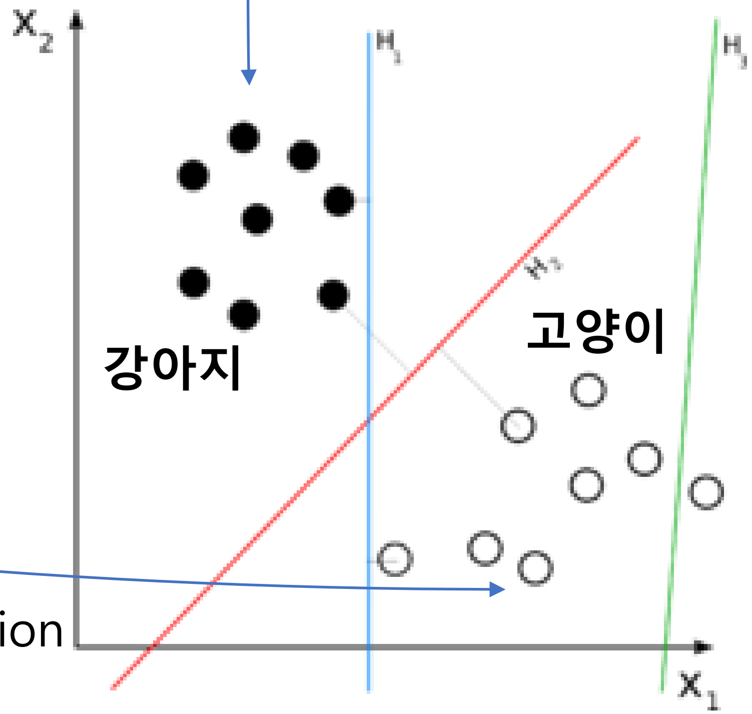
CV Study – Week 1

Neural Network & Convolutional Network

Linear Classifier

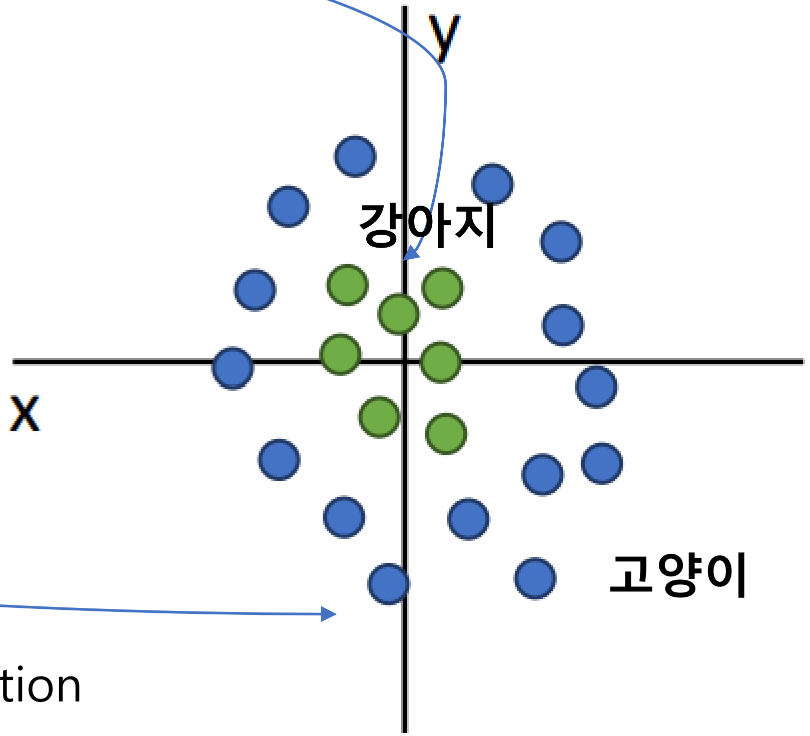
Feature extraction



Feature extraction

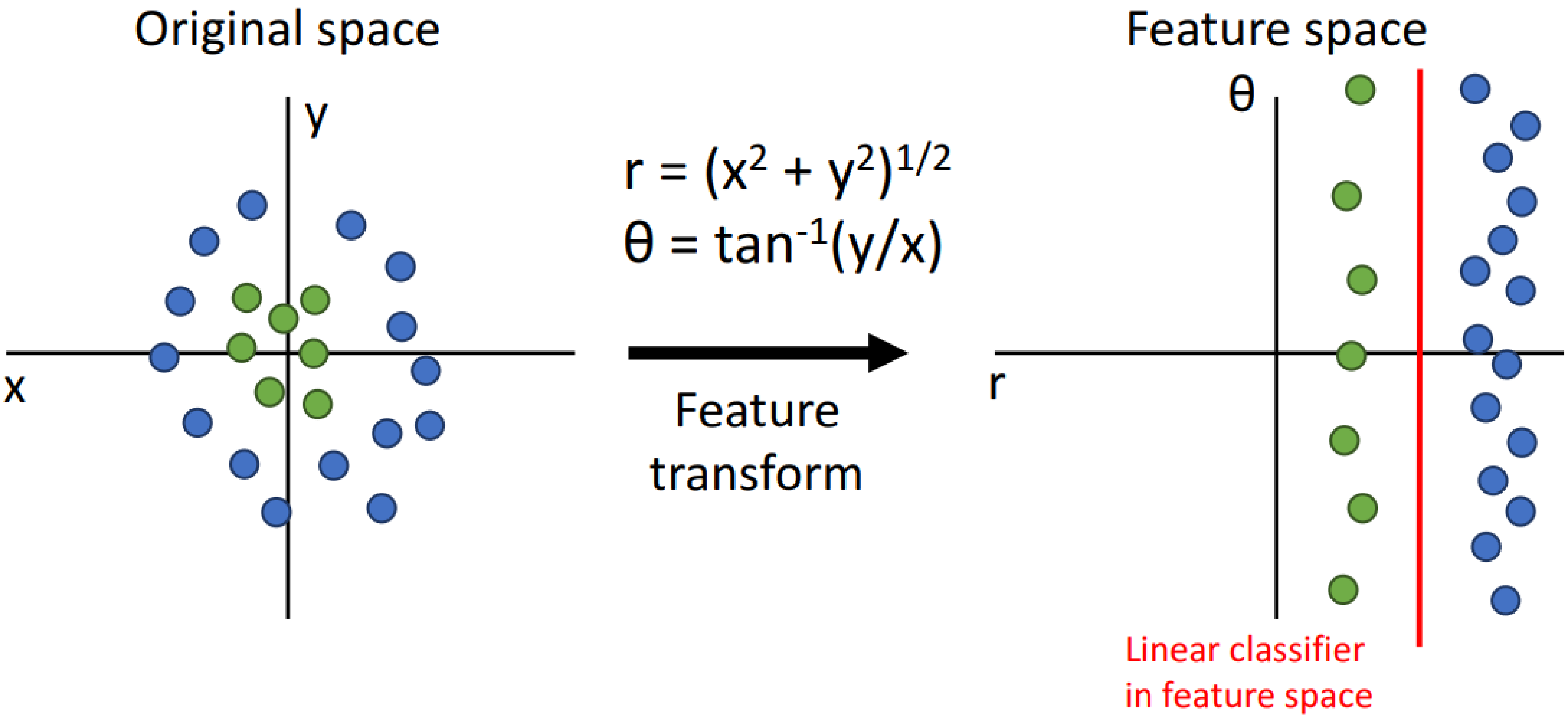
Linear Classifier

Feature extraction

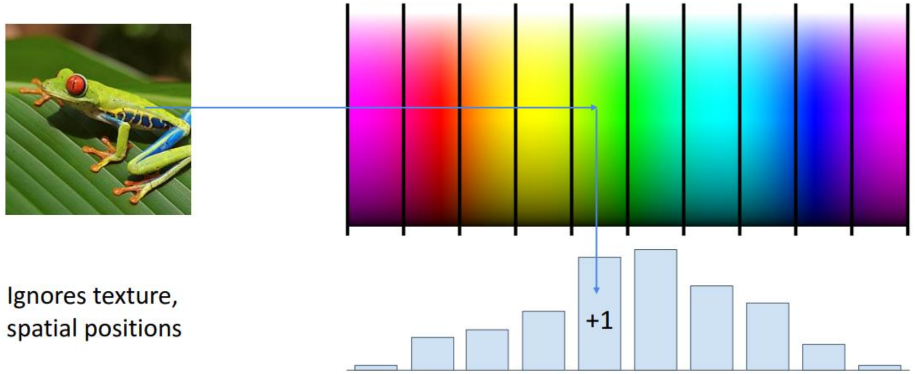


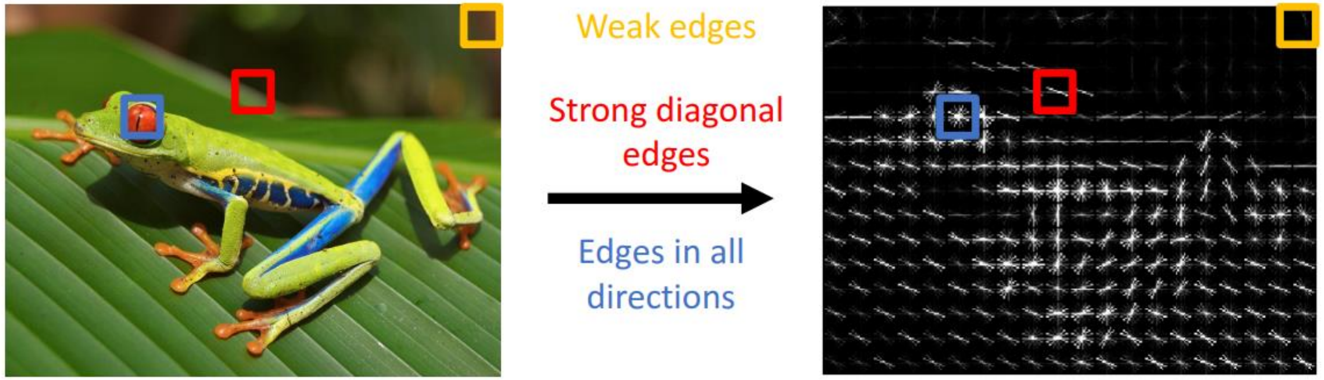
Feature extraction

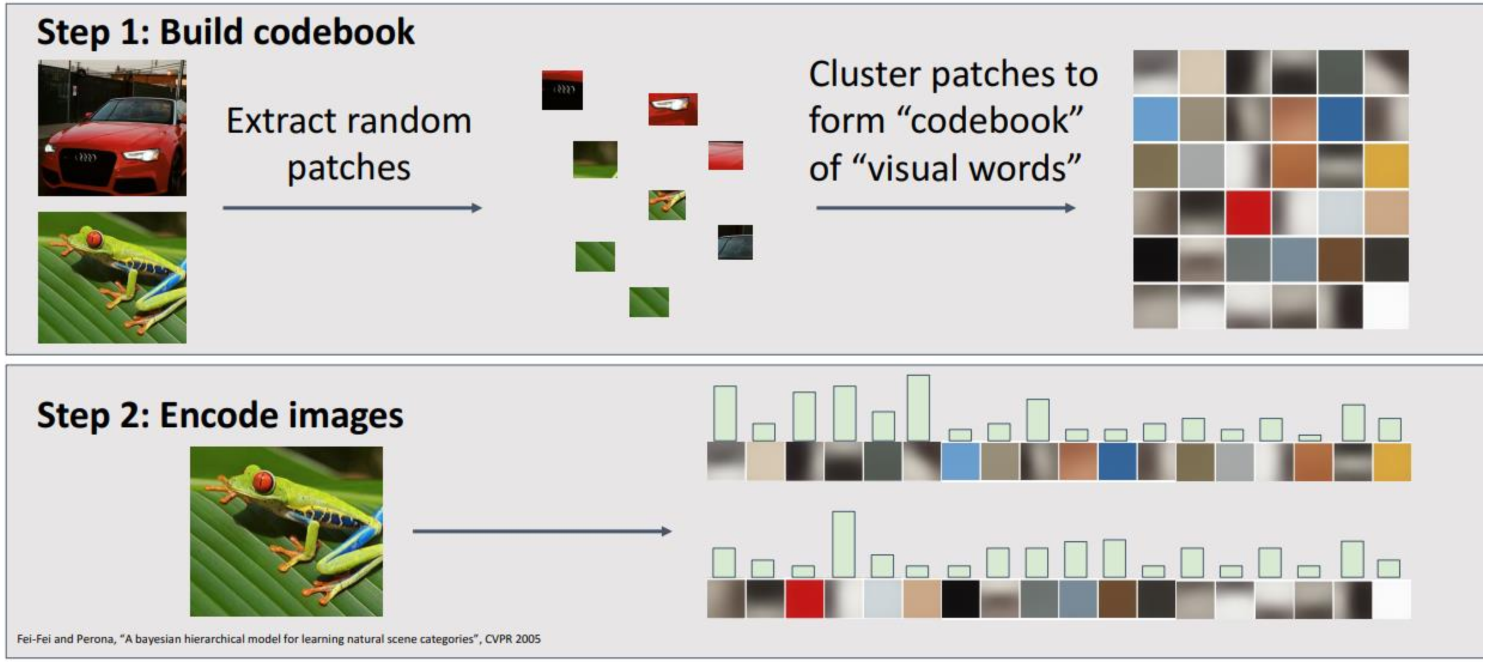
Linear Classifier



Before Neural Network





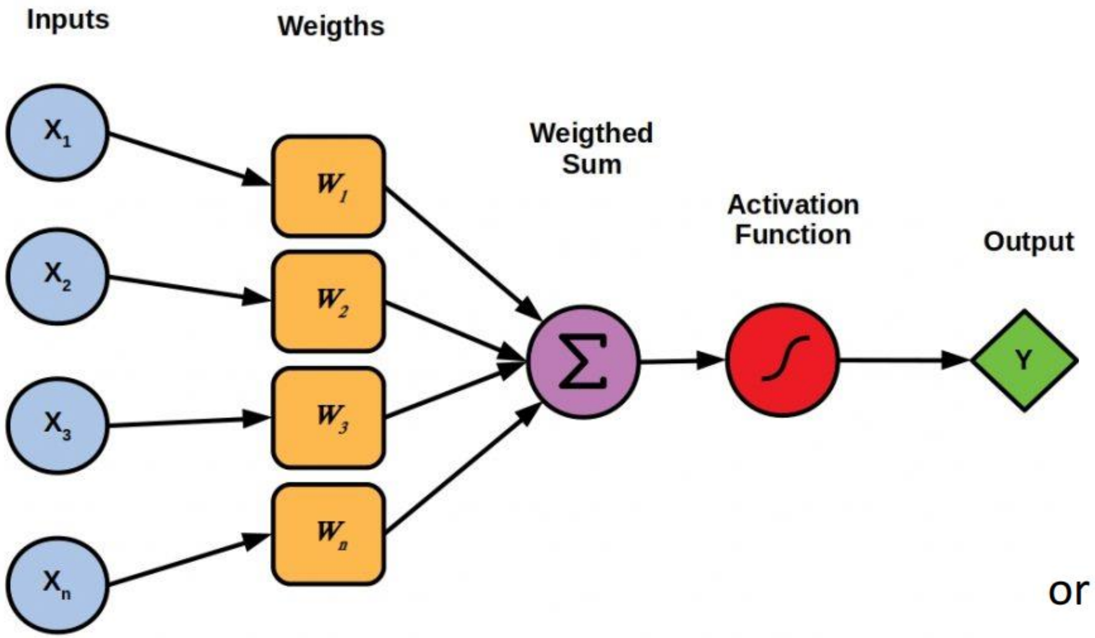


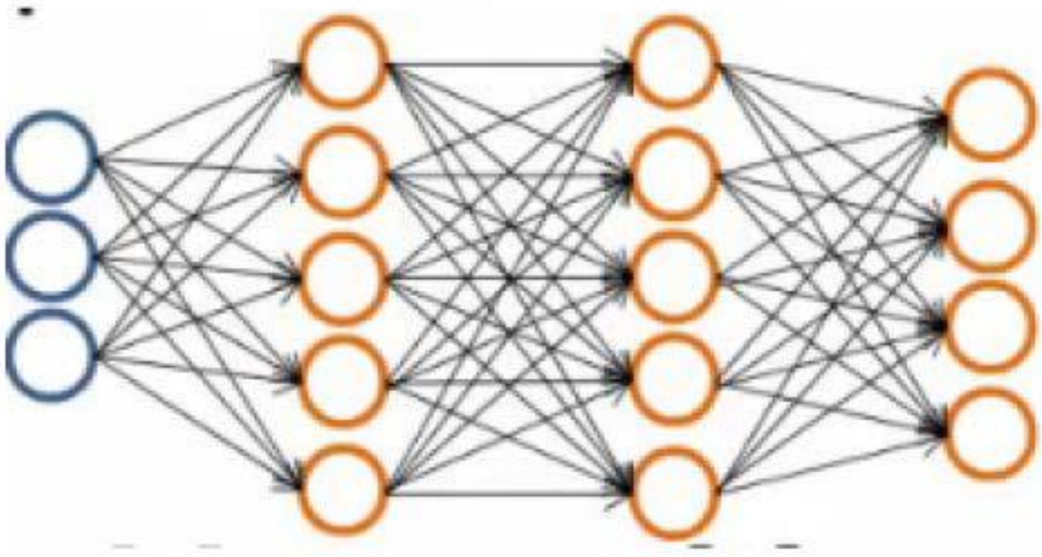
Before Neural Network

Neural Network

$$f = Wx$$

$$x \in \mathbb{R}^D, W \in \mathbb{R}^{C \times D}$$





$$f = W_2 \max(0, W_1 x)$$

$$W_2 \in \mathbb{R}^{C \times H} \quad W_1 \in \mathbb{R}^{H \times D} \quad x \in \mathbb{R}^D$$

or 3-layer Neural Network

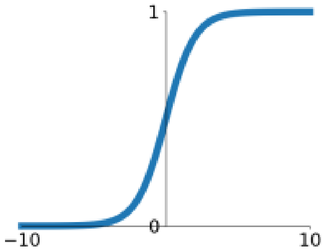
$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$$W_3 \in \mathbb{R}^{C \times H_2} \quad W_2 \in \mathbb{R}^{H_2 \times H_1} \quad W_1 \in \mathbb{R}^{H_1 \times D} \quad x \in \mathbb{R}^D$$

Neural Network – Activation Function

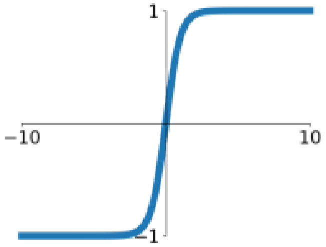
Sigmoid

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



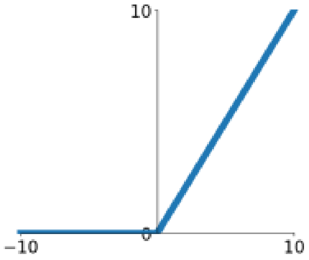
tanh

$$\tanh(x)$$



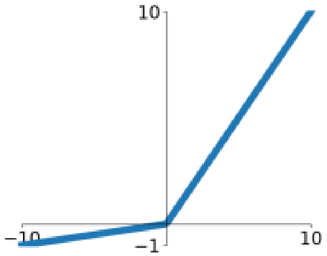
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

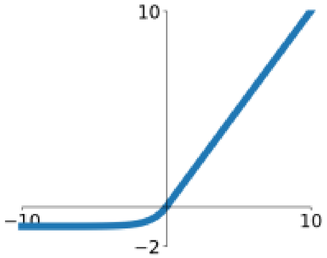


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

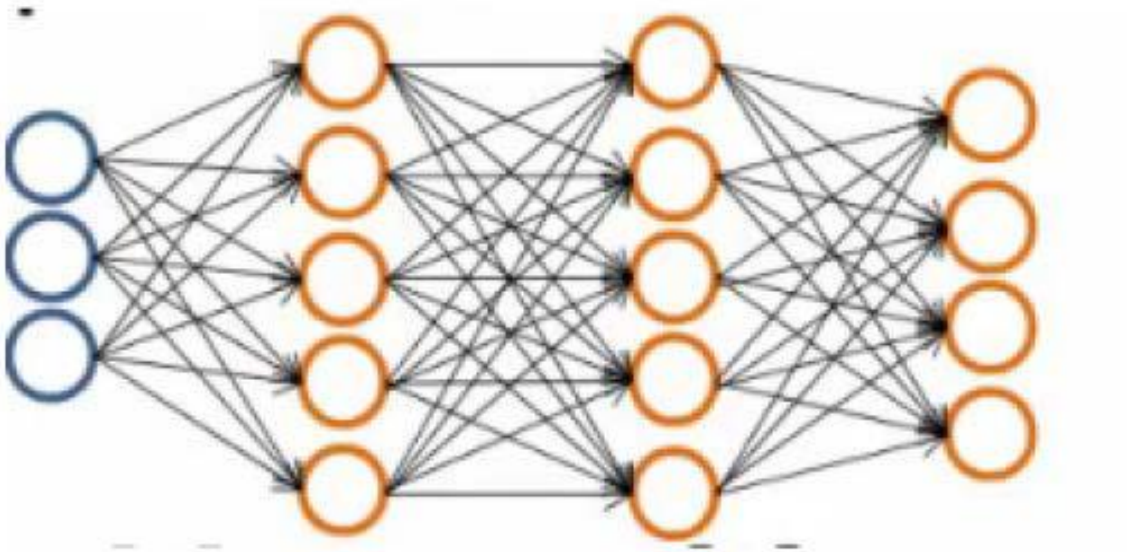
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Neural Network

Linear classifier: One template per class

Neural net: first layer is bank of templates;
Second layer recombines templates

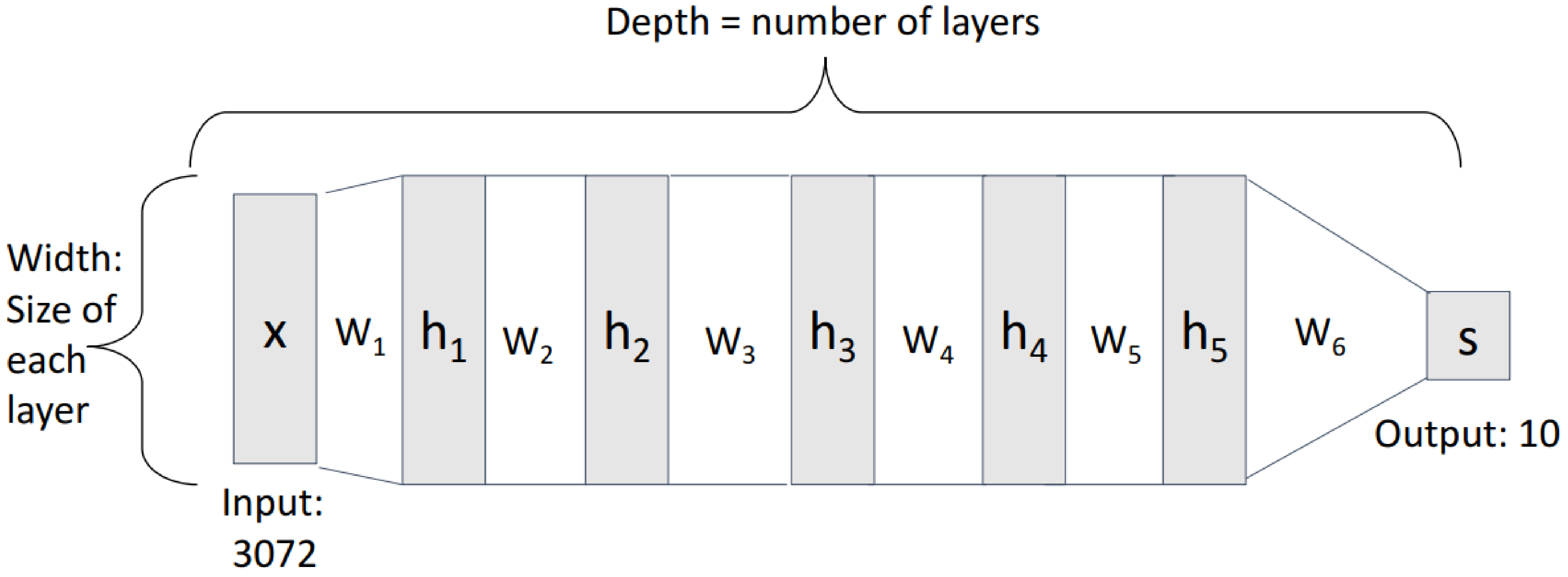


or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$$W_3 \in \mathbb{R}^{C \times H_2} \quad W_2 \in \mathbb{R}^{H_2 \times H_1} \quad W_1 \in \mathbb{R}^{H_1 \times D} \quad x \in \mathbb{R}^D$$

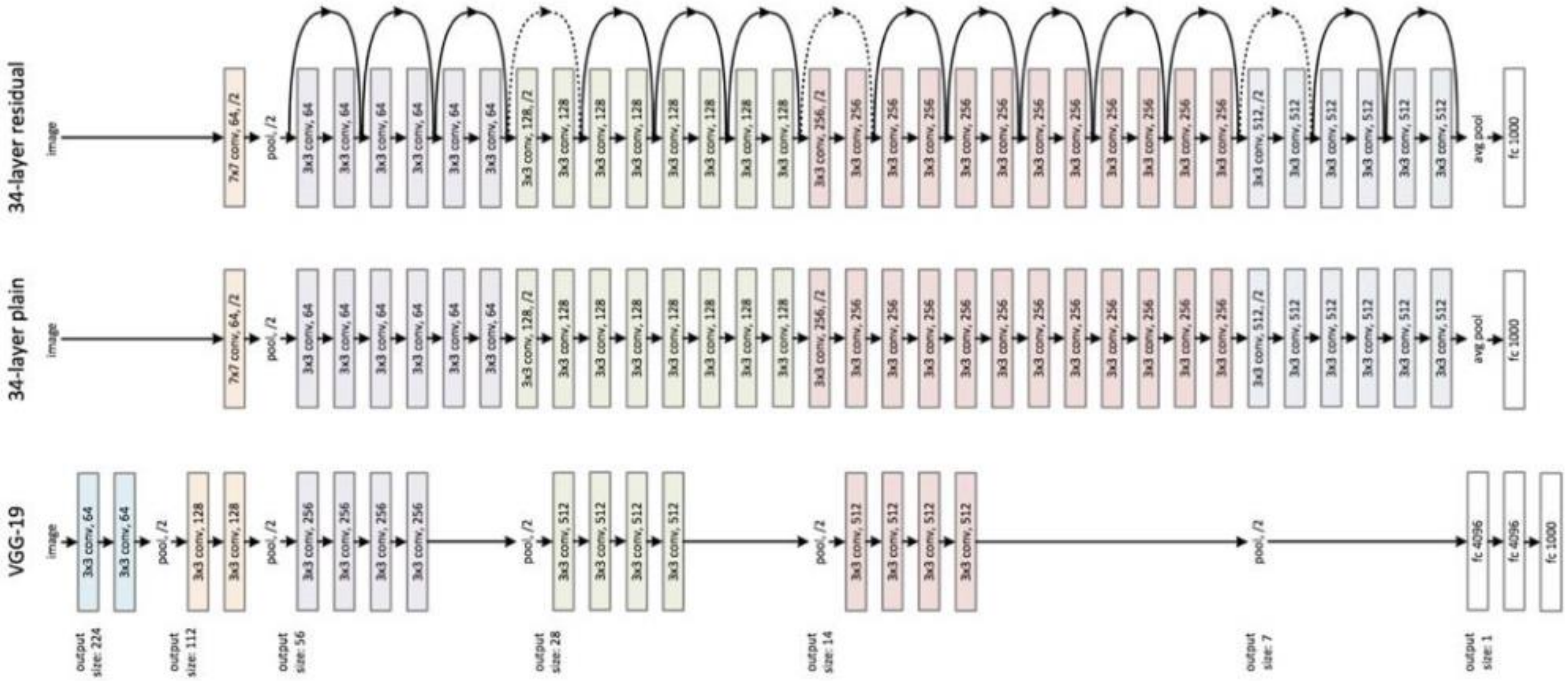
Deep Neural Network



$$s = W_6 \max(0, W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x))))))$$

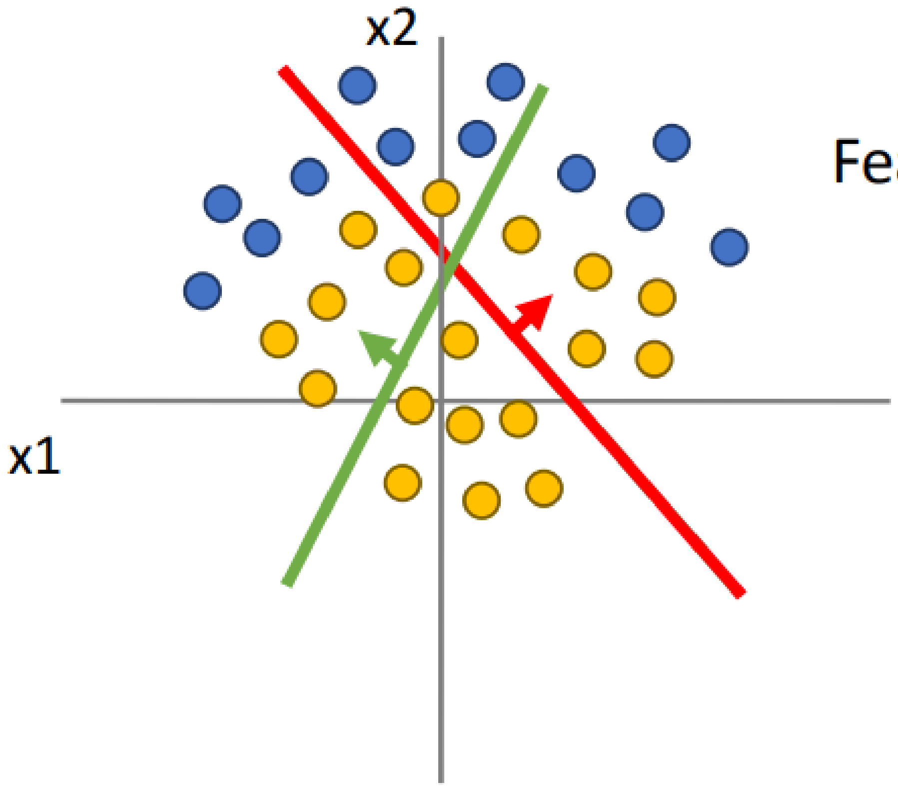
Deep Neural Network

The great gradient highway.



Space Warping

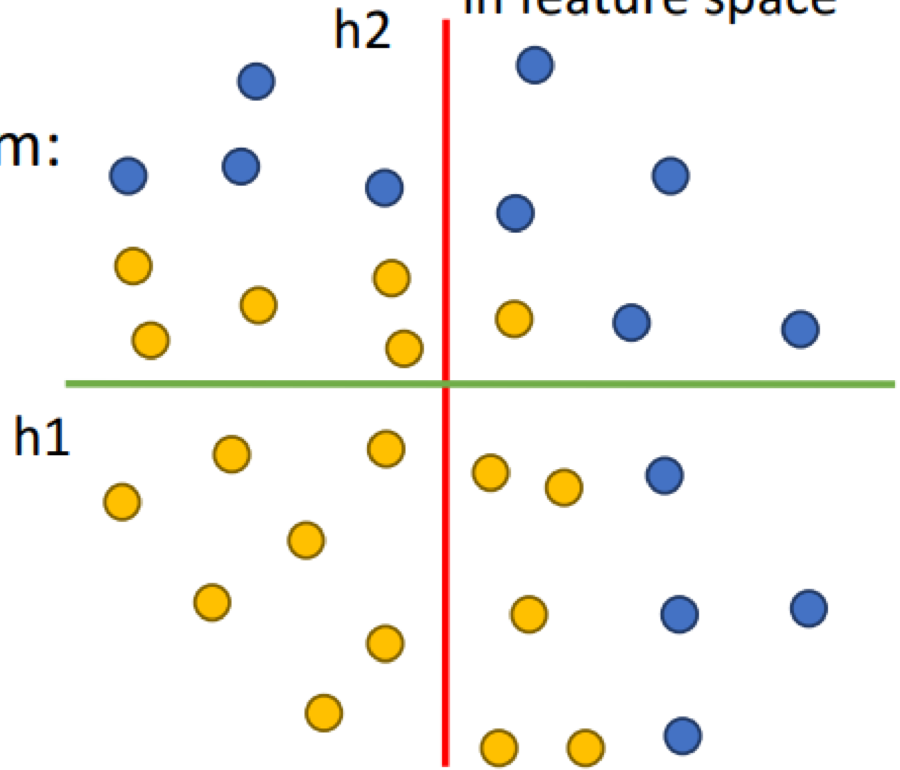
Points not linearly separable in original space



Consider a linear transform: h = Wx
Where x, h are both 2-dimensional

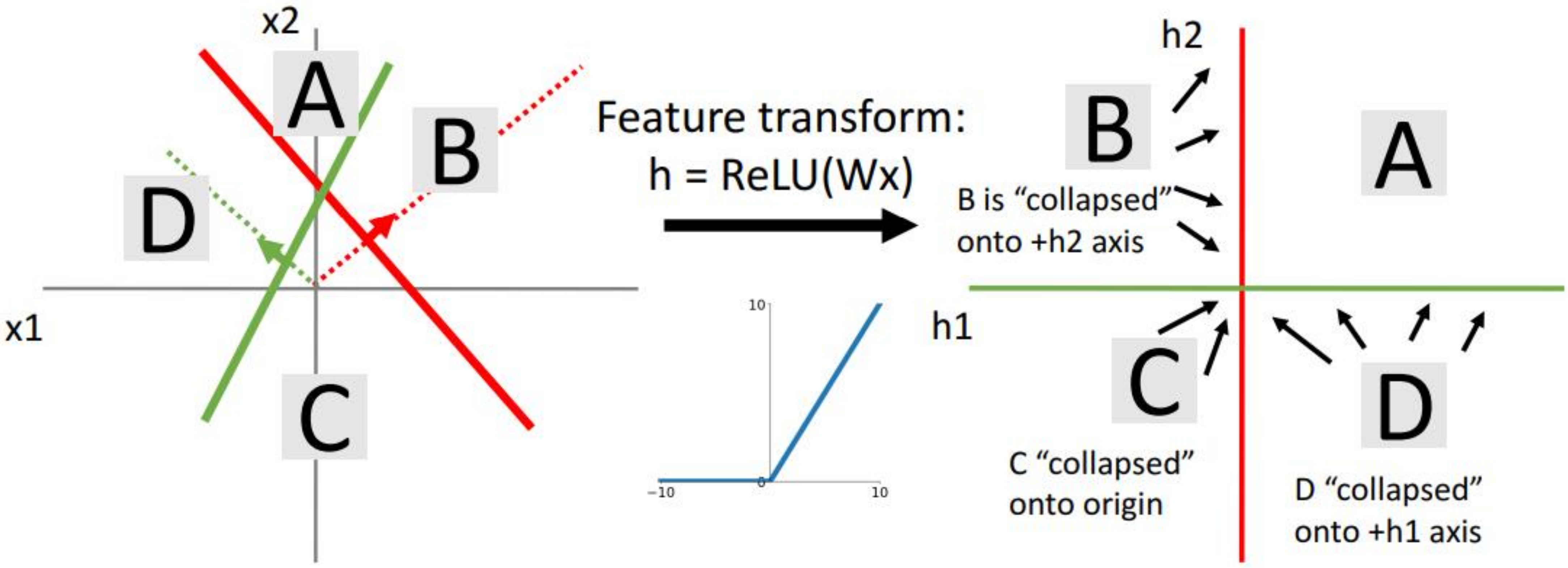
Feature transform:
h = Wx

Not linearly separable in feature space



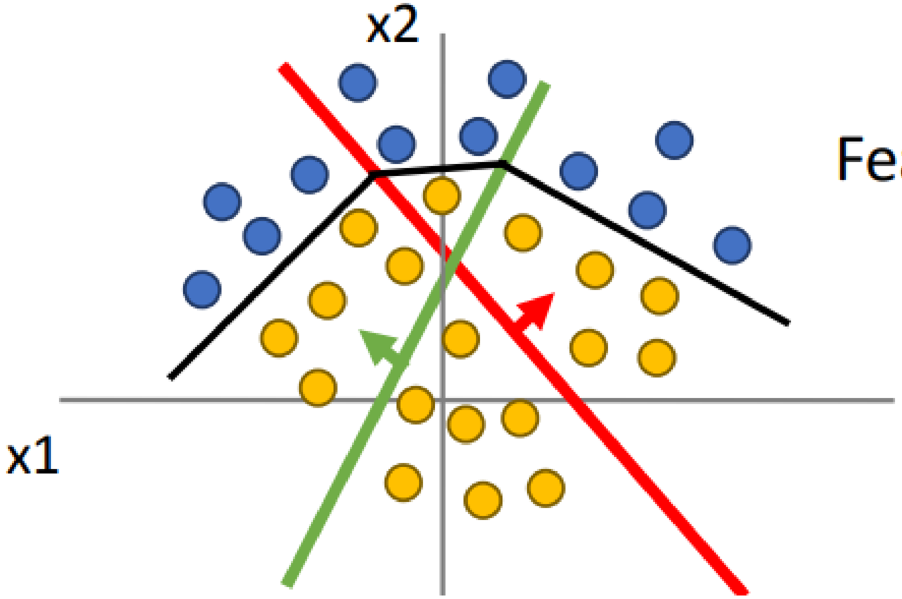
Space Warping

Consider a neural net hidden layer:
h = ReLU(Wx) = max(0, Wx)
Where x, h are both 2-dimensional



Space Warping

Points not linearly separable in original space



Linear classifier in feature space gives nonlinear classifier in original space

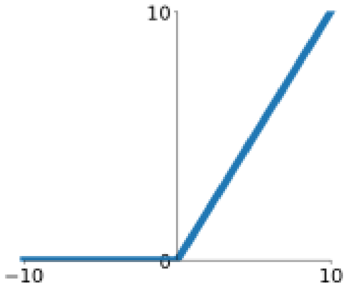
Consider a neural net hidden layer:

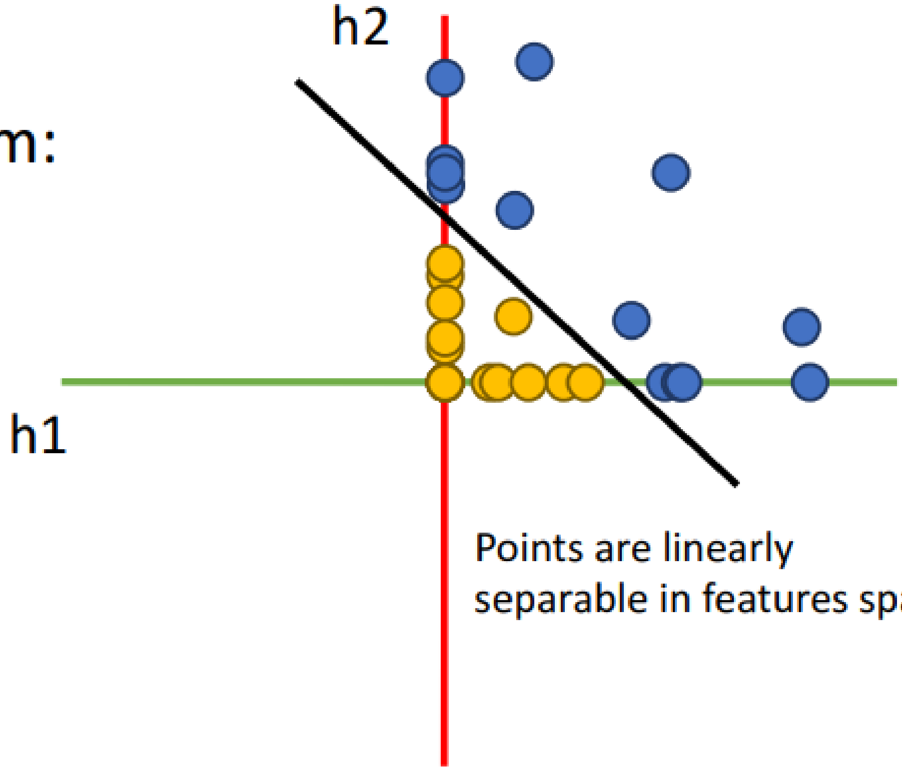
h = ReLU(Wx) = max(0, Wx)

Where x, h are both 2-dimensional

Feature transform:

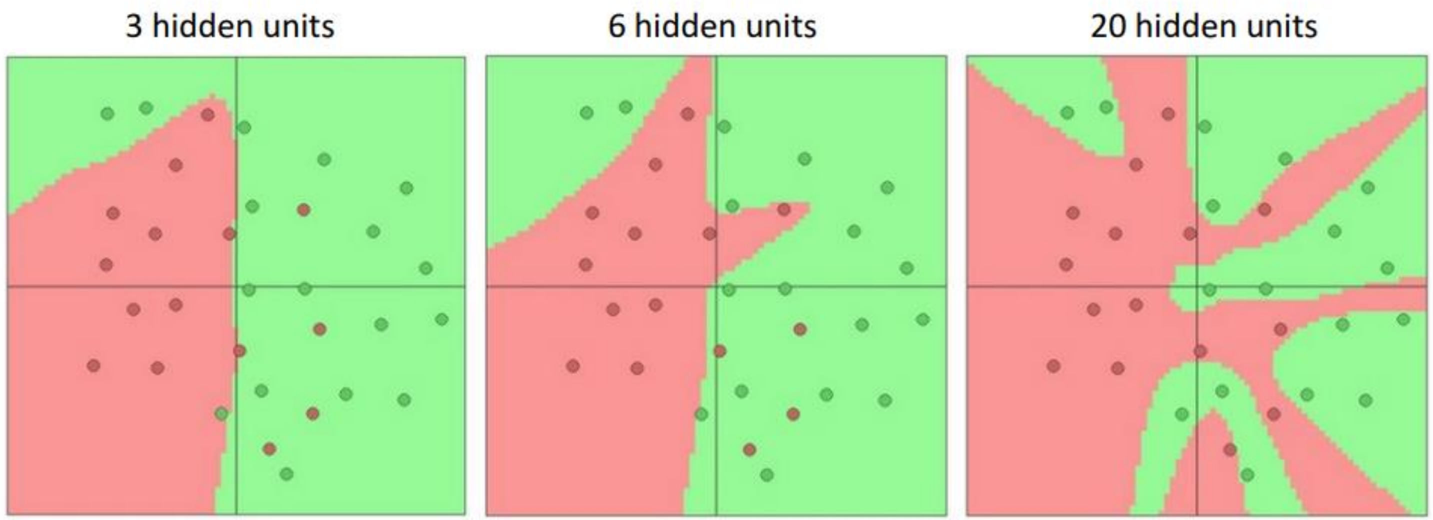
h = ReLU(Wx)



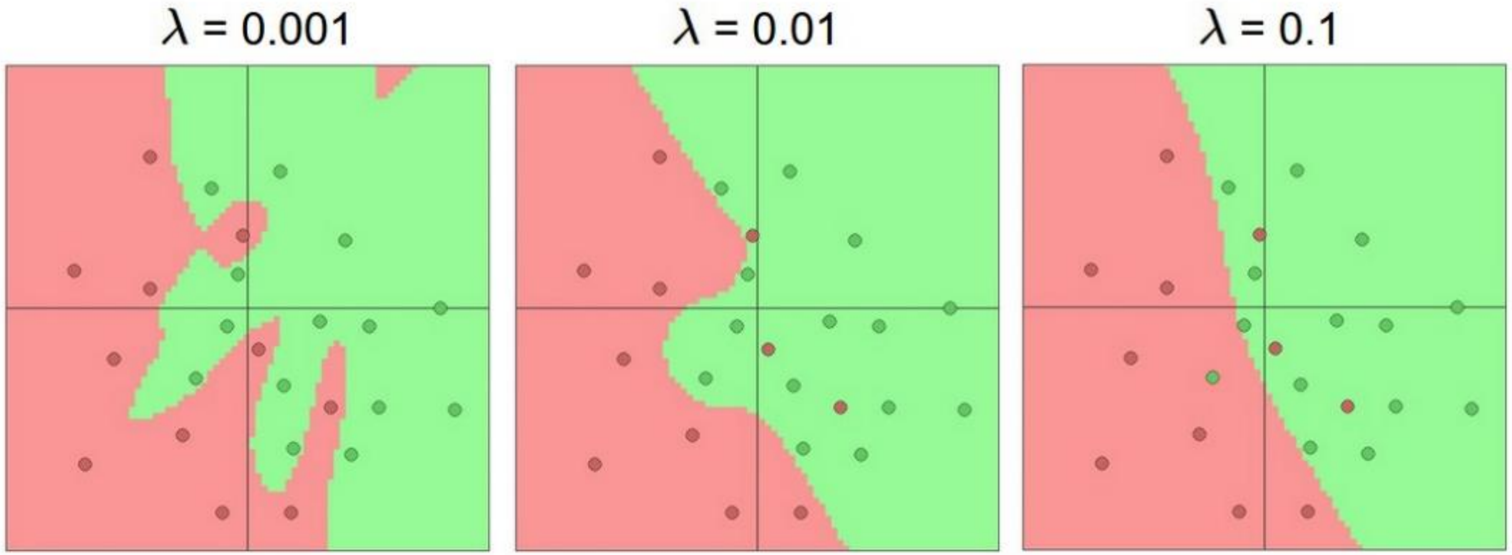


Points are linearly separable in features space!

Setting the number of layers and their sizes



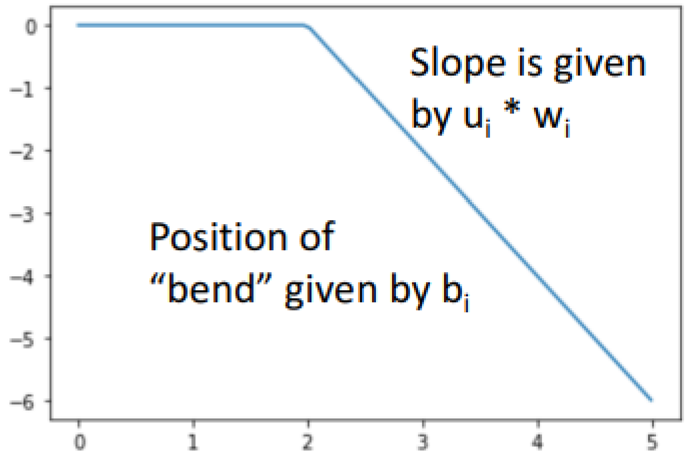
Don’t regularize with size; instead use stronger L2



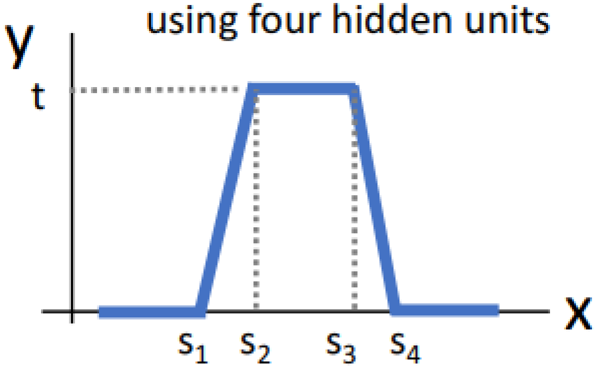
Deep Neural Network – Universal Approximation

A neural network with one hidden layer can approximate any function f: R^N -> R^M with arbitrary precision*

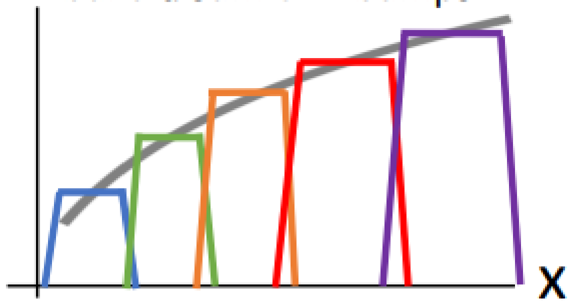
Flip left / right based on sign of w_i



We can build a “bump function” using four hidden units

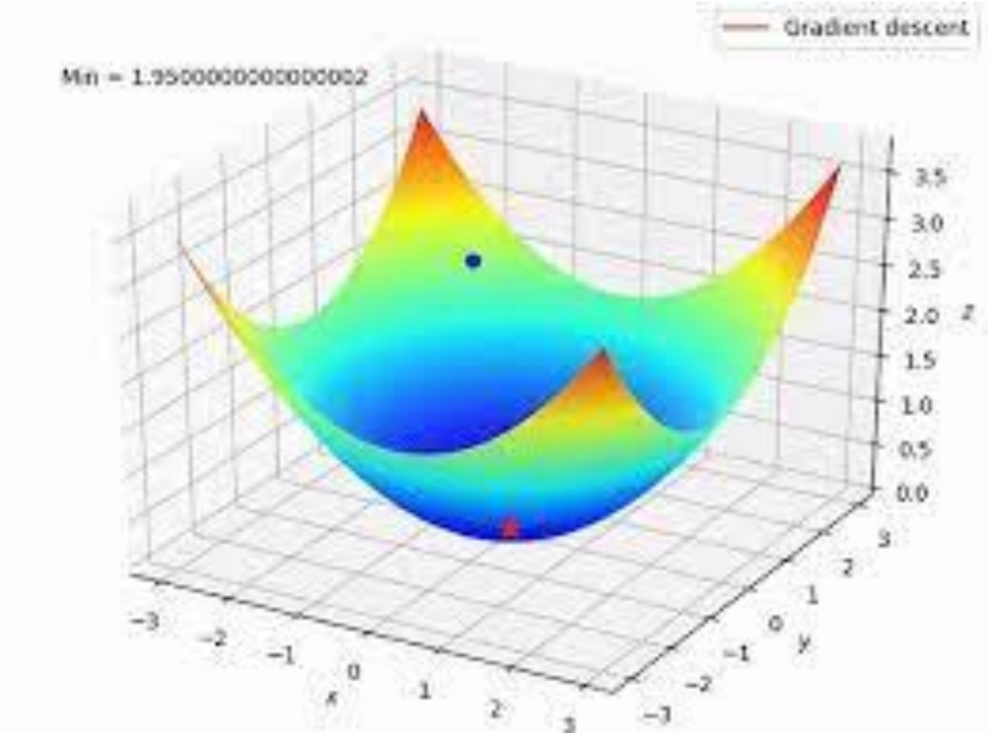


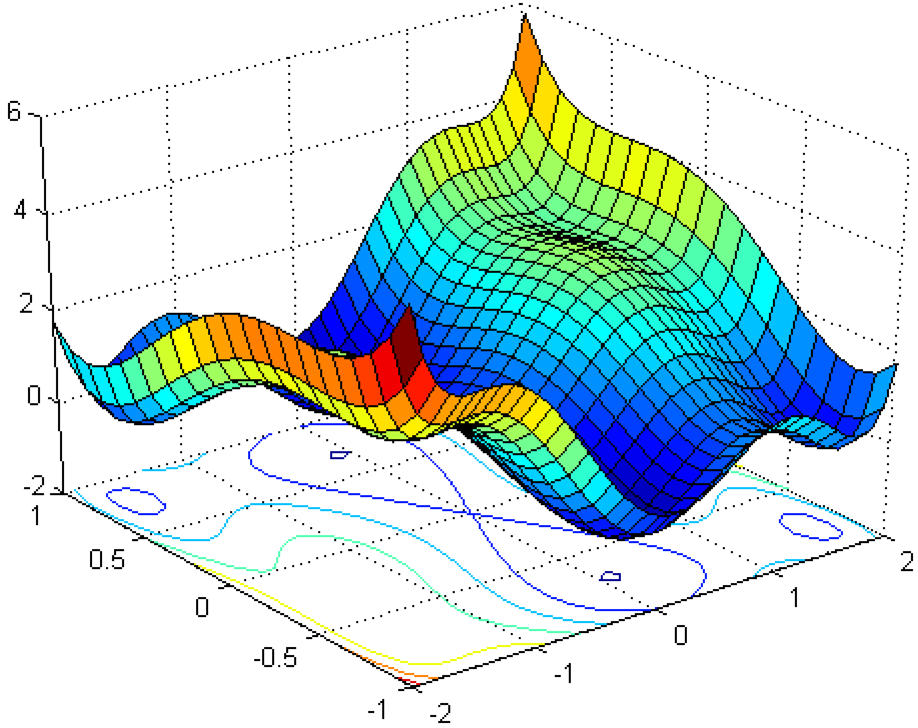
With 4K hidden units we can build a sum of K bumps



Approximate functions with bumps!

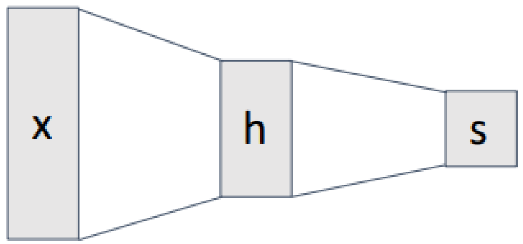
Convex Function



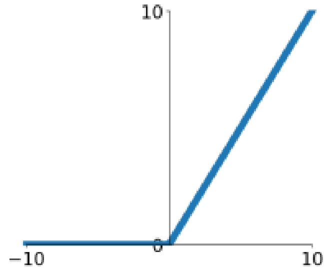


Convolutional Network

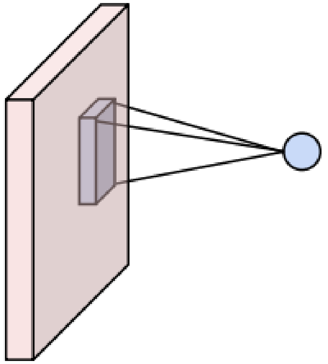
Fully-Connected Layers



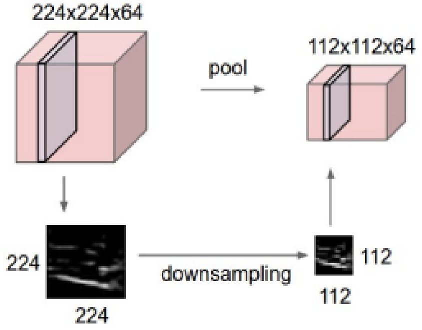
Activation Function



Convolution Layers



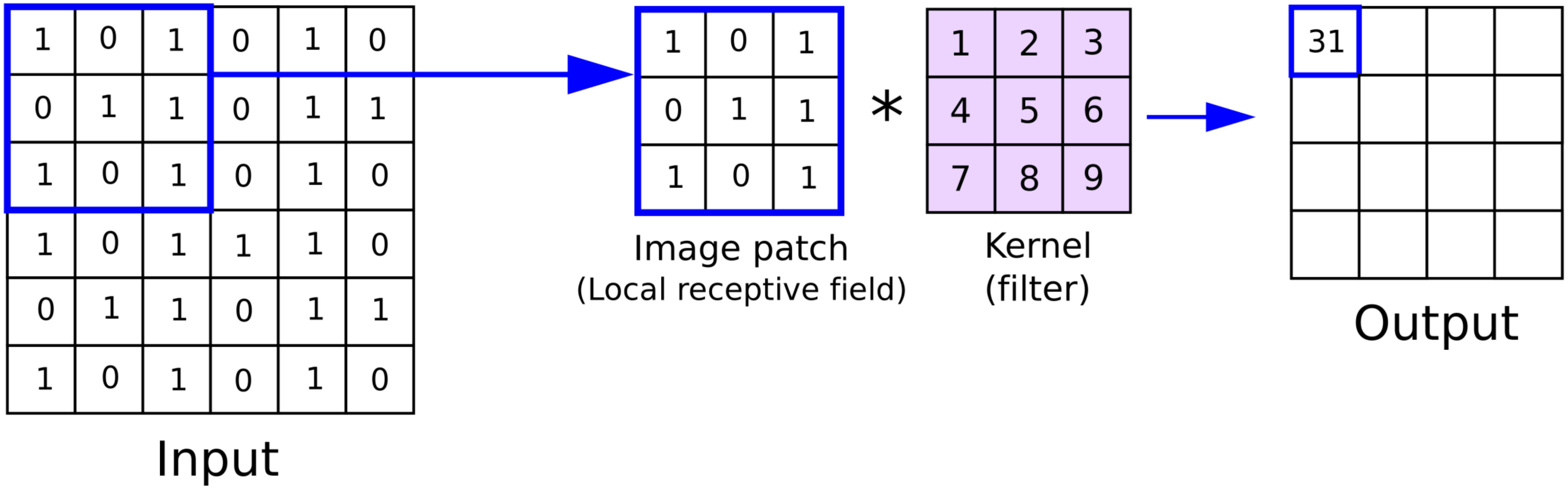
Pooling Layers



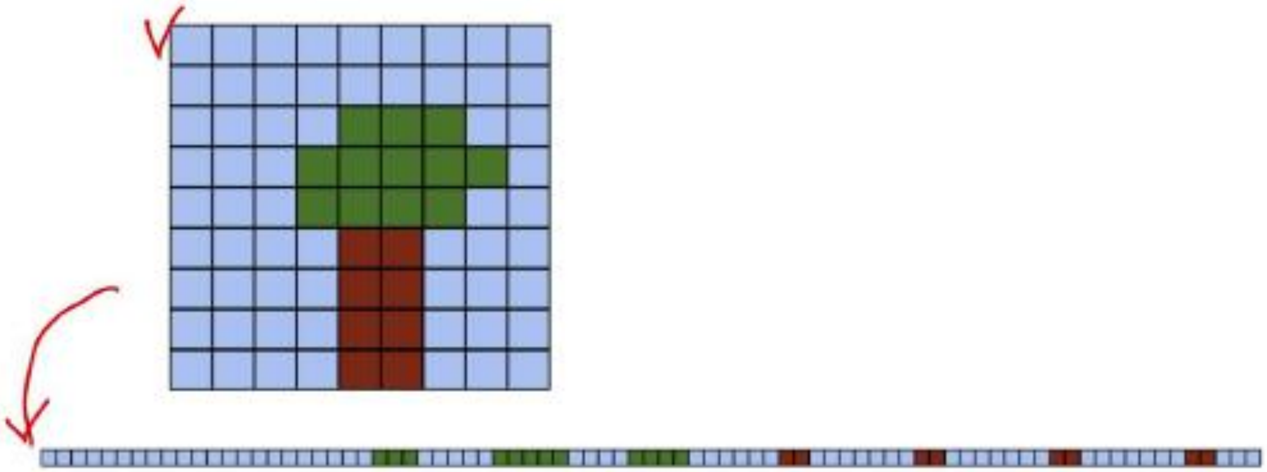
Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Convolutional Network



Why Convolutional Network?



단점)
- img가 가지고 있는 위치적 특성 날라감 - 위에 선이 뭔 사진인데 그래서?
- param 개수가 ㄷㄷ(원래 img pixel 수가 많으므로) - overfitting 위험성뿐만 아니라 연산 속도 느림

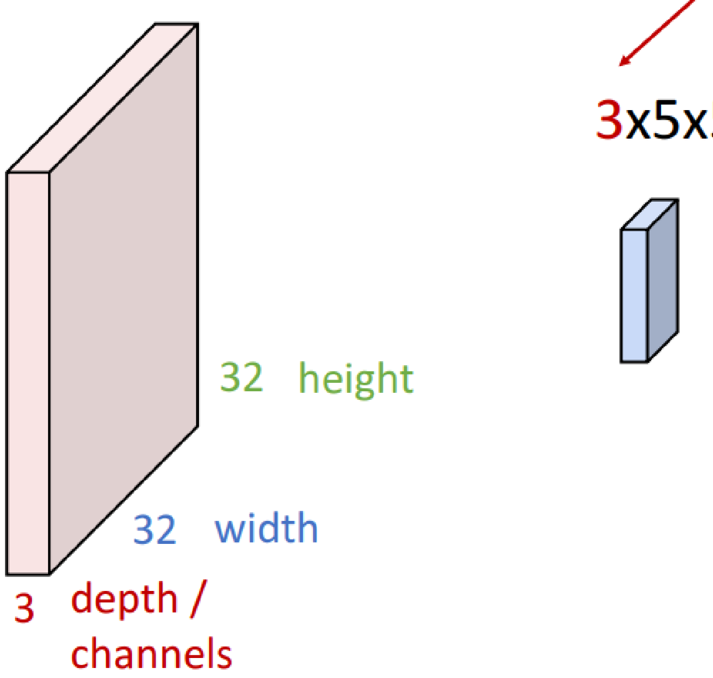
Convolutional Network

Convolution Layer

Filters always extend the full depth of the input volume

3x32x32 image

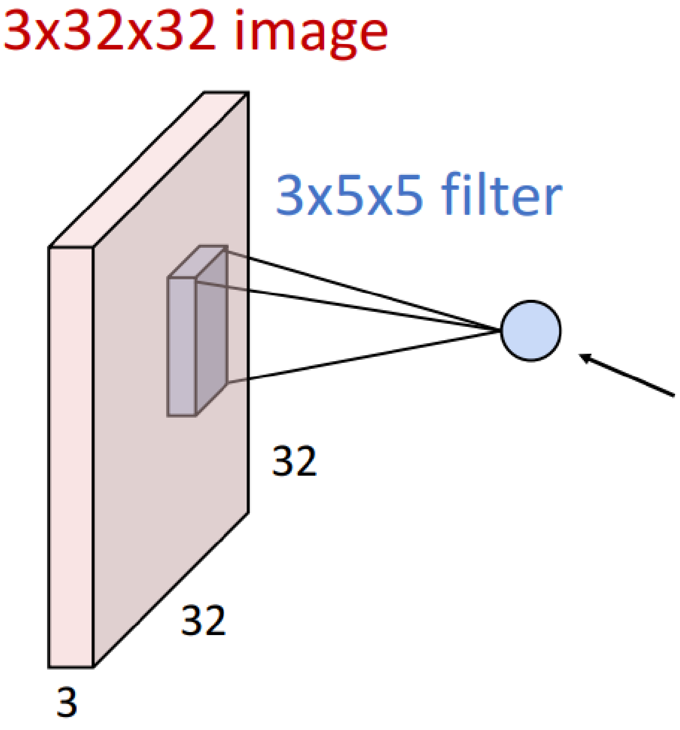
3x5x5 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolutional Network

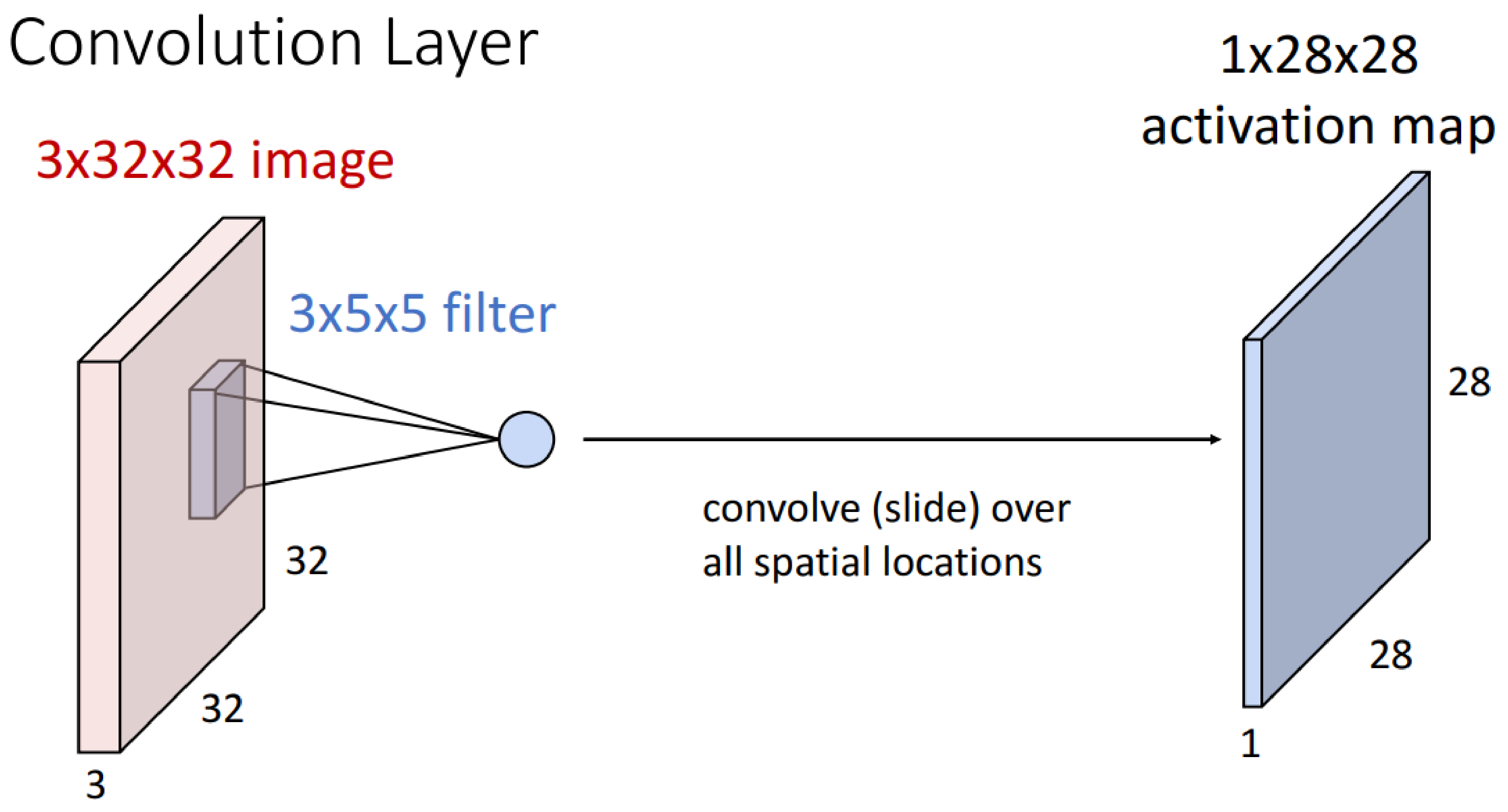
Convolution Layer



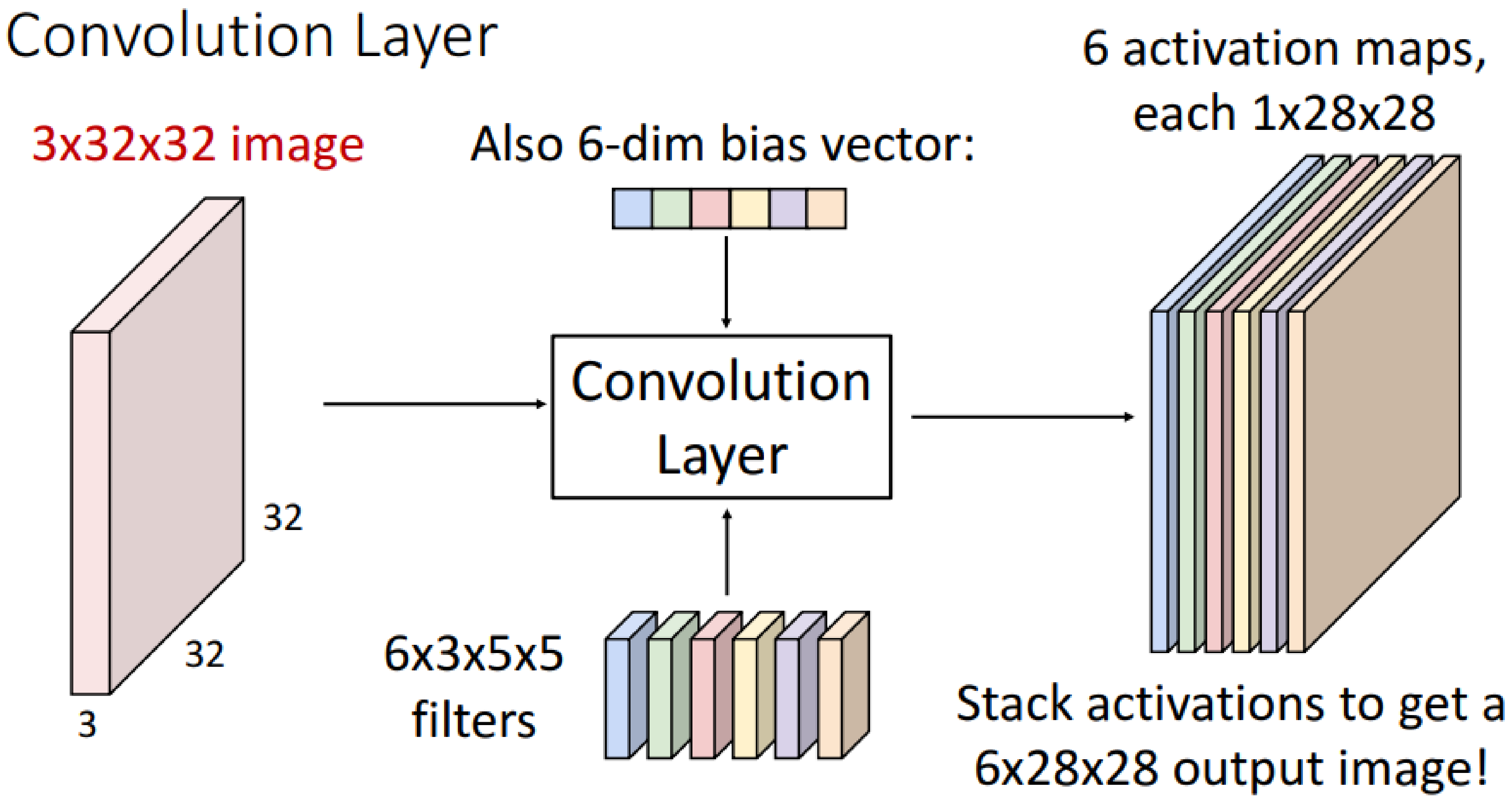
1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

$$w^T x + b$$

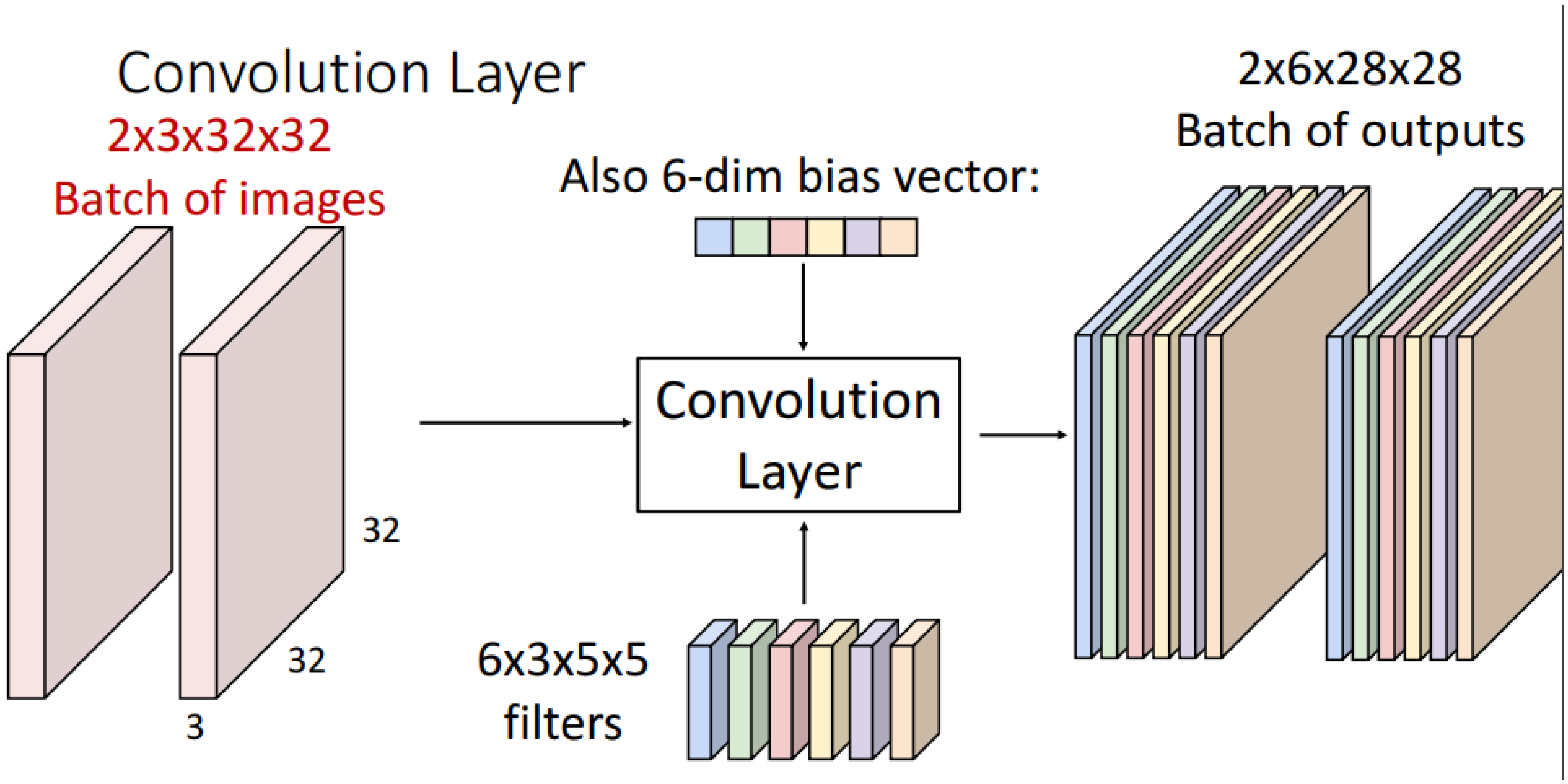
Convolutional Network



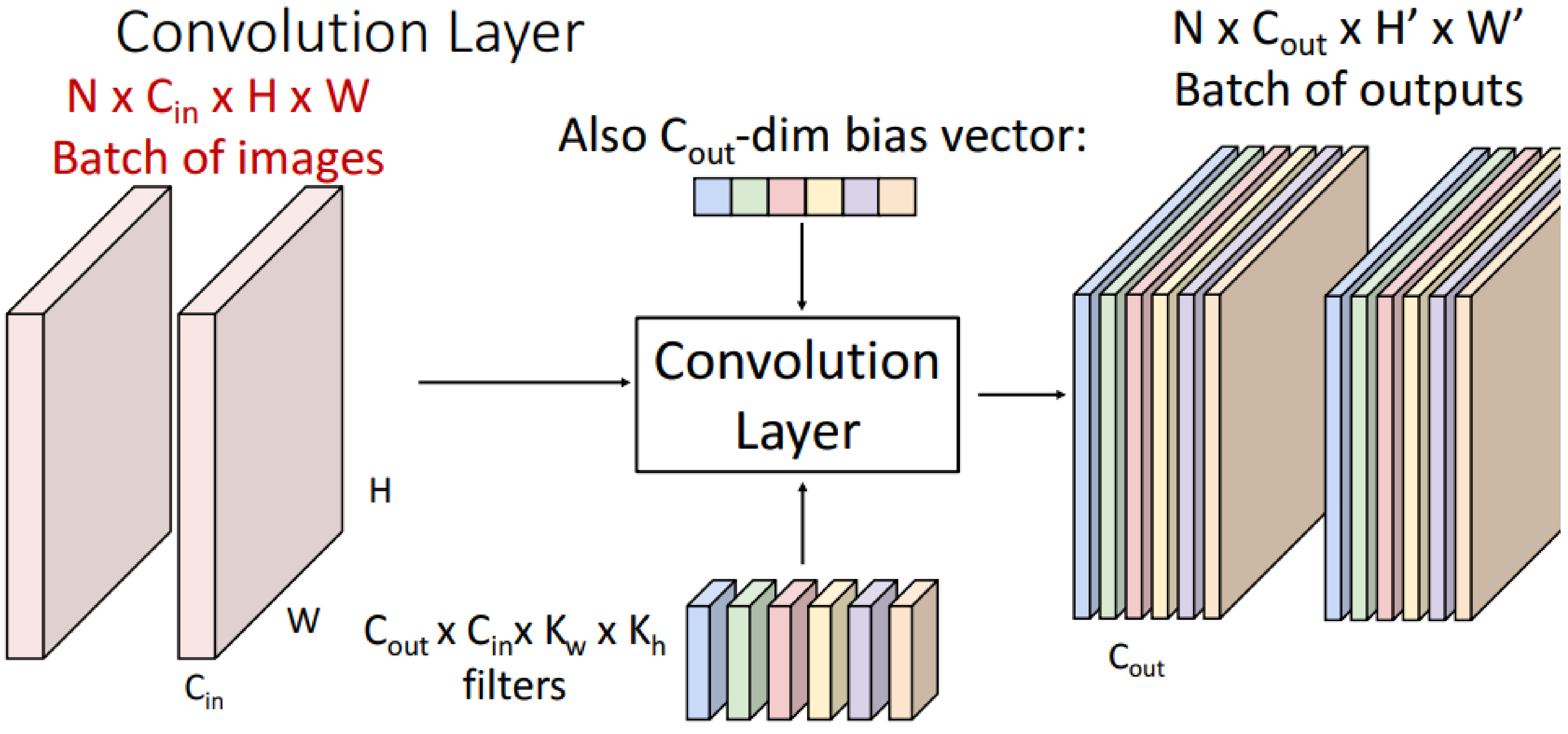
Convolutional Network



Convolutional Network



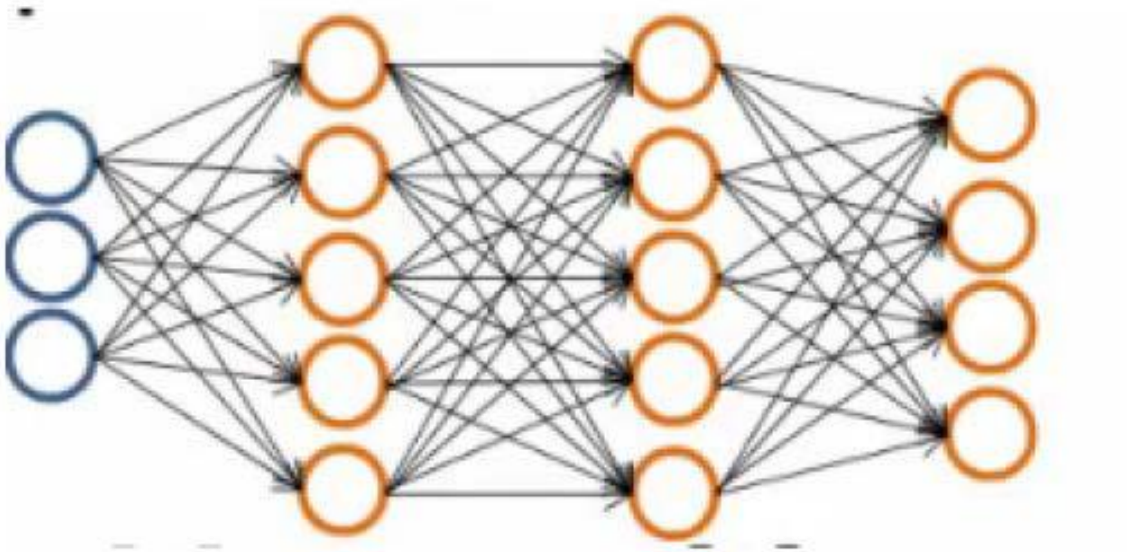
Convolutional Network



Back to the Neural Network

Linear classifier: One template per class

Neural net: first layer is bank of templates;
Second layer recombines templates



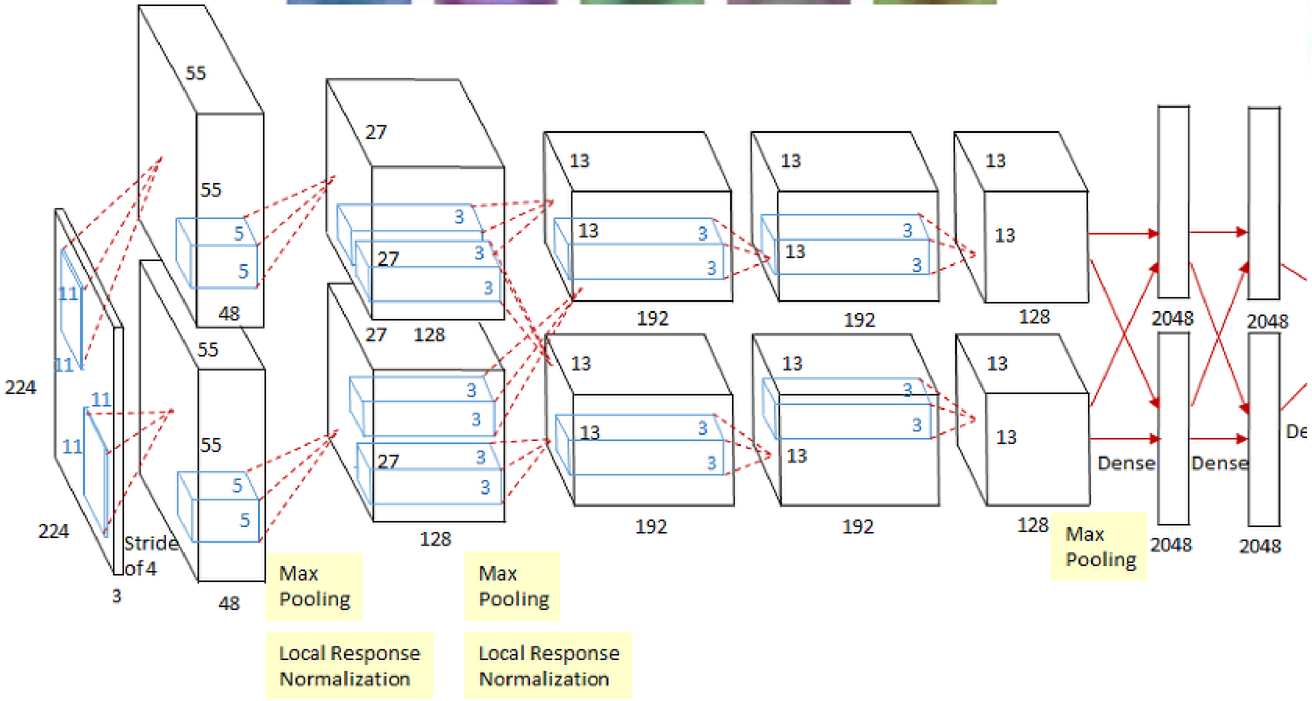
or 3-layer Neural Network

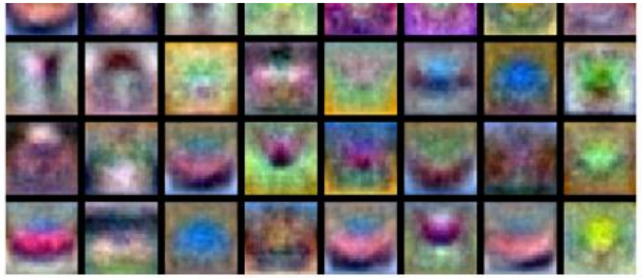
$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$W_3 \in \mathbb{R}^{C \times H_2}$ $W_2 \in \mathbb{R}^{H_2 \times H_1}$ $W_1 \in \mathbb{R}^{H_1 \times D}$ $x \in \mathbb{R}^D$

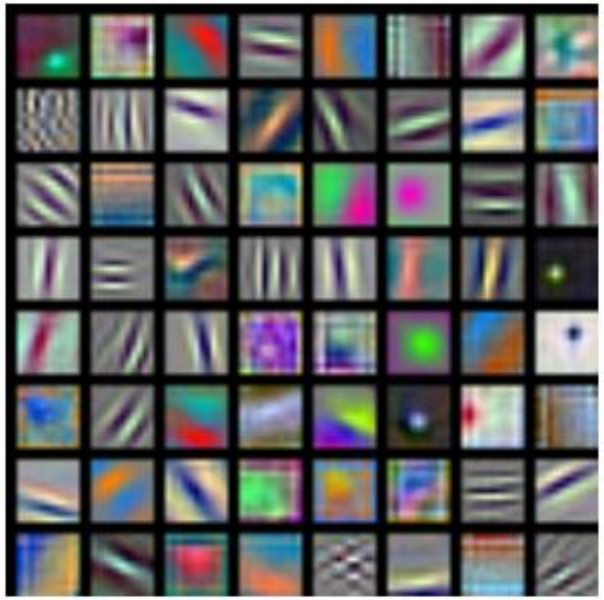
What do convolutional filters learn?

Linear classifier: One template per class



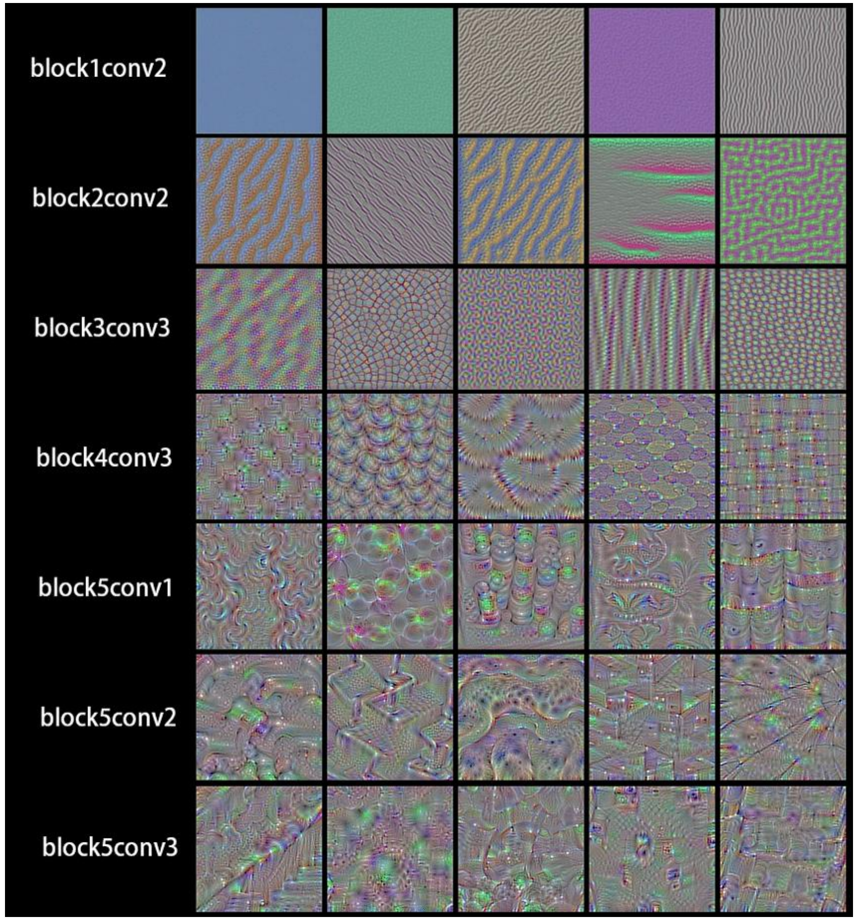


First-layer conv filters: local image templates
(Often learns oriented edges, opposing colors)



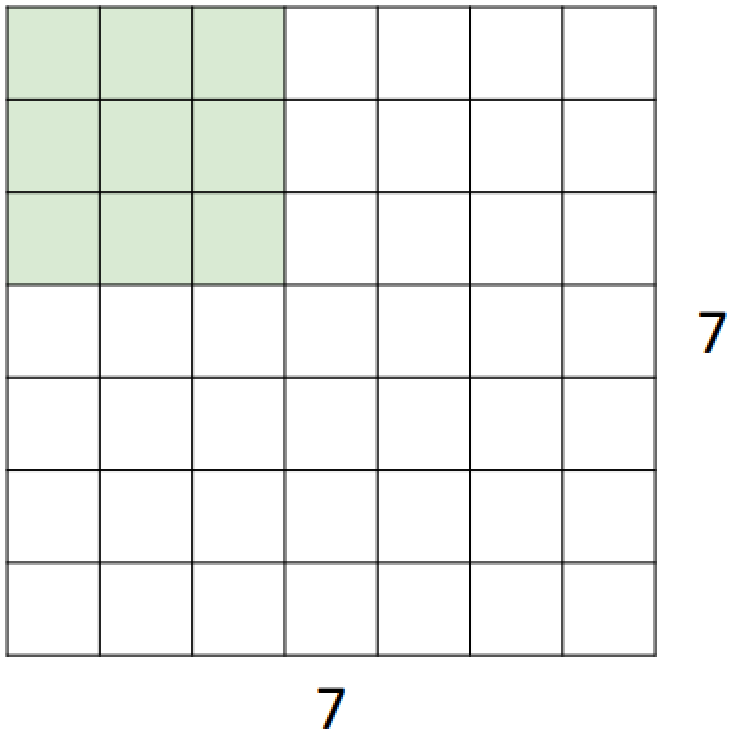
AlexNet: 64 filters, each 3x11x11

What do convolutional filters learn?



Visualization of Convolution Filters. With the increase of depth in the convolutional neural network using convolution layers and pooling layers, the pattern searched by convolution filters becomes larger in scale and tend to be more sophisticated.

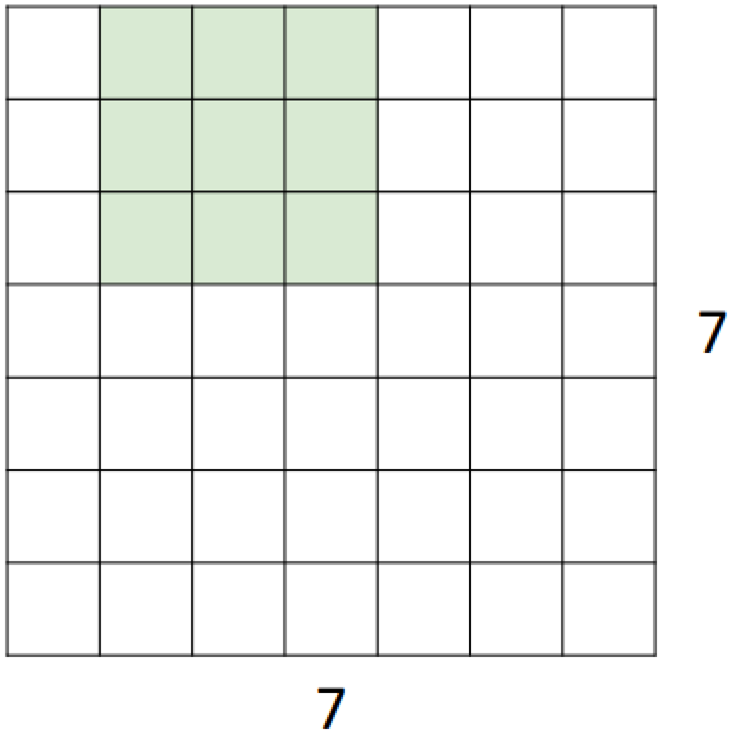
Convolutional Network



Input: 7x7

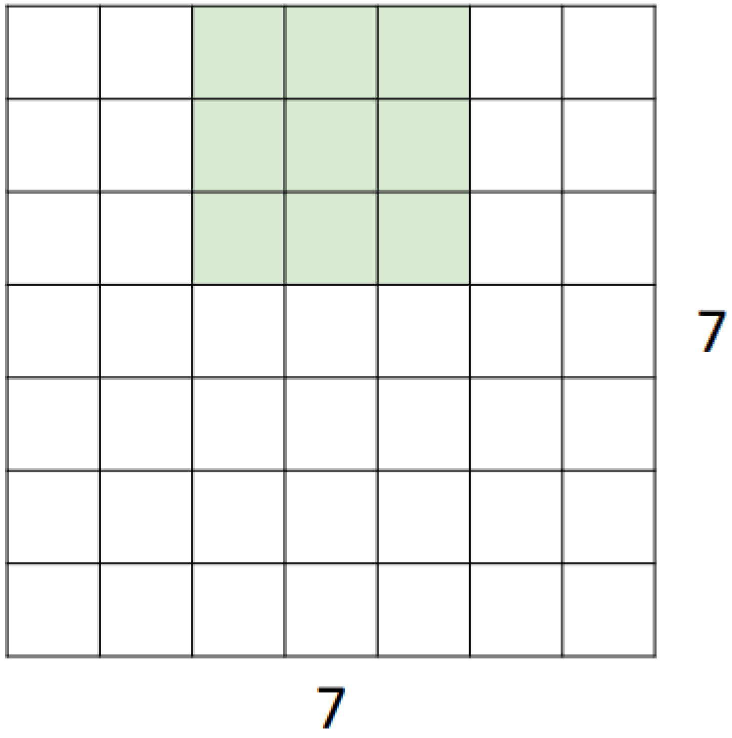
Filter: 3x3

Convolutional Network



Input: 7x7
Filter: 3x3

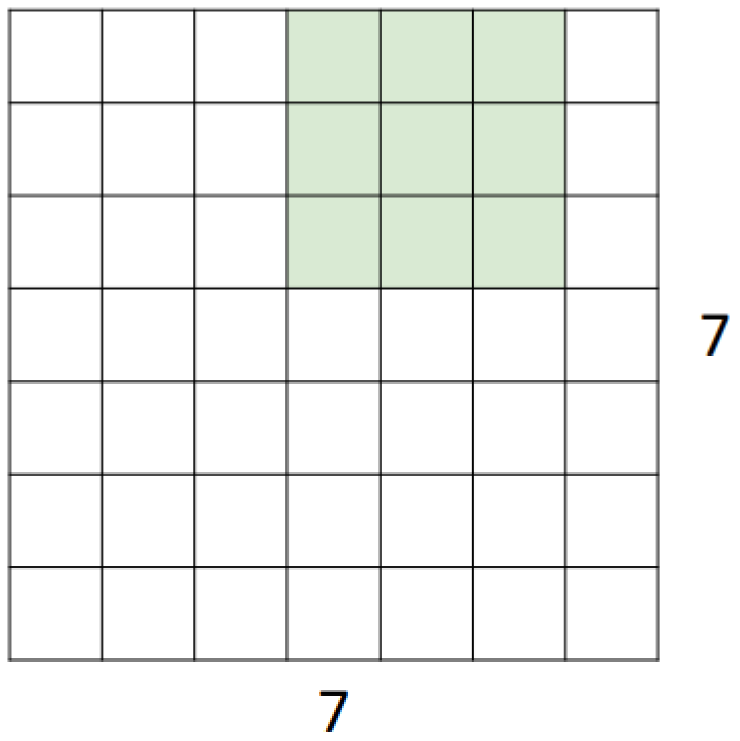
Convolutional Network



Input: 7x7

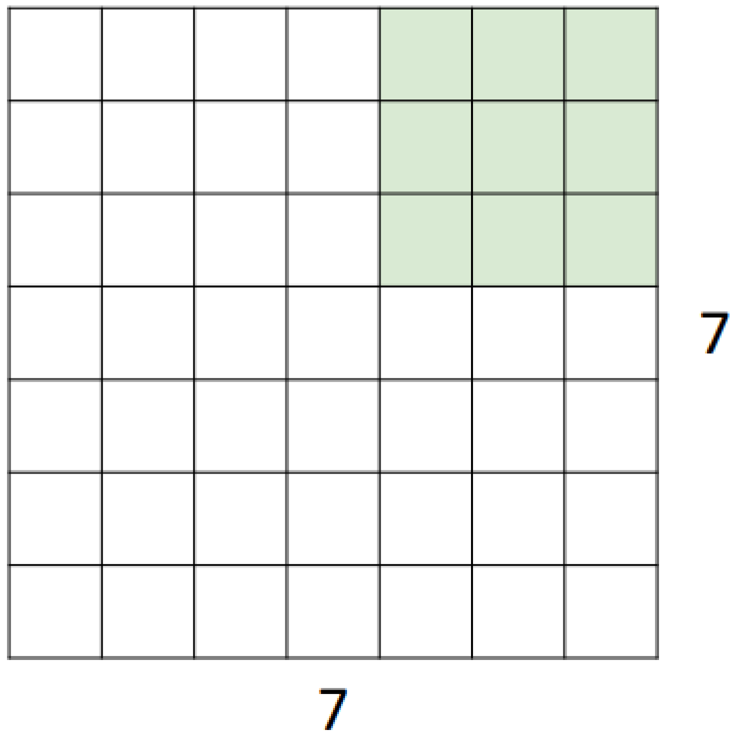
Filter: 3x3

Convolutional Network



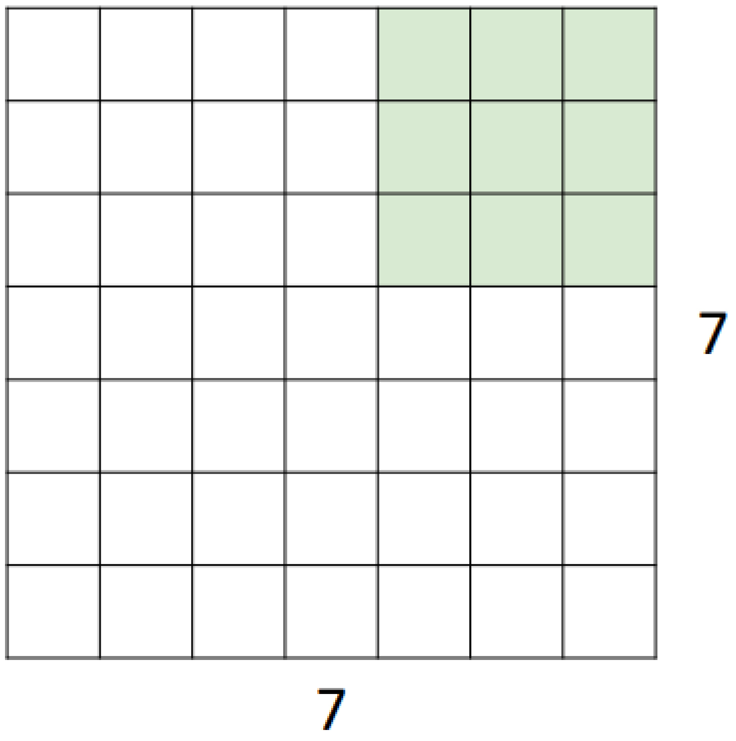
Input: 7x7
Filter: 3x3

Convolutional Network



Input: 7x7
Filter: 3x3
Output: 5x5

Convolutional Network



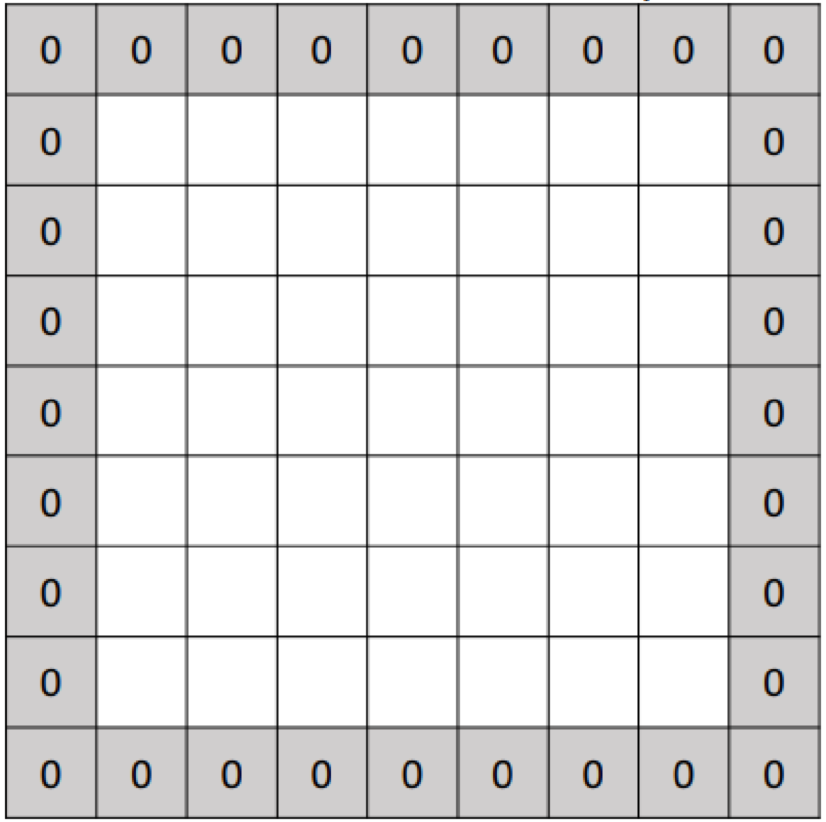
Input: 7x7
Filter: 3x3
Output: 5x5

In general:
Input: W
Filter: K
Output: W – K + 1

Problem: Feature maps “shrink” with each layer!

Convolutional Network - Padding

A closer look at spatial dimensions



Input: 7x7
Filter: 3x3
Output: 5x5

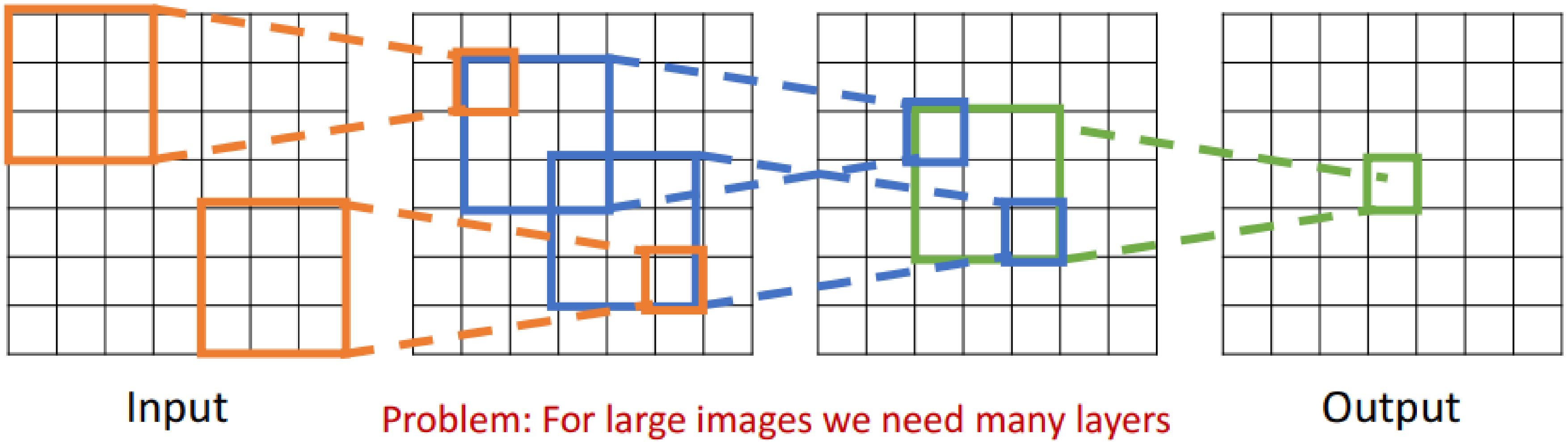
In general:
Input: W
Filter: K
Padding: P
Output: W – K + 1 + 2P

Very common:
Set P = (K – 1) / 2 to make output have same size as input!

Convolutional Network

Receptive Fields

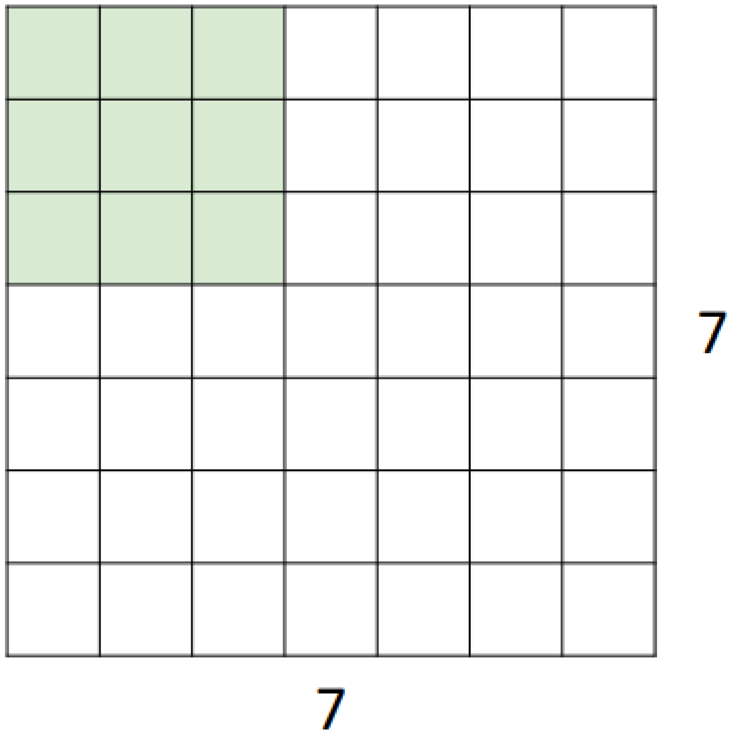
Each successive convolution adds K – 1 to the receptive field size
With L layers the receptive field size is 1 + L * (K – 1)



Problem: For large images we need many layers for each output to "see" the whole image image

Solution: Downsample inside the network

Convolutional Network - Stride

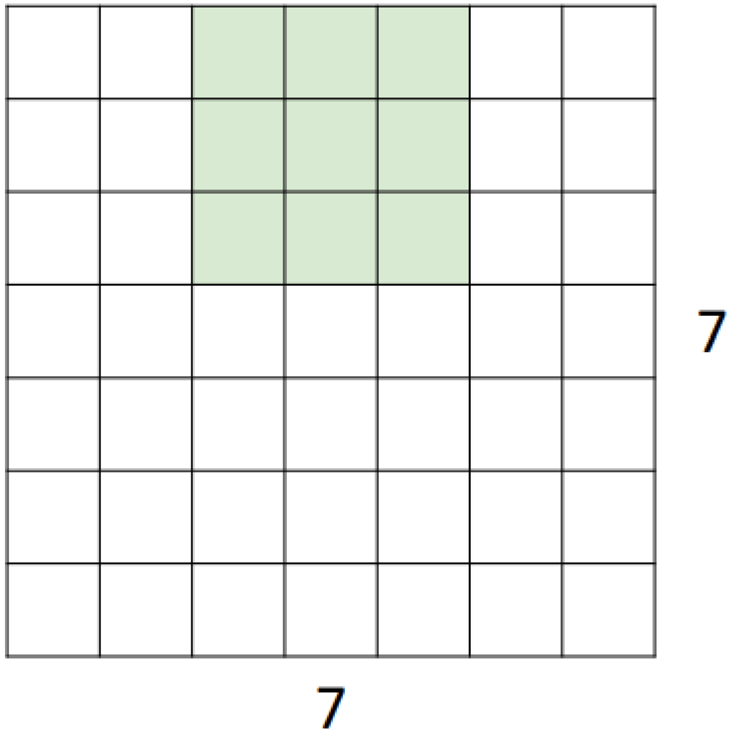


Input: 7x7

Filter: 3x3

Stride : 2

Convolutional Network - Stride

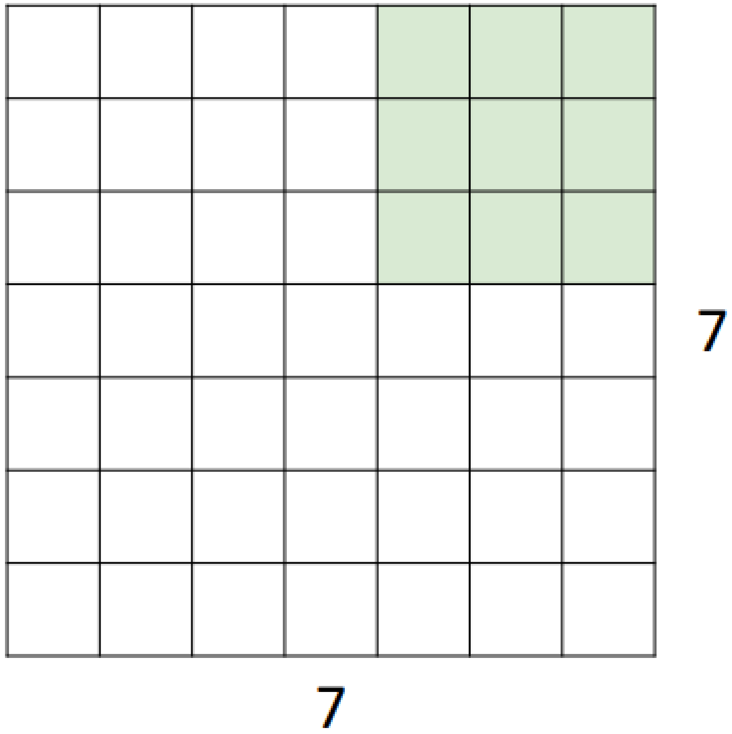


Input: 7x7

Filter: 3x3

Stride : 2

Convolutional Network - Stride



Input: 7x7

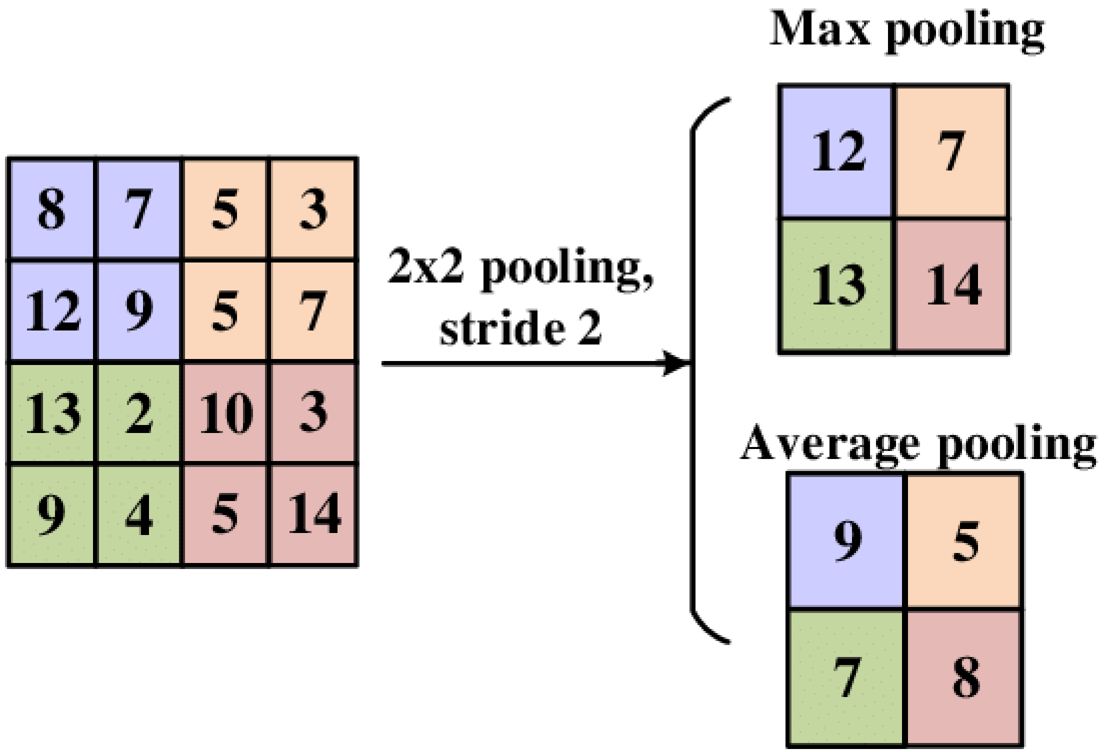
Filter: 3x3

~~Output: 5x5~~ **Output : 3 x 3**

Stride : 2

Output : (W – K + 2P) / S + 1

Convolutional Network - Pooling



Pros and Cons – Stride & Pooling

convolution with stride 방식의 장단점

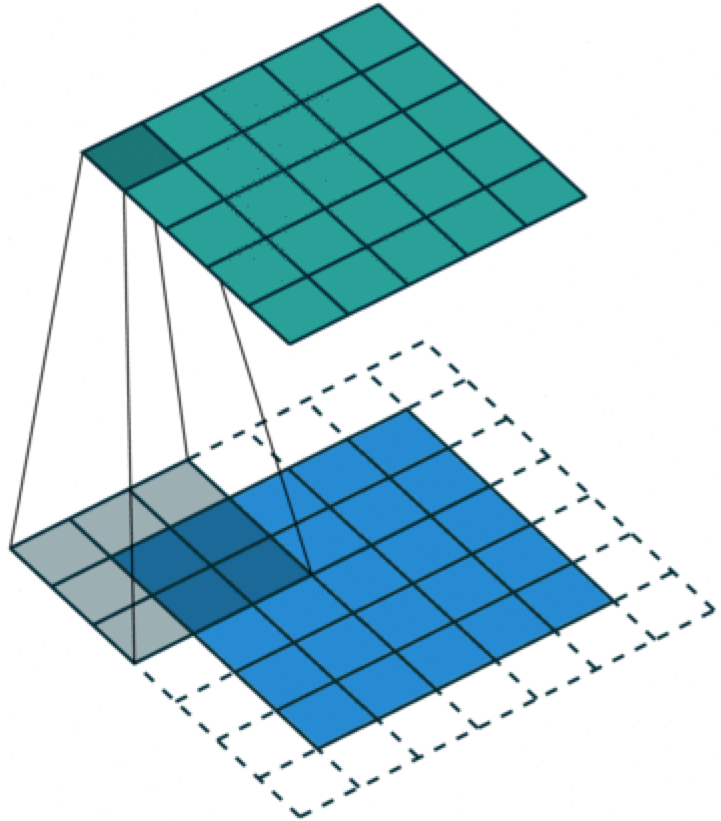
- ① 학습 가능한 파라미터가 추가되므로 네트워크가 resolution을 잘 줄이는 방법을 학습할 수 있어서 pooling보다 성능이 좋습니다.
- ② feature를 뽑기 위한 Convolution Layer와 Downsampling을 위한 stride를 동시에 적용할 수 있습니다. 이 경우 같은 3 x 3 크기의 필터를 사용하더라도 stride가 적용되기 때문에 **더 넓은 receptive field**를 볼 수 있습니다.
- ③ STRIVING FOR SIMPLICITY: THE ALL CONVOLUTIONAL NET에서는 모든 Pooling을 Convolution with stride로 변경 시 성능 상승의 효과가 있는 것을 확인하였습니다.
  - `We find that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks`

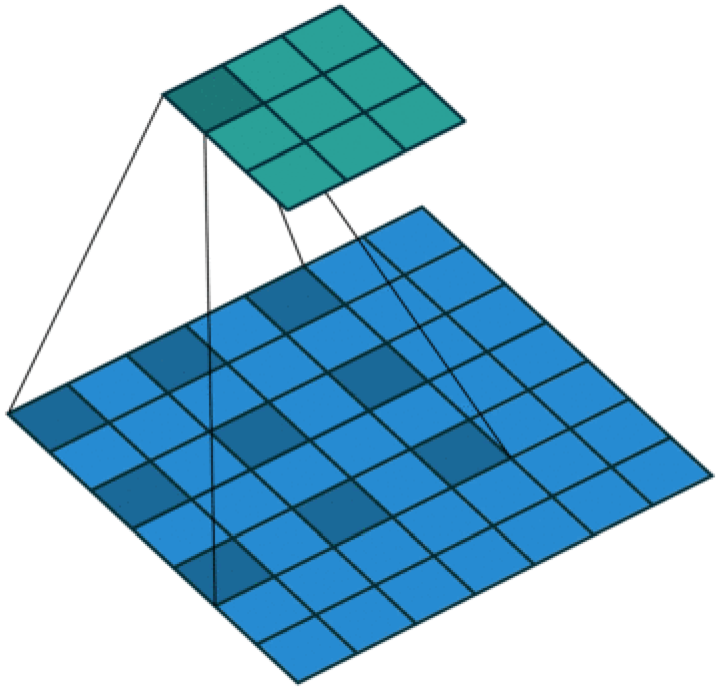
pooling 방식의 장단점

- ① convolution 연산 대비 연산량이 적으며 저장해야 할 파라미터의 숫자도 줄어드므로 학습 시간도 상대적으로 줄일 수 있고 인퍼런스 시 시간도 줄일 수 있습니다.
- ② FishNet 에서 제안한 내용 중에 Skip Connection에서 Convolution layer가 계속 추가되면 backpropagation 시 gradient가 잘 전달이 안될 수 있다고 하여 단순히 Pooling만을 사용한 기법이 적용됩니다. 즉, **layer를 줄여서 gradient 전파에 초점을 두려고 할 때** pooling을 사용하는게 도움이 될 수 있습니다.

https://gaussian37.github.io/dl-concept-stride_vs_pooling/

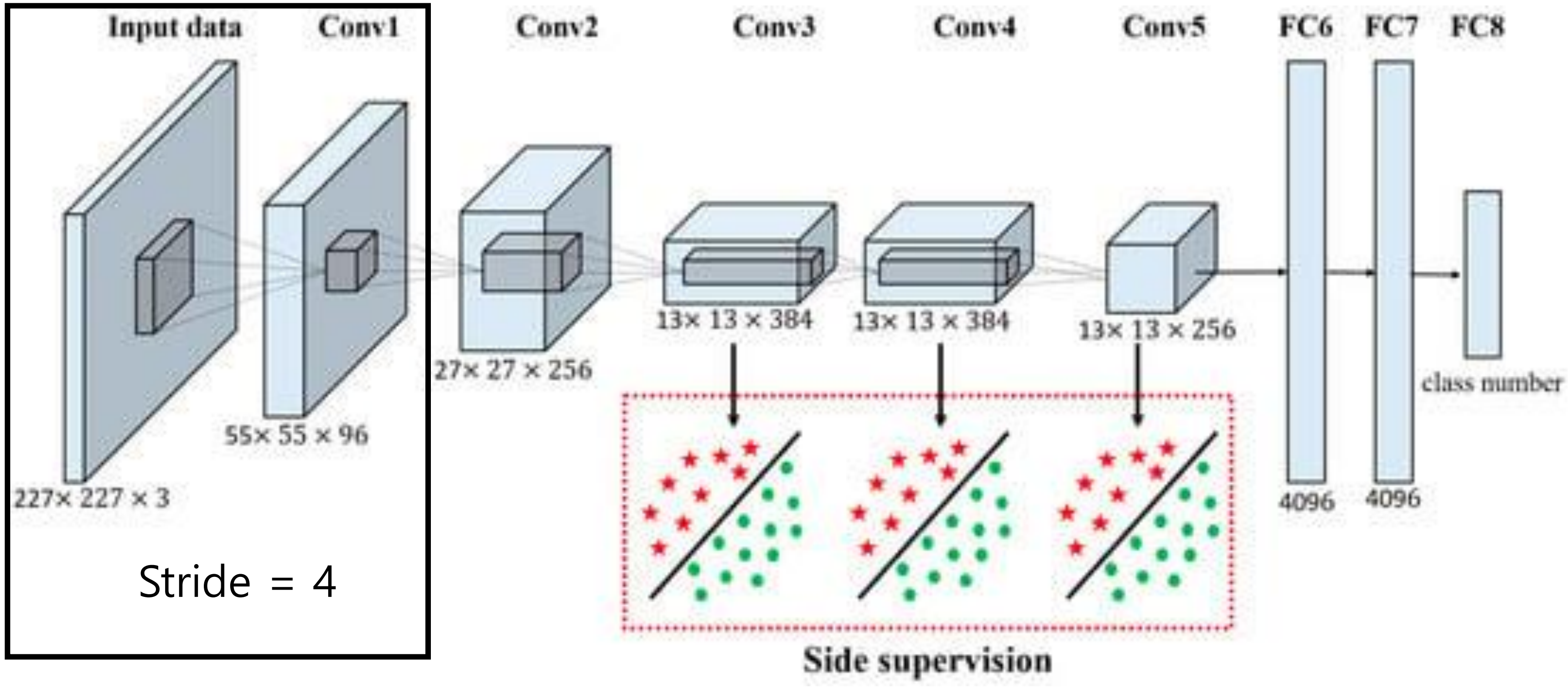
Convolutional Network – Atrous Convolution



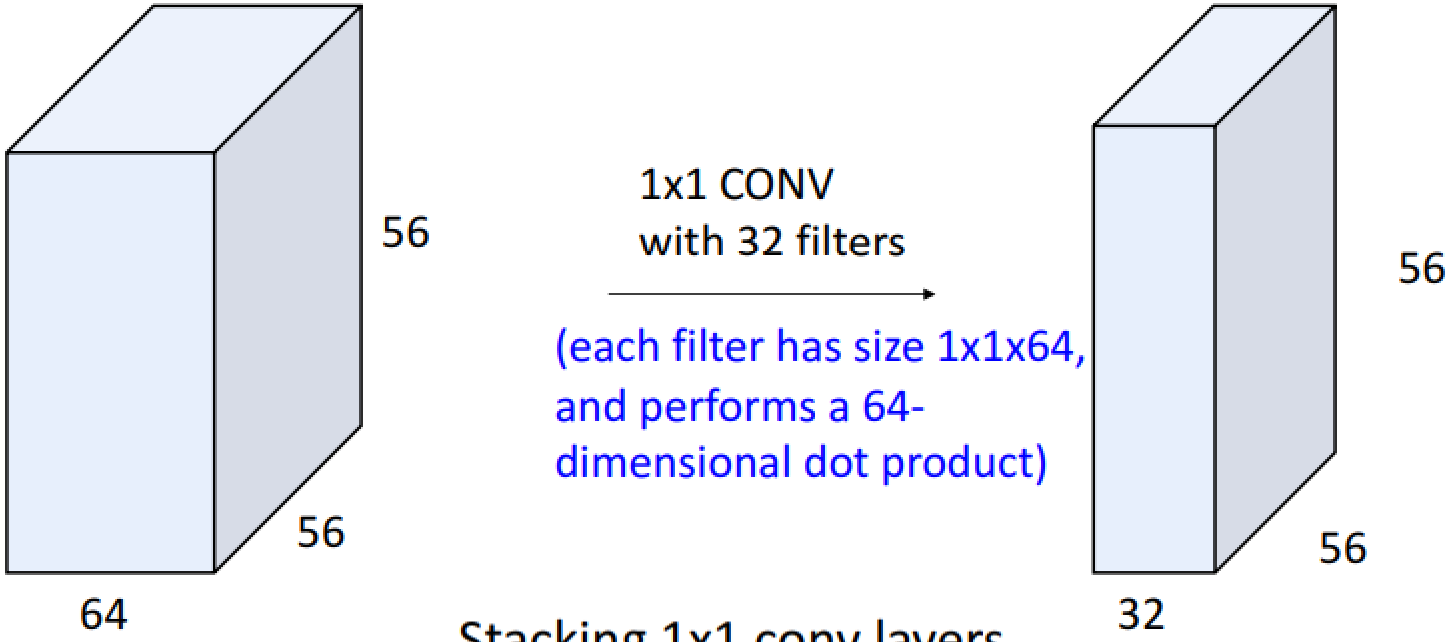


https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d

Convolutional Network – Calculate Parameters

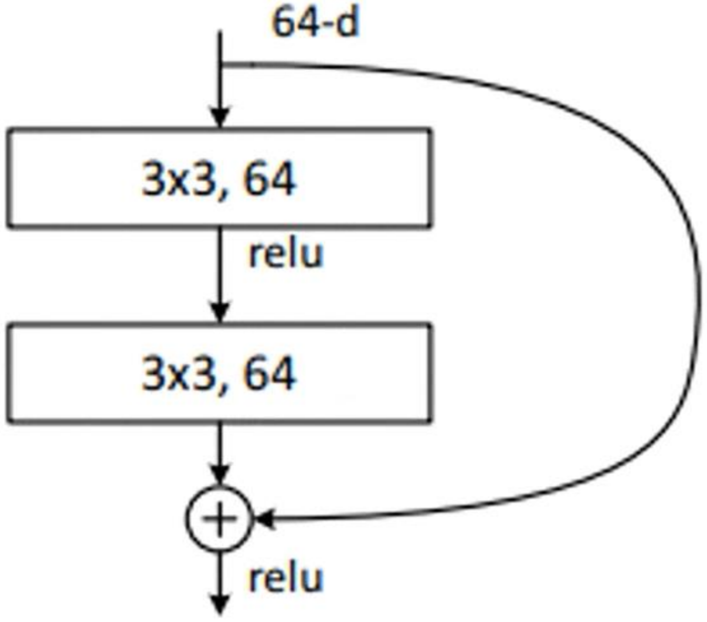


Convolutional Network – 1x1 Convolution

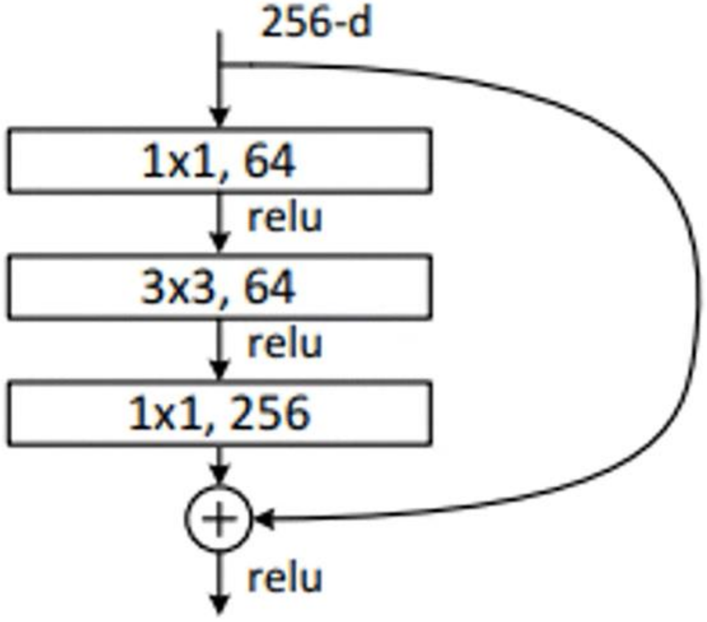


Stacking 1x1 conv layers
gives MLP operating on
each input position

Convolutional Network – 1x1 Convolution



Standard

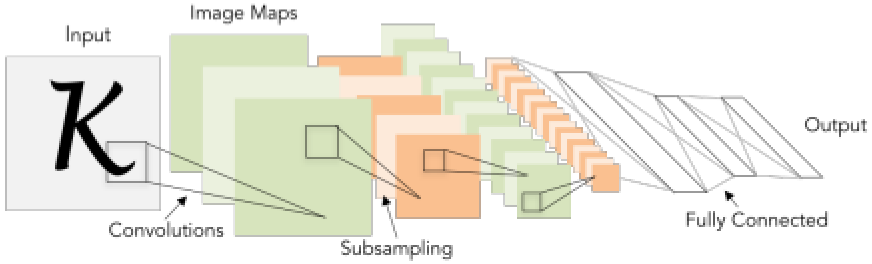


BottleNeck

Convolutional Network – LeNet-5

Example: LeNet-5

Layer	Output Size	Weight Size
Input	1 x 28 x 28	
Conv (C_{out}= , K= P=2, S=1)	20 x 28 x 28	
ReLU	20 x 28 x 28	
MaxPool(K=2, S=2)	20 x 14 x 14	
Conv (C_{out}= , K= P=2, S=1)	50 x 14 x 14	
ReLU	50 x 14 x 14	
MaxPool(K= S=	50 x 7 x 7	
Flatten	2450	
Linear (2450 -> 500)	500	
ReLU	500	
Linear (500 -> 10)	10	



As we go through the network:

Spatial size **decreases**
(using pooling or strided conv)

Number of channels **increases**
(total "volume" is preserved!)

Convolutional Network – Normalization

Internal Covariance Shift 현상은 위 그림처럼 아무리 input layer에서 정규분포를 가지는 입력을 줘도 **hidden layer**를 지나면서 그 분포가 점점 정규분포를 벗어나는 것을 의미한다.

Batch Normalization for convolutional networks

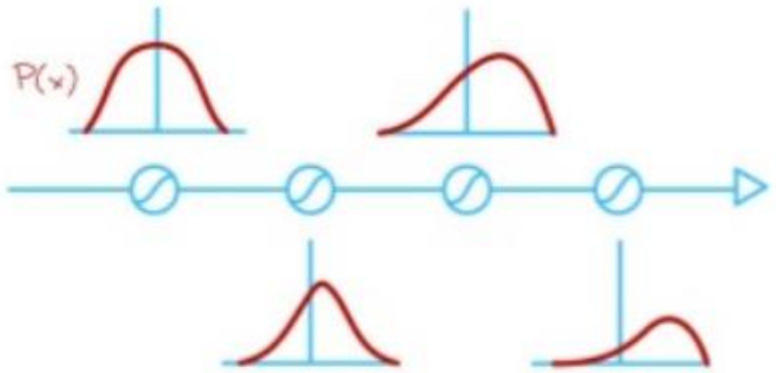
$$x: \; N \times C \times H \times W$$

Normalize

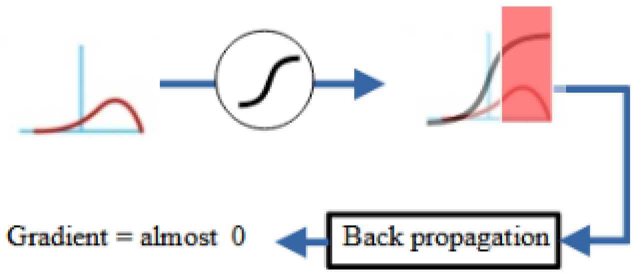
$$\mu, \sigma: \; 1 \times C \times 1 \times 1$$

$$\gamma, \beta: \; 1 \times C \times 1 \times 1$$

$$y = \gamma (x - \mu) / \sigma + \beta$$



위 그림으 4번째 노드의 경우는 상당히 우측으로 치우쳤는데, 저 상태에서 back propagation을 하면 대부분의 분포가 gradient = 0부근(빨간 구역)에 집중되어 있기 때문에 **gradient vanishing** 현상이 발생한다.



Convolutional Network – Normalization

Batch Normalization for convolutional networks

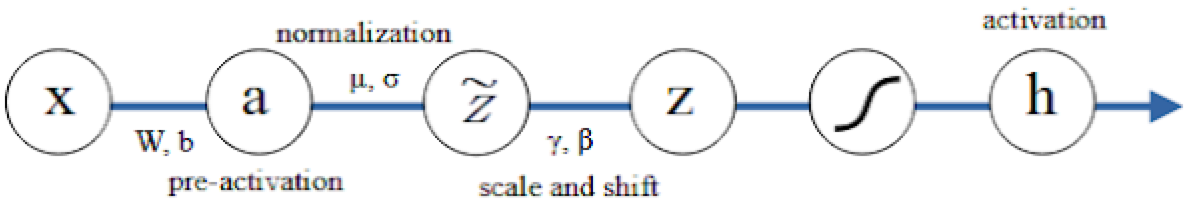
$$\mathbf{x}: \; N \times C \times H \times W$$

Normalize

$$\boldsymbol{\mu}, \boldsymbol{\sigma}: \; 1 \times C \times 1 \times 1$$

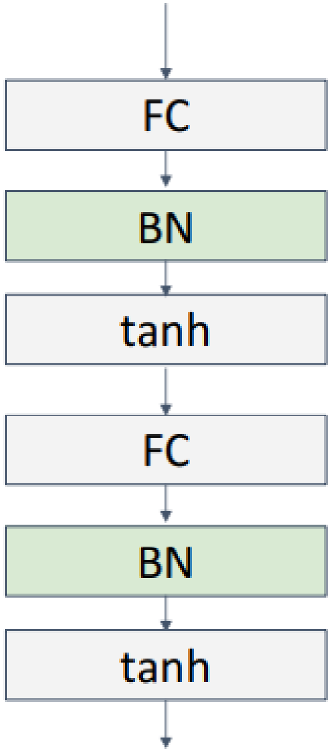
$$\gamma, \beta: \; 1 \times C \times 1 \times 1$$

$$y = \gamma(x - \mu)/\sigma + \beta$$

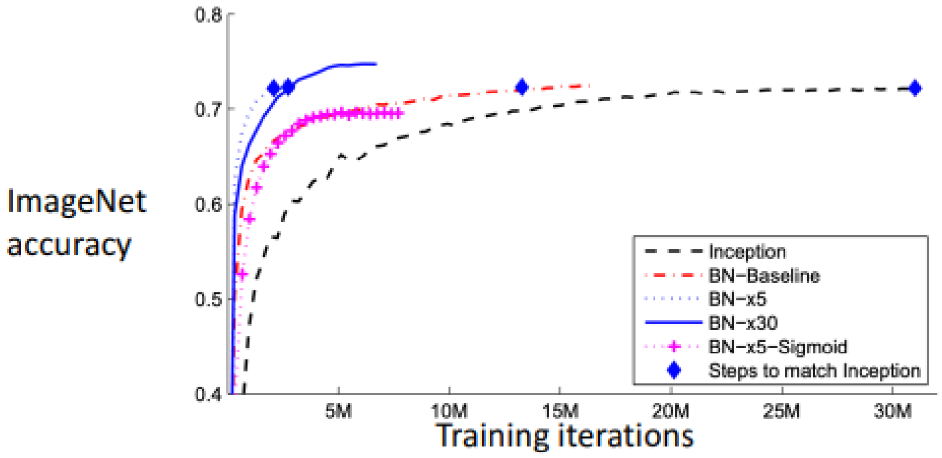


Convolutional Network – Normalization

Batch Normalization



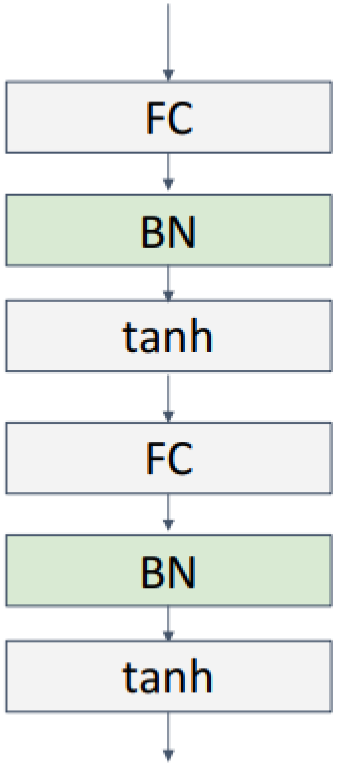
- Makes deep networks **much** easier to train!
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Zero overhead at test-time: can be fused with conv!



edy, "Batch normalization: Accelerating deep

Convolutional Network – Normalization

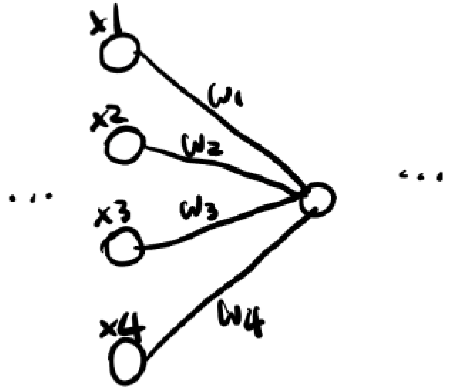
Batch Normalization



- Makes deep networks **much** easier to train!
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Zero overhead at test-time: can be fused with conv!
- Not well-understood theoretically (yet)
- Behaves differently during training and testing: this is a very common source of bugs!

Convolutional Network – DropOut

x1 값 신경이나 쓸까? No -> 이런 상황이 바로 overfitting!
이럴때 weight 값을 줄여주기 위해 loss function에 term을 추가해 해결했던 방식이 L1, L2 Regularization



L1 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M}|W_j|}_{\text{Regularization Term}}$$

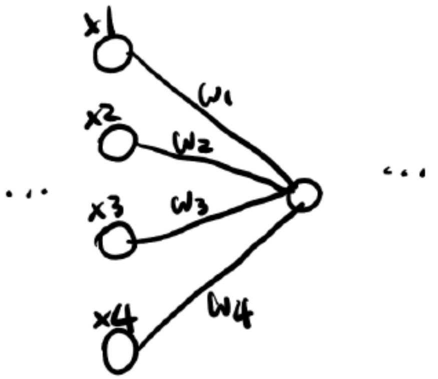
L2 Regularization

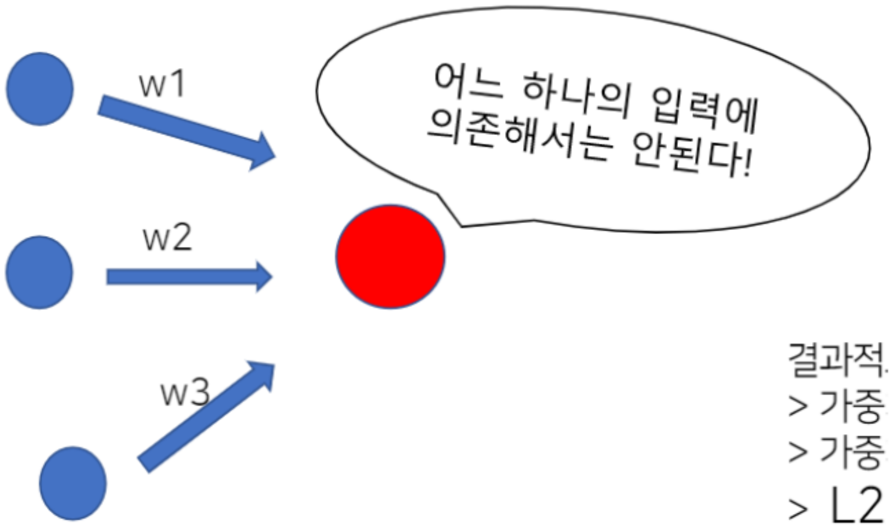
$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Weights	오버피팅이 나타난 경우	정상적인 경우
W1	0.35	0.13
W2	100002342	-0.05
W3	234.2	1.2
W4	-11313434	0.45

Convolutional Network – DropOut

해당 상황 다르게 말하면 '몇몇 node 값에 극단적으로 의존해버리는 상황 발생'
-> 이를 해결하고자 하는 것이 Dropout (loss function의 수정 없이) How?





결과적
> 가중
> 가중
> L2

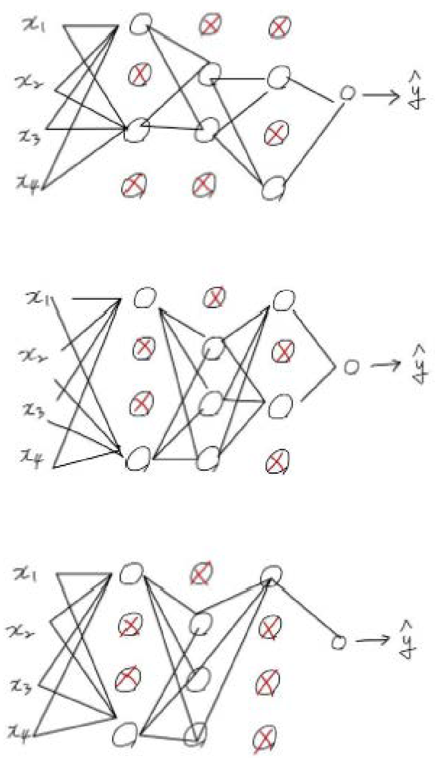
Weights	오버피팅이 나타난 경우	정상적인 경우
W1	0.35	0.13
W2	100002342	-0.05
W3	234.2	1.2
W4	-11313434	0.45

Convolutional Network – DropOut

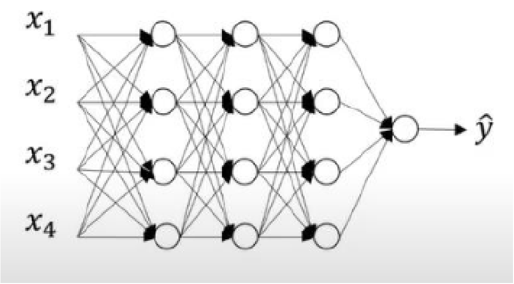
학습 할때마다 매번 랜덤으로 node 골라서 그거 빼고 train해보자!
-> 원래 같았으면 의존했을 node를 아예 빼고 train한 것도 있으니 아까 상황 방지 가능

train 하고나서 test (inference) 할때는 모든 node 사용

(Ensemble 느낌이 있음)

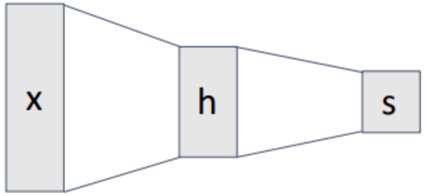


훈련결과를 다 모아서 만든 최종 모델

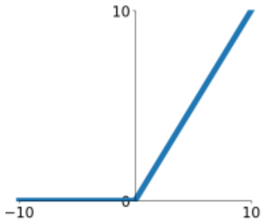


Convolutional Network – Feature Extraction

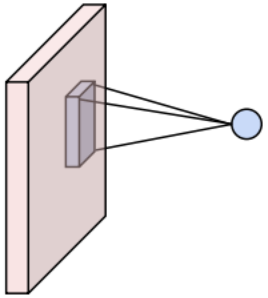
Fully-Connected Layers



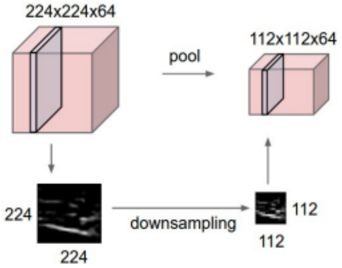
Activation Function



Convolution Layers



Pooling Layers

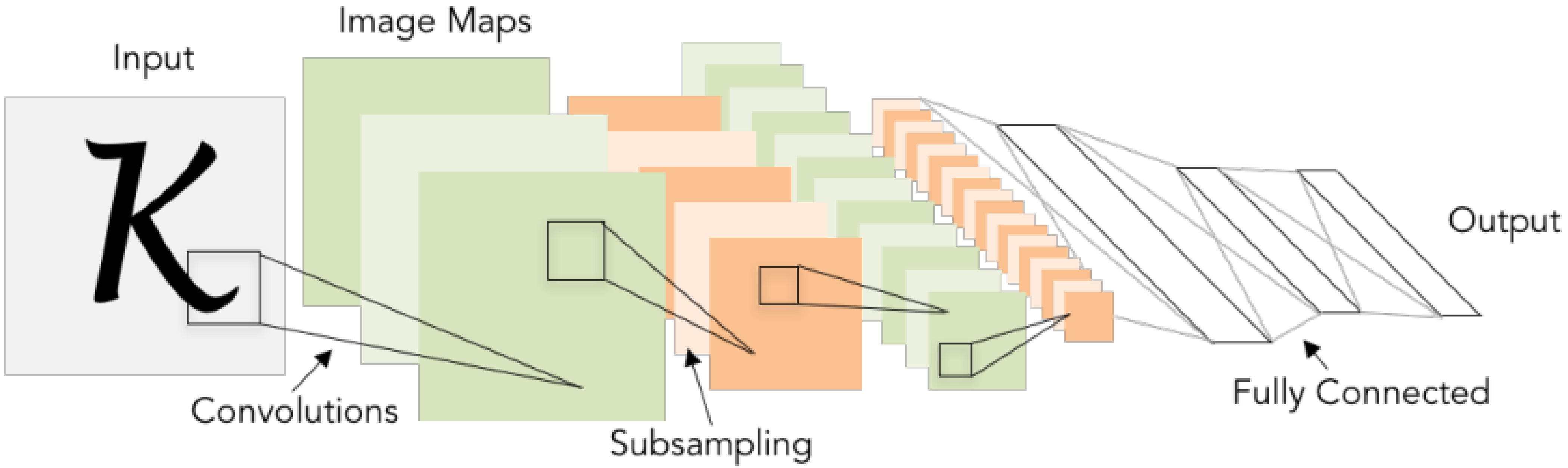


Normalization

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Conv layer ➡ Batch Normalization ➡ ReLU ➡ (Dropout) ➡ (Pooling)

Convolutional Network – Feature Extraction



Conv layer ➡ **Batch Normalization** ➡ **ReLU** ➡ **(Dropout)** ➡ **(Pooling)**

Reference

- EECS 498-007 / 598-005 Deep Learning for Computer Vision, Lecture 5, Lecture 7 slides
- DSL 7기 전재현 CNN 강의안
- https://gaussian37.github.io/
- https://sonsnotation.blogspot.com/
- Yang, Zhuoqian & Dan, Tingting & Yang, Yang. (2018). Multi-Temporal Remote Sensing Image Registration Using Deep Convolutional Features. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2853100.
- https://coding-yoon.tistory.com/116
- https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d