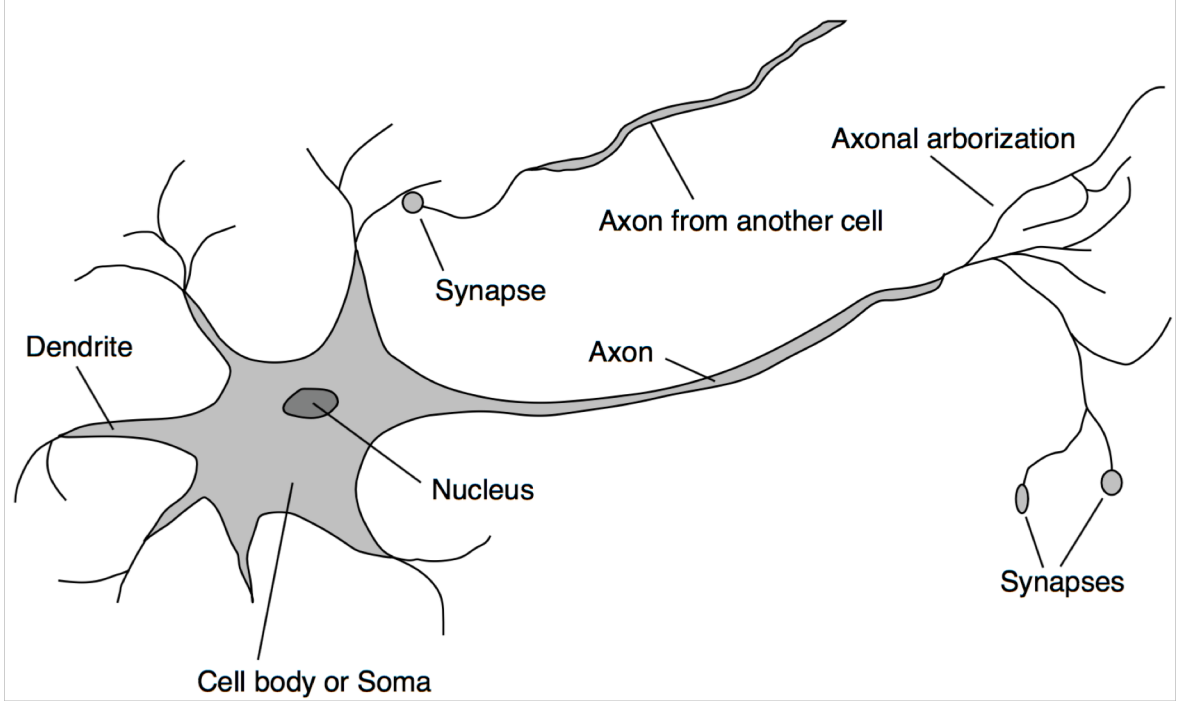
CSCM77
From NN to CNN (an overview)

Prof. Xianghua Xie
x.xie@swansea.ac.uk
http://csvision.swan.ac.uk

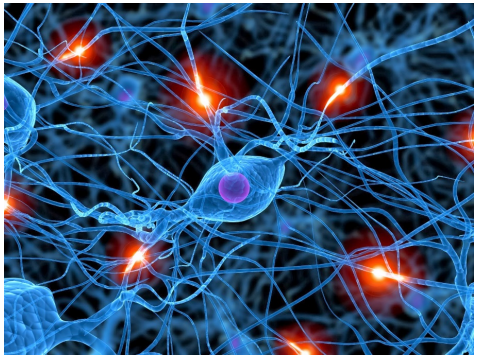
Human Brain

- Number of neurons: 10^{11}; >20 types; 10^{14} synapses
- Neuron switching time: 1~10ms cycle time
- Signals are noisy “spike trains” of electrical potential
- Scene recognition speed: ~0.1s



Human Brain

- Each neuron receives input from ~1,000 others
- Impulses arrive simultaneously
  - added together
  - an impulse can either increase or decrease the possibility of nerve pulse firing
- If sufficiently strong, a nerve pulse is generated
- The pulse forms the input to other neurons.



Artificial Neural Network

- ANN: a bottom-up attempt to model the functionality of brain

- ANNs are biological inspired, but not necessarily biologically plausible

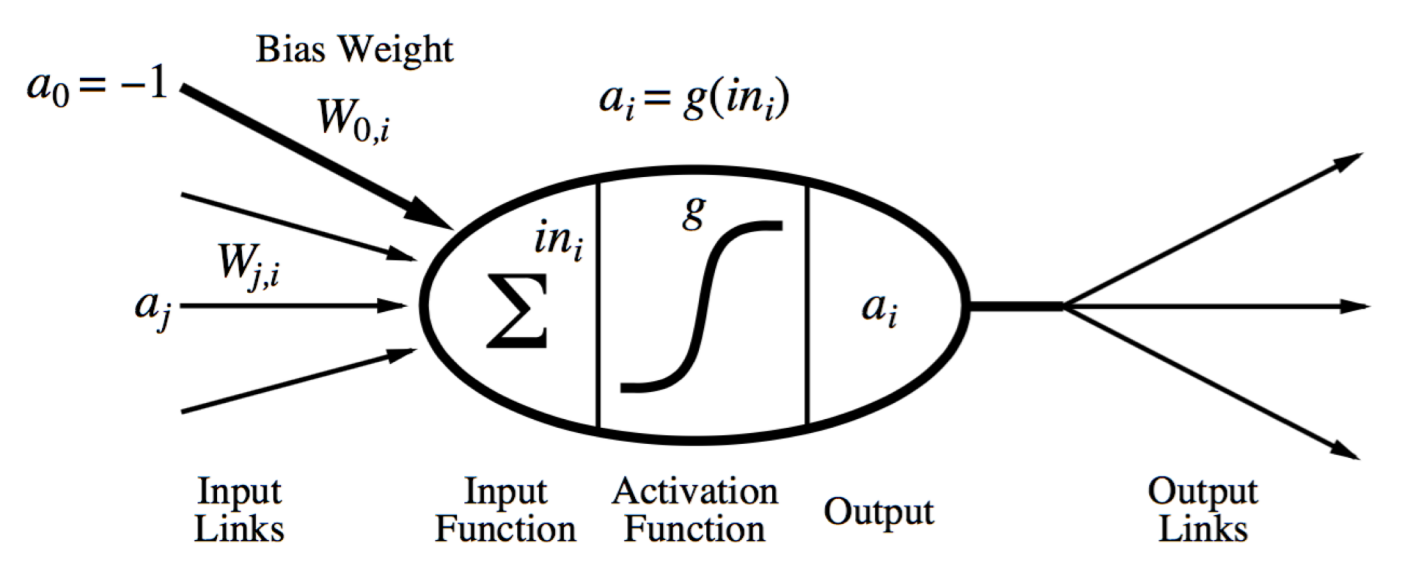
- ANN properties
  - Many neuron-like switching units
  - Many weighted interconnections among units
  - Highly parallel, distributed process (GPU)
  - Emphasis on tunning weights automatically

Extra Session on Coursework

- 2-3PM Tuesday

Artificial Neural Network

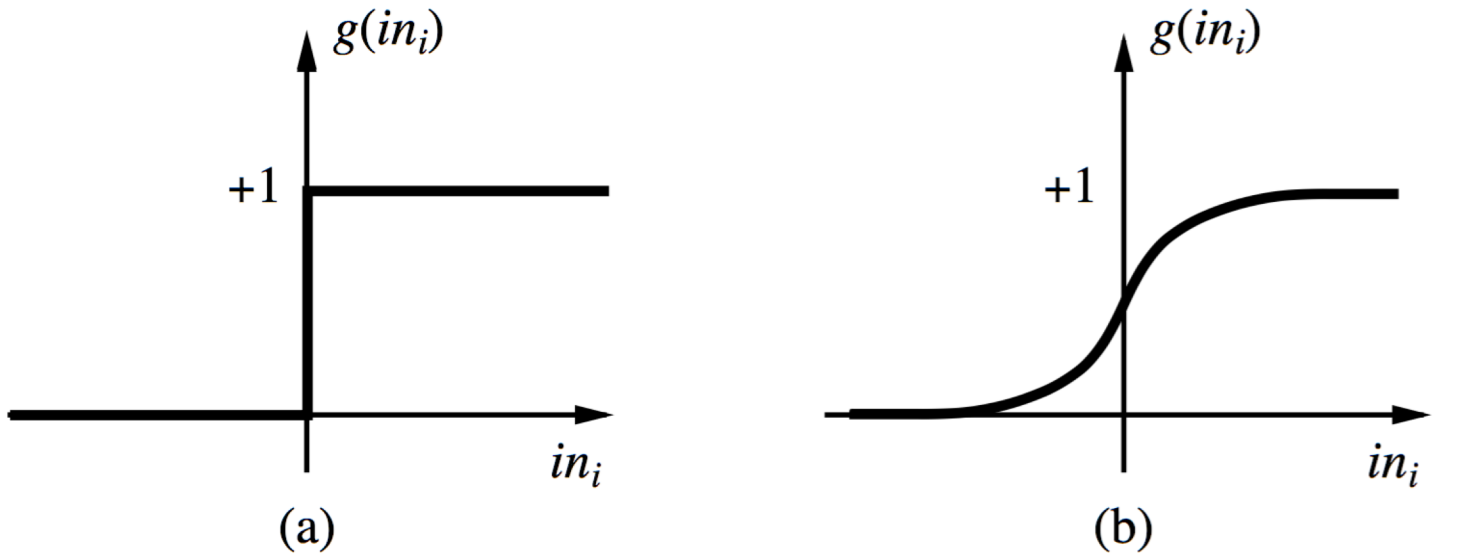
- Perceptron
  - A gloss simplification of real neurons; synthetic interpretation of biological neurons
  - Its purpose is to develop understanding of the capabilities of various networks of such simple units



$$a_i \leftarrow g(in_i) = g\left(\Sigma_j W_{j,i} a_j\right)$$

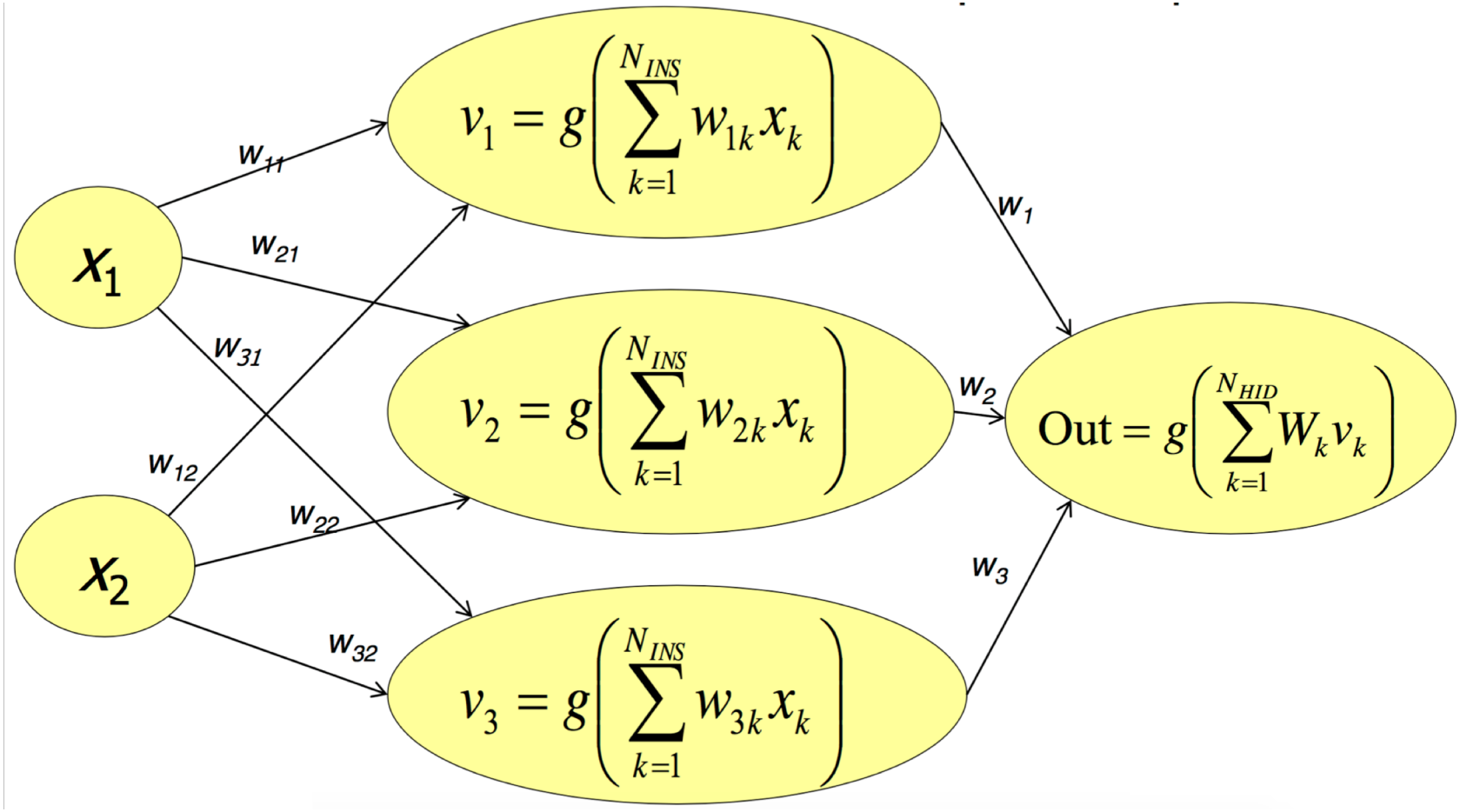
Artificial Neural Network

- Activation function (firing of the neurons)
  - (a): a step function or threshold function
  - (b): a sigmoid function $1/(1+e^{-x})$
  - The bias weight moves the location of the function



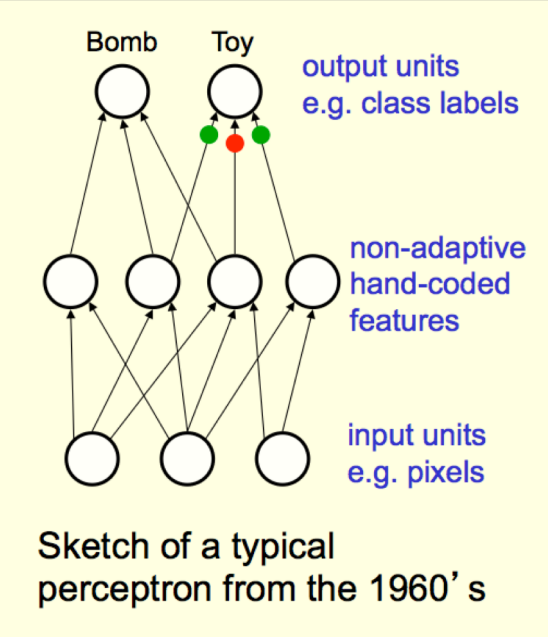
Multilayer Neural Network

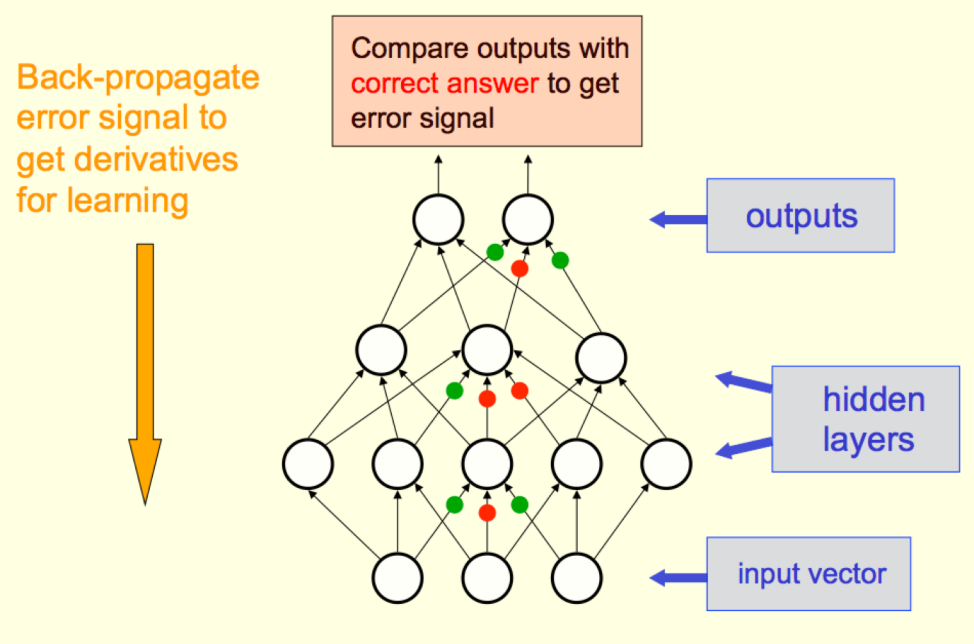
- Many different ways to connect the perceptrons in a network
- An example multilayer neuron network with one hidden layer



Neural Networks Learning (1960s – 1990s)

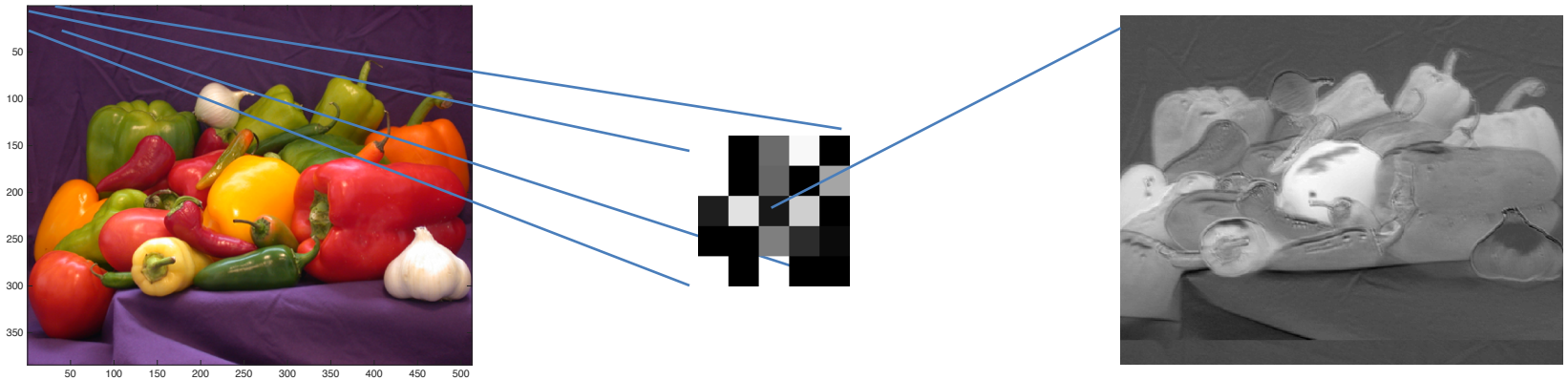
- 1st generation: hand crafted (hard coded) feature
- 2nd generation: back-propagation based learning
  - Back propagate the training error from the output
  - Requires large amount of labelled training data
  - Computationally very expensive
  - Initialisation of the weights is important





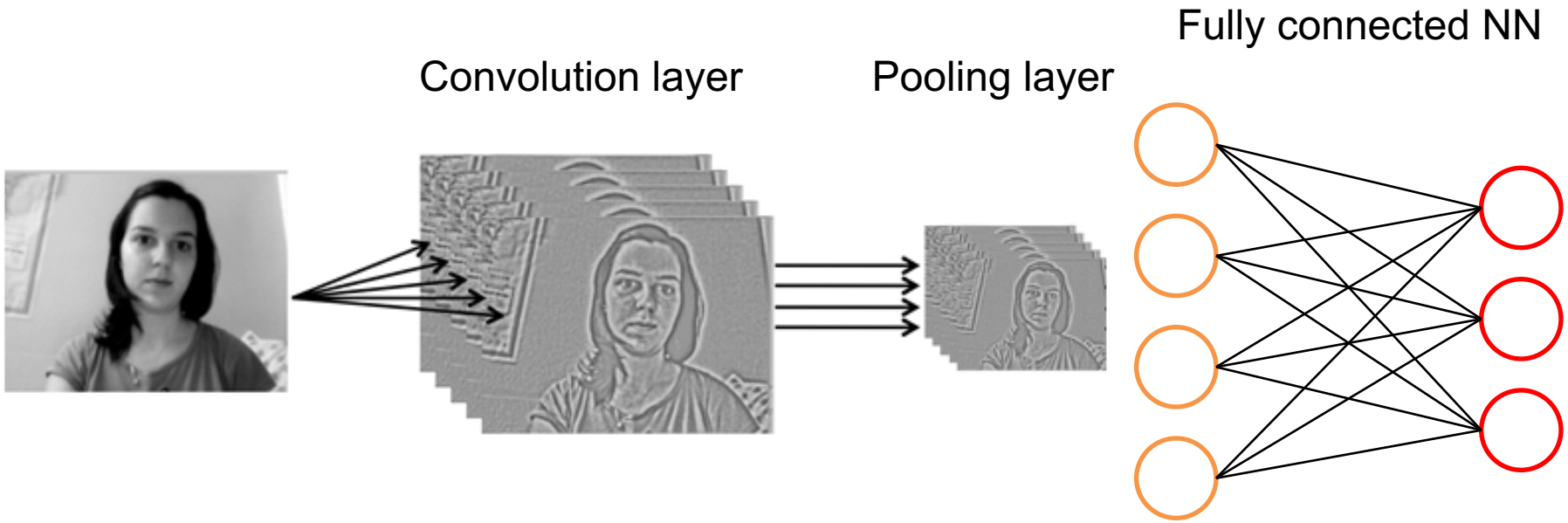
CNN: Convolutional Neural Network

- NNs disadvantages:
  - non-localised features,
  - many parameters to learn,
  - doesn't scale well (difficult to train with several hidden layers)
- Image domain features are often locally aggregated



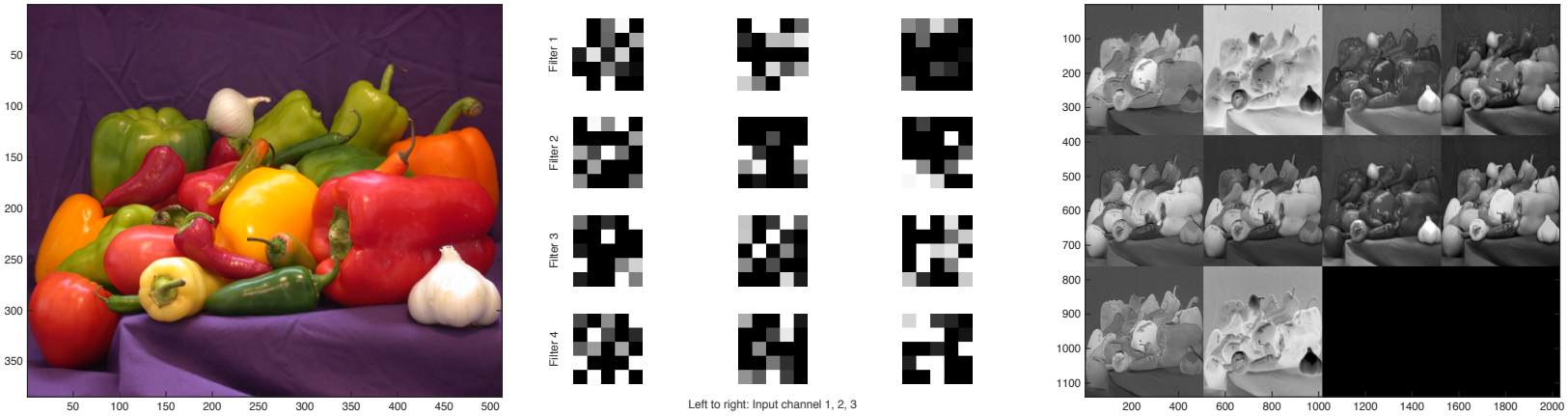
CNN: Convolutional Neural Network

- CNN
  - Filter kernel identifies features within a receptive region of input
  - Convolution layer generates feature map of input layer
  - Pooling generalises feature maps



Convolution Layer

- Convolve bank of locally receptive filters across image spatial domain
- Random filters work well – Optimise mixture by using the “back propagation” algorithm
- Greatly reduce parameters by sharing weights for a given filter
  - Single weight per filter, replaces weight for each input-node connection

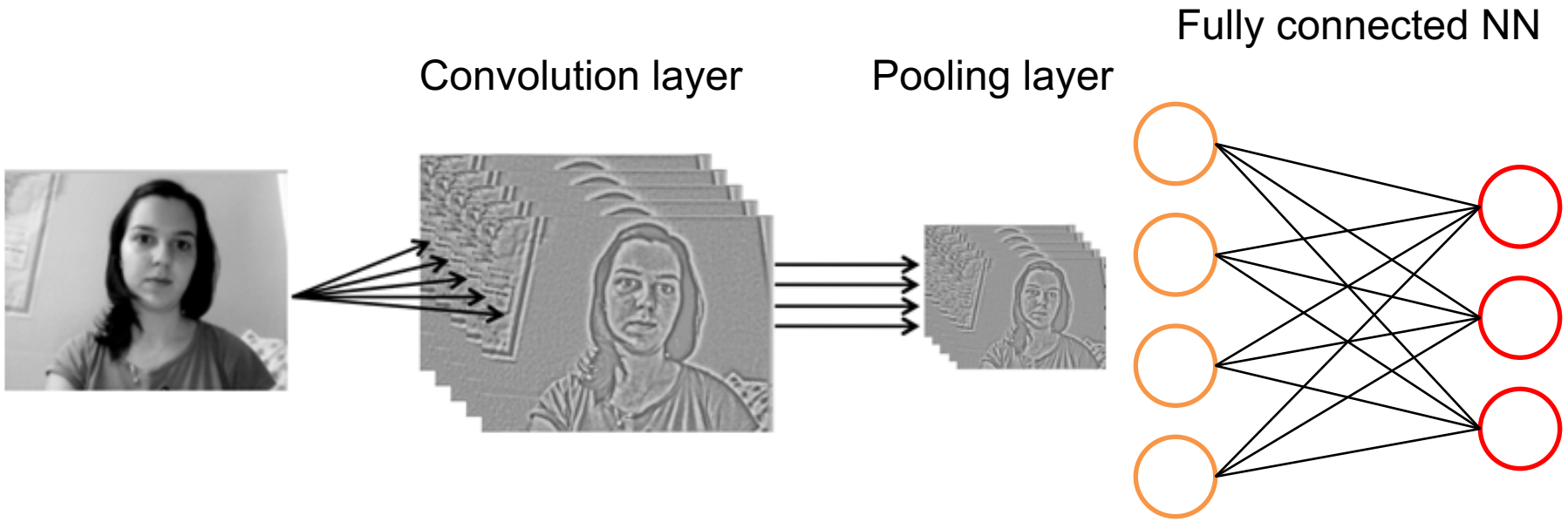


Pooling Layer

- Feature maps from convolution contain high amount of spatial information
- Pooling reduces resolution and generalises feature maps
- Locally receptive filter passed over each feature map
- Common pooling techniques
  - Max-pooling
  - Min-pooling
  - Mean-pooling

Learning

- Same feedforward and back propagation gradient descent as NNs
- Reduction of weight parameters saves computation
- Allows deeper networks (multiple convolutions and pooling) to learn higher level descriptors



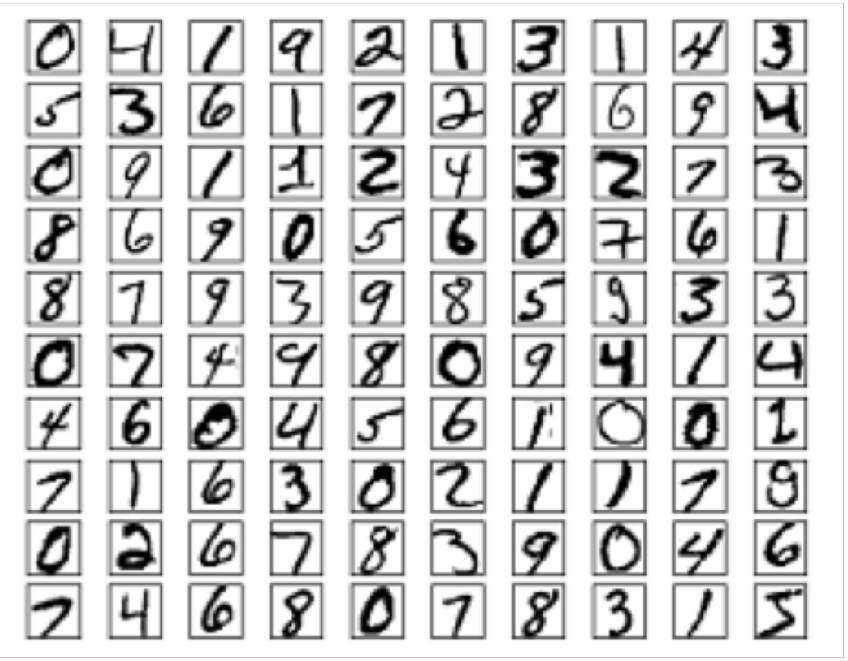
Example of CNN on Digit Recognition

- MNIST hand-written numerical digit dataset
- 50,000 training samples and 10,000 testing samples
- 28x28 pixel grayscale images of digits 0-9
- Benchmark for classifier performance – dominated by CNNs

http://yann.lecun.com/exdb/mnist/

Example of CNN on Digit Recognition

- Input layer connected to **n** = 32*32 = 1024 pixels
- Output layer connected to **c** = 10 digit labels
- Use pre-trained CNN “LeNet-5”
- LeNet-5 Network structure:
  - Input – Conv – Pool – Conv – Pool – Conv – Output

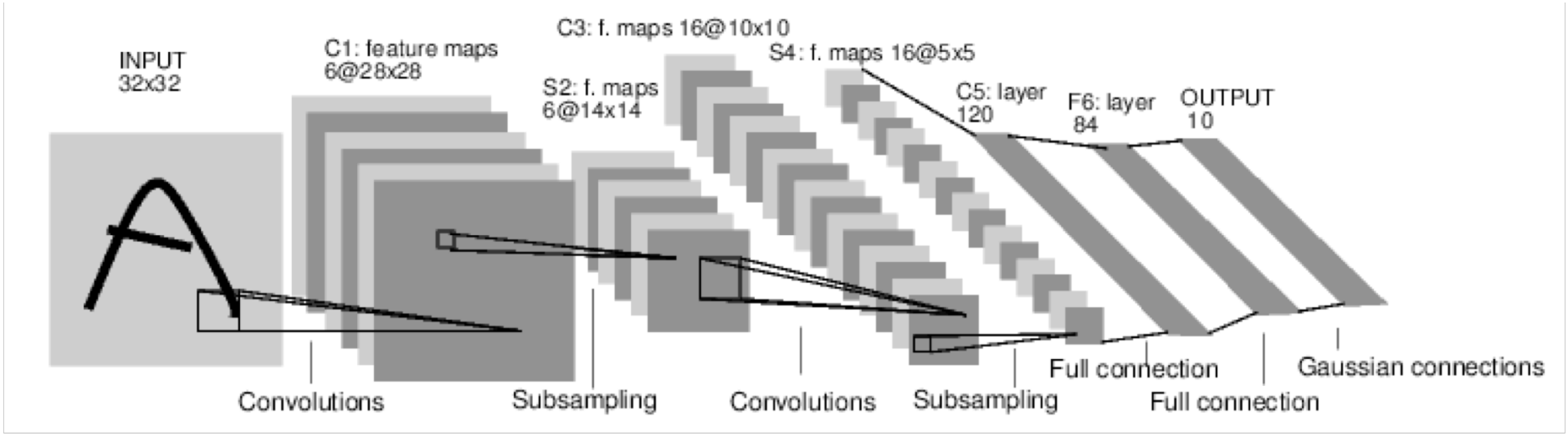


Lab CNNs for MNIST - myCNN Demo

- myCNN (stands for "my Convolutional Neural Network") Matlab implementation of convolutional neural network (CNN).
- Digits Recognition Demo
  - @myCNN, implementation of CNN
  - demo_myCNN.m
    - GUI of the digits recognition software
  - Mnist2matlab.m
    - Convert the MINST dataset (within ./MINST/) to matlab format
  - example_myLeNet5_SDLM_training.m
    - Train the CNN model
  - myLeNet5-example.mat
    - Pre-trained CNN model

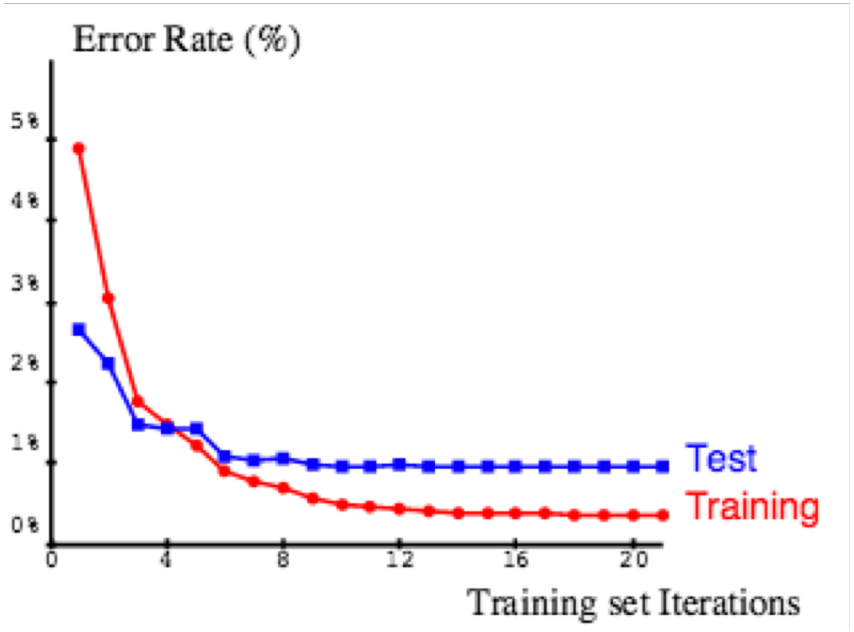
LeNet5 CNN Architecture:

- Input: 32*32*1 pixel intensities
- 1C - **in**: 32*32, **filter**: 5*5*1*6, **stride**: 1, **pad**: 1, **out**: 6@28*28
- 2P - **in**: 28*28**, filter**: 2*2, **stride**: 5, **pad**: 5, **out**: 6@14*14
- 3C - **in**: 14*14, **filter**: 5*5*1*16, **stride**: 1, **pad**: 1, **out**: 16@10*10
- 4P - **in**: 10*10, **filter**: 2*2, **stride**: 5, **pad**: 5, **out**: 16@5*5
- 5C - **in**: 5*5, **filter**: 5*5*120, **stride**: 1, **pad**: 1, **out**: 120@1*1
- 6F - **in**: 1*1, **fully connected NN**, **out**: 10@1*1



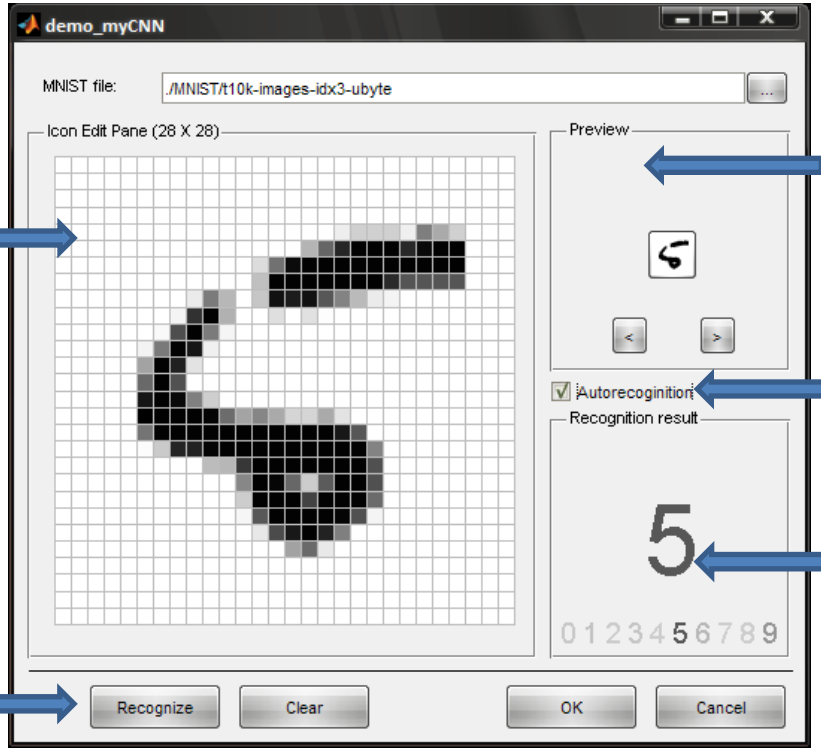
demo_myCNN

- Loads pretrained LeNet-5 architecture
- Provides an interface to test new observations
- Figure shows the falling Mean Square Error as training epochs progress
  - Training error falls with gradient descent based optimisation
  - Testing error falls as training learns best general features for batch learning



demo_myCNN Gui interface

Icon Edit Pane:
Draw new digit observations using the mouse



Preview:
Shows MNIST sample
Cycle using left and right arrows

Autorecognition:
Obtain classification on the fly

Recognition result:
Final output layer.
Overall classification is the large number.
Possible classifications are then ranked below.

Recognize: Perform classification
Clear: Clear the Icon Edit Pane