Machine Learning saves Computer Vision

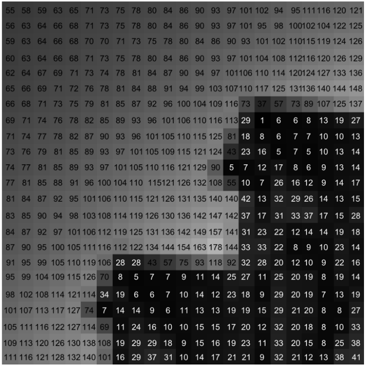
Dima Damen
Dima.Damen@bristol.ac.uk

Department of Computer Science,
University Of Bristol,
Bristol, United Kingdom.

June 22, 2019

What is Computer Vision?

- A digital image is just a bunch of samples (pixels) and quantised values (colour)



55	58	59	63	65	71	73	75	78	80	84	86	90	93	97	101	102	94	95	111	116	120	121
56	63	64	66	68	71	73	75	78	80	84	86	90	93	97	101	95	98	100	102	104	122	125
59	63	64	66	68	70	70	71	73	75	78	80	84	86	90	93	101	102	110	115	119	124	126
60	63	64	66	68	71	73	75	78	80	84	86	90	93	97	101	104	108	112	116	120	126	129
62	64	67	69	71	73	74	78	81	84	87	90	94	97	101	106	110	114	120	124	127	133	136
65	66	69	71	72	76	78	81	84	88	91	94	99	103	107	110	117	125	131	136	140	144	148
66	68	71	73	75	79	81	85	87	92	96	100	104	109	116	73	37	57	73	89	107	125	137
69	71	74	76	78	82	85	89	93	96	101	106	110	116	113	29	1	6	6	8	13	19	27
71	74	77	78	82	87	90	93	96	101	105	110	115	125	81	18	8	6	7	7	10	10	13
73	76	79	81	85	89	93	97	101	105	109	115	121	124	43	23	16	5	7	5	10	13	14
74	77	81	85	89	93	97	101	105	110	116	121	129	90	5	7	12	17	8	6	9	13	14
77	81	85	88	91	96	100	104	110	115	121	126	132	108	55	10	7	26	16	12	9	14	17
81	84	87	92	95	101	106	110	115	121	126	131	135	140	140	42	13	32	29	26	14	13	15
83	85	90	94	98	103	108	114	119	126	130	136	142	147	142	37	17	31	33	37	17	15	28
84	87	92	97	101	106	112	119	125	131	136	142	149	157	141	31	23	22	12	14	14	19	18
87	90	95	100	105	111	116	112	122	134	144	154	163	178	144	33	33	22	8	9	10	23	14
91	95	99	105	110	119	106	28	28	43	57	75	93	118	92	32	28	20	12	10	9	22	16
95	99	104	109	115	126	70	8	5	7	7	9	11	14	25	27	11	25	20	19	8	19	14
98	102	108	114	121	114	34	19	6	6	7	10	14	12	23	18	9	29	20	19	7	13	19
101	107	113	117	127	74	7	14	14	9	6	11	13	13	19	19	15	29	21	20	8	8	27
105	111	116	122	127	114	69	11	24	16	10	10	15	15	17	20	12	32	20	18	8	10	33
109	113	120	126	130	138	108	19	29	29	18	9	15	16	19	23	11	33	20	15	8	25	38
111	116	121	128	132	140	101	16	29	37	31	10	14	17	21	21	9	32	21	12	13	38	41

Dima Damen
Dima.Damen@bristol.ac.uk

What is Computer Vision?

- It all started with a summer project!

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

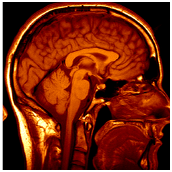
THE SUMMER VISION PROJECT

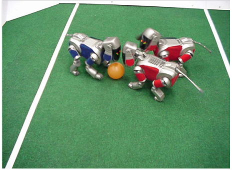
Seymour Papert

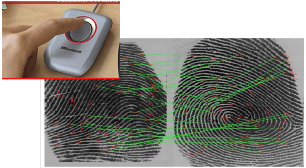
The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

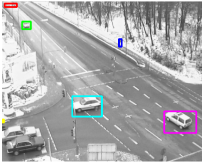
What is Computer Vision?

- Now covers a wide range of applications
- In the second lecture, we will investigate video understanding (particularly action recognition) my area of research and expertise.





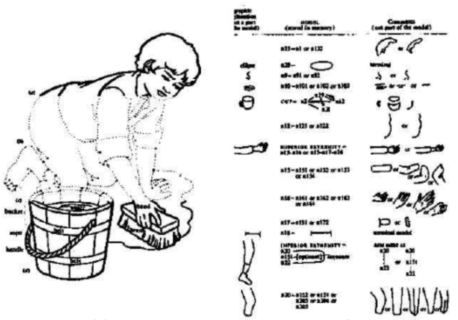




Dima Damen
Dima.Damen@bristol.ac.uk

Early Vision Attempts

- Early computer vision methods tried to model the world, without using training data

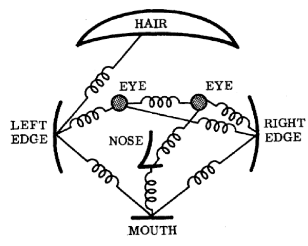


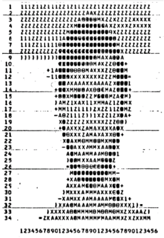
b) c) [1]

[1] A. Guzman (1971). Analysis of curved line drawings using context and global information. Machine Intelligence 6

Early Vision Attempts

- Early computer vision methods tried to model the world, without using training data





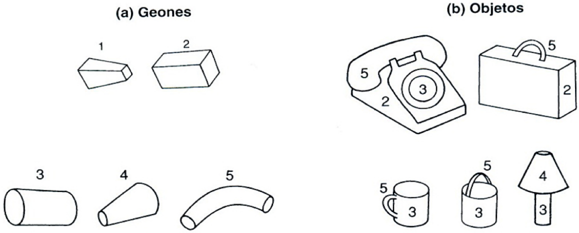
Original picture.

HAIR WAS LOCATED AT (7, 23)
L/EDGE WAS LOCATED AT (17, 13)
R/EDGE WAS LOCATED AT (17, 26)
L/EYE WAS LOCATED AT (14, 17)
R/EYE WAS LOCATED AT (14, 23)
NOSE WAS LOCATED AT (20, 20)
MOUTH WAS LOCATED AT (22, 20) [2]

[2] Fischler, M.A.; Elschlager, R.A. (1973). The Representation and Matching of Pictorial Structures. IEEE Transactions on Computers: 67.

Early Vision Attempts

- Early computer vision methods tried to model the world, without using training data



[3]

[3] Biederman, I. (1987) Recognition-by-components: a theory of human image understanding. Psychol Rev. 1987;94(2):115-47.

Meanwhile, Machine Learning was rising

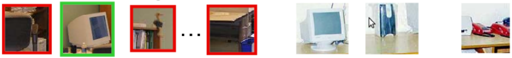
- In the meanwhile, another field was being developed.
- In 1959, Arthur Samuel defined Machine Learning (ML) as a "Field of study that gives computers the ability to learn without being explicitly programmed"[4]
- In 1990s, several attempts started to formulate vision as a **learning problem**

[4] Wikipedia

Dima Damen
Dima.Damen@bristol.ac.uk

Computer Vision as a Learning Problem

- Formulation: binary classification



Features $x =$	x_1	x_2	x_3	$\cdots$	x_N	x_{N+1}	x_{N+2}	$\cdots$	x_{N+M}
Labels $y =$	-1	+1	-1		-1	?	?		?

Training data: each image patch is labeled as containing the object or background

Test data

- Classification function

$\hat{y} = F(x)$ Where $F(x)$ belongs to some family of functions

- Minimize misclassification error

(Not that simple: we need some guarantees that there will be generalization)

What is Computer Vision?

- Methods that use training data quickly outperformed modelling approaches (1990+).
- Machine learning is now a core part of computer vision.
- Nearly every machine learning algorithm has been used in one way or another in computer vision.
- Visual data (images and videos) is a new source for machine learning scientists.
- In fact the rise of Deep Learning is due to the success it had on vision (images) data.

Early Success - Viola & Jones Face Detector (2001)

BBC NEWS

LIVE BBC NEWS CHANNEL

News services
Your news when you want it

News Front Page
World
UK
England
Northern Ireland
Scotland
Wales
Business
Market Data
Your Money
Economy
Companies
Politics
Health
Education
Science & Environment
Technology
Entertainment

Last Updated: Monday, 6 February 2006, 14:29 GMT

E-mail this to a friend   Printable version

Face-hunting cameras boost Nikon

Japanese camera maker Nikon has tripled its profits on the back of strong sales of digital cameras that automatically focus on human faces.

Operating profit for the three months to 31 December was 19.8bn yen ($167m; £95m), up from 5.9bn yen in 2004.

Face recognition cameras like the Coolpix L1 are popular

Nikon said that sales of compact digital cameras had been boosted by the success of new face recognition models.

It had also seen strong sales of its digital "SLR" cameras with interchangeable lenses and bodies

SEE ALSO:
- Nikon to focus on digital cameras
  12 Jan 06 | Business
- Digital trouble hits Nikon shares
  10 Feb 04 | Business
- Why digital cameras = better photographers
  20 Jan 04 | Magazine
- R.I.P. 35mm Camera
  15 Jan 04 | Magazine

RELATED INTERNET LINKS:
- Nikon

The BBC is not responsible for the content of external internet sites

TOP BUSINESS STORIES
- Unemployment dips to 2.47

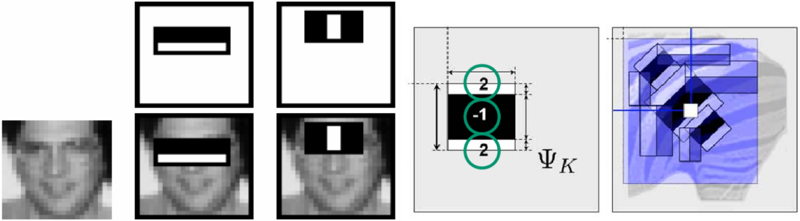
Early Success - Viola & Jones Face Detector (2001)

Early Success - Viola & Jones Face Detector (2001)

Paul Viola
MIT (1996-2000)
MERL (2001-2002)
Microsoft (2002 - 2014)
Amazon (2014 - now)

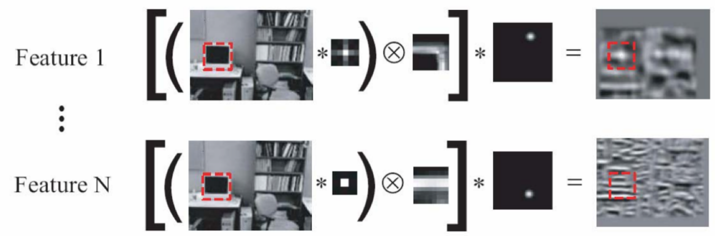
Michael Jones
Compaq (-2000)
MERL (2001-now)

Early Success - Viola & Jones Face Detector (2001)

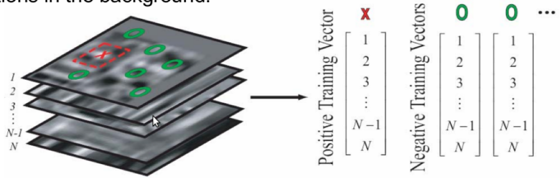


Early Success - Viola & Jones Face Detector (2001)

First we evaluate all the N features on all the training images.



Then, we sample the feature outputs on the object center and at random locations in the background:



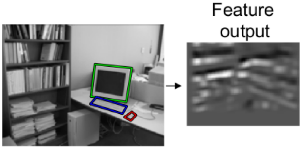
Early Success - Viola & Jones Face Detector (2001)

- Training Data + 10,000 negative examples were selected by randomly picking sub-windows from 9500 images which did not contain faces

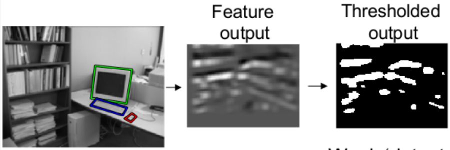
Early Success - Viola & Jones Face Detector (2001)

- AdaBoost Classification - a method for supervised learning
- Weak classifiers: classifiers that perform slightly better than chance. (error < 0.5)
- Boosting is an iterative algorithm that repeatedly constructs a hypothesis aimed at correcting mistakes of the previous hypothesis
- Introduced by Freund & Shapire (1995)

Early Success - Viola & Jones Face Detector (2001)

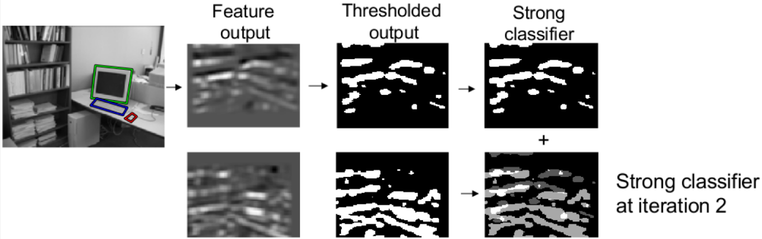


Early Success - Viola & Jones Face Detector (2001)

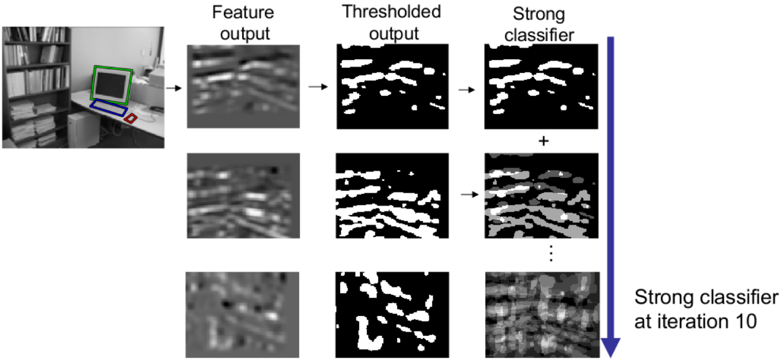


Weak 'detector'
Produces many false alarms.

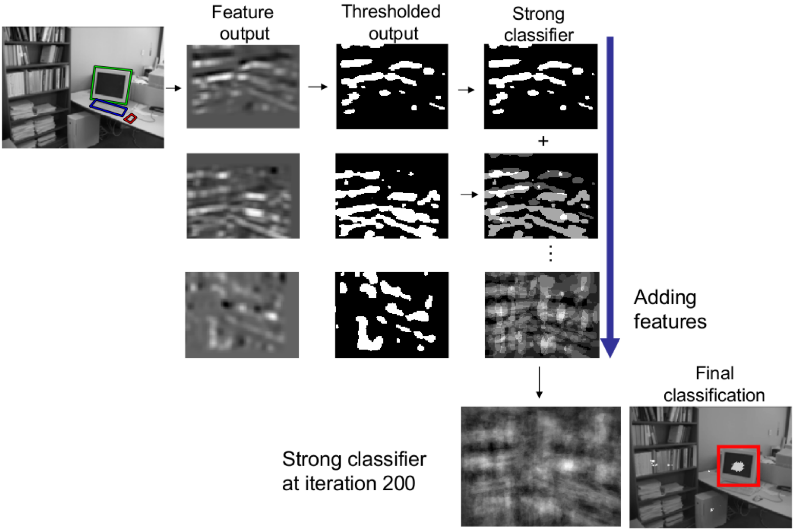
Early Success - Viola & Jones Face Detector (2001)



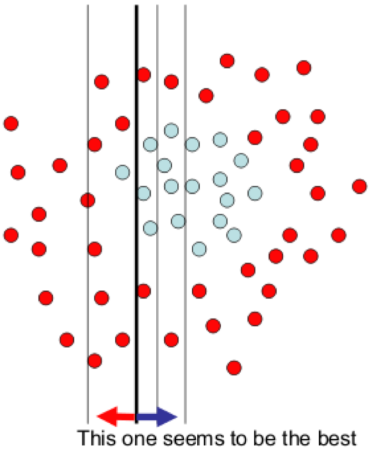
Early Success - Viola & Jones Face Detector (2001)



Early Success - Viola & Jones Face Detector (2001)



Early Success - Viola & Jones Face Detector (2001)



Each data point has

a class label:

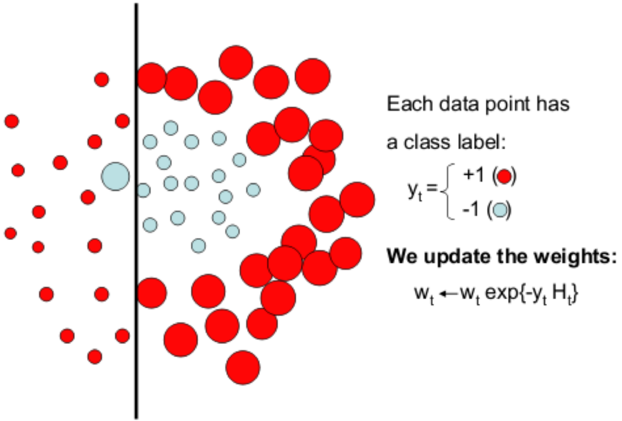
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

and a weight:

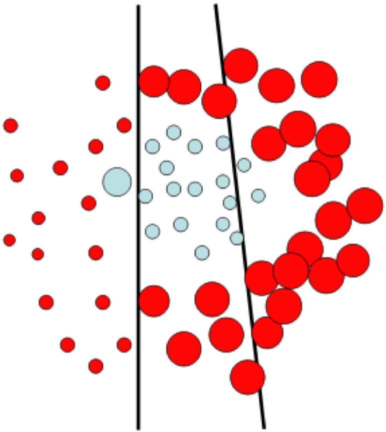
$w_t = 1$

This is a '**weak classifier**': It performs slightly better than chance.

Early Success - Viola & Jones Face Detector (2001)



Early Success - Viola & Jones Face Detector (2001)



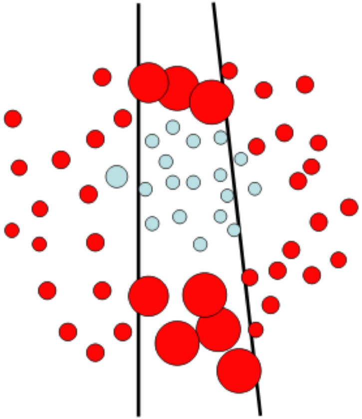
Each data point has a class label:

$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Early Success - Viola & Jones Face Detector (2001)



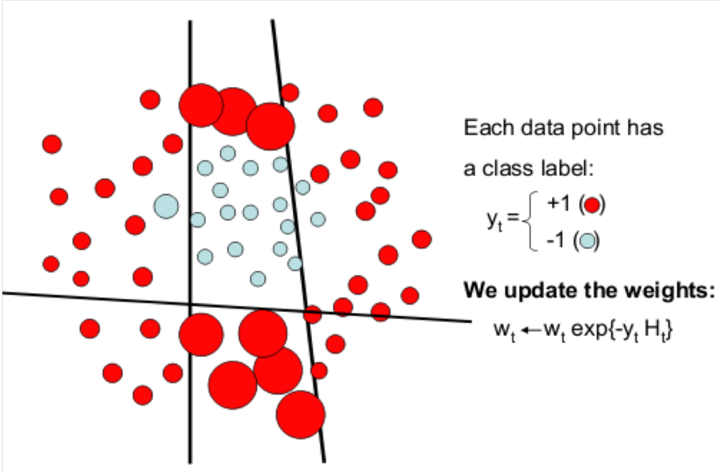
Each data point has a class label:

$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

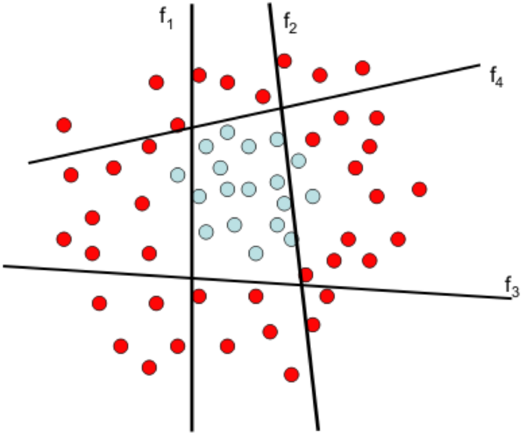
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

Early Success - Viola & Jones Face Detector (2001)



Early Success - Viola & Jones Face Detector (2001)

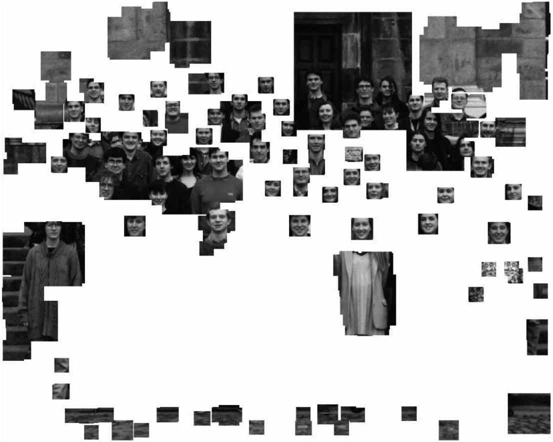


The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Early Success - Viola & Jones Face Detector (2001)

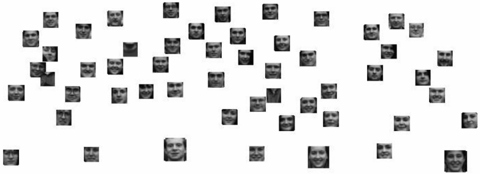
Early Success - Viola & Jones Face Detector (2001)

Early Success - Viola & Jones Face Detector (2001)



Early Success - Viola & Jones Face Detector (2001)

Early Success - Viola & Jones Face Detector (2001)



Early Success

- The processing time of a 384 by 288 pixel image on a conventional personal computer (back in 2001) about 0.067 seconds.
- Free implementation of it available as part of OpenCV
- The age of Computer Vision has begun
- Machine Learning has saved Computer Vision
- With it, came the need for **labeled training data**

Collecting Training Data!

- The PASCAL Challenge
- In 2006 - 5,304 images, - 9,507 objects

Collecting Training Data!

- Challenges and evaluation metrics are the base for publishing

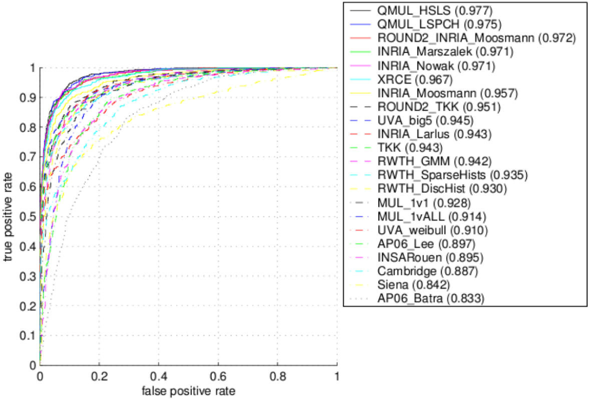
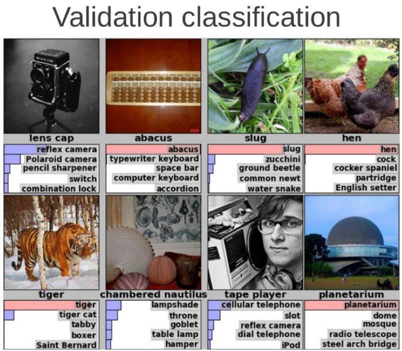


Figure 5: Competition 1.3: car (all entries)

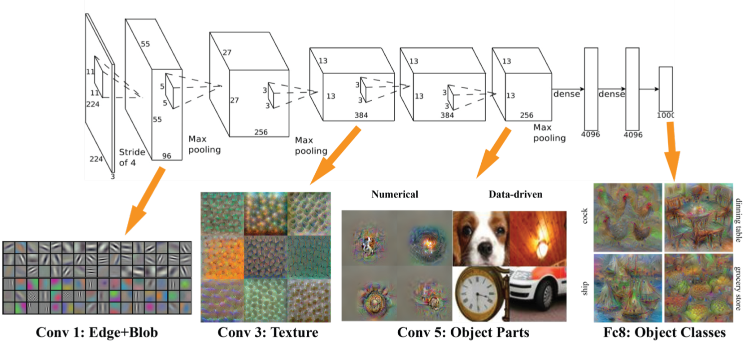
Collecting Training Data!

- IMAGENET
- In 2009 - 10M labeled images depicting 10,000 object categories



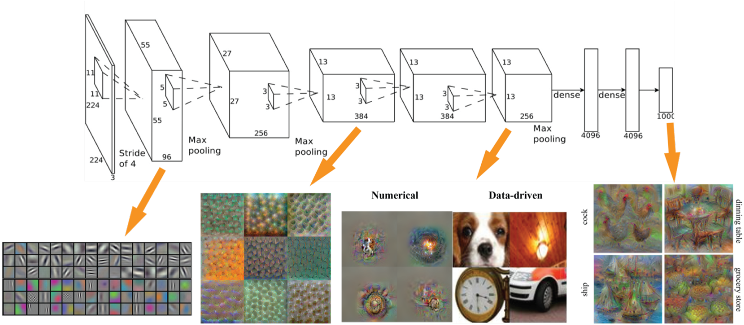
ImageNet and Convolutional Neural Networks

- Until 2012, methods that won the ImageNet challenge followed one strategy:
  - Step 1: Feature Extraction (choose the right features)
  - Step 2: Classification
- In 2012, both Computer Vision and Machine Learning have been transformed by CNNs



ImageNet and Convolutional Neural Networks

- From then, methods that use CNNs outperformed previous approaches on every problem tackled.
- Deep Machine Learning saved Computer Vision once again.



Conv 1: Edge+Blob Conv 3: Texture Conv 5: Object Parts Fc8: Object Classes

What is a Convolutional Neural Network (CNN)

- Not every Deep Neural Network (DNN) is a Convolutional Neural Network (CNN)
- By the end of this summer school you will be familiar with 3 types of DNNs
  - Fully-Connected DNN
  - Convolutional DNN
  - Recurrent DNN
- CNNs could be credited for the recent success of Neural Networks[5]
- The term was first used by LeCun in his technical report: “Generalization and network design strategies” (1989).

[5] Arguably!

When to use CNNs?

- CNNs expect the **input** data x has a known **grid-like topology**
- Typical examples:
  - Audio: 1-D recurring data at regular time intervals
  - Images: 2-D grid of pixels
  - Video: 3-D (sequence of 2-D grid of pixels)
- As the input is grid-like, operations might apply to individual or groups of grid cells.
- Accordingly, CNN is a neural network that uses *convolution* in place of general matrix multiplication *in at least one of its layers*.

Kernels vs Tensors

- The *convolution* operation is typically denoted with $*$

$$\mathrm{x} * \omega \tag{1}$$

  where x is the **input** and ω is the **kernel**, also known as the **feature map**
- Traditionally, these kernels were manually defined for specific purposes, e.g. edges in Viola & Jones face detector
- In CNNs, kernels are trained/learnt from data, for one or multiple tasks
- Moreover, multiple *dependent* kernels are trained/learnt in one go
- In CNN, x is a multidimensional array of data, and ω is a multidimensional array of kernels - referred to as **a tensor**

Convolution vs Correlation

- Using the convolution operator, for x and ω , the result S would be

$$S(i,j) = (\mathrm{x} * \omega)(i,j) = \sum_m \sum_n \mathrm{x}(m,n)\omega(i-m,j-n) \tag{2}$$

- A main property of convolution is that it is cummutative

$$S(i,j) = (\mathrm{x} * \omega)(i,j) = (\omega * \mathrm{x})(i,j) = \sum_m \sum_n \mathrm{x}(i-m,j-n)\omega(m,n) \tag{3}$$

- The commutative property of the *convolution* operator is because we have **flipped** the kernel relative to the input - when m increases, the index of x increases but the index of ω decreases
- *The only reason to flip the kernel is to obtain the cummutative property - helpful in writing proofs*

Convolution vs Correlation

- The convolution operator is used for theoretical proofs

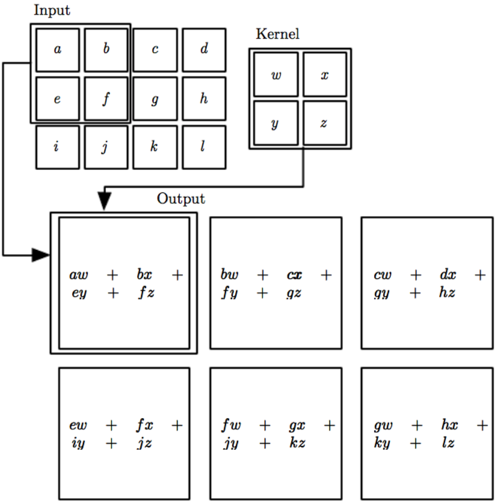
$$S(i,j) = (\mathrm{x} * \omega)(i,j) = \sum_m \sum_n \mathrm{x}(m,n)\omega(i-m,j-n) \tag{4}$$

- However, *most* DNN libraries implement the convolution as a **cross-correlation** operation, without flipping the kernel[6]

$$S(i,j) = (\mathrm{x} * \omega)(i,j) = \sum_m \sum_n \mathrm{x}(i+m,j+n)\omega(m,n) \tag{5}$$

[6] We do not have a good reason to call them CNNs really!

Convolution vs Correlation



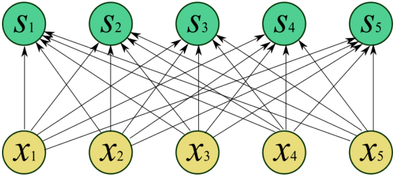
Reference: Goodfellow et al (2016) p325

Convolutional Neural Networks

- And now... to the main attraction **Convolutional Neural Networks (CNN)**
- Three primary properties distinguish fully-connected networks from convolutional neural networks:
  1. Sparse Interactions
  2. Parameter Sharing
  3. Equi-variant Representations

CNN Properties: 1- Sparse Interactions[7]

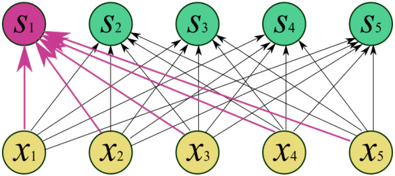
- A major difference between fully connected neural networks and CNNs are the contributions of input units to output units.
- Consider this two-layer fully-connected network, with 5 input units,



[7] Also referred to as multi-scale interactions

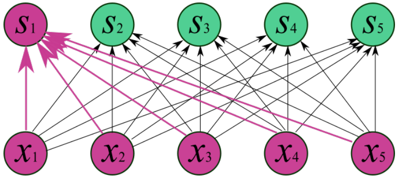
CNN Properties: 1- Sparse Interactions

- A major difference between fully connected neural networks and CNNs are the contributions of input units to output units.
- For one output unit s_1,



CNN Properties: 1- Sparse Interactions

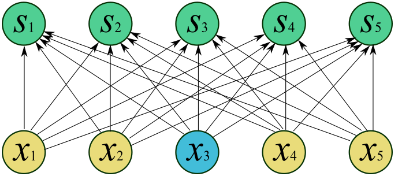
- A major difference between fully connected neural networks and CNNs are the contributions of input units to output units.
- its value is decided from all 5 input units
  $s_1 = f(x_1, x_2, x_3, x_4, x_5; \omega_1, \omega_2, \omega_3, \omega_4, \omega_5)$.



Dima Damen
Dima.Damen@bristol.ac.uk

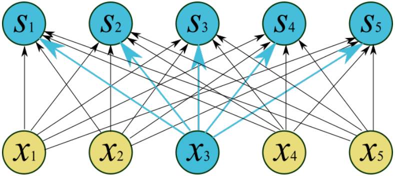
CNN Properties: 1- Sparse Interactions

- A major difference between fully connected neural networks and CNNs are the contributions of input units to output units.
- similarly, each input unit, e.g. x_3,



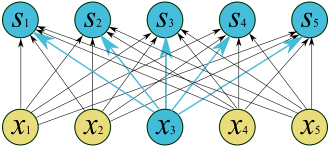
CNN Properties: 1- Sparse Interactions

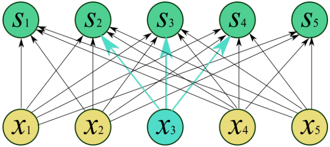
- A major difference between fully connected neural networks and CNNs are the contributions of input units to output units.
- similarly, each input unit, e.g. x_3, contributes to all output units



CNN Properties: 1- Sparse Interactions

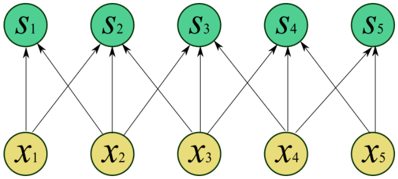
- In **CNNs**, due to the grid structure, it is sufficient to limit the number of connections from each input unit unit to k,
- See the connections from x_3





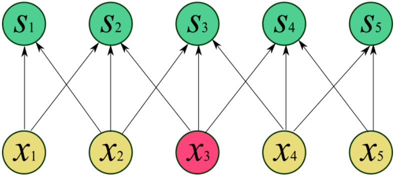
CNN Properties: 1- Sparse Interactions

- In **CNNs**, due to the grid structure, it is sufficient to limit the number of connections from each input unit unit to k,
- resulting in sparse weights - and sparse interactions between input and output



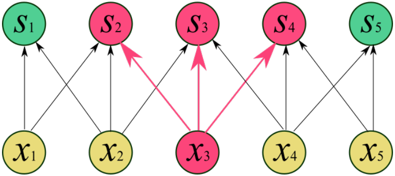
CNN Properties: 1- Sparse Interactions

- In **CNNs**, one input unit x_3,



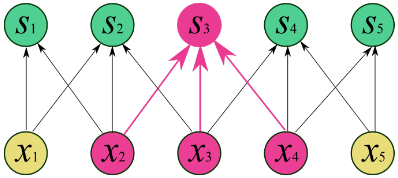
CNN Properties: 1- Sparse Interactions

- In **CNNs**, one input unit x_3, affects a limited number of output units



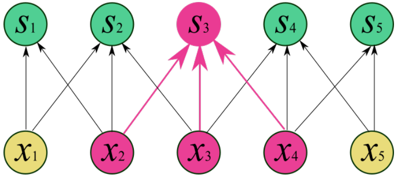
CNN Properties: 1- Sparse Interactions

- Similarly, the input units affecting a certain output unit (e.g. s_3),



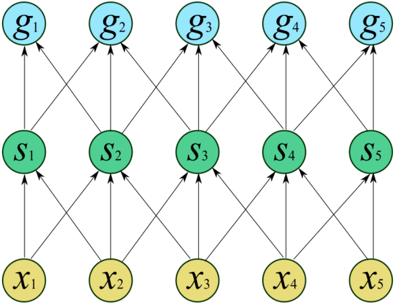
CNN Properties: 1- Sparse Interactions

- The input units affecting a certain output unit (e.g. s_3), are known as the unit's **receptive field**.



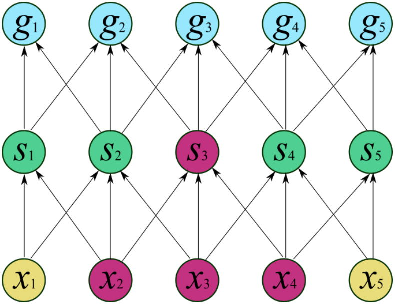
CNN Properties: 1- Sparse Interactions

- Interestingly, as more layers are added,



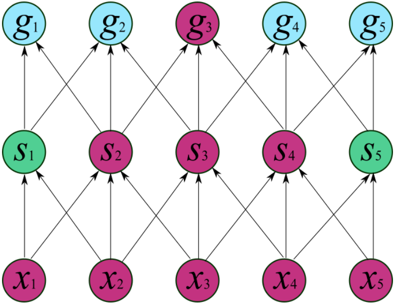
CNN Properties: 1- Sparse Interactions

- Interestingly, as more layers are added,



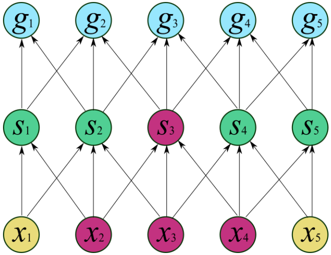
CNN Properties: 1- Sparse Interactions

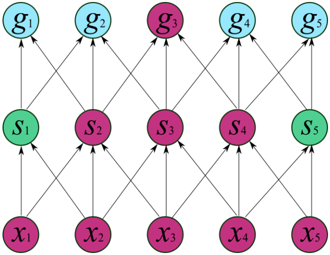
- Interestingly, as more layers are added,



CNN Properties: 1- Sparse Interactions

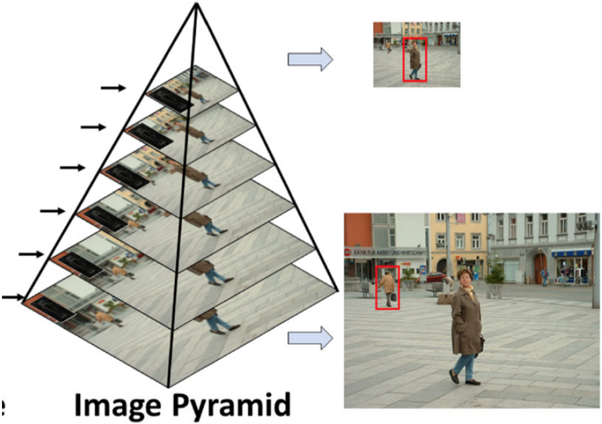
- **The receptive field of the units in the deeper layers of a CNN is larger than the receptive field of the units in the shallow layers**





CNN Properties: 1- Sparse Interactions

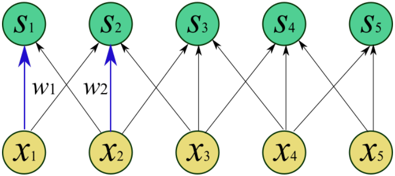
- Is this new?



Suleiman and Sze, An Energy-Efficient Hardware Implementation of HOG-Based Object Detection, 2015

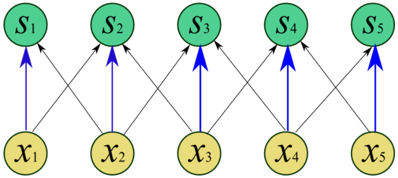
CNN Properties: 2- Parameter Sharing

- **Paramter sharing** refers to using the same parameter for more than one function in the network
- You can consider this as tying two paramters w_1 and w_2 together, so they can only have the same value
- You have dropped the number of parameters you need to train by 1 (!)



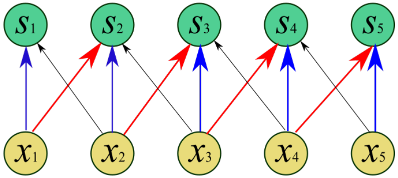
CNN Properties: 2- Parameter Sharing

- You can similarly think about sharing more parameters



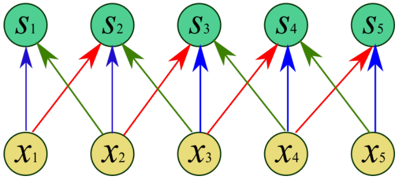
CNN Properties: 2- Parameter Sharing

- You can similarly think about sharing more parameters



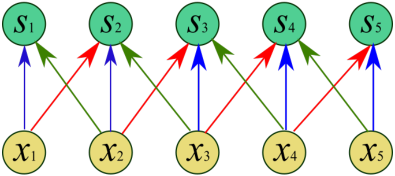
CNN Properties: 2- Parameter Sharing

- You can similarly think about sharing more parameters



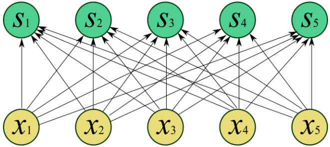
CNN Properties: 2- Parameter Sharing

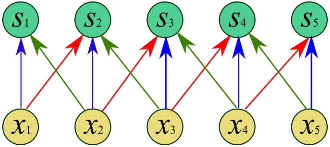
- Though parameter sharing on this network - with sparse interactions - the number of parameters to train is... 3 !!!



CNN Properties: 2- Parameter Sharing

- Compare the number of parameters in the fully-connected network to this CNN with sparse interactions and parameter sharing!
- Only 12% !!! :-)



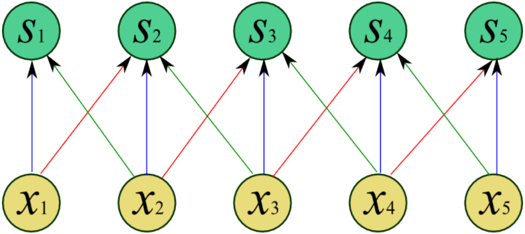


CNN Properties: 2- Parameter Sharing

- Parameter sharing is also known as **tied weights**, because the weight applied to one input is **tied** to the weight applied elsewhere.
- Does not affect the runtime of the forward pass
- Does significantly reduce the memory requirements for the model
- You have significantly less parameters to train, and thus you need less data
- But only works on the assumption that the data is grid-like and thus sharing the weights is a sensible idea!

CNN Properties: 2- Parameter Sharing

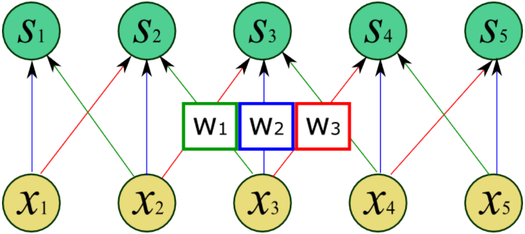
- It this new??



Dima Damen
Dima.Damen@bristol.ac.uk

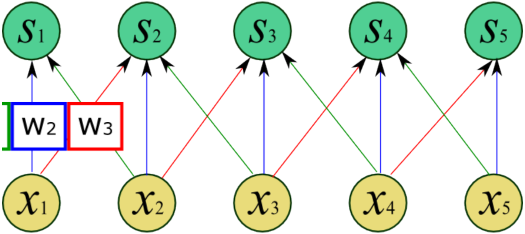
CNN Properties: 2- Parameter Sharing

- It this new??



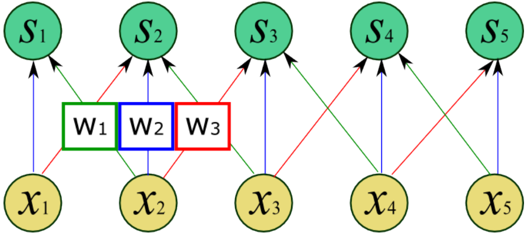
CNN Properties: 2- Parameter Sharing

- It this new??



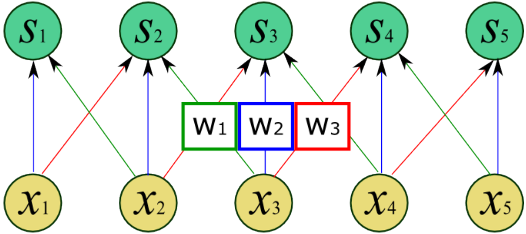
CNN Properties: 2- Parameter Sharing

- It this new??



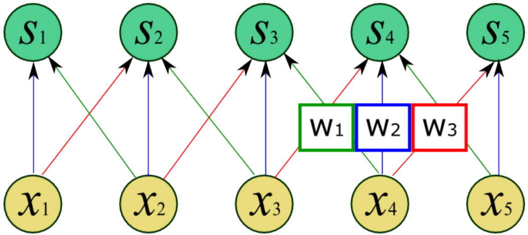
CNN Properties: 2- Parameter Sharing

- It this new??



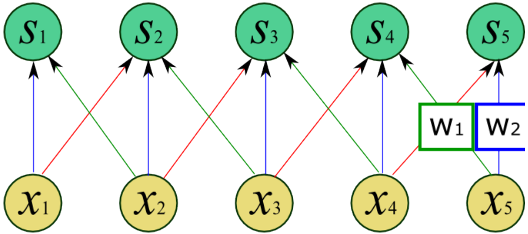
CNN Properties: 2- Parameter Sharing

- It this new??



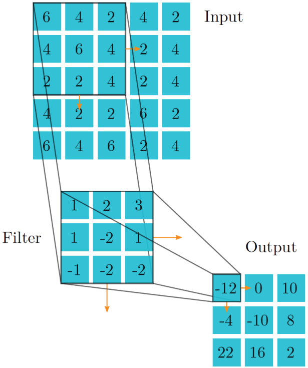
CNN Properties: 2- Parameter Sharing

- It this new?? **CONVOLUTION!!!** - or cross-correlation :-)



CNN Properties: 2- Parameter Sharing

- And in 2-D



Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

CNN Properties: 3- Equi-variant Representations

- As a result of the first two properties, CNNs exhibit some equivariance properties.
- A function is equivariant if when the input changes (or shifts) in a certain way, the output also changes in exactly the same way.
- CNNs are equi-variant to... translation
- This is of immense value in images for example. If an object moves in the input, its representation will move in the output in the same direction.
- However CNNs are NOT equivariant to... rotation or scale

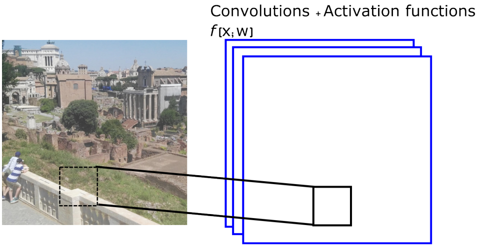
Your first CNN Layer

- Multiple convolutional layers $\rightarrow$ You can learn multiple features, e.g.

Source: Rob Fergus, NN, MLSS2015 Summer School Presentation

Your first CNN Layer

- Multiple convolutions can be piled
- Convolving a single kernel can extract one kind of feature
- We want to extract many kinds of features at many locations



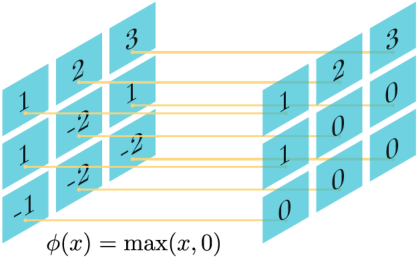
Dima Damen
Dima.Damen@bristol.ac.uk

Your first CNN Layer

- However, to make the most of the input, particularly around the edges/borders, one essential feature of any CNN implementation is **zero padding** the input to make it wider
- Without zero-padding, the input shrinks by one pixel less than the kernel width at each layer
- With zero-padding, the input and output are of the same size, unlike example below
- Without zero-padding, the number of convolutional layers that can be included in a network will be capped

Your first CNN Layer

- The convolutions are directly followed by activation functions, in the same fashion as fully-connected CNNs
- RELU activation function is shown in the example below

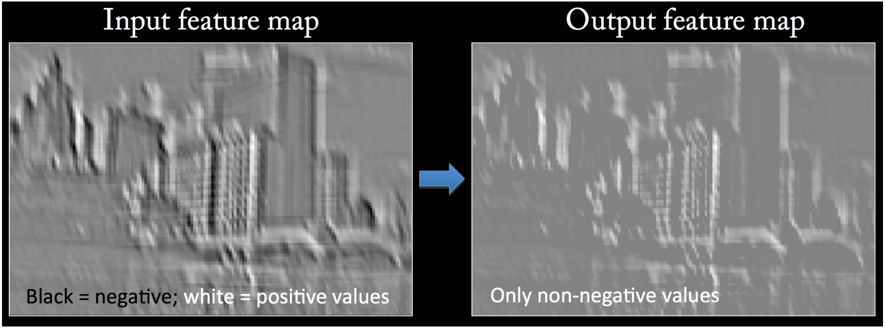


[8]

[8] Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

Your first CNN Layer

- The convolutions are directly followed by activation functions, in the same fashion as fully-connected CNNs
- RELU activation function is shown in the example below

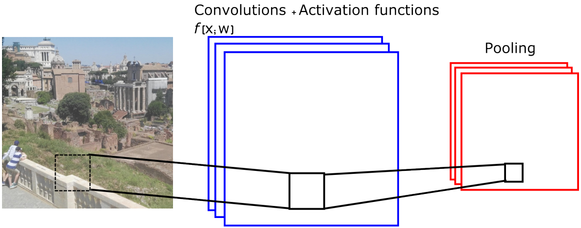


[9]

[9] Source: Rob Fergus, NN, MLSS2015 Summer School Presentation

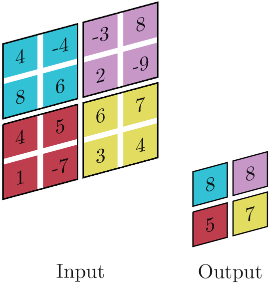
Your first CNN Layer

- **Pooling functions** are added to modify the output layer further, typically its size.
- A pooling function **replaces** the output of the net at a certain location, with a **summary** of the outputs in nearby outputs.



Your first CNN Layer

- **Max pooling**[10], for example, takes the maximum output within a rectangular neighbourhood.
- Pooling is almost always associated with downsampling,

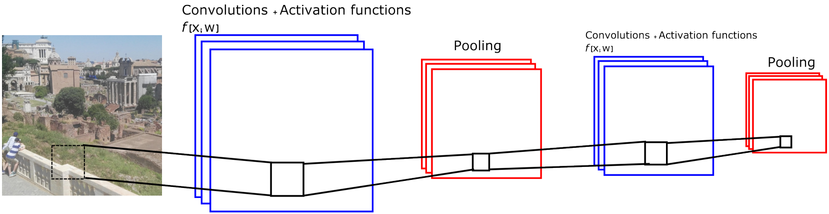


[10] First proposed by Zhou and Chellappa, 1988

Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

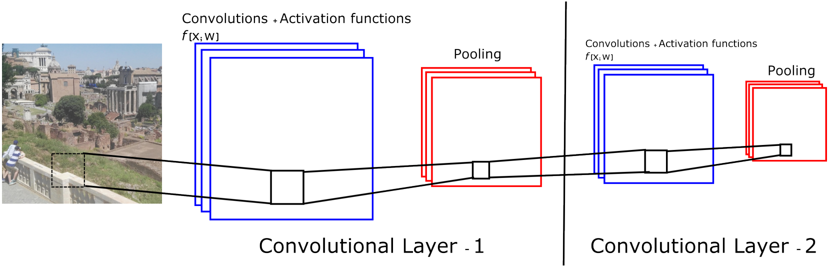
Your first CNN Architecture

- Further convolution $\rightarrow$ activation and $\rightarrow$ pooling with downsampling layers can be added



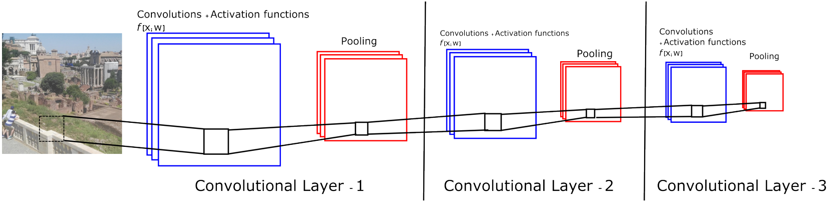
Your first CNN Architecture

- Technically, we would refer to these as the first and second convolutional layers of a deep CNN



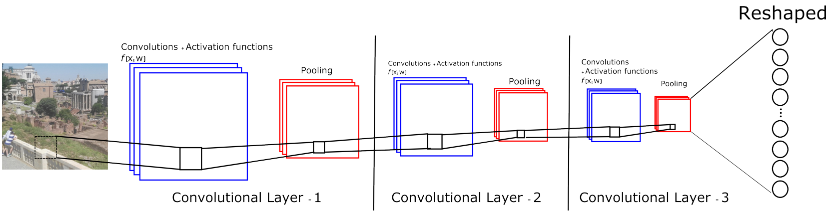
Your first CNN Architecture

- As multiple convolutional layers as added, filters with larger receptive fields are learnt



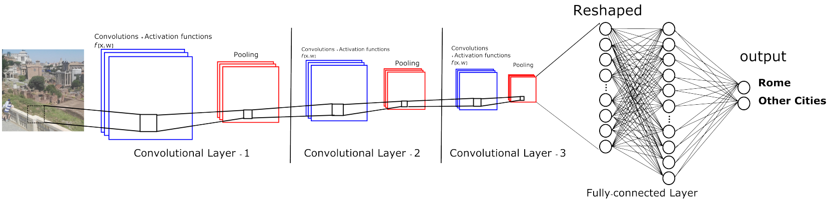
Your first CNN Architecture

- However, CNN architectures do not only have convolutions layers, they also have fully connected layers
- To use fully connecter layers, matrices are usually reshaped into 1-D



Your first CNN Architecture

- One or more fully connected layers can then be added

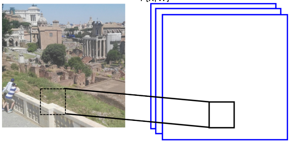


CNN Architecture Considerations

- In images for example, we have 3 channels (R/G/B)
- This means the input is 3D, and thus our convolutions are necessarily 3-D tensors

CNN Architecture Considerations

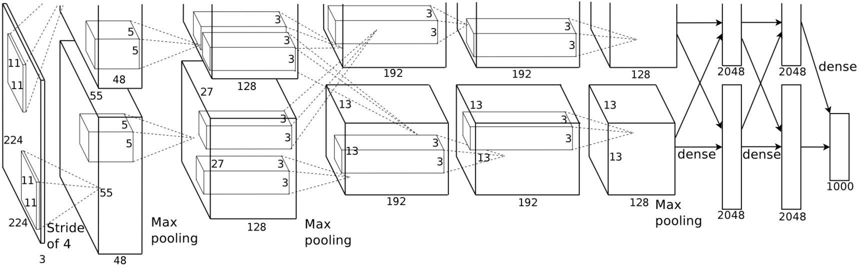
- In the previous example, the input and output sizes of the convolution+activation layers were equal



- However, practically we do not convolve densely, but instead we move the convolution skipping certain pixels.
- The number of pixels we skip, is referred to as the **stride** of the layer
- This results in downsampled convolutions
- It is possible to use separate strides for each dimension

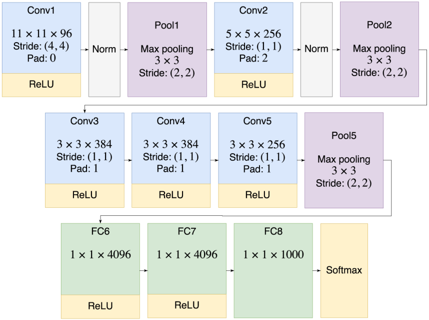
CNN Architectures - AlexNet

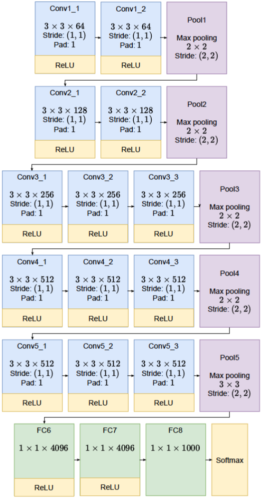
- When AlexNet won the most challenging computer vision task - Classifying 1000 classes by training from 10,000,000 images (The ImageNet Challenge), the new wave of CNN architectures started



Alex Krizhevsky, Sutskever and Hinton (2012) ImageNet Classification with Deep Convolutional Neural Networks, NIPS

CNN Architectures - AlexNet vs VGG-16





Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

CNN Architectures - Further architectures

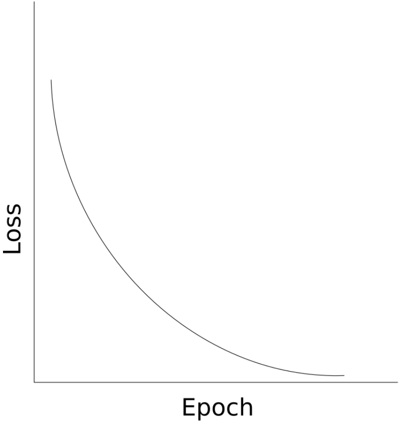
- See live demos at: `http://cs231n.stanford.edu`
- Visualise recent architectures at: `http://josephpcohen.com/w/visualizing-cnn-architectures-side-by-side-with-mxnet/`

Training CNNs

- The most expensive part of CNN training is learning the features
- The fully-connected layers are usually relatively inexpensive to train because of the small number of features provided as input
- When performing gradient descent, every gradient step requires a complete run of forward propagation and backward propagation through the entire network
- Several approaches have been proposed to solve this:
  - Greedily training one layer at a time, freezing the others
  - Use pre-trained features, only training the last convolutional layer with the fully-connected layers
  - Use random features, only training the fully-connected layers
  - Selected hand-crafted features (not recommended)
  - Apply k-means clustering to image patches and use the cluster centres for convolutions (Coates et al 2011)
- All these approaches were popular before mega-size datasets were introduced, and remain relevant for problems with small data sizes

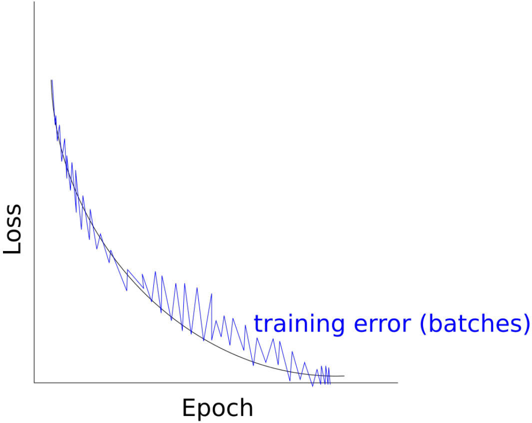
Training CNNs

- Once you built an architecture, defined your loss function, prepared your data, it's time to train a CNN
- Optimally, the loss decreases after each iteration



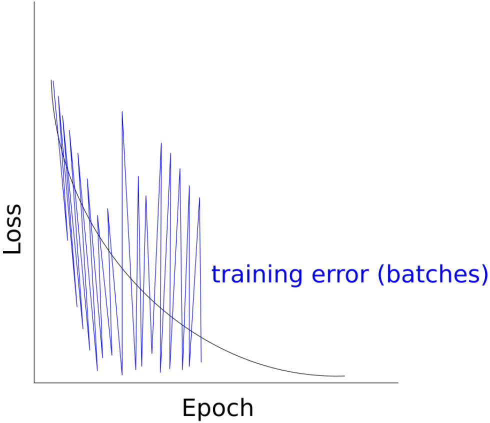
Training CNNs

- Practically, due to mini-batch optimisation, you see this wiggly curve



Training CNNs

- Training error wiggles A LOT?!
- Training error goes up?!
- It is all about your hyper-parameters...



Selecting hyperparameter values

- For each hyperparameters, one must understand the relationship between its value and each of the following:
  - Training error/loss
  - Testing error/loss (generalisation)
  - Computational resources (memory and runtime)

Dima Damen
Dima.Damen@bristol.ac.uk

1. Batch-size Training

- There are two extremes when performing gradient descent
  - Calculating the gradient from a single example
  - Calculating the gradient for the whole dataset
- Neither is ideal, thus we typically calculate the gradient from a number of data points
- This number is referred to as the ‘batch size’
- To correctly approximate the loss from a batch, it is crucial that samples are selected randomly
- However, there are approaches that sample data for a purpose **(read about hard mining)**

1. Batch-size Training

- The effect of changing the batch-size on the accuracy of a dataset, depends on the dataset
- The amount of wiggle in the training loss is related to the batch size.
- However there are general guidelines:
  - Larger batches indeed provide better approximation of the full-dataset gradient
  - However, as batch size increases, the increase in accuracy or the decrease in training time is NOT linear
  - Due to parallel processing, the typical limitation to batch size is the GPU memory
  - To make the most of the GPU architectures, batches that are a power of 2 provide the best runtime - 128, 256, ...
  - For small-sized data samples, batches between 32 and 256 are typical
  - For large-sized data samples, batches of 8 or 16 are common

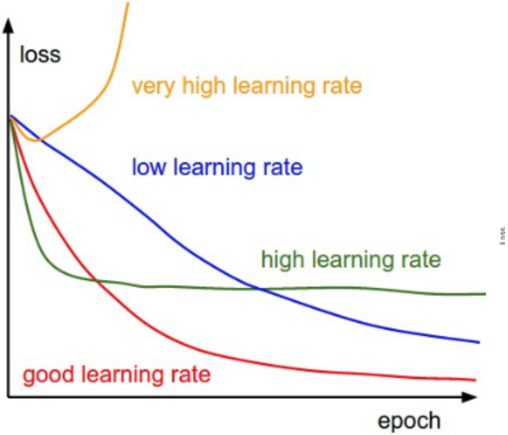
2. Learning Rate

- The learning rate is the **most important hyperparameter** to set

"If you have time to tune only one parameter, tune the learning rate"[11]

[11] Goodfellow et al, p 424

2. Learning Rate



http://cs231n.github.io/neural-networks-3/

3. Batch Normalisation

- Another powerful practical modification to the baseline training algorithm is known as **batch normalisation**
- Recall the standardisation approach to data

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{\sigma}$$

- The distribution of $\hat{\mathbf{x}}$ would then be zero-meaned with a standard deviation of 1
- When training using $\hat{\mathbf{x}}$ instead of $\mathbf{x}$, the training converges faster.
- You can compensate for that effect on the output by learnt variables

$$y = \boldsymbol{\lambda}\hat{\mathbf{x}} + \boldsymbol{\beta}$$

3. Batch Normalisation

- When applied to **each** layer in the network, all layers are given the chance to converge faster
- Using batch normalisation, higher learning rates can be used

Training CNNs

- Training CNNs to replicate a paper, is a good starting point
- Training a CNN for a new problem (new data, new loss function, ...) is far from trivial
- But it's not a dark art

Conclusion

- Well-done! You've been through a crash-course on Convolutional Neural Networks
- Importantly, you know more about Computer Vision - its origin and major turning points
- Next Lecture takes you beyond the basics
- During the lab, you will have hands-on expertise to build your first CNN

Further Reading

- Deep Learning

  Ian Goodfellow, Yoshua Bengio, and Aaron Courville
  MIT Press, ISBN: 9780262035613.
  - Chapter 9 – Convolutional Networks