Deep Learning

Machine Learning

Hamid R Rabiee – Zahra Dehghanian

Spring 2025

Sharif University of Technology

History

- 1940s–1960s:
  - development of theories of biological learning
  - implementations of the first models
    - perceptron (Rosenblatt, 1958) for training of a single neuron.

- 1980s-1990s: back-propagation algorithm to train a neural network with more than one hidden layer
  - too computationally costly to allow much experimentation with the hardware available at the time.

- 2006 “Deep learning” name was selected
  - ability to train deeper neural networks than had been possible before
    - Although began by using unsupervised representation learning, later success obtained usually using large datasets of labeled samples

Deep Learning

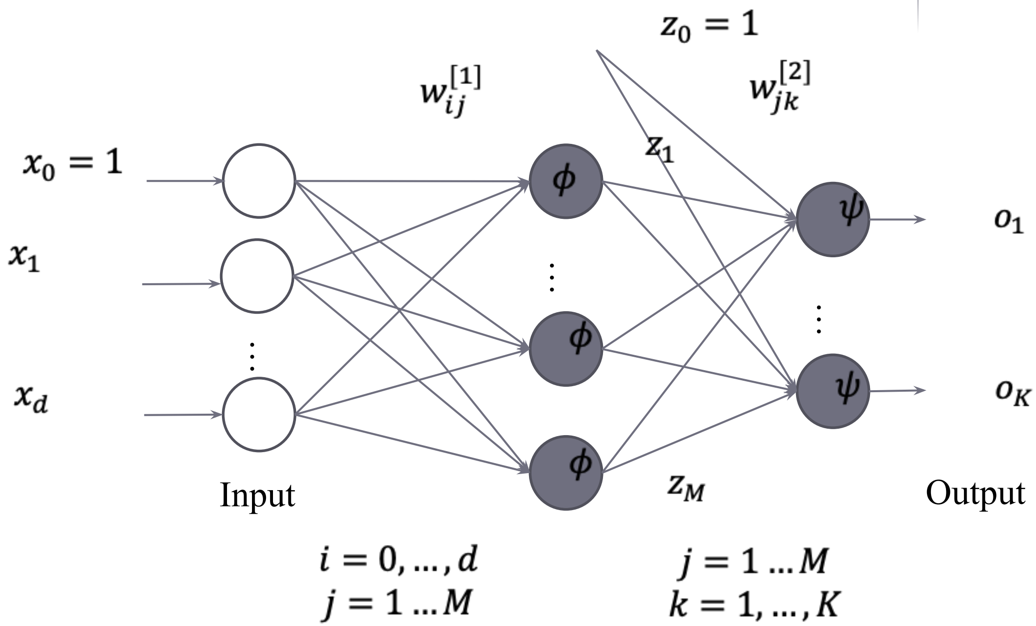
- Learning a computational models consists of multiple processing layers
  - learn representations of data with multiple levels of abstraction.

- Dramatically improved the state-of-the-art in many speech, vision and NLP tasks (and also in many other domains)

MLP with single hidden layer

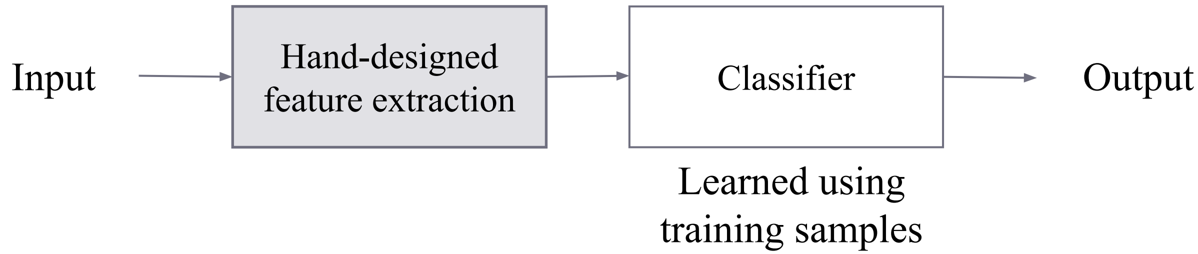
- Two-layer MLP (Number of layers of adaptive weights is counted)

$$o_k(\boldsymbol{x}) = \psi\left(\sum_{j=0}^{M} w_{jk}^{[2]} z_j\right) \Rightarrow o_k(\boldsymbol{x}) = \psi\left(\sum_{j=0}^{M} w_{jk}^{[2]} \underbrace{\phi\left(\sum_{i=0}^{d} w_{ij}^{[1]} x_i\right)}_{z_j}\right)$$



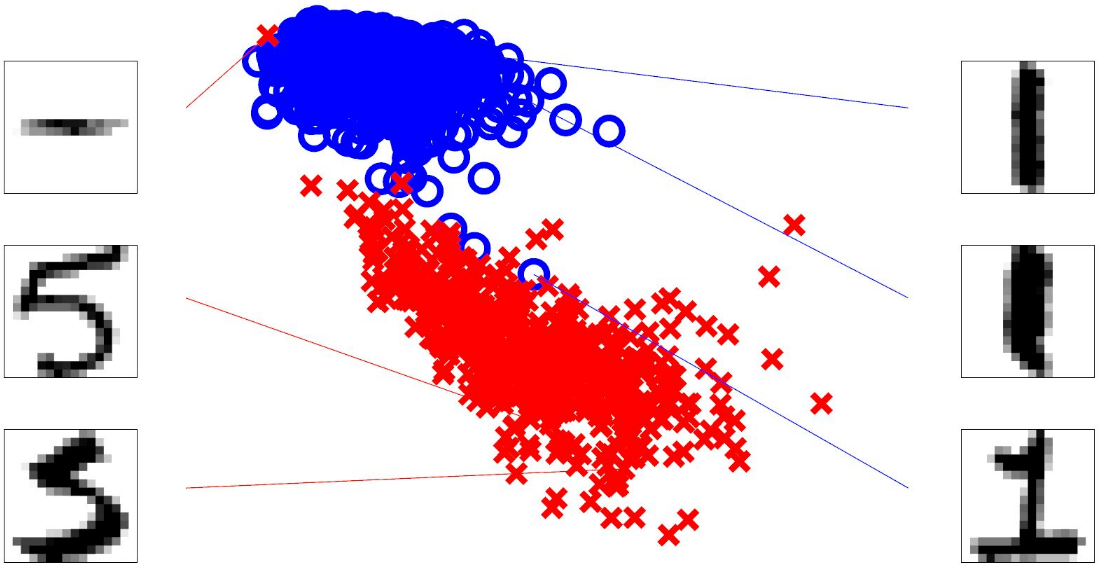
Machine Learning Methods

- Conventional machine learning methods:
  - try to learn the mapping from the input features to the output by samples
  - However, they need appropriately designed hand-designed features



Example

- x_1: intensity
- x_2: symmetry



[Abu Mostafa, 2012]

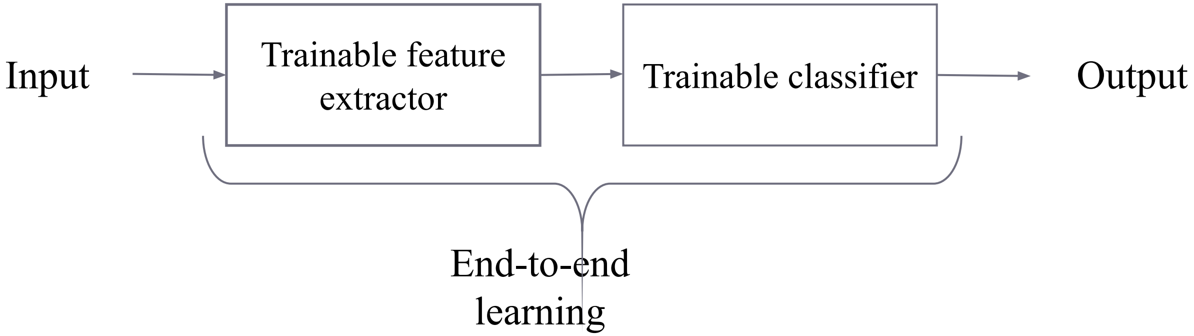
Representation of Data

- Performance of traditional learning methods depends heavily on the representation of the data.
  - **Most efforts were on designing proper features**

- However, designing hand-crafted features for inputs like image, videos, time series, and sequences is not trivial at all.
  - It is difficult to know which features should be extracted.
    - Sometimes, it needs long time for a community of experts to find (an incomplete and over-specified) set of these features.

Representation Learning

- Using learning to discover both:
  - the representation of data from input features
  - and the mapping from representation to output

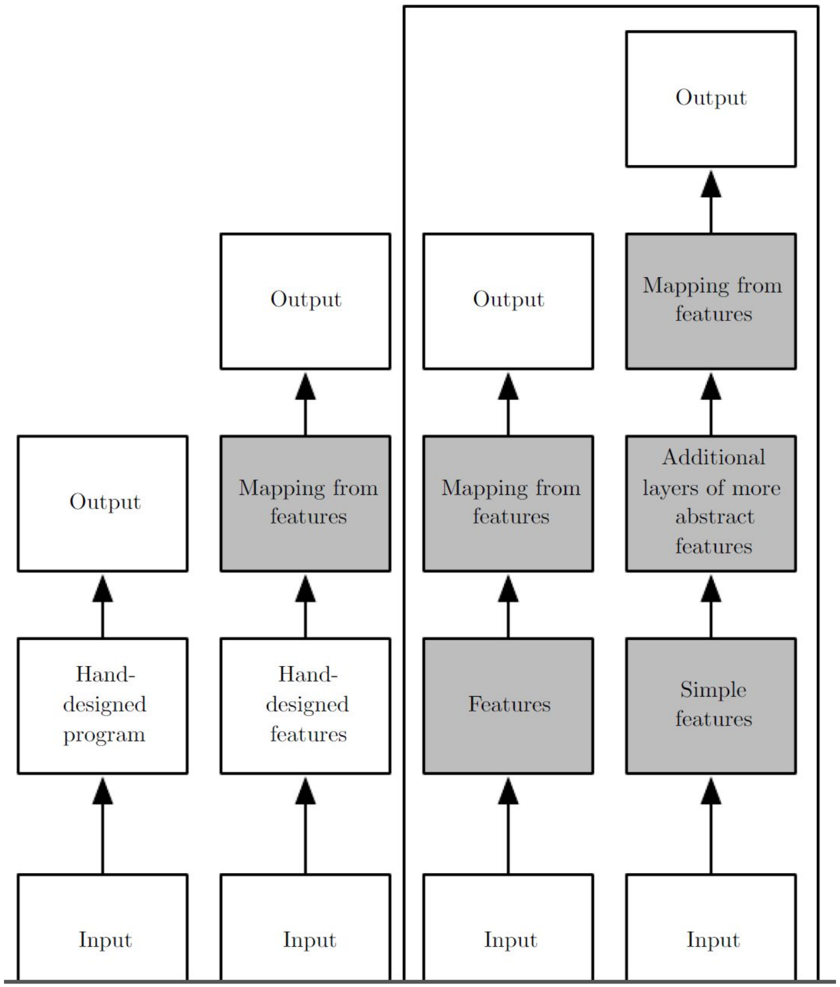


Deep networks

- Deeper networks (with multiple hidden layers) can work better than a single-hidden-layer networks is an empirical observation
  - despite the fact that their representational power is equal.

- In practice usually 3-layer neural networks will outperform 2-layer nets, but going even deeper may not help much more.
  - This is in stark contrast to Convolutional Networks

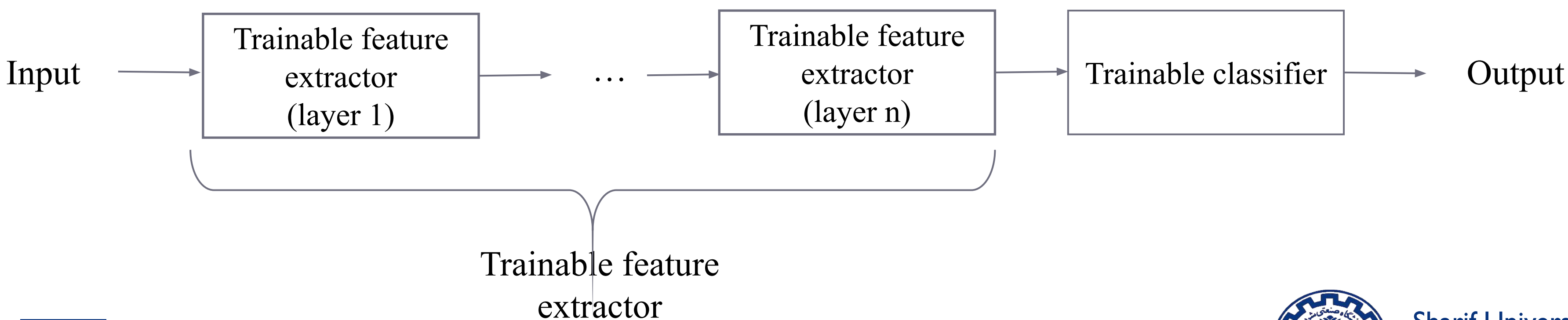
Deep Learning Approach



Deep Learning Approach

- Deep breaks the desired complicated mapping into a series of nested simple mappings
  - each mapping described by a layer of the model.
  - each layer extracts features from output of previous layer

- shows impressive performance on many Artificial Intelligence tasks



Deep Representations: The Power of Compositionality

- Compositionality is useful to describe the world efficiently
  - Learned function seen as a composition of simpler operations
  - Hierarchy of features, concepts, leading to more abstract factors enabling better generalization
    - each concept defined in relation to simpler concepts
    - more abstract representations computed in terms of less abstract ones.
  - Again, theory shows this can be exponentially advantageous

- Deep learning has great power and flexibility by learning to represent the world as a nested hierarchy of concepts

This slide has been adopted from Yoshua Bengio's slides

Deep learning

- Use networks with many layers

- A single hidden layer with enough units can approximate any target network
  - More layers more closely mimics human learning
  - We may need far less number of nodes when we use deep networks

- A hierarchy of internal representations for the input.
  - The first layer constructs a low-level representation;
  - More complex representations in terms of simpler representation of the previous layer

Boolean functions

- Input: N Boolean variable
- How many neurons in a one hidden layer MLP is required?
- More compact representation of a Boolean function
  - "Karnaugh Map"
    - representing the truth table as a grid
    - Grouping adjacent boxes to reduce the complexity of the Disjunctive Normal Form (DNF) formula

W,Z \ X,Y	00	01	10	11
00	1	1	1	1
01				
10		1	1	
11	1			1

Worst case

- Which truth tables cannot be reduced further simply?

- Largest width needed for a single-layer Boolean network on N inputs
  - Worst case: 2^{N-1}
    - Example: Parity function

W,Z \ X,Y	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

$X \oplus Y \oplus Z \oplus W$

Using deep network: Parity function on N inputs

- Simple MLP with one hidden layer:

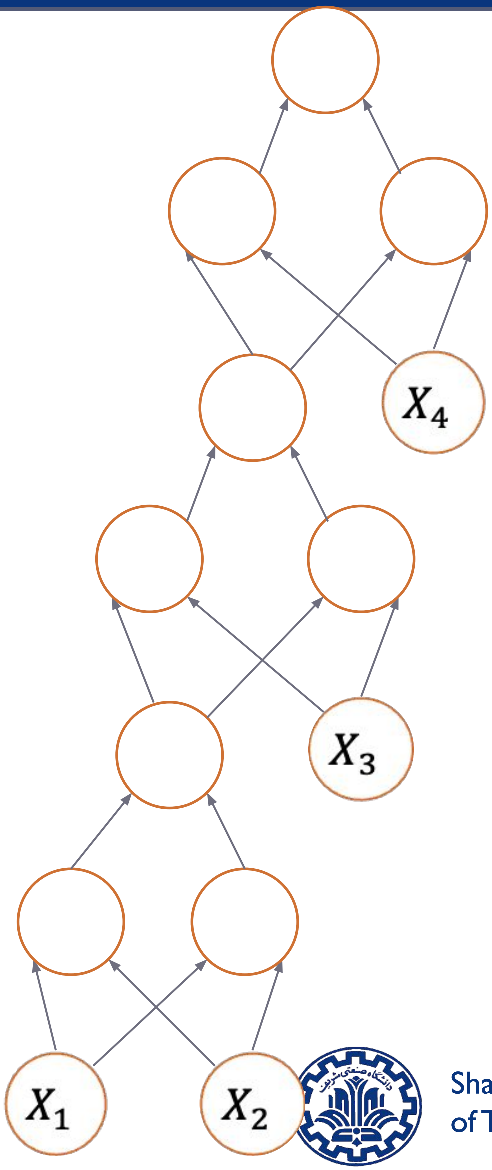
  2^{N-1} Hidden units

  $(N+2)2^{N-1}$ Weights and biases

- $f = X_1 \oplus X_2 \oplus \cdots \oplus X_N$

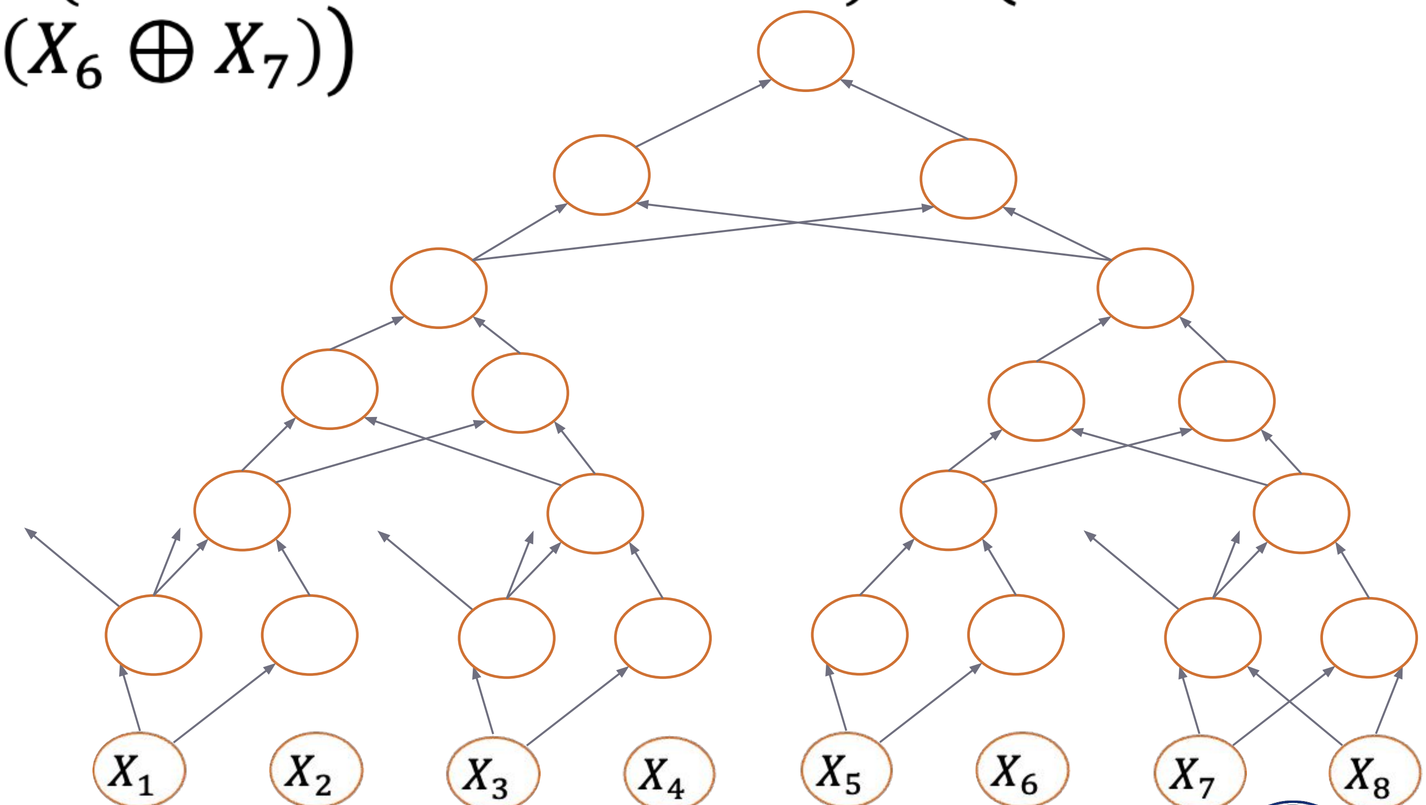
  $3(N-1)$ Hidden nodes

  $9(N-1)$ Weights and biases



A better architecture

- Only requires $2\log N$ layers
- $f = \left(\left(X_1 \oplus X_2\right) \oplus \left(X_3 \oplus X_4\right)\right) \oplus \left(\left(X_4 \oplus X_5\right) \oplus \left(X_6 \oplus X_7\right)\right)$



Boolean function: Wide vs. deep network

- MLP with a single hidden layer is a universal Boolean function

- However, a single-layer network might need an exponential number of hidden units w.r.t. the number of inputs

- Deeper networks may require far fewer neurons than the single hidden layer network
  - Linear w.r.t. the number of inputs when that is deep enough

Why does deep learning become popular?

- Large datasets
- Availability of the computational resources to run much larger models
- New techniques to address the training issues

Training issues

- The backpropagation algorithm is an efficient way of computing the derivative of the cost function w.r.t. each of the weights

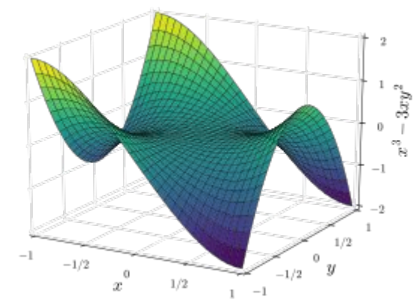
- However, many issues must be considered to have successful training:
  - Optimization issues
  - Generalization issues

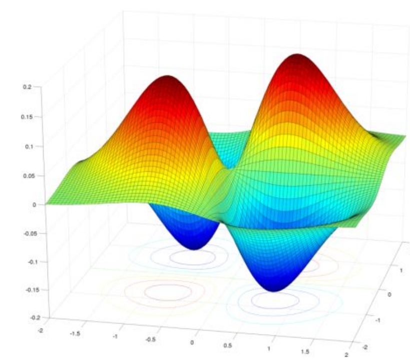
Optimization issues

- Problems with gradient descent
  - saddle points
  - Platueaux
  - poor conditioning
  - …

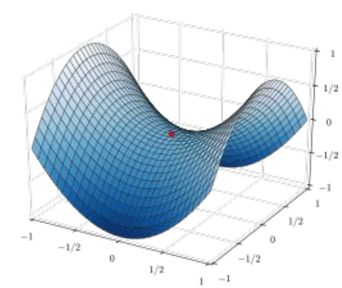
Saddle point

- **Popular hypothesis**:
  - In large networks, saddle points are far more common than local minima
  - This is not true for small networks





- **Saddle point:** A point where
  - The slope is zero
  - The surface increases in some directions, but decreases in others
  - Gradient descent algorithms often get "stuck" in saddle points

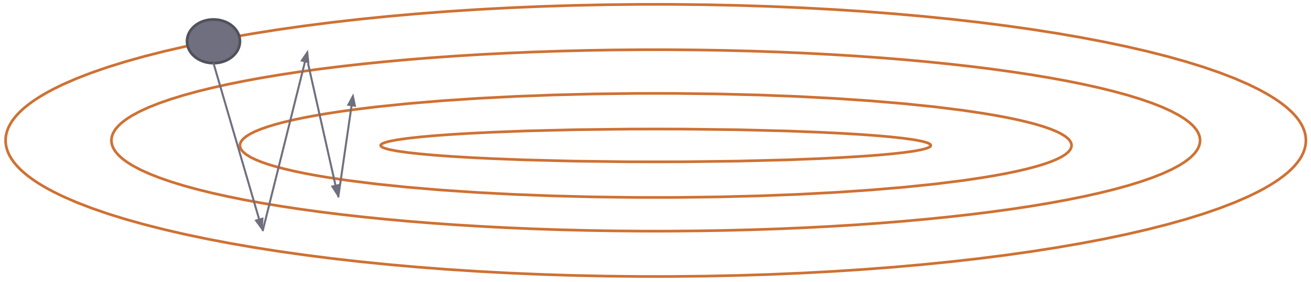


Plateaux

- A flat region of cost function
  - When the gradient is always close to zero in a region, then the weights will not change.

- Saturated units can lead to plateau
  - The derivative of this units in the saturation region is close to zero.

Poor conditioning



- We need greeter gradients in the horizontal direction but we receive a larger gradient in the vertical direction

Optimization issues

- Problems with gradient descent
  - saddle points
  - Platueaux
  - poor conditioning
  - …

Optimization issues

- Problems with gradient descent
  - saddle points
  - Plateaux
  - poor conditioning
  - …
- Choices affecting optimization
  - Learning rate (and learning rate decay)
  - Batch size
  - Weight Initialization
  - (Input) Normalization
  - Activation functions
  - …

Weight initialization

- Initialize weights near zero
  - Thus, network (with sigmoid activation function) initially is near linear and can gradually get non-linear

- Small random numbers (e.g. $w \sim N(0,0.01)$)
  - Doesn't work with deeper networks.
    - After some layers all activations become (near) zero

Xavier initialization

- To have similar variances for neurons outputs:
  - neurons with larger number of inputs the incoming weights are scaled down to reach comparable variance for different nodes

$$z = w_1 x_1 + \cdots . + w_r x_r$$

- Initialization: Gaussian with **zero mean** and **$1/\sqrt{\text{fan\_in}}$ standard deviation**
  - fan_in for fully connected layers = number of neurons in the previous layer
  - Thus, scale down weights variances when there exist higher fan in

- Helps to reduce exploding and vanishing gradient problem

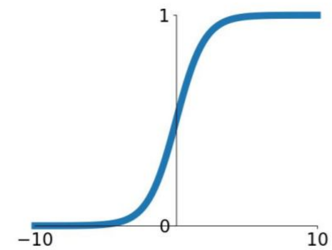
[Glorot et al., 2010]

Input normalization

- Normalize inputs to zero mean and unit variance
- Batch normalization was introduced to normalize the activation of hidden units too
- To alleviate poor conditioning or ravines in the optimization landscape

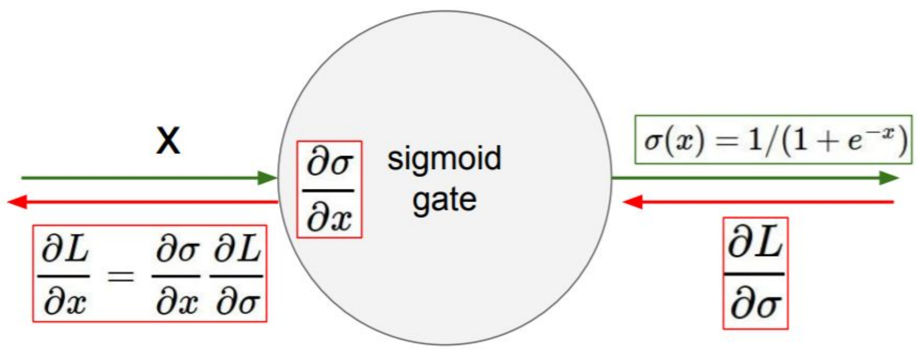
Activation functions: sigmoid

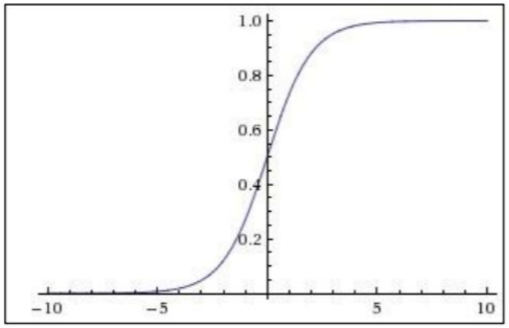
- Squashes numbers to range [0,1]



Sigmoid

- Saturated neurons "kill" the gradients





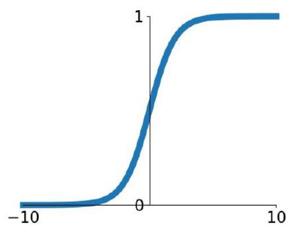
$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$

Activation functions

- Sigmoid and tanh are traditional activation functions
- Many new activation function since 2012

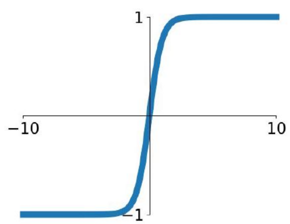
Sigmoid

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



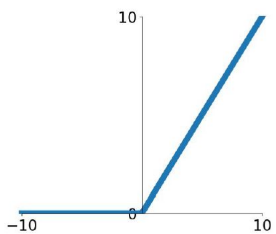
tanh

$$\tanh(x)$$



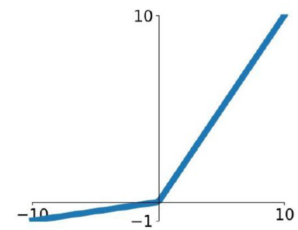
ReLU

$$\max(0, x)$$



Leaky ReLU

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

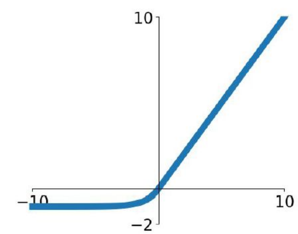


Maxout

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

ELU

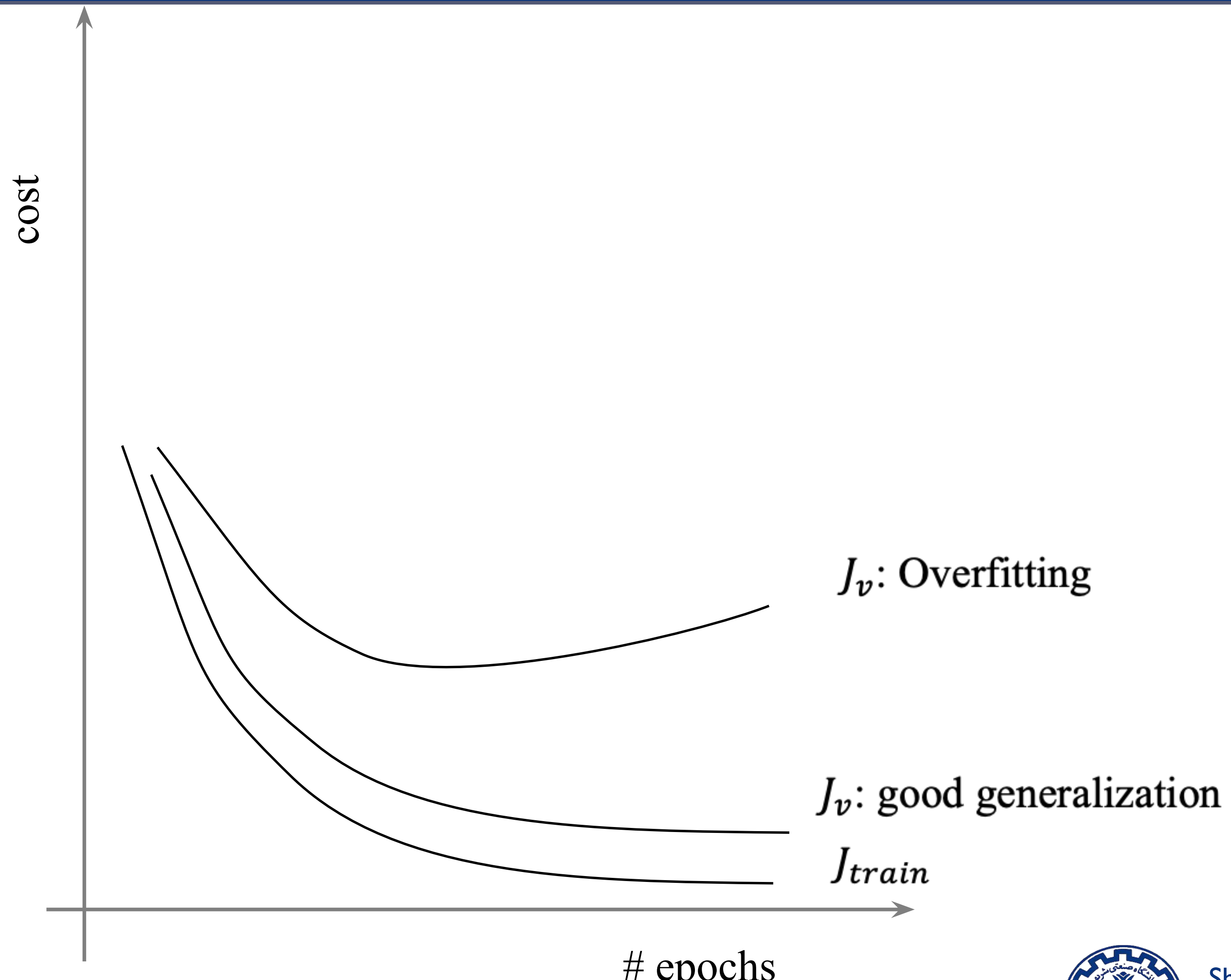
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Generalization techniques

- Regularization or weight-decay
- Hyper-parameter tuning
- Early stopping
- Model ensemble
- Weight-sharing
  - e.g., CNNs
- Pre-training
- Data augmentation
- Dropout
- Batch Normalization
- …

Generalization



Regularization

-

$$J(W) = \frac{1}{N}\sum_{n=1}^{N} L^{(n)}(W) + \lambda R(W)$$

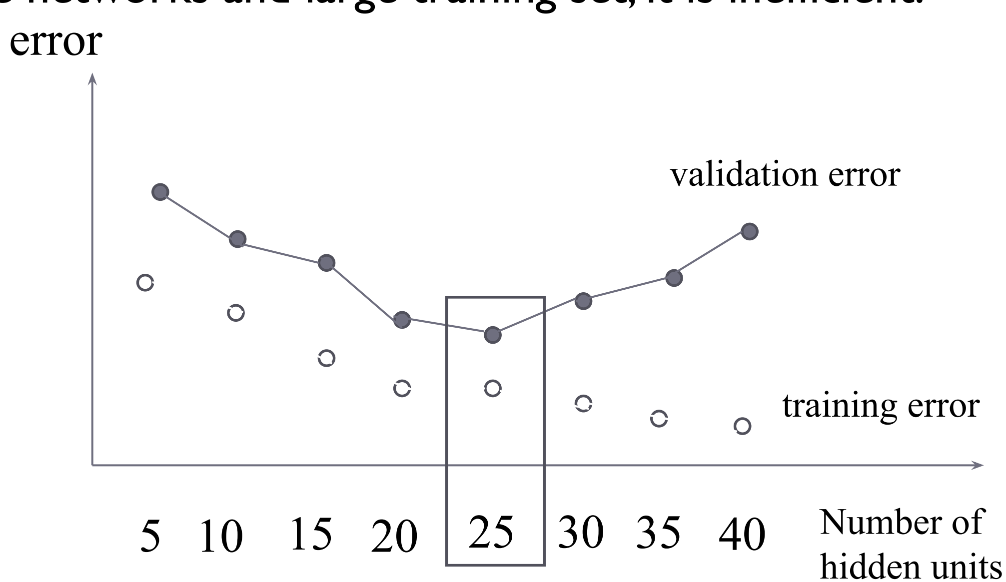
- $R(W)$: is defined based on the norm of the weights vectors
  - Example: $R(W) = \sum_l \sum_{i,j} {w_{ij}^{[l]}}^2$

Regularization can prevent overfitting

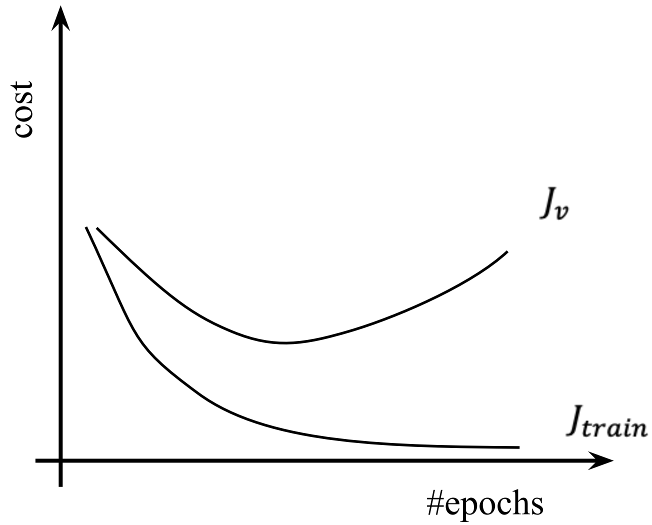
- Small W leads to linear regime of activation functions like sigmoids
- A deep network with small W can also act as a near linear function

Hyper-parameter tuning: Example Number of Hidden Units

- Shows the expressive power the network
  - Can specify the total numbers of weights that are the number of freedom degree
- Select among networks with different no. of hidden units by training these networks and then evaluating them on a validation set
  - For large networks and large training set, it is inefficient.



Early stopping

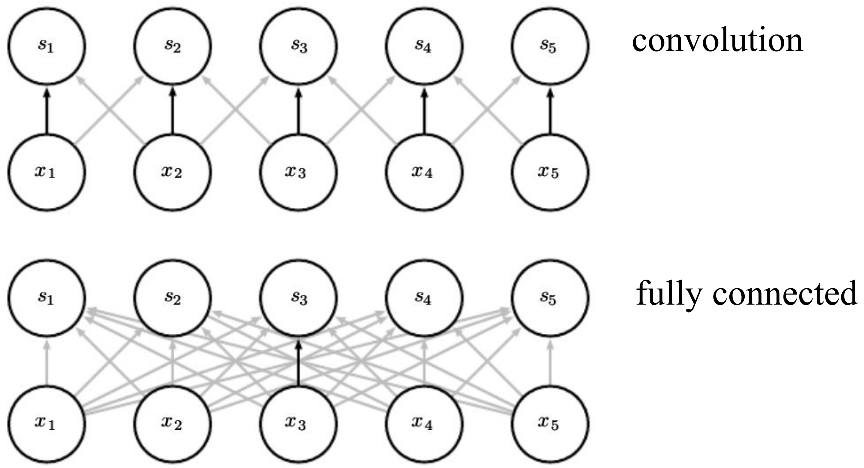


- Stopping gradient descent early (instead of finding the best number of epochs after trying a wide range)
  - However, it separates generalization and approximation issues

Convolutional layer vs. fully connected layer

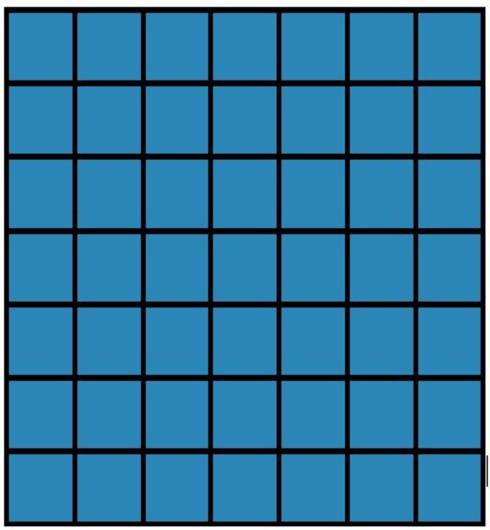
- **Parameter sharing**

(Black arrows indicate the connections that use a particular parameter in two different models)

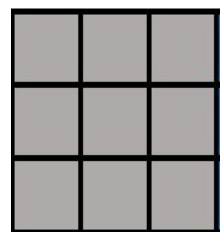


[Goodfellow et al. 2016]

Convolutional filter

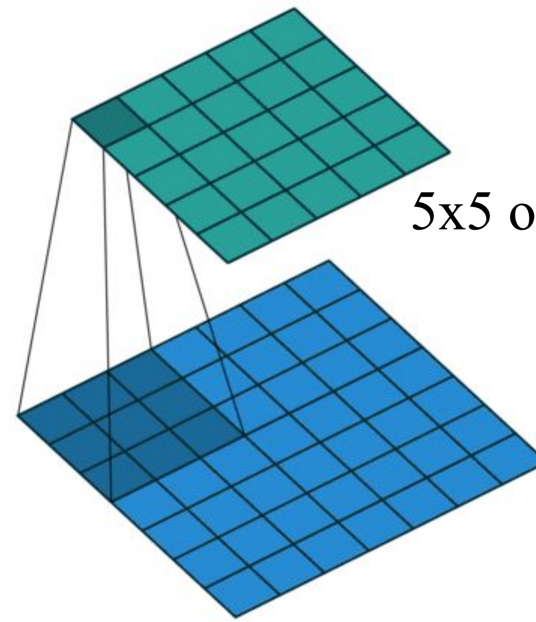


7x7 input



3x3 filter

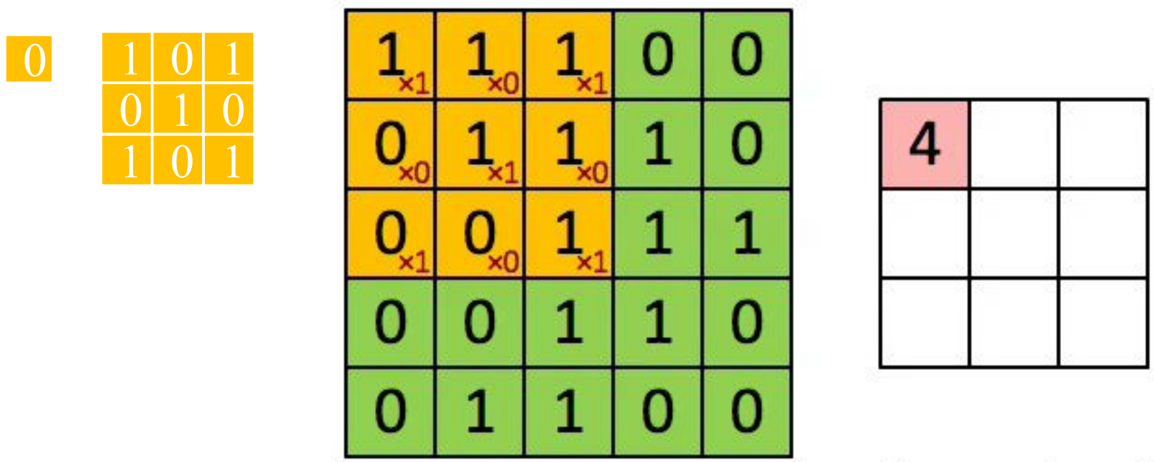
Gives the responses of that filter at every spatial position



5x5 output

Source:
http://iamaaditya.github.io/2016/03/one-by-one-convolution/

What is a convolution

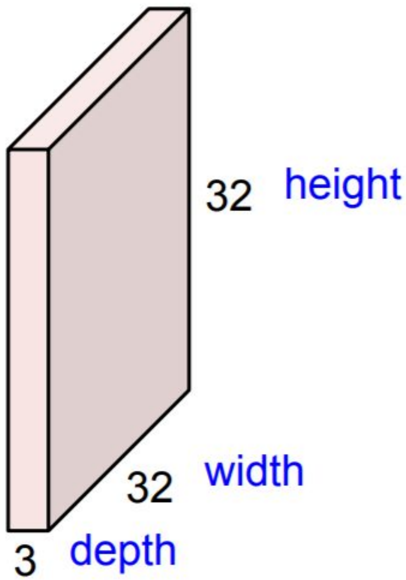


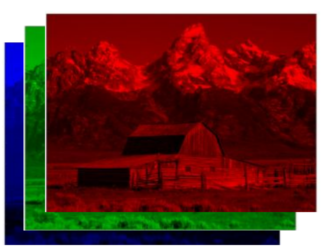
Convolved Feature

- Scanning an image with a “filter”
  - Note: a filter is really just a perceptron, with weights and a bias
  - At each location, the “filter and the underlying map values are multiplied component wise, and these are added along with the bias

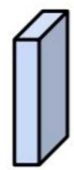
Convolution

32x32x3 image -> preserve spatial structure



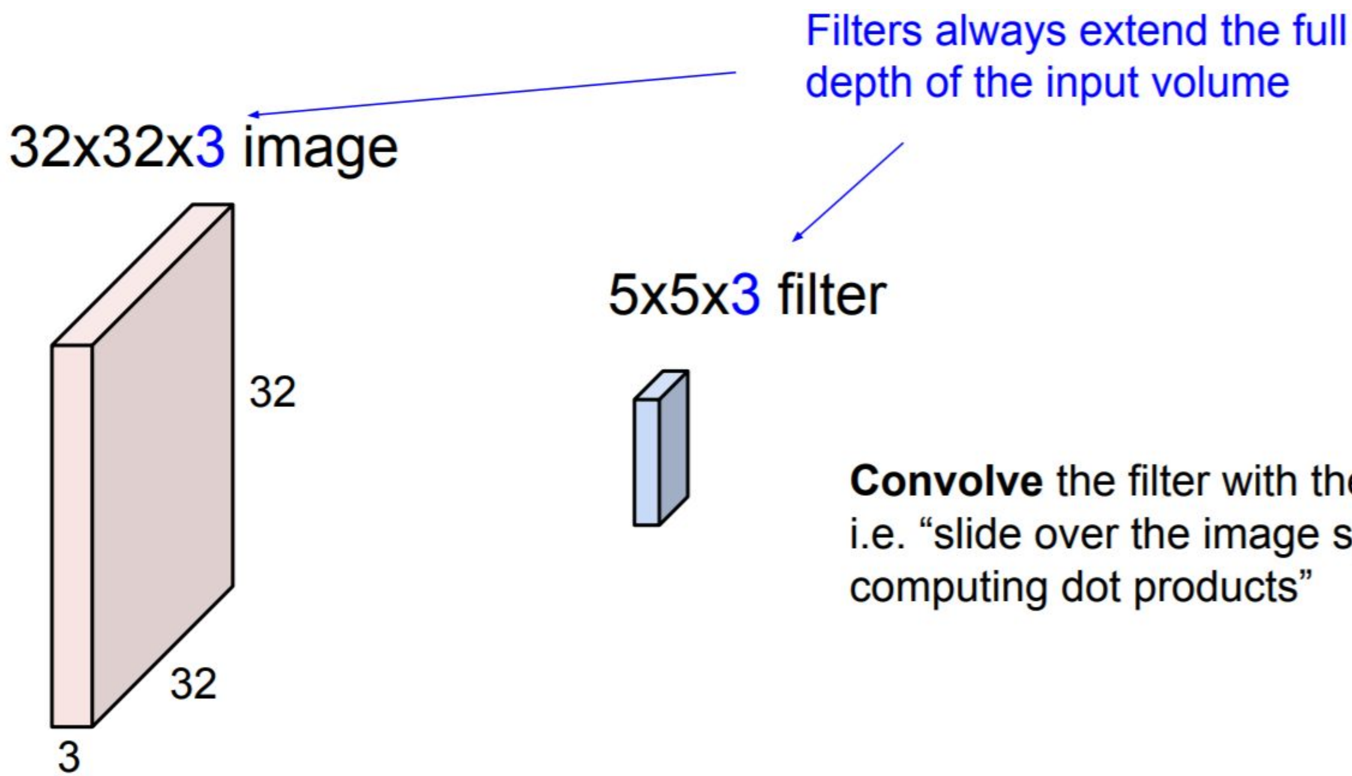


5x5x3 filter



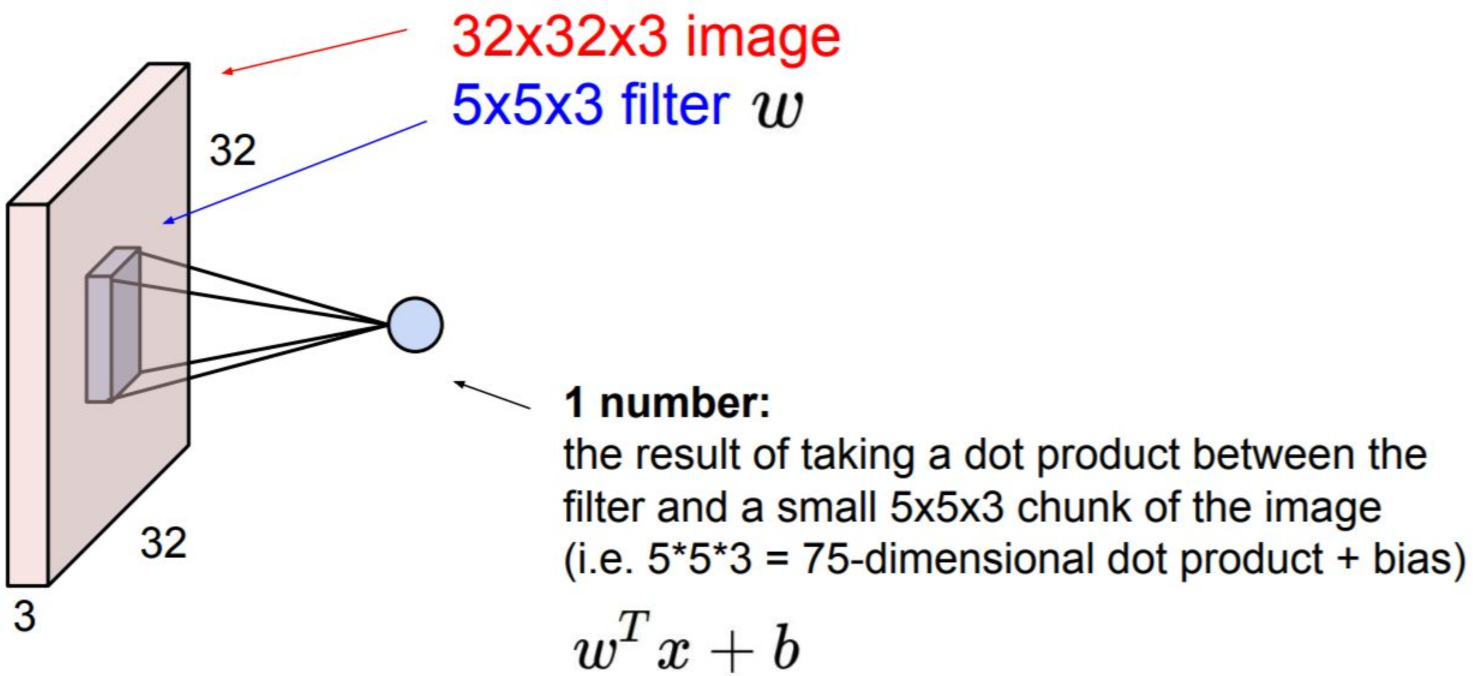
Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution



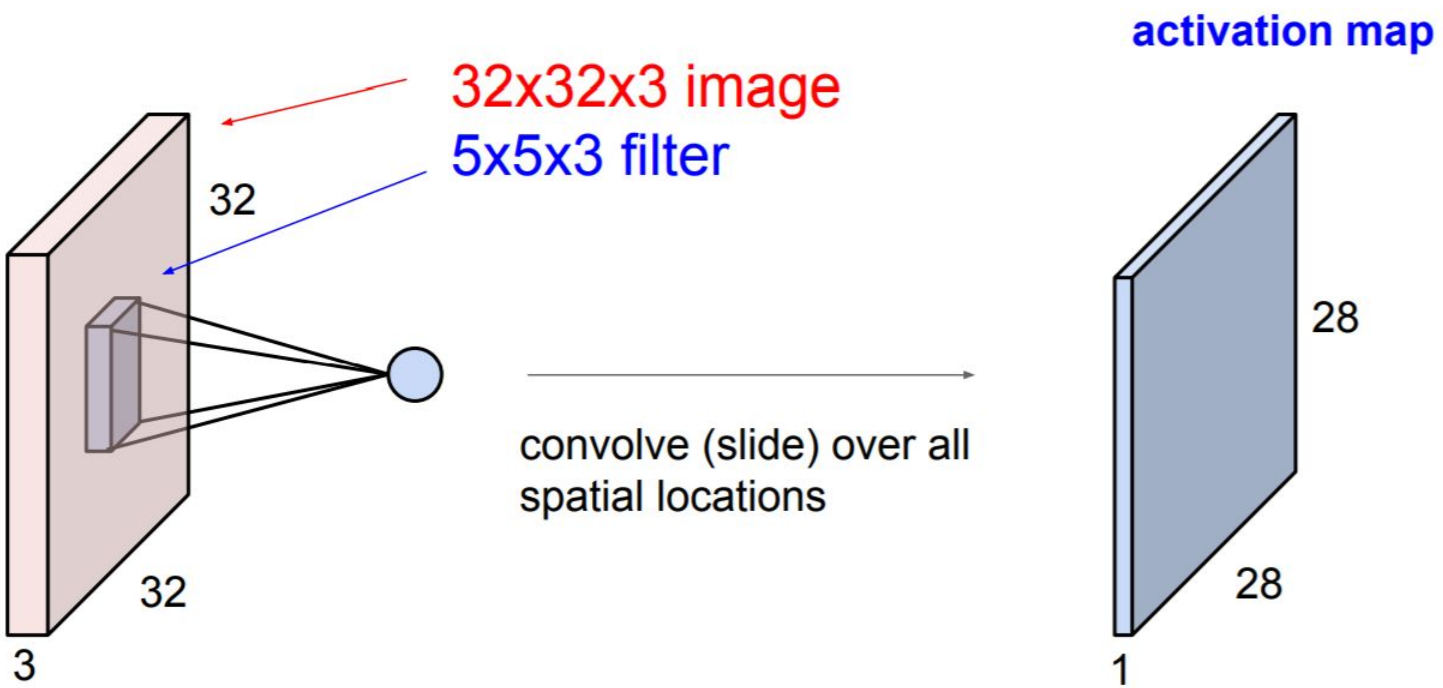
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution

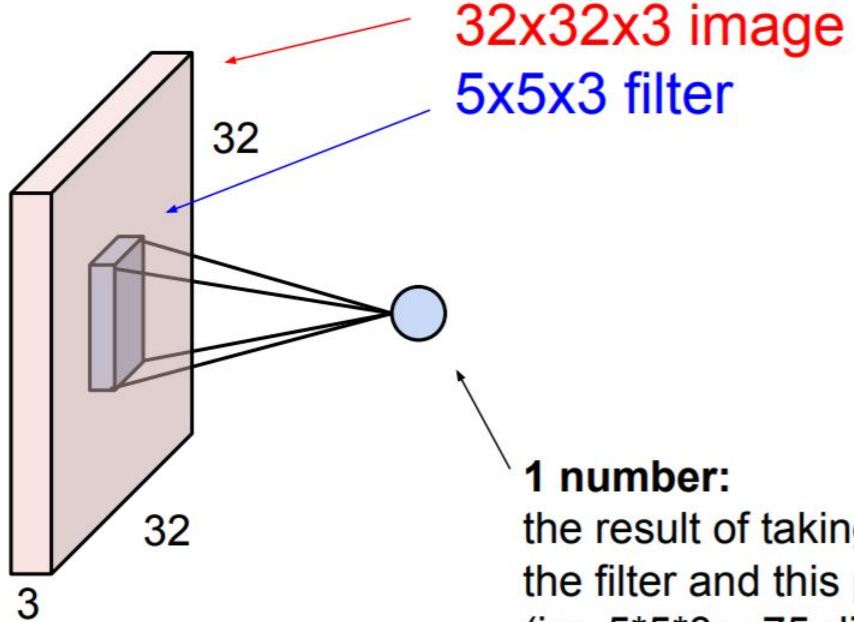


Local connections spatially but full along the entire depth of the input volume.

Convolution



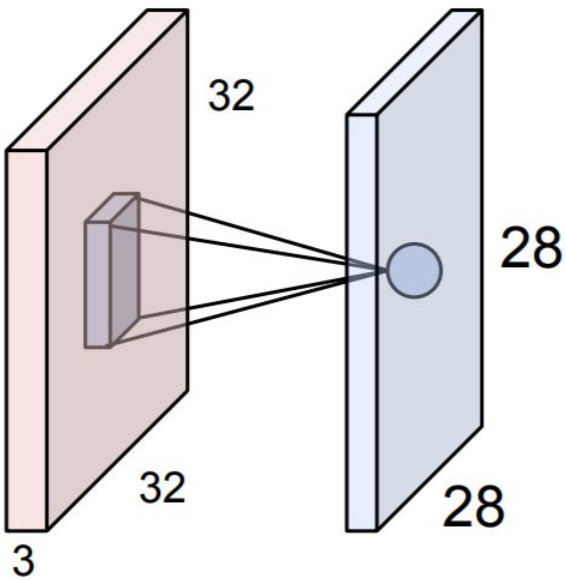
Convolutional layer: neural view



It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Convolutional layer: neural view



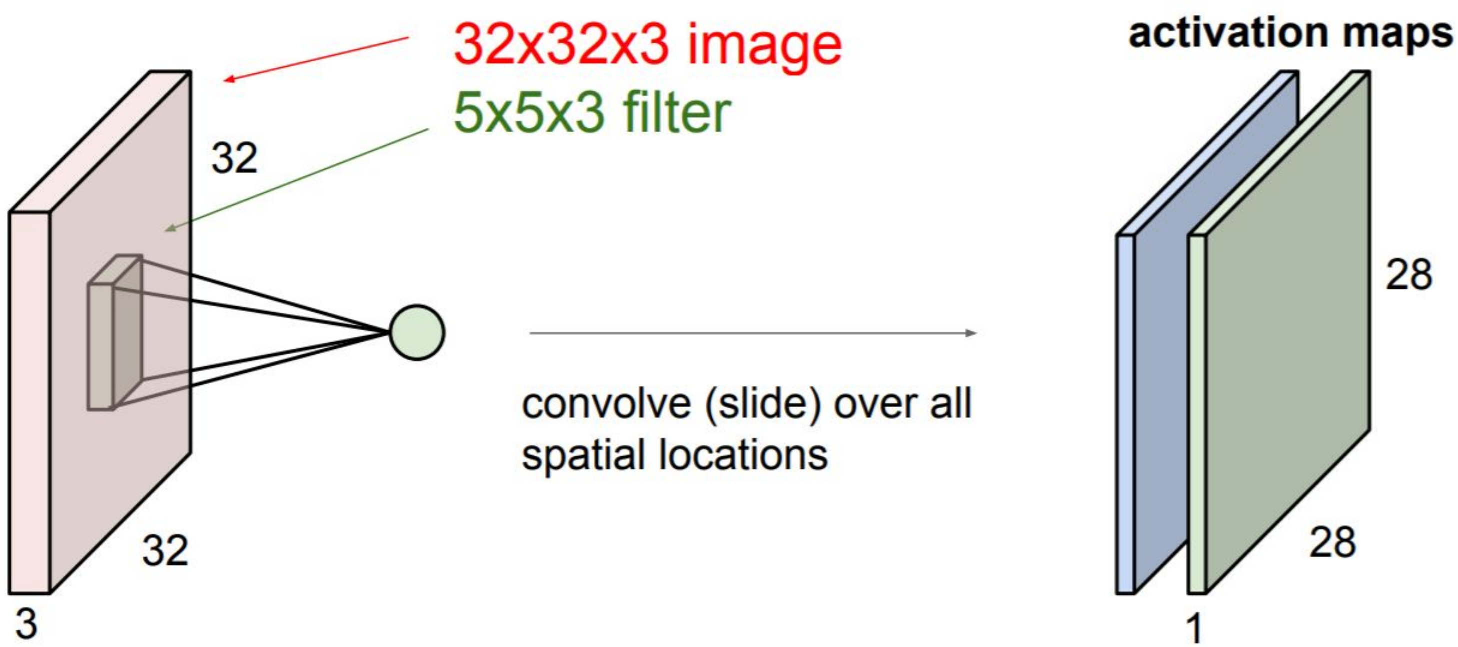
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters "5x5x3"

"5x5 filter" => "5x5 receptive field for each neuron"

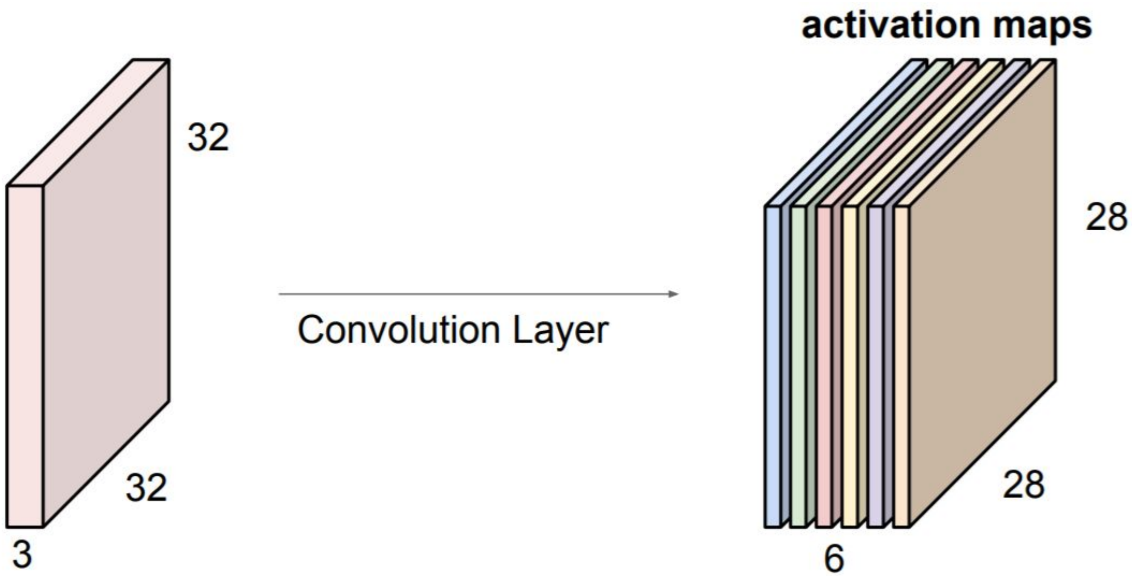
Convolution: Feature maps or activation maps

consider a second, green filter



Convolution: Feature maps or activation maps

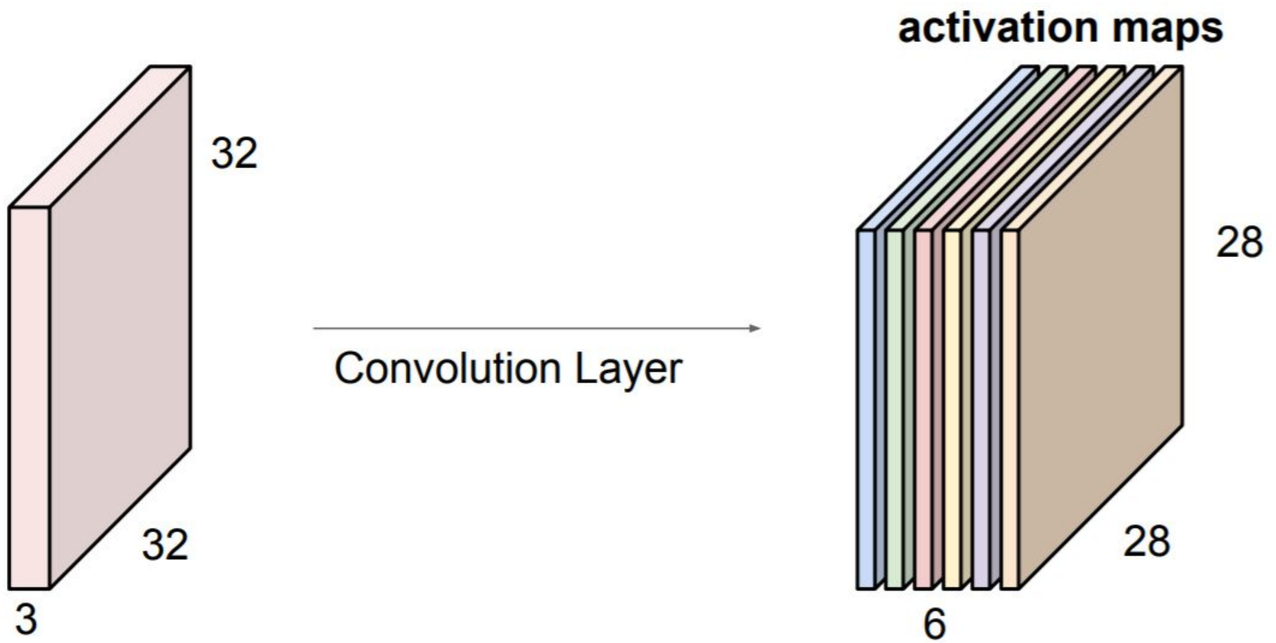
- If we had 6 5x5 filters, we'll get 6 separate activation maps:



- We stack these up to get a "new image" of size 28x28x6!
  - **depth** of the output volume equals to the number of filters

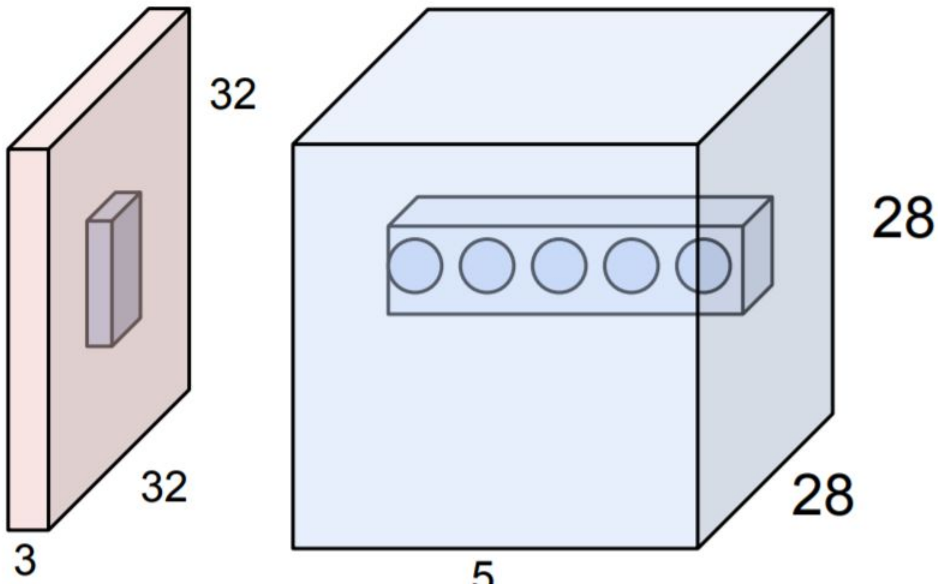
Convolutional layer: neural view

- If we had 6 “5x5 filters”, we’ll get 6 separate activation maps:



There will be 6 different neurons all looking at the same region in the input volume
constrain the neurons in each depth slice to use the same weights and bias

Convolutional layer: neural view



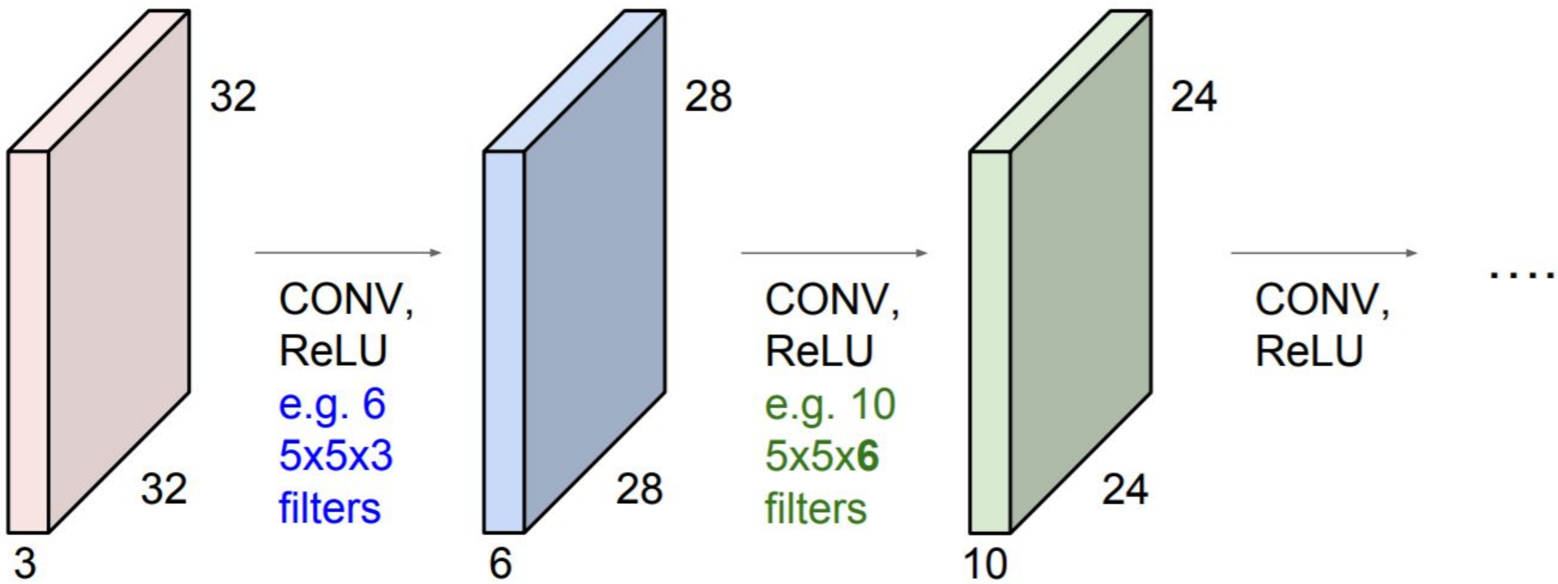
set of neurons that are all looking at the same region of the input as a **depth column**

E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

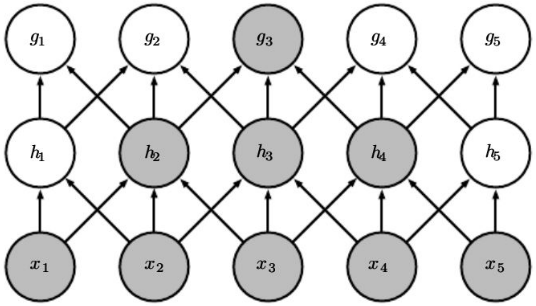
ConvNet

- Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Receptive Field

How big of a region in the input does a neuron on the second conv-layer see?

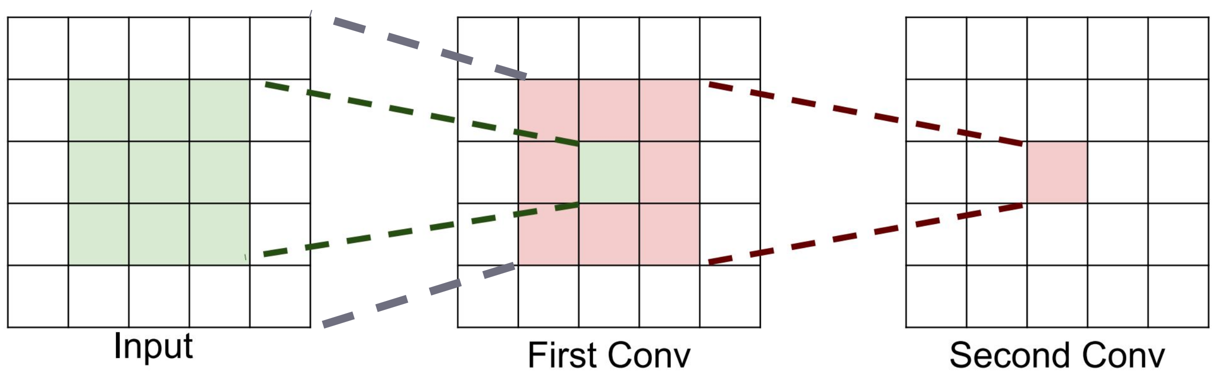


units in the deeper layers can be indirectly
connected to all or most of the input image.

[Goodfellow et al. 2016]

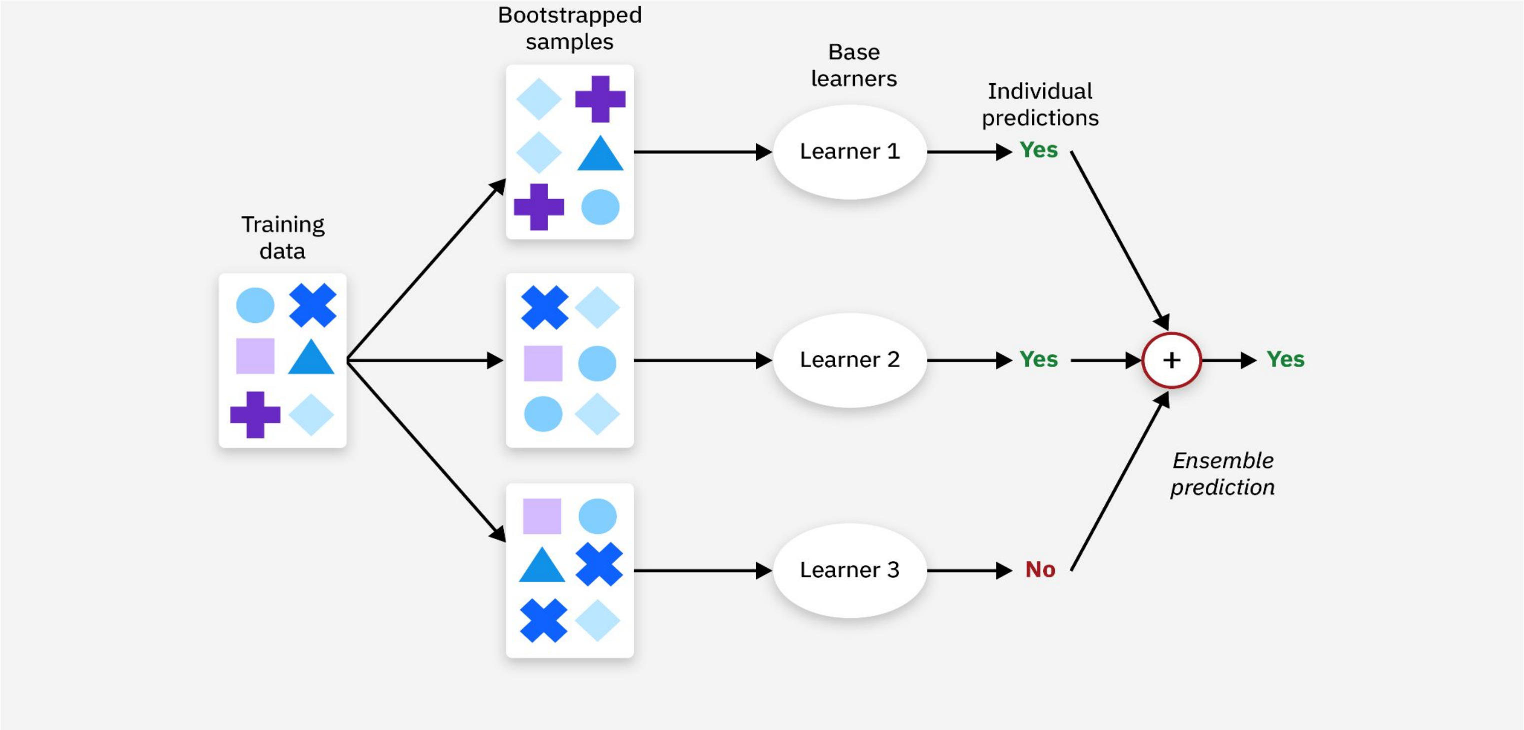
Receptive Field

How big of a region in the input does a neuron on the second conv-layer see?



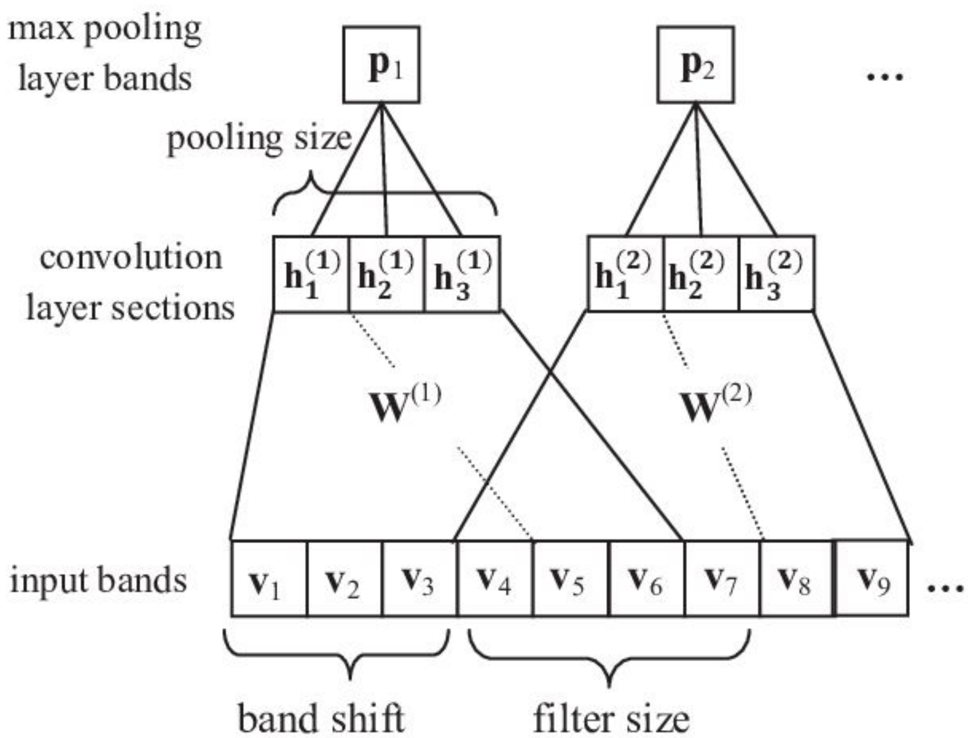
Model ensemble

Ensemble models are a machine learning approach that combine multiple individual models (known as base estimators) in the prediction process. Ensemble models offer a solution to overcome the technical challenges of building a single estimator.



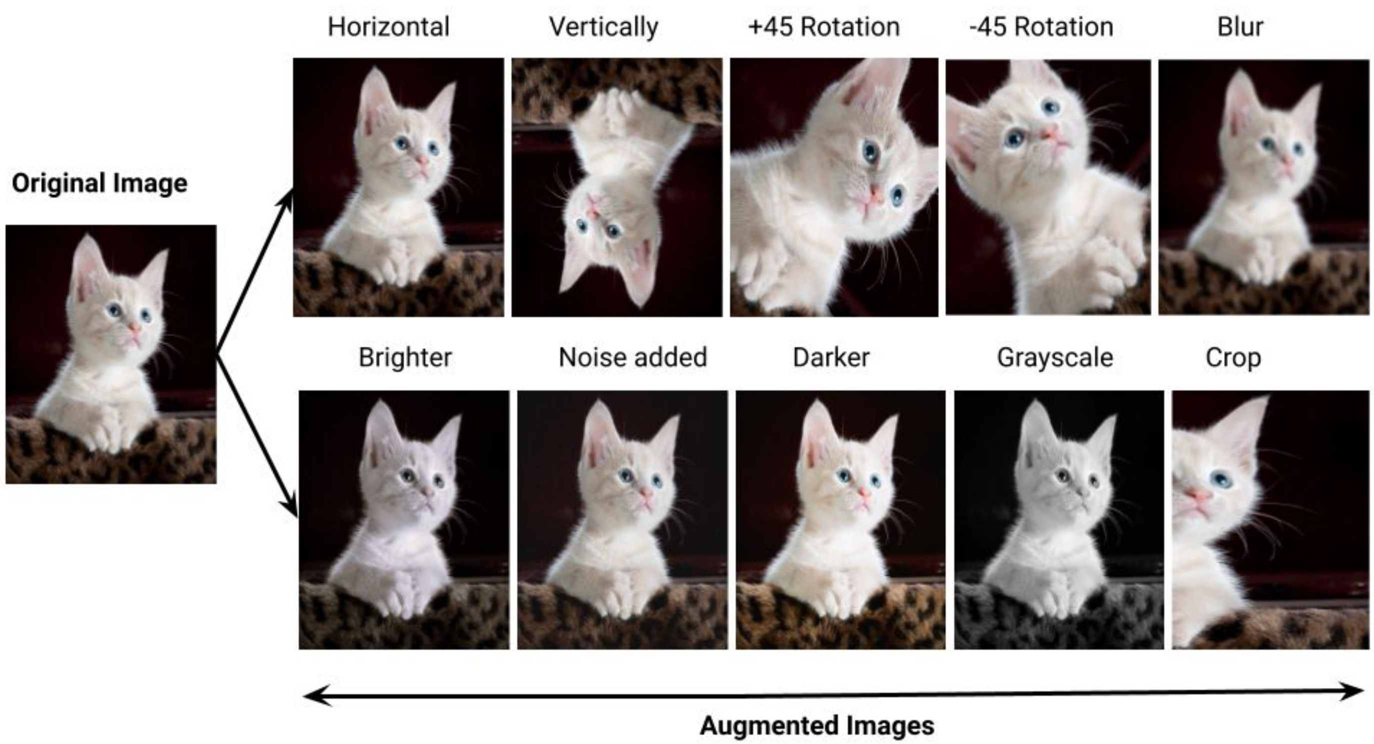
Weight sharing

Weight sharing is an old-school technique for reducing the number of weights in a network that must be trained; it was leveraged by LeCunn-Net circa 1998. It is exactly what it sounds like: the reuse of weights on nodes that are close to one another in some way.



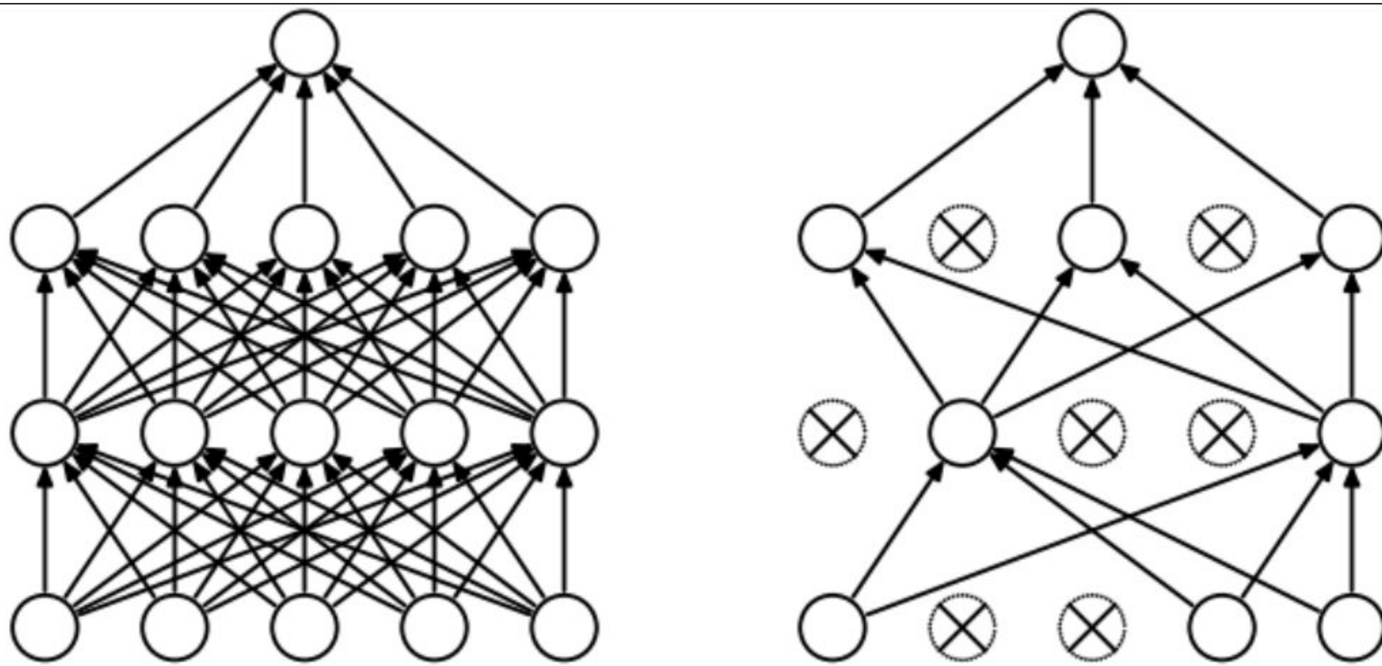
Data augmentation

Data augmentation is the process of artificially generating new data from existing data, primarily to train new machine learning (ML) models.



Dropout

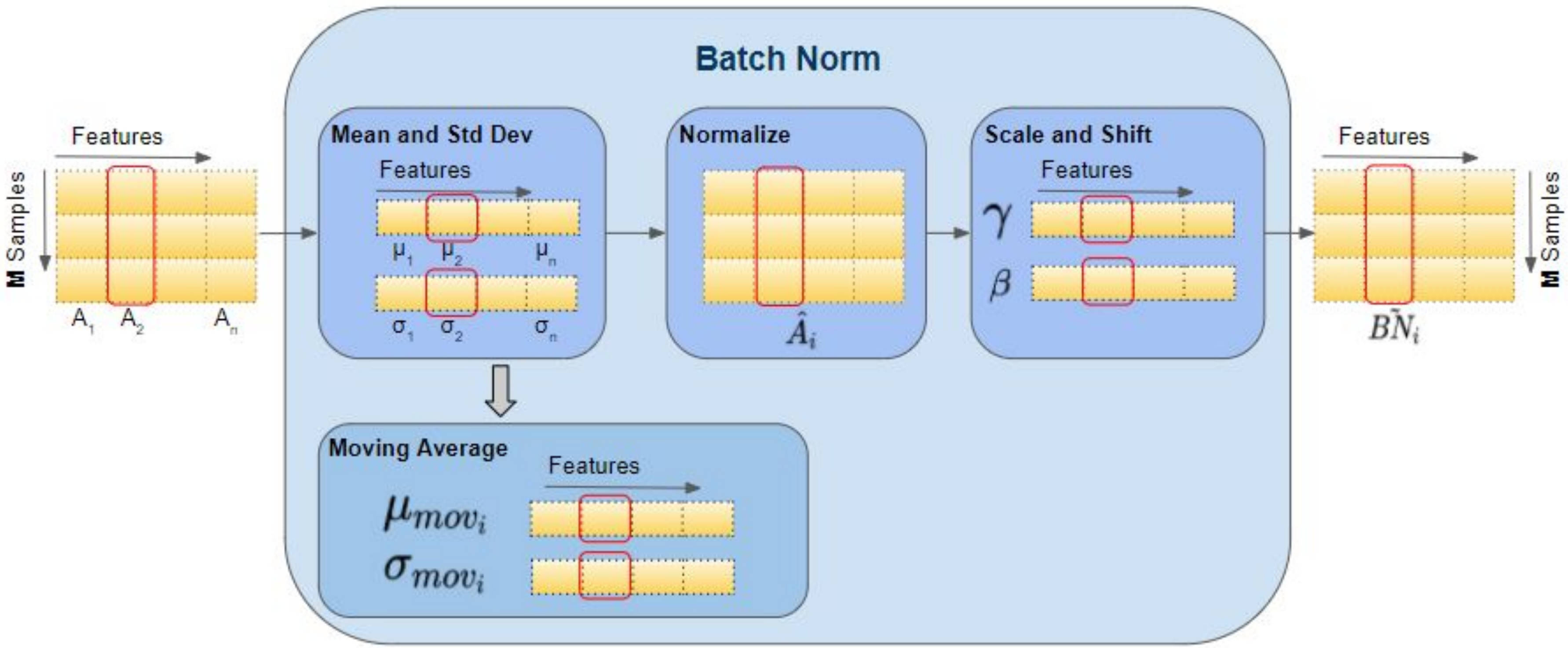
Dropout and dilution are regularization techniques for reducing overfitting in artificial neural networks by preventing complex co-adaptations on training data.



(a) Standard Neural Net (b) After applying dropout.

Batch Normalization

Batch Normalization is used to reduce the problem of internal covariate shift in neural networks. It works by normalizing the data within each mini-batch. This means it calculates the mean and variance of data in a batch and then adjusts the values so that they have similar range.



Summary

- Neural nets are universal approximators
- Backpropagation is a training algorithm for neural nets
- Training issues must be considered
  - Optimization and generalization issues
- Convolutional layers as an example of inductive bias that improves generalization are introduced.