Machine Learning
CS342

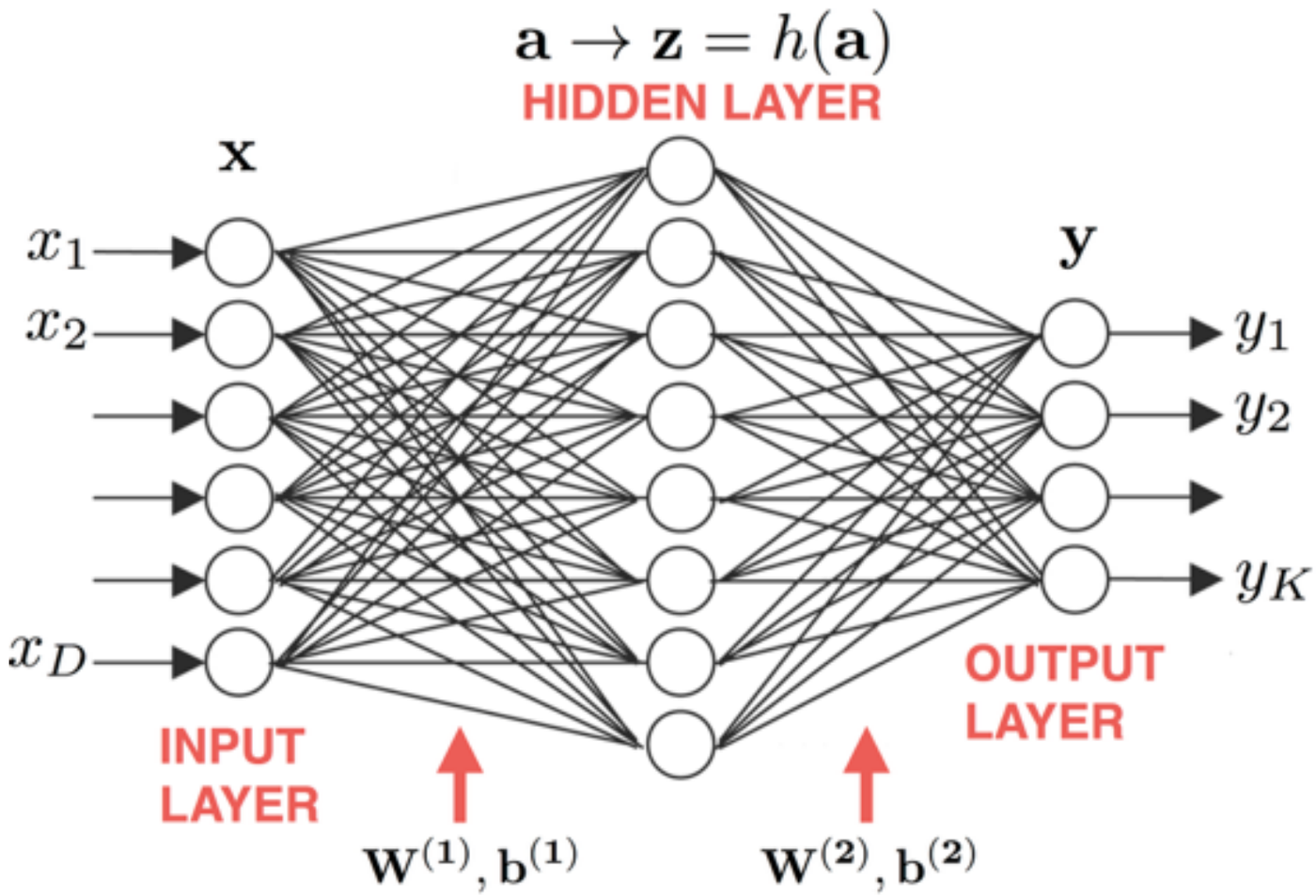
Lecture 15: Artificial Neural Networks (ANNs): CNNs and Deep Learning

Dr. Theo Damoulas

T.Damoulas@warwick.ac.uk

Office hours: Mon & Fri 10-11am @ CS 307

MLPs: Feed-forward ANNs



Fully connected

$W^{(1)}$ DxM matrix of connection weights (parameters) between input-hidden

$W^{(1)}$ MxK matrix of connection weights (parameters) between hidden-output

b “bias” term {x=1} like our intercept in LinReg

Forward propagation of information

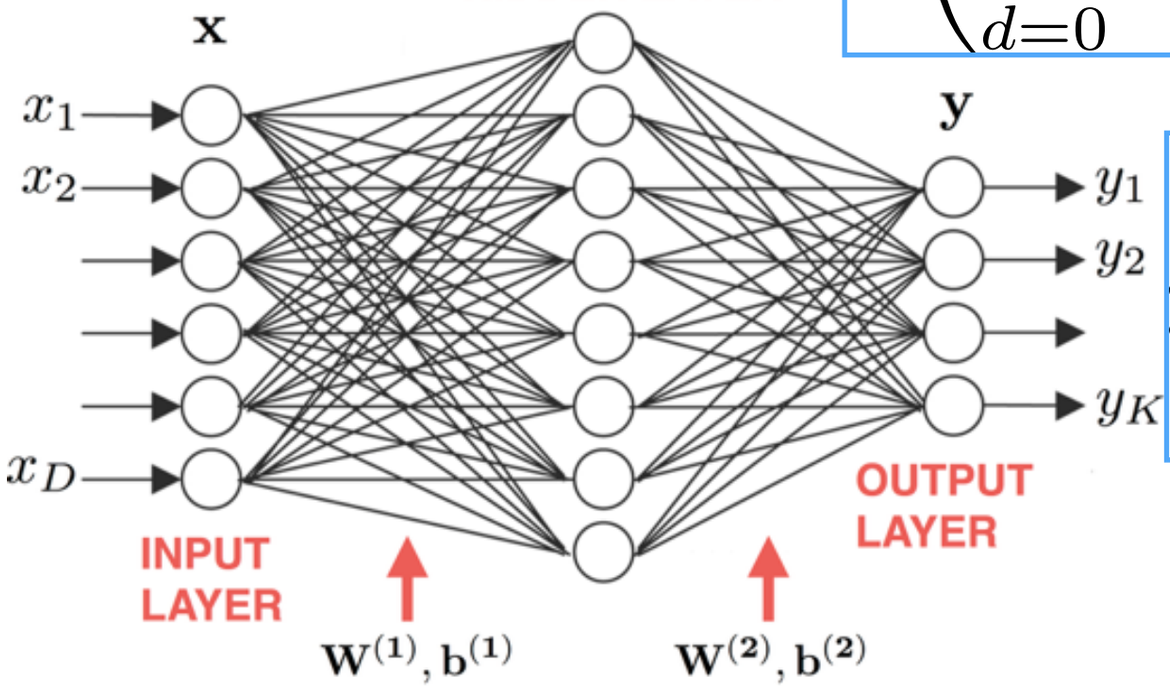
1 training example

$$\mathbf{x} \in \mathbb{R}^D$$

$$h\left(\sum_{d=0}^{D} x_d w_{dj}^{(1)}\right)$$

$$E_n = \frac{1}{2}\sum_k (t_{nk} - y_{nk})^2$$

$$y_k = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{d=0}^{D} x_d w_{dj}^{(1)}\right)\right)$$



Output layer above is linear unit for classification

$$\sum_{d=0}^{D} x_d w_{dj}^{(1)}$$

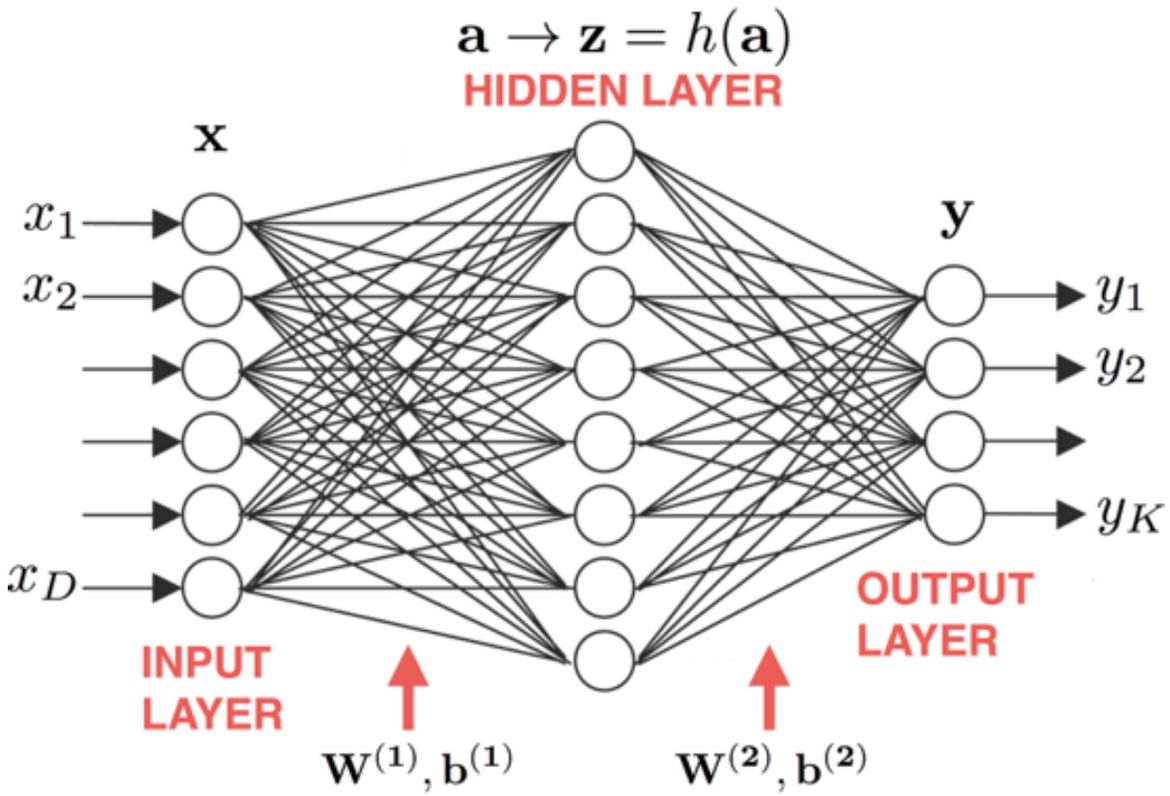
$$\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{d=0}^{D} x_d w_{dj}^{(1)}\right)$$

Forward Propagation of Information

Backward propagation of errors (Backprop)

(Stochastic) Gradient Descent: Find derivatives and update!

Propagate back error based on contributions to error from each w



e.g. Squared Error

$$E_n = \frac{1}{2}\sum_k (t_{nk} - y_{nk})^2$$

Total Error:

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

Derivatives:

$$\frac{\partial E}{\partial \mathbf{W}^{(1)}} \quad \frac{\partial E}{\partial \mathbf{W}^{(2)}}$$

Successive application of Chain Rule

$$\frac{\partial E}{\partial \mathbf{W}^{(1)}} = \frac{\partial E}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial \mathbf{W}^{(1)}}$$

Backward propagation of errors (Backprop)

- A message passing technique that propagates back the error
- Larger activations and weights W more responsible for outcome
- As usual in ANNs we look for error derivatives to perform (S)GD
- SGD will find local minimum solutions - no guarantees
- Empirically shown to work well
- The parameter inference workhorse of ANNs and Deep Learning

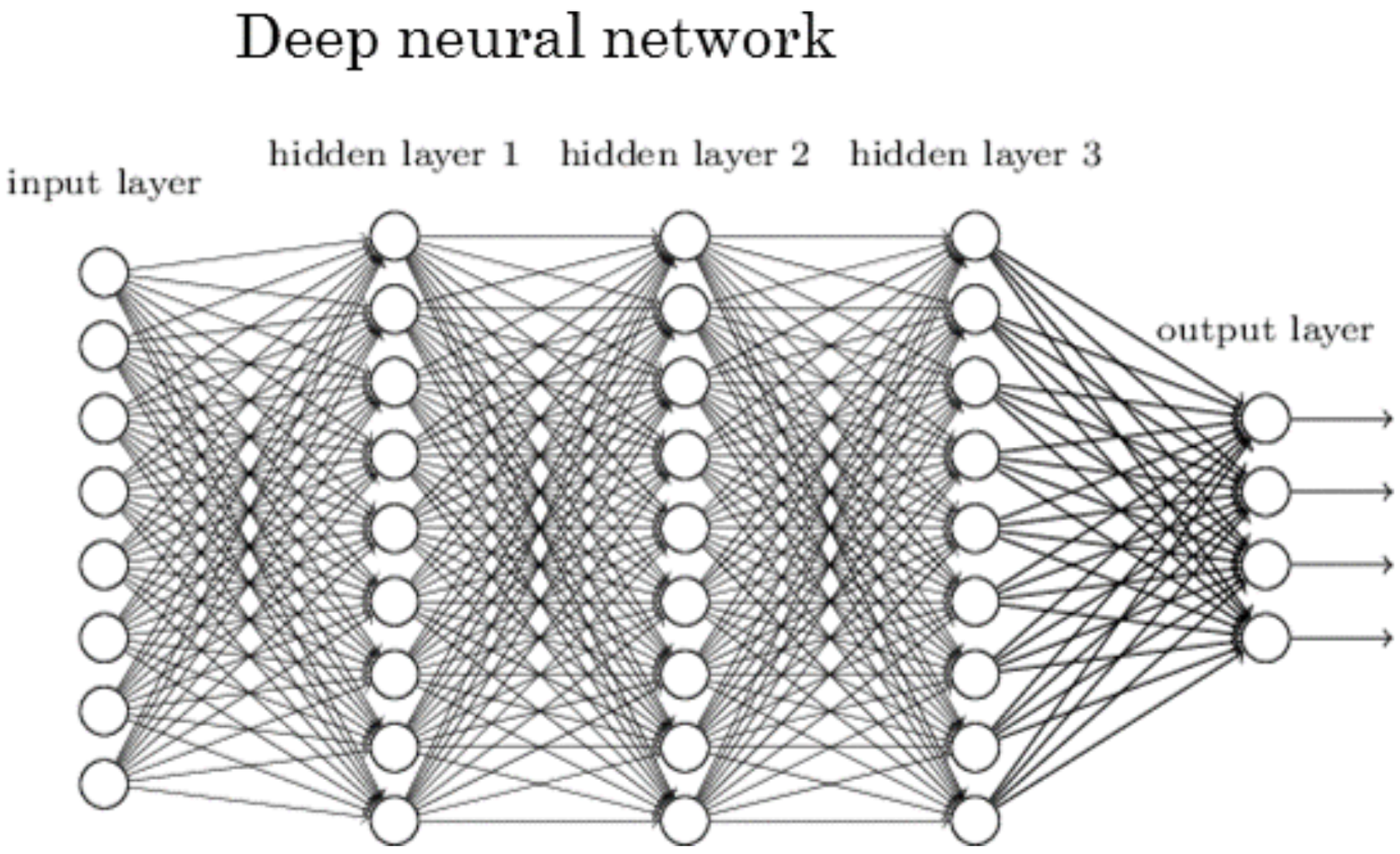
If you go through the derivative derivation you will discover the beauty of backprop and its message (error) passing nature

A nice backdrop explanation video:

https://www.youtube.com/watch?v=GlcnxUlrtek

C.Bishop and T.Mitchell full derivations

Deep Learning



MLP with many hidden layers and tricks for AFs, regularisation, and training

Deep Learning beyond ANNs: “Deep architectures”: Feature Learning

Deep Learning: Feature Learning

Wait… an ANN with 2-layers is a ***Universal Approximator*** right?
So why increase the complexity and number of hidden layers?

More hidden layers = More parameters = Higher model complexity

Some Deep Learning models:

- Deep feed-forward NNs
- Convolutional NNs
- Deep recurrent NNs
- Deep Gaussian Processes
- Deep <insert favourite model>

Feature learning: include feature construction/extraction in the model and infer the "best" one… learn building blocks of complex representations

Deep Learning: Feature Learning

1-hidden layer MLP

$$y_k = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{d=0}^{D} x_d w_{dj}^{(1)}\right)\right)$$

$$y_k = \sigma\left(\mathbf{W}^{(1)} h\left(\mathbf{X}\mathbf{W}^{(2)}\right)\right)$$

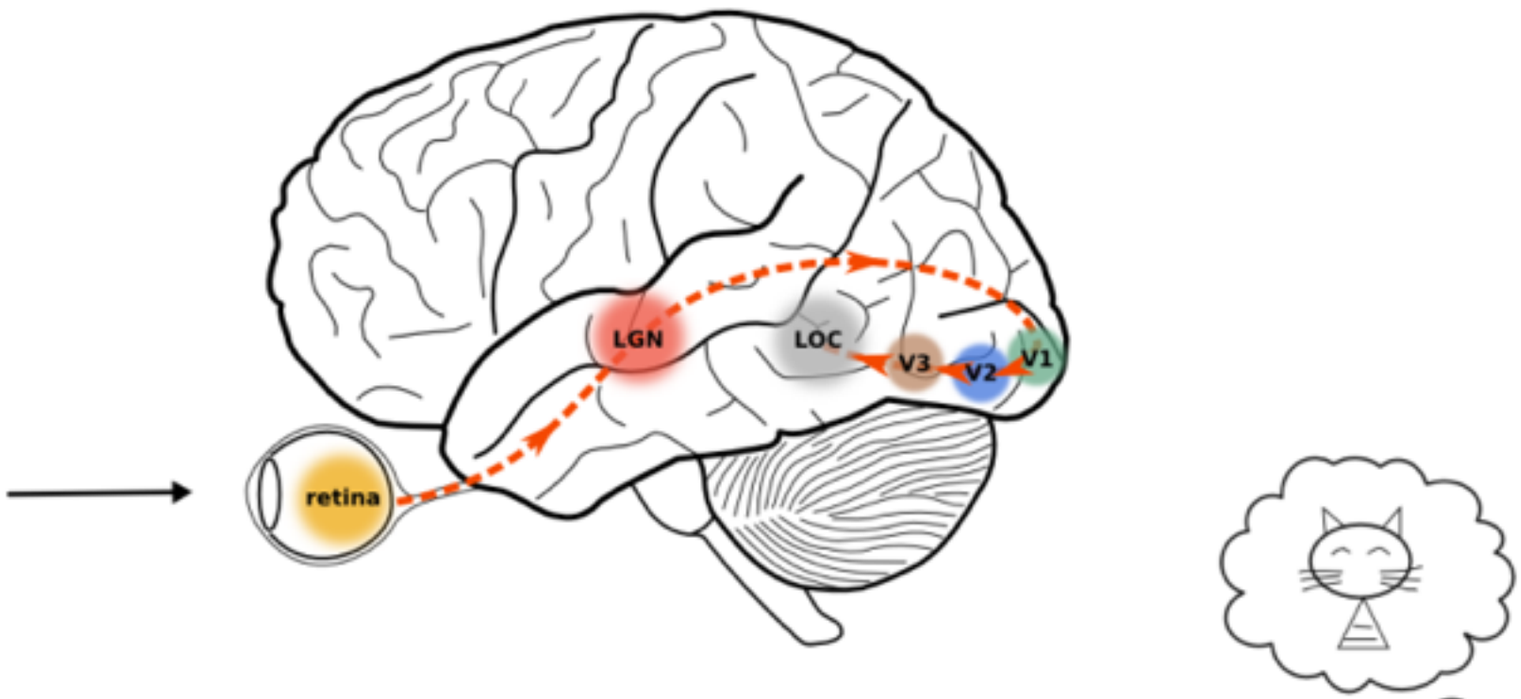
Here is a 4-layer MLP (3 hidden layers) with some hidden-layer AFs: h,f,g

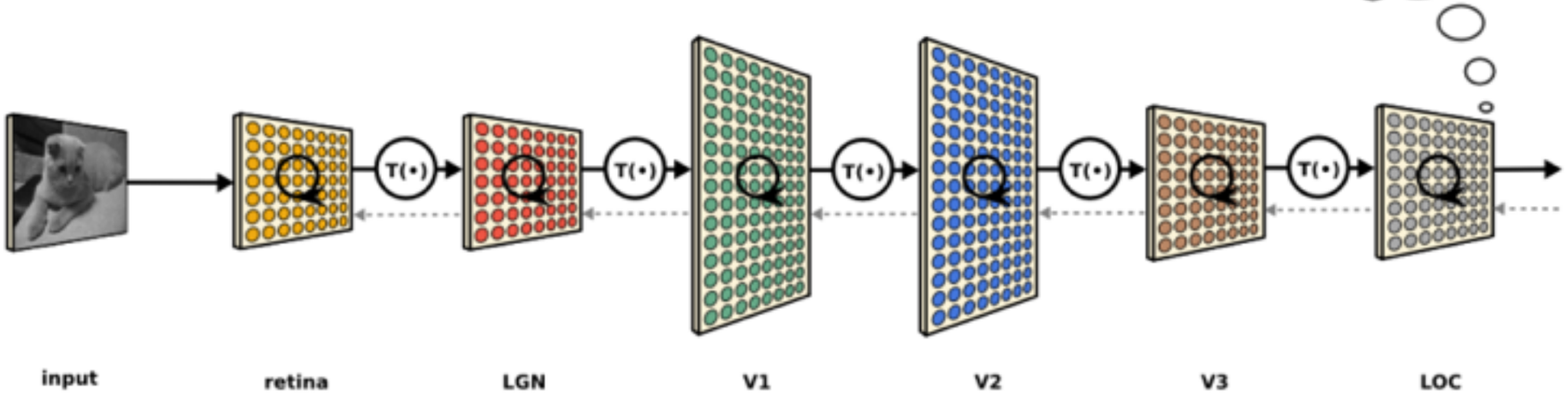
$$y_k = \sigma\left(\mathbf{W}^{(1)} h\left(\mathbf{W}^{(2)} f\left(\mathbf{W}^{(3)} g\left(\mathbf{X}\mathbf{W}^{(4)}\right)\right)\right)\right)$$

Our data (training examples) only inform us of the final function output
The hidden layers perform “feature learning” from a broad class of functions

Injecting prior knowledge into these layers for feature engineering

Deep Learning: Convolutional Neural Networks

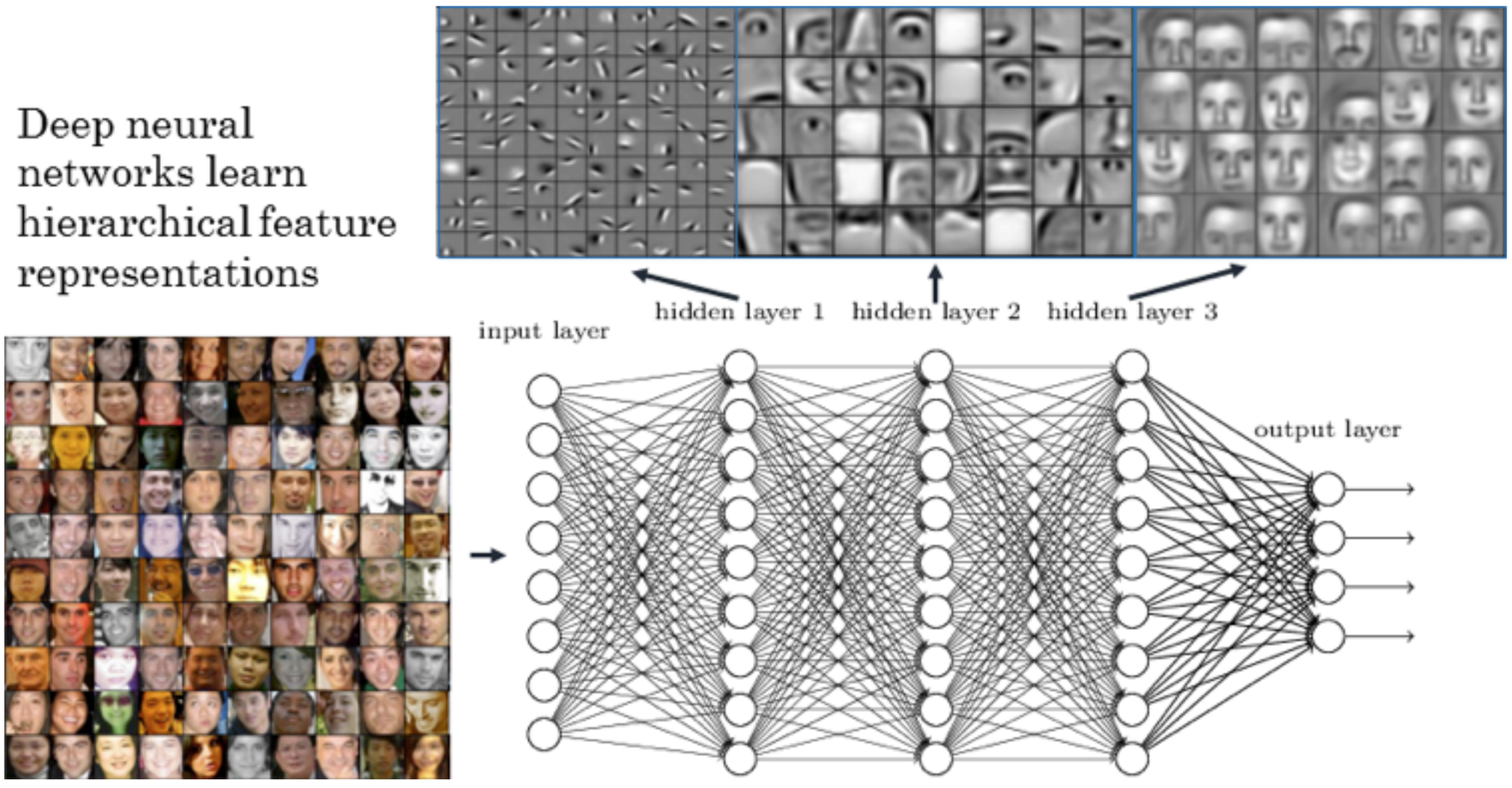




Visual cortex is also hierarchical: progressively computing richer representations/features (from pixel to gradient to edges to shapes etc.)

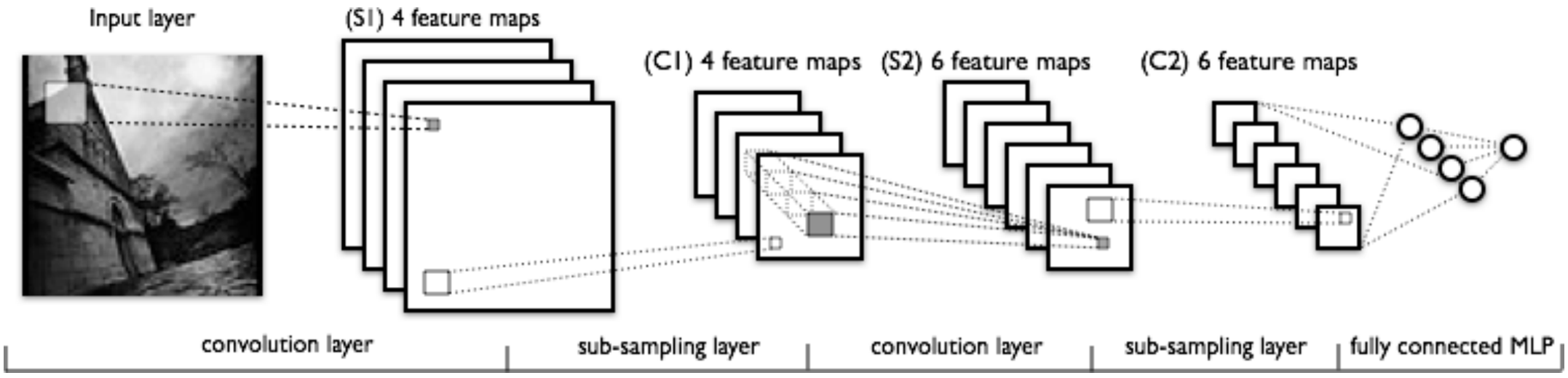
Deep Learning: Human and Computer Vision

Deep learning really started from CV need/drive for Feature Engineering



Exploit prior knowledge (e.g. spatial correlation, human vision, visual cortex) into model structure (layers, operations, connectivity)

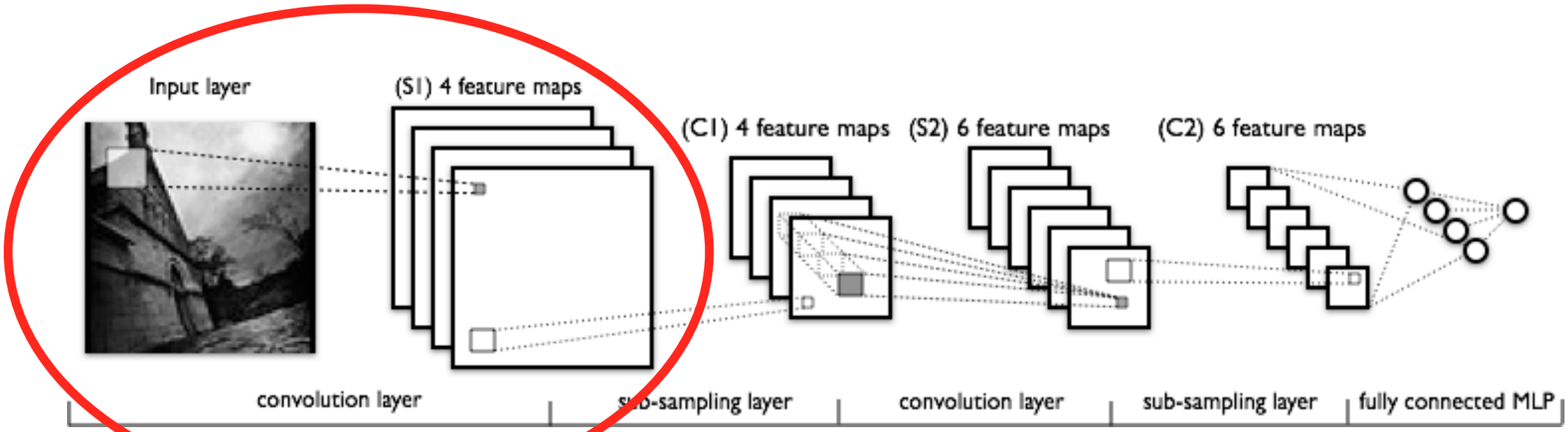
Deep Learning: Convolutional Neural Networks



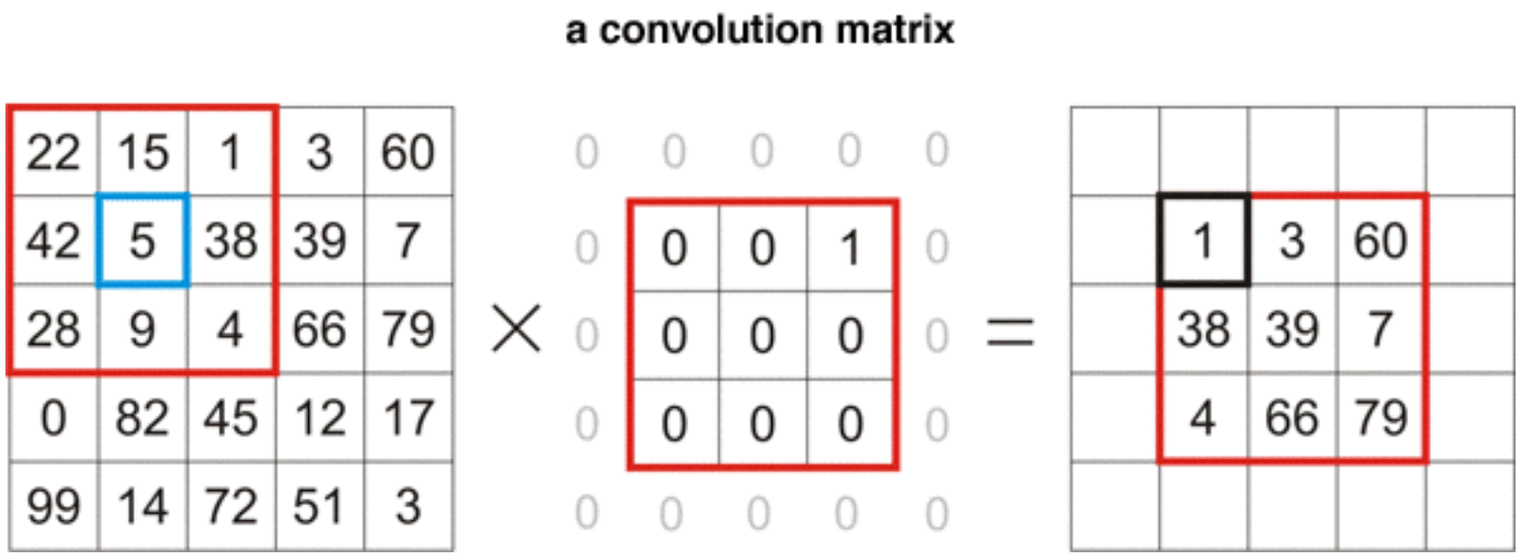
“Ingredients” of CNNs

- Convolutional layers
- Not fully connected network
- Subsampling layers - Pooling (max-pooling, mean-pooling)
- Rectified linear unit activation function
- Backprop

Deep Learning: Convolutional Neural Networks



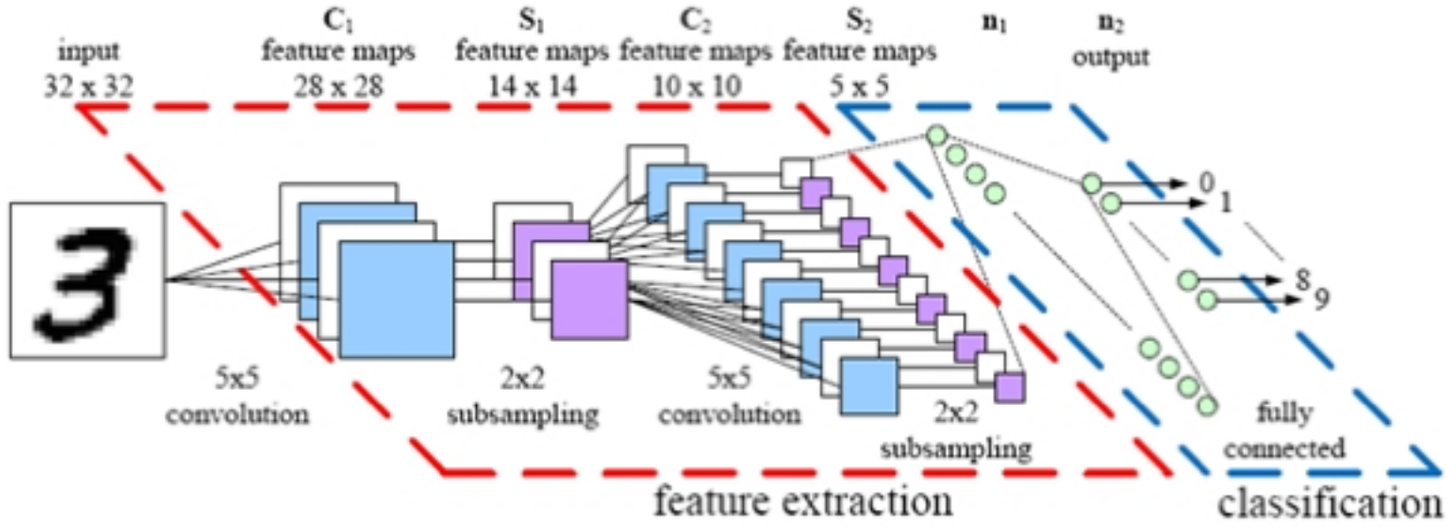
Extracting features from images via **convolution** and subsampling



Sparse interactions
Parameter sharing
Equivariant representations

Deep Learning: Convolutional Neural Networks

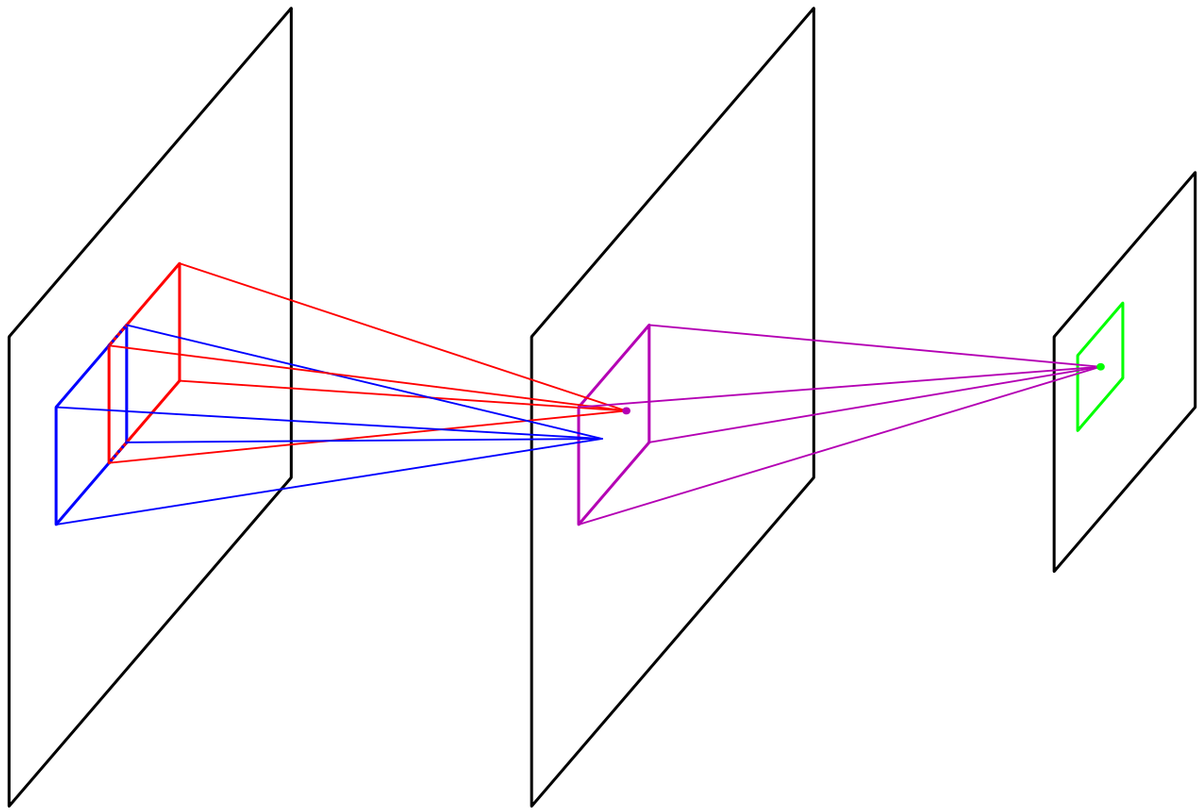
Digit recognition



Identity of digit is invariant under translation, rotation and scaling

- Could train a standard (fully connected) MLP on the pixel values, feed a lot of examples and fingers crossed…
- This ignores **spatial correlation** of pixel values and requires digit in many possible variations
- Why not exploit this spatial correlation and extract **local features**?
- That local information can later (deeper) be merged to construct higher-order features

Deep Learning: Convolutional Neural Networks



3 mechanisms for exploiting that knowledge:

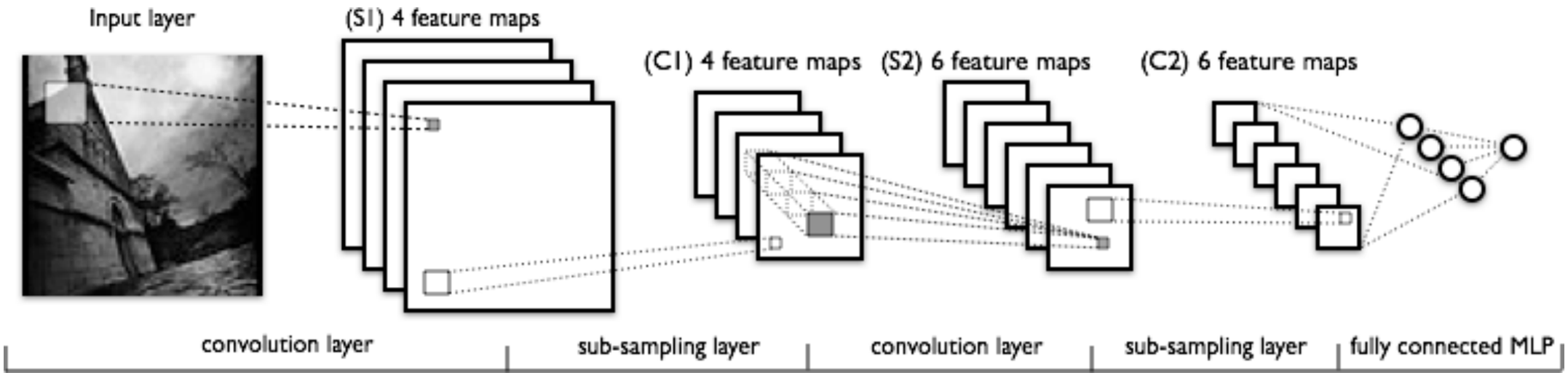
1) Local receptive fields (every unit in CL only sees a patch)
2) Weight sharing (all units in CL share the same weights)

Will detect same pattern at different location

Leading to a convolution operation

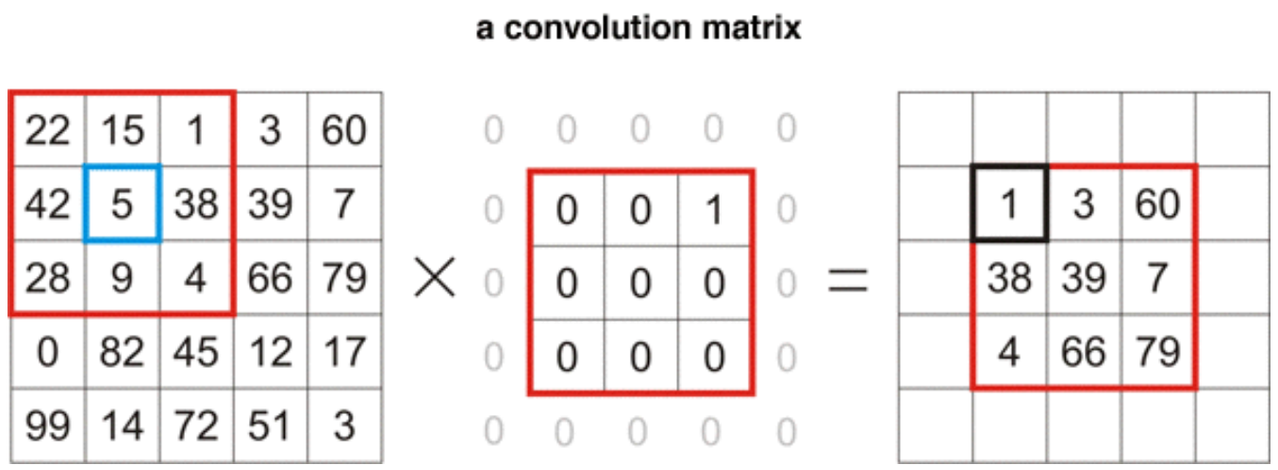
3) Subsampling

Deep Learning: Convolutional Neural Networks



Every “feature map” is the application of a different convolution operator

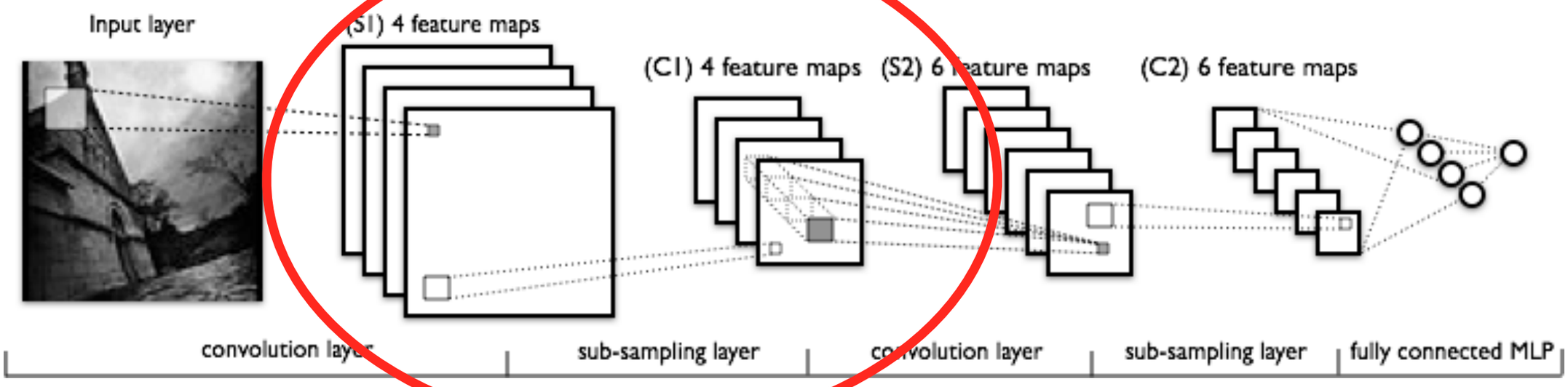
- Convolution achieved via weight sharing within units of a feature map
- Detect multiple features via different weights = convolution kernel



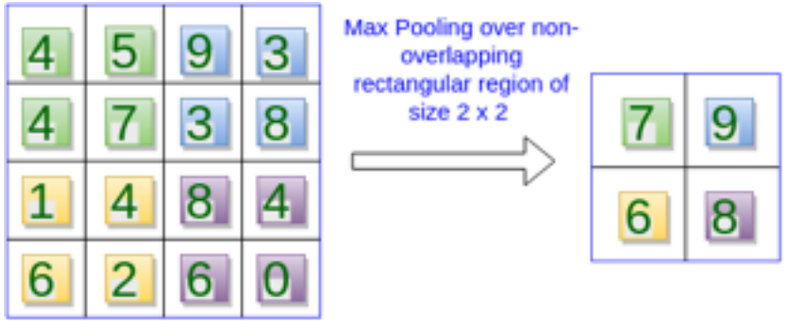
Department of COMPUTER SCIENCE

Deep Learning: Convolutional Neural Networks

Output of Convolutional layer becomes input to Subsampling layer

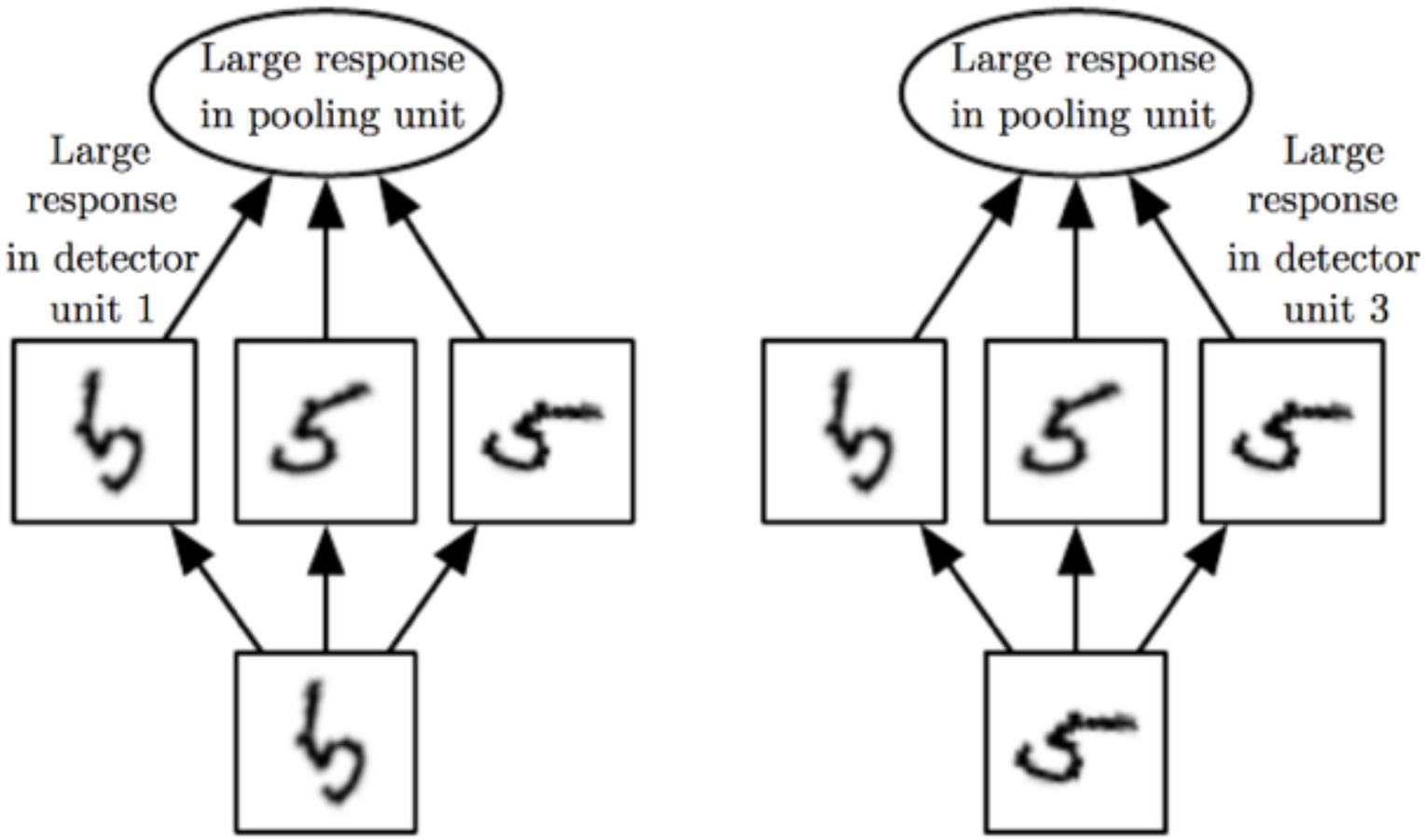


- One subsampling feature map per convolutional feature map
- Pooling layer e.g. average of 2x2 window values or max (max-pooling)
- Receptive field is non-overlapping (contrary to convolutional layers)
- Typically half the resolution of CL
- Progressively we pick up higher-order features



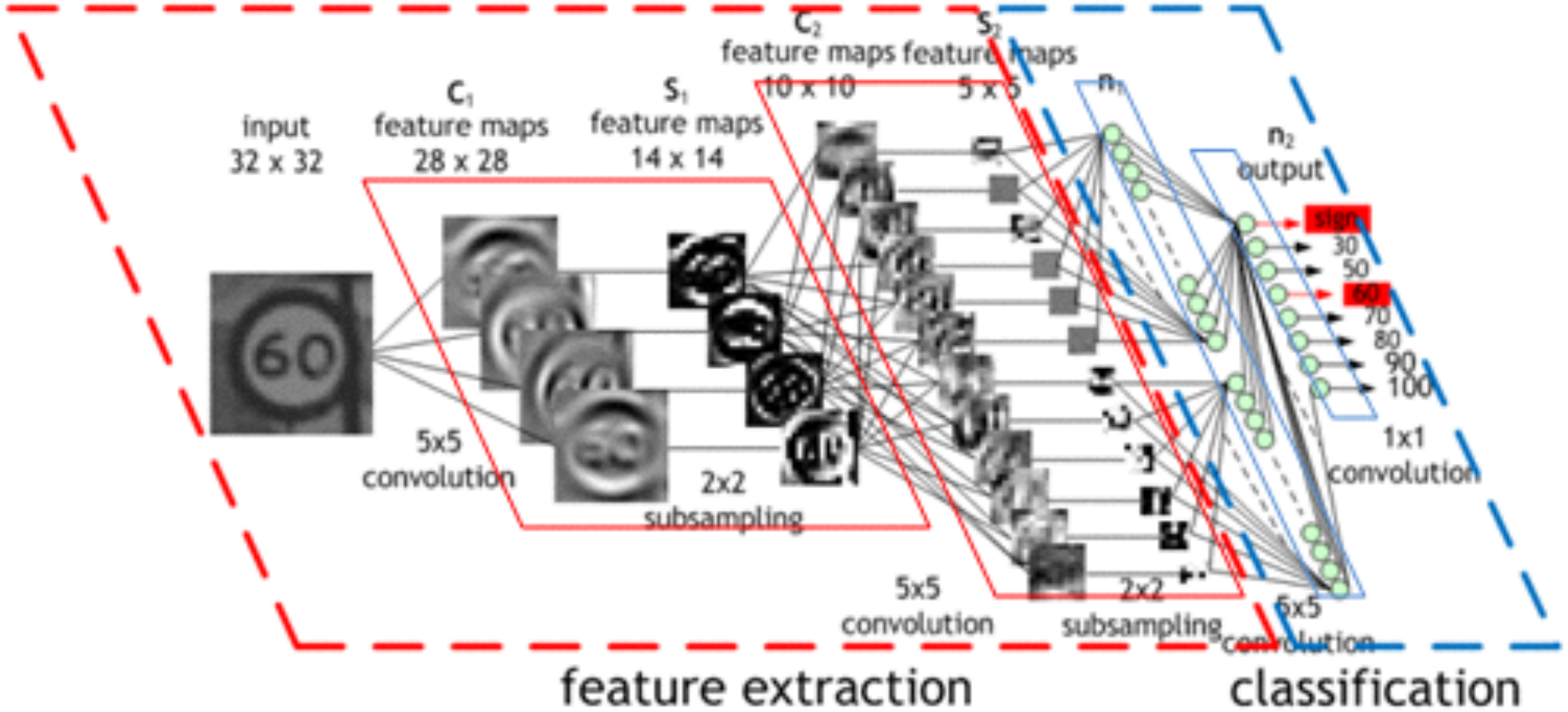
Deep Learning: Convolutional Neural Networks

When subsampling the pooling operator is applied to the output of the previous CL



So pooling gives us some further invariance

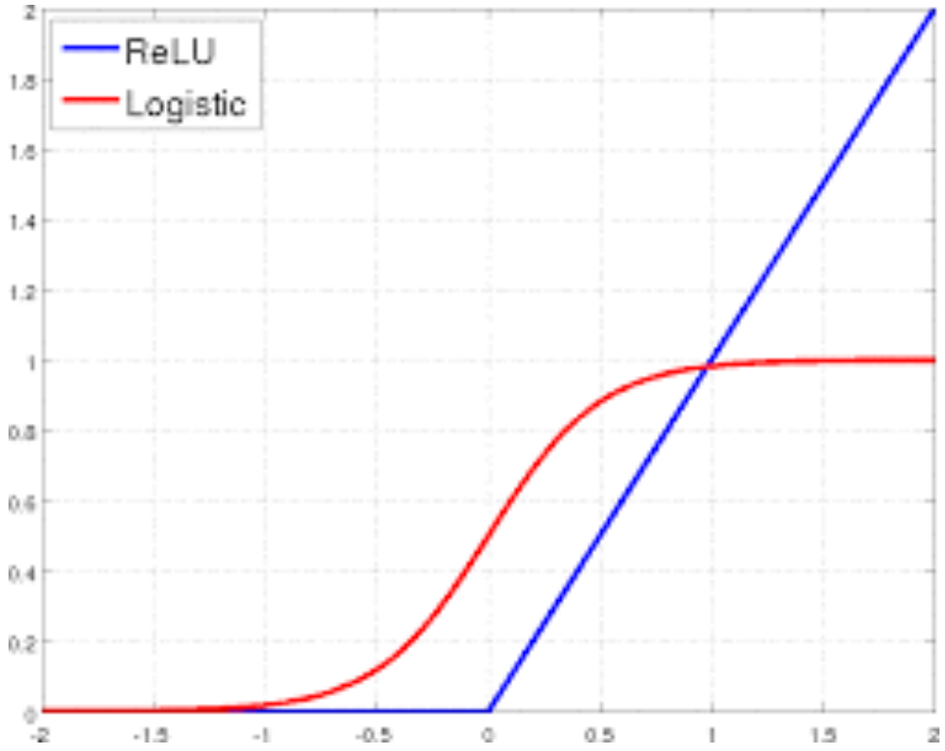
Deep Learning: Convolutional Neural Networks



Deep Learning: Convolutional Neural Networks

Rectified Linear Unit (ReLU) activation function

AFs so far:
Step function
Logistic
Softmax
ReLU
None (linear unit)



$$f(y) = \max\{0, y\}$$

In CNNs typically an ReLU function is used for mathematical ease
In terms of parameter inference again use Back-propagation
Due to local receptive fields and weight sharing = less parameters
Regularisation to prevent overfitting: dropout (randomly “drop out” neurons during the learning process)

Deep Learning

Ok so what are the fundamental characteristics of deep learning?

- Hierarchical feature representations of inputs
- **Learn features** by having an embedded feature extraction component
- Thousands to Millions of parameters:
  - Danger of overfitting
  - Big Data enables fitting of these complex models
  - Try to control overfitting by “coupling parameters” - less DoF
  - Try to control overfitting by weight sharing and subsampling
  - Try to control overfitting by not fully-connected layers
  - Try to control overfitting by dropout (its an ensemble of models)
- Inference? **Back propagation**
- ReLU is typical but also logistic activation functions used
- Convolutional layer for images and spatial correlation
- Subsampling to pool information and “blur” for higher-level info
- Pooling also can handle invariance

Deep Learning

A massive ensemble model of interconnected computational units
Natural for GPU computations!

Can be thought of as including some **very strong prior** information

- Shared parameters
- Specific convolutions (weights) that look for e.g. edges etc
- Strong prior over connectivity (synapses)
- **Priors over parameters and network structure**

Very hot area of research
A significant engineering effort in practice (convolutional filters, num of layers, num of units per layer, type of pooling, type of AF)
Little theoretical insight or guarantees
Amazing empirical performance on images and other domains soon