Image Analysis (FMAN20)
Lecture 7, 2019

MAGNUS OSKARSSON

Support Vector Machines

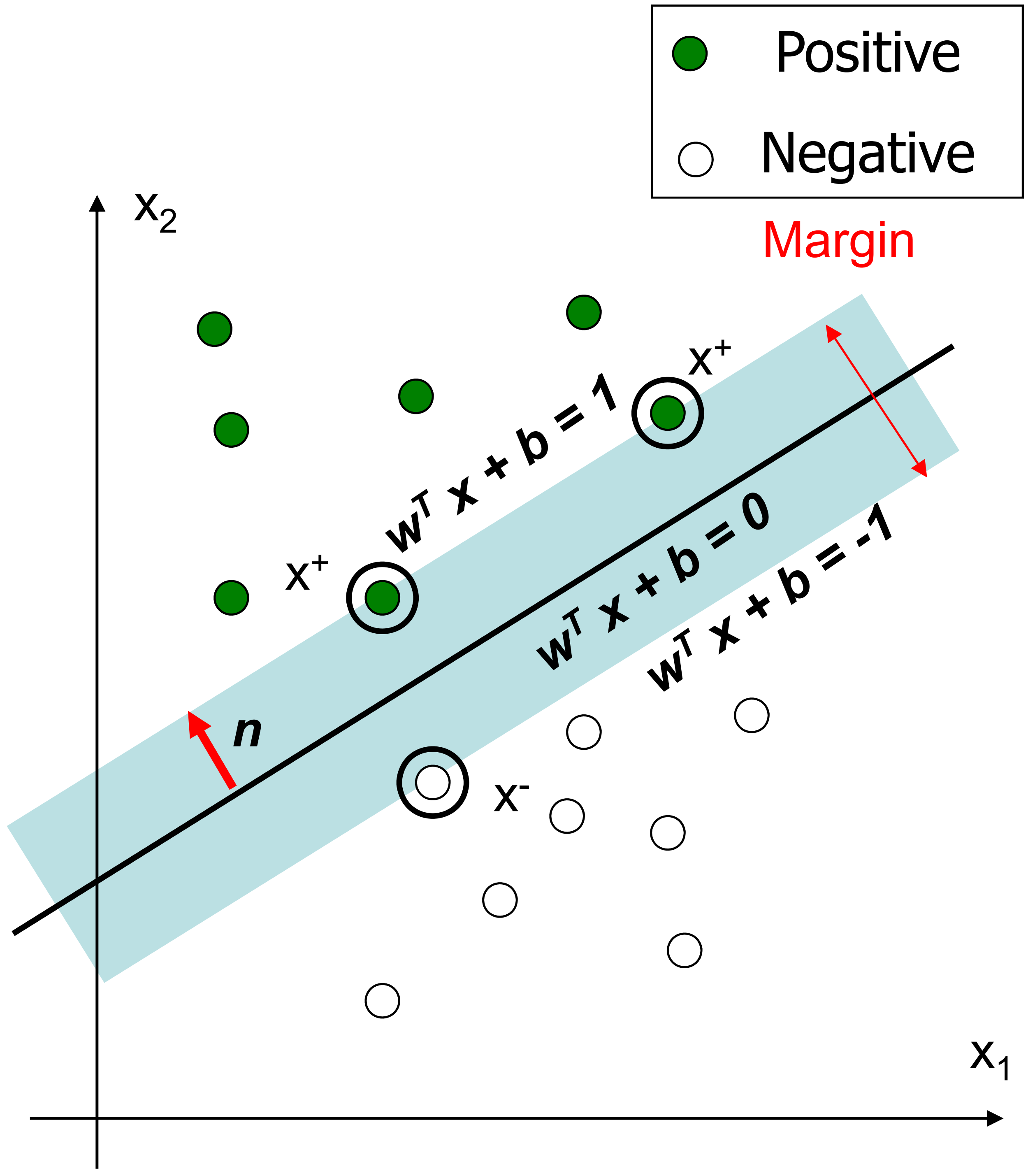
- Formulation:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

such that

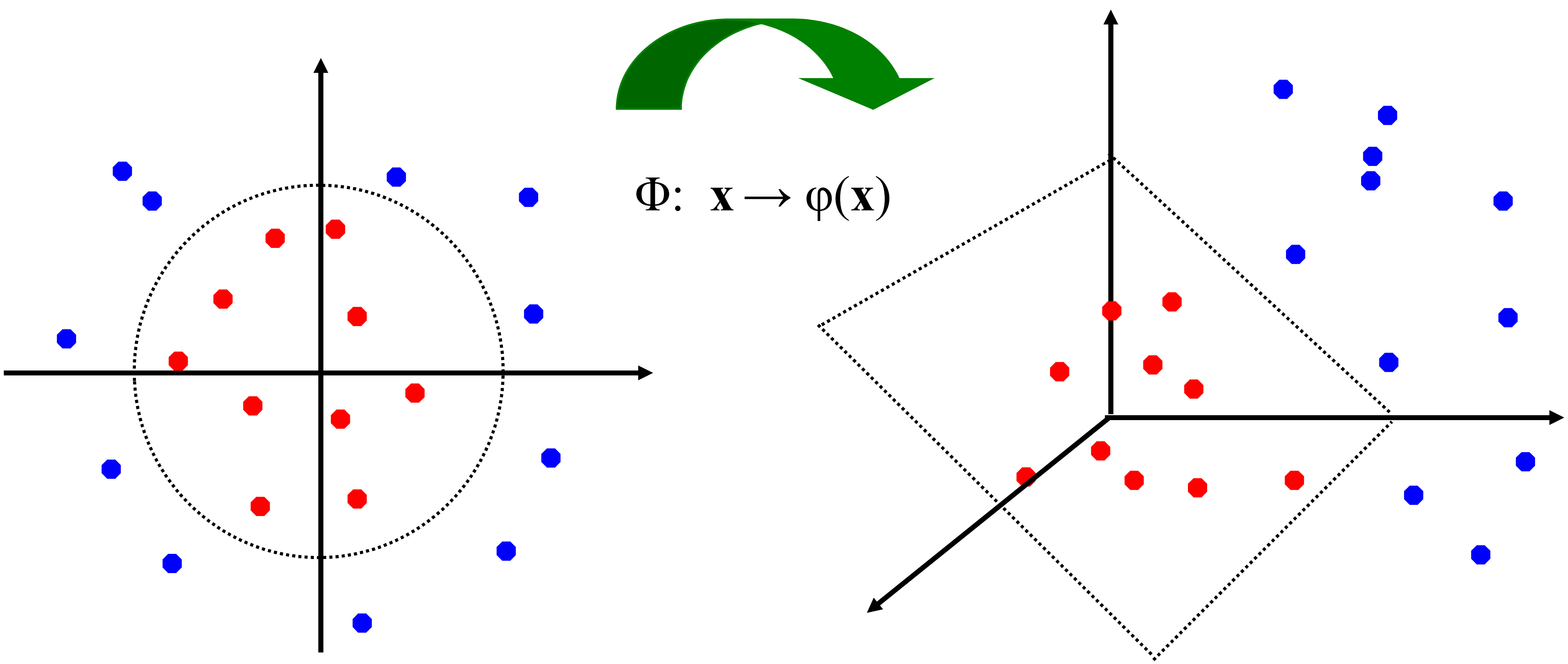
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

- Quadratic program with linear constraints



Non-linear SVMs: Feature Space

General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable



Support Vector Machine: Algorithm

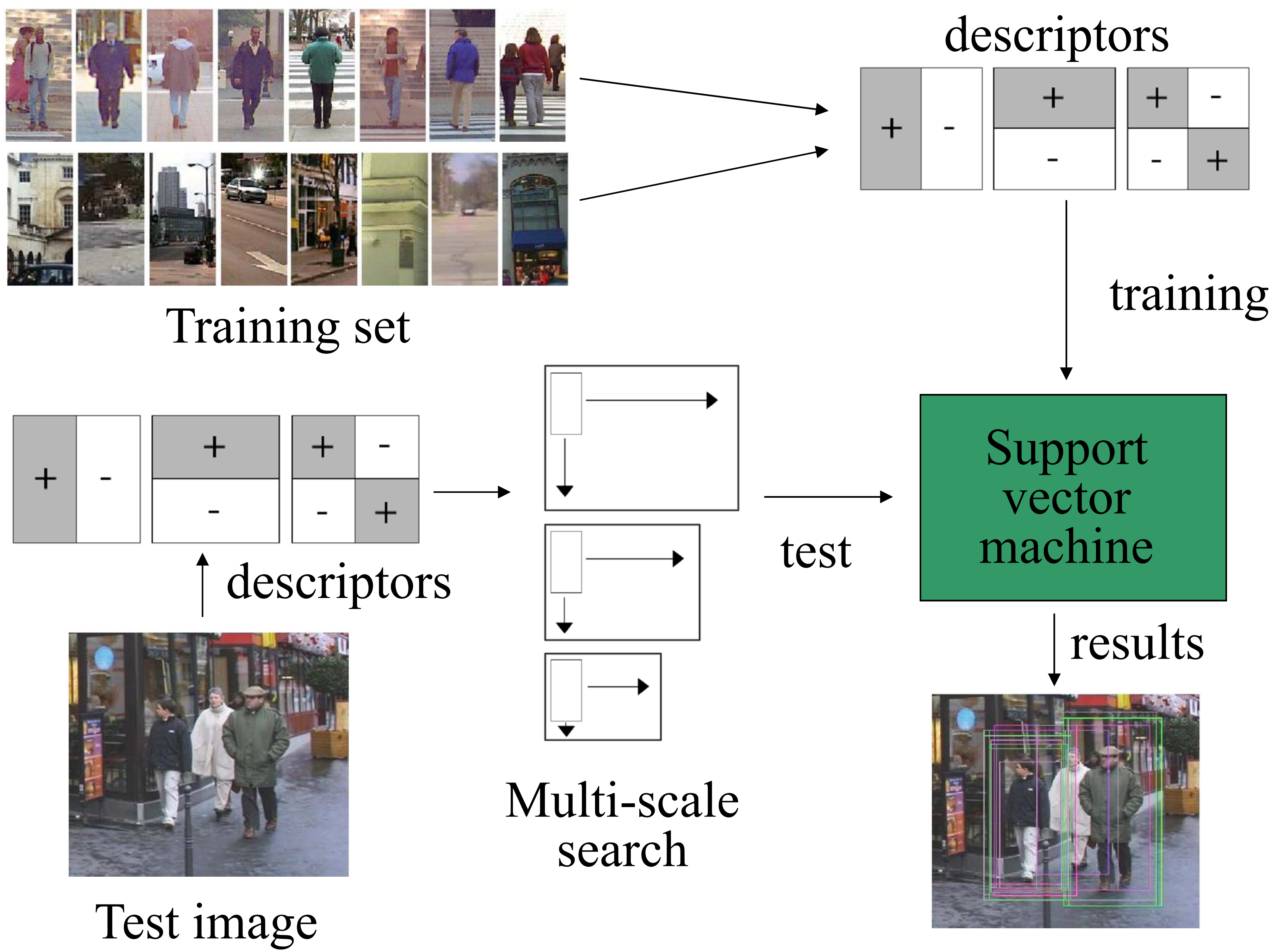
1. Choose a kernel function

2. Choose a value for C

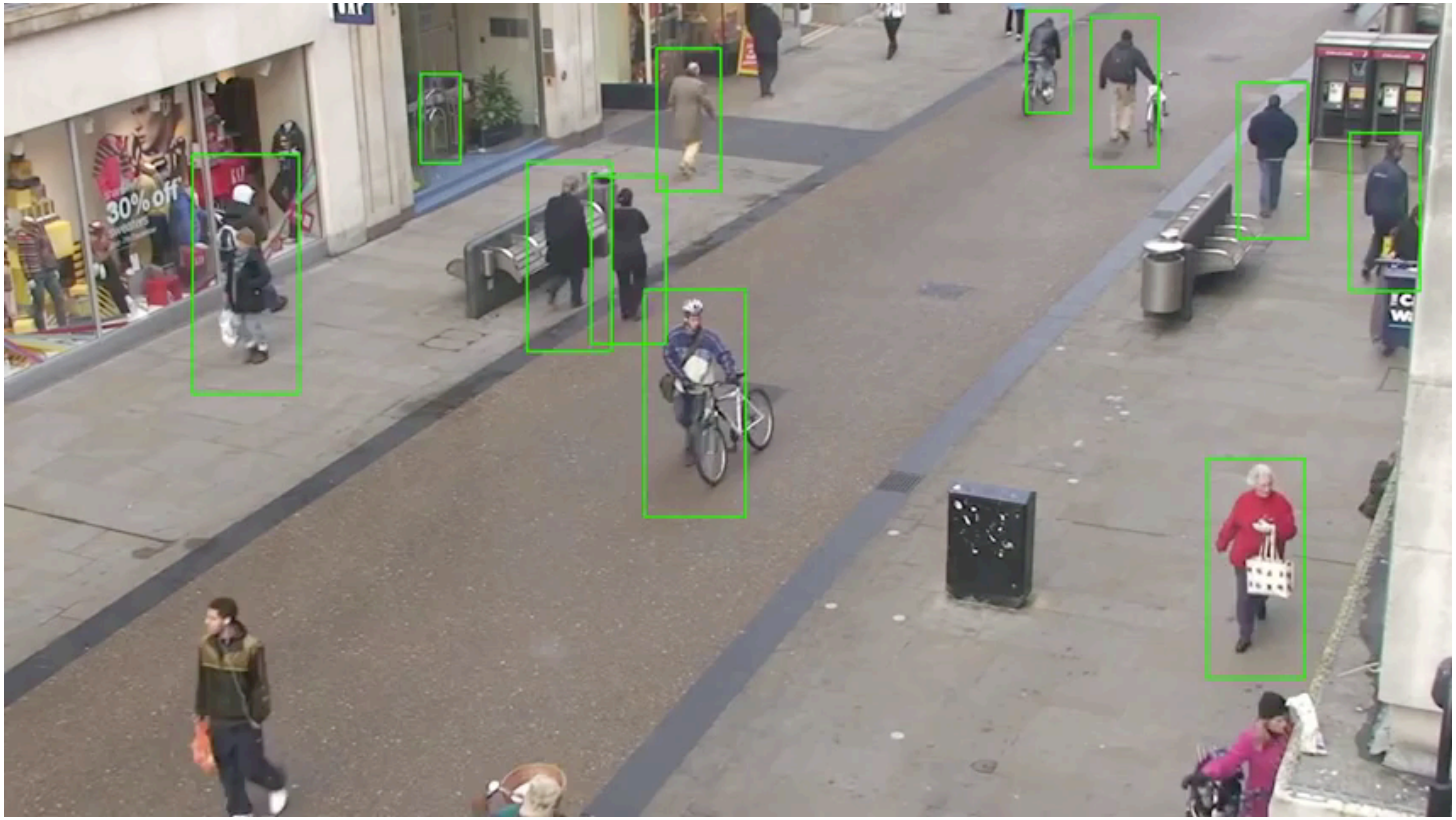
3. Solve the quadratic programming problem
   (many software packages available, e.g. libsvm)

4. Construct the discriminant function from the support vectors

Support Vector Machine Detector



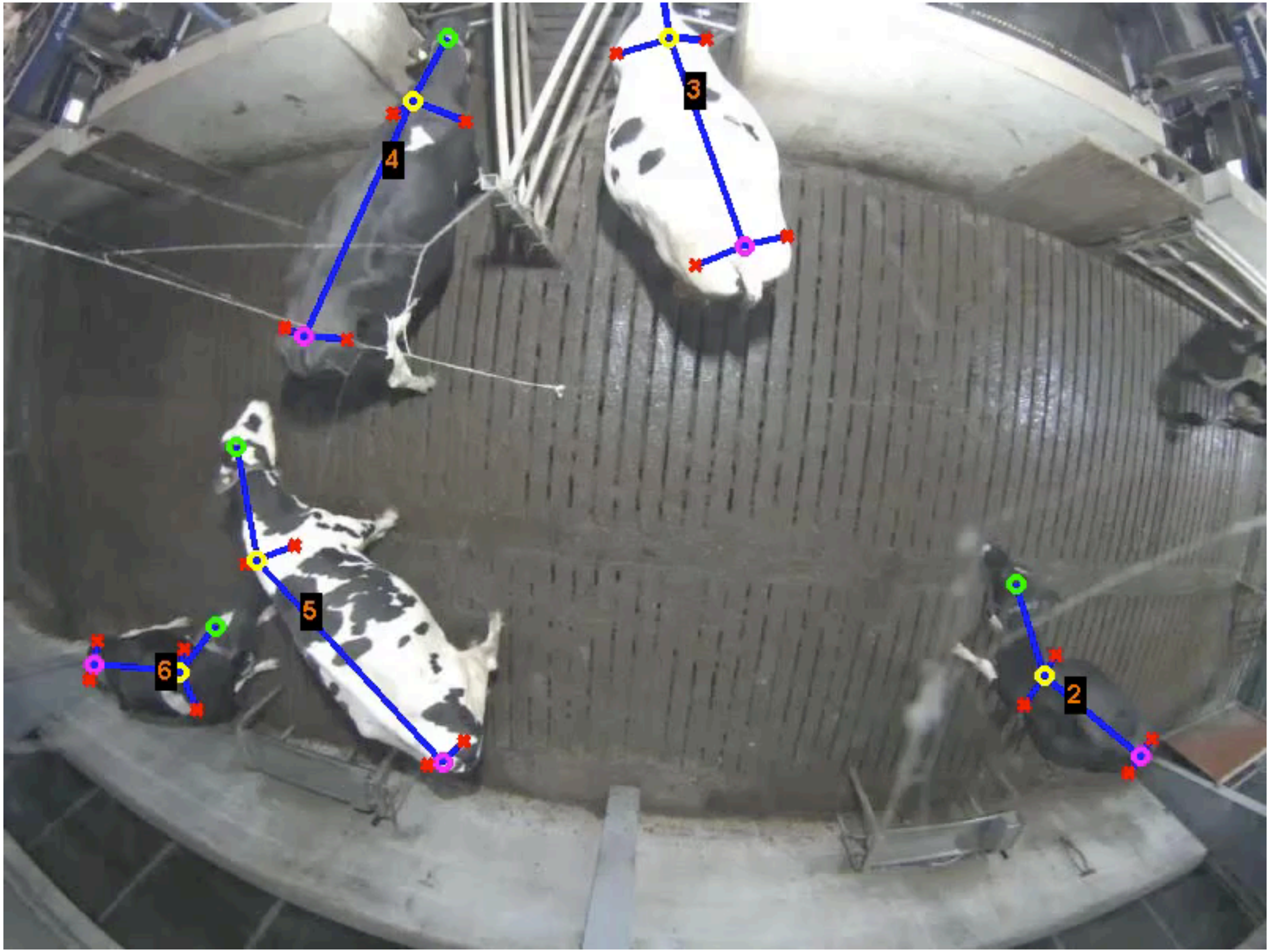
SVM – Pedestrian detection



Overview – Deep Learning

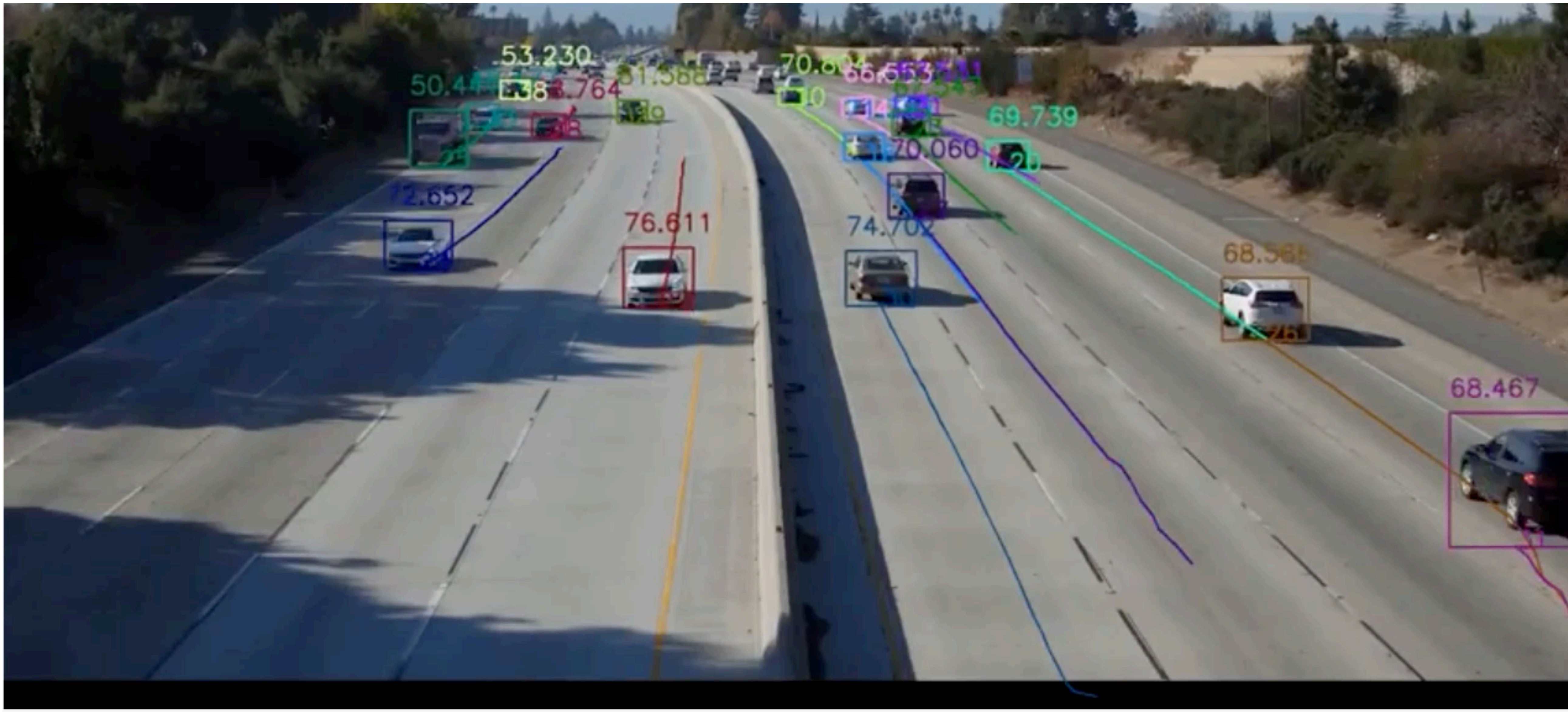
1. History and Motivation
2. Components of Deep Learning
    1. Convolution
    2. Non-linear
    3. Max pooling
    4. Soft-Max
3. Network Design
4. Training
5. Examples

Example: Precision Livestock Farming



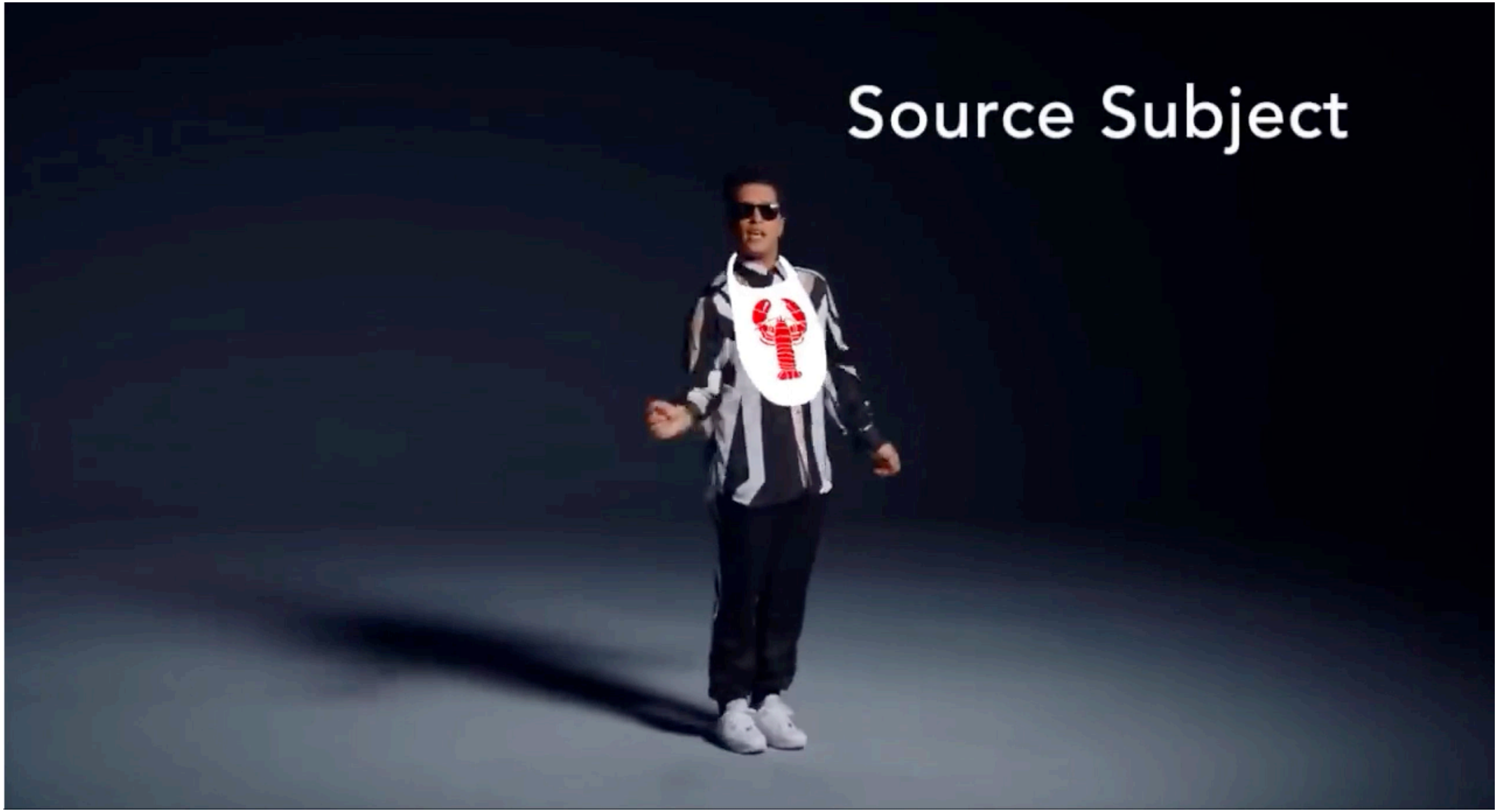
Example: Semantic segmentation

Example: Tracking and measuring speed



Demo of vehicle tracking and speed estimation for the AI City Challenge Workshop at CVPR 2018

Example: Transfer of motion



ImageNet Challenge 2012

Task 1: Classification

Car

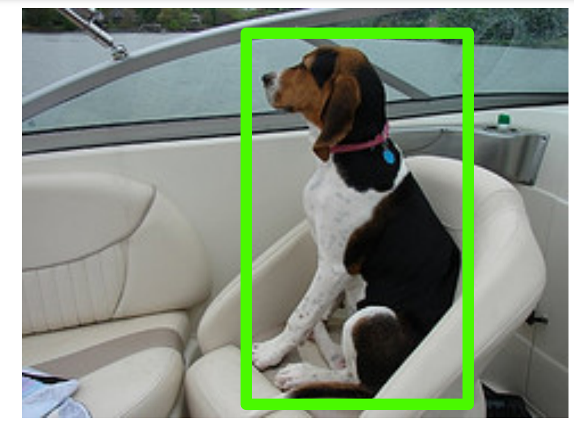
- Predict a class label
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 50% of training.

Task 2: Detection (Classification + Localization)

classification Car

- Predict a class label and a bounding box
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 40% of training.

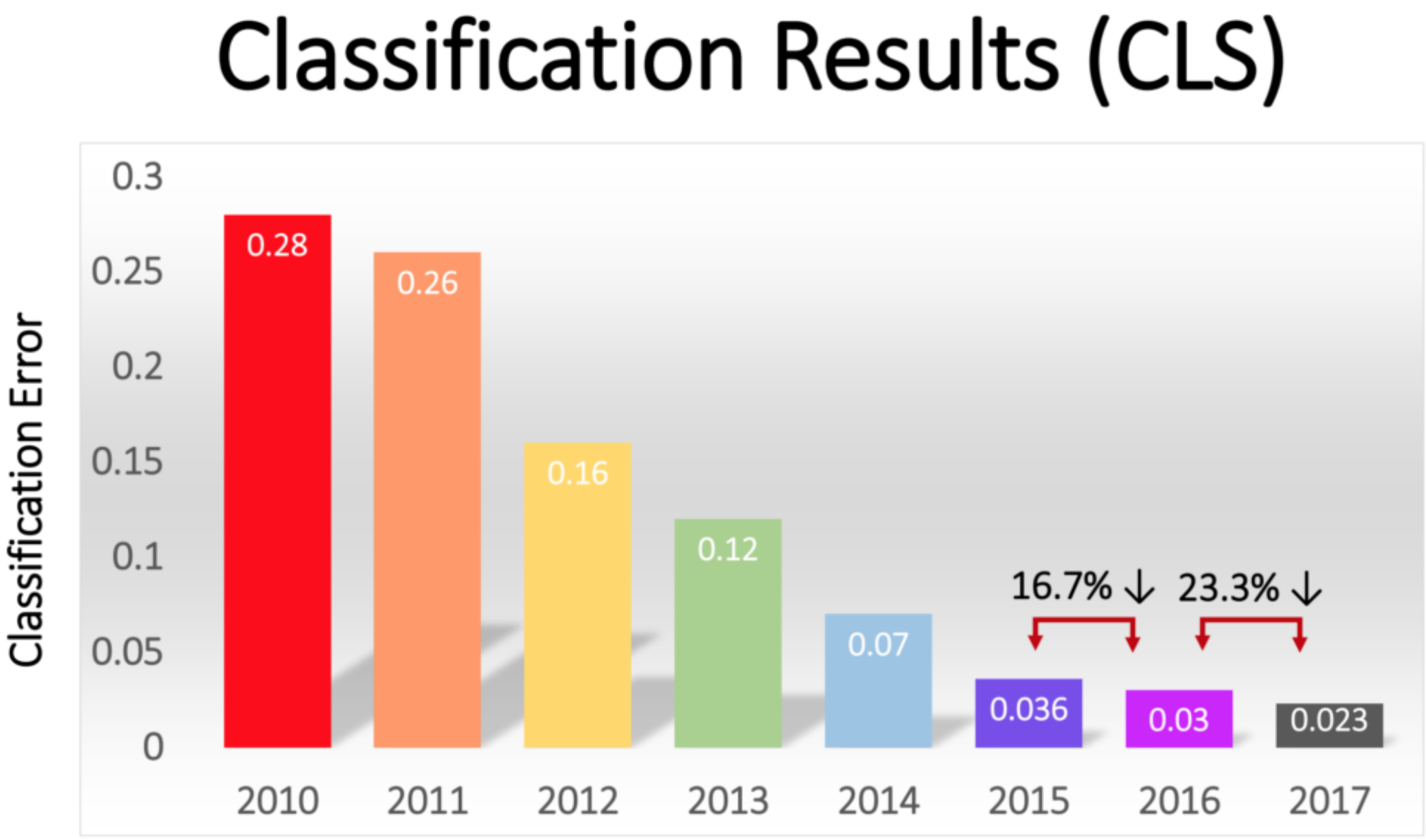
Task 3: Fine-grained classification



classification Walker hound

- Predict a class label given a bounding box in test
- 1 prediction / image
- 120 dog classes (subset)
- ~200 images per class for training (subset)
- Bounding boxes for 100% of training

AlexNet on Imagenet classification challenge 2012



AlexNet on Imagenet classification challenge 2012

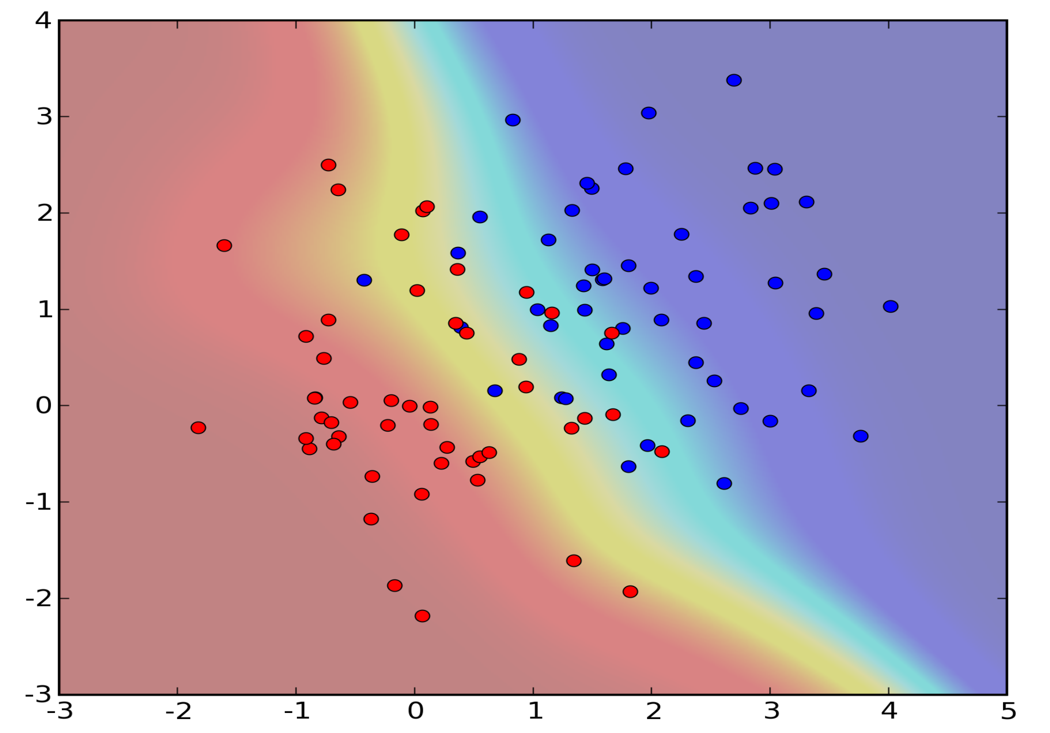
Why does it suddenly work better?

- Larger models
- Requires larger training sets
- Requires computational power - GPUs

Deep learning
Convolutional Neural Networks

- Slides and material from
- http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf
- MatConvNet
- http://www.robots.ox.ac.uk/~vgg/practicals/cnn/
- Gabrielle Flood’s master’s thesis
- Anna Gummeson’s master’s thesis

Components for deep learning



- One neuron
  - Example: Logistic regression
  - Classification model (x feature vector, (w,b) parameters, s smooth thresholding

$$x \in R^d, w \in R^d, b \in R, f(x) = s(w^T x + b)$$

  - Logistic regression

$$s(z) = \frac{1}{1 + e^{-z}}$$

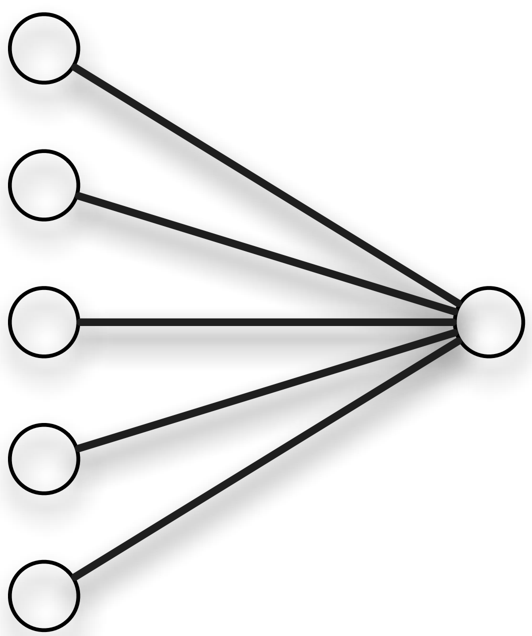
  - ML estimate of parameters (w,b) is a convex optimization problem

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\log(1 + e^{-y_i\boldsymbol{w}^T\boldsymbol{x}_i}).$$

Single Layer Neural Networks One Neuron

- One neuron

$$x \in R^d, w \in R^d, b \in R, f(x) = s(w^T x + b)$$



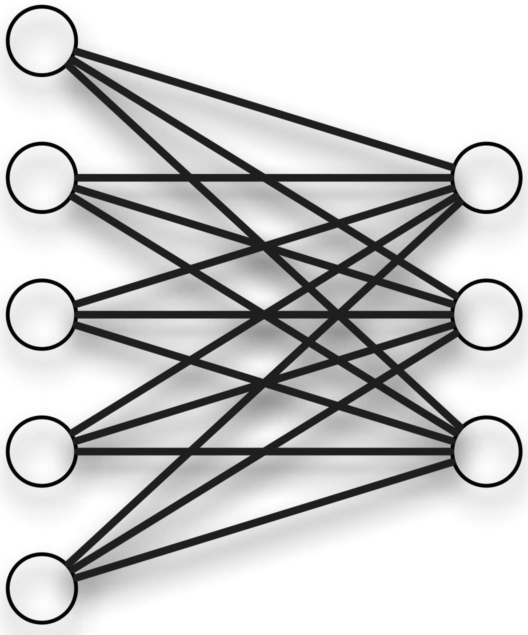
Single Layer Neural Networks Several Neurons

- Several parallell neurons

$$x \in R^d, y \in R^k, B \in R^k, W - k \times d\text{matrix}$$

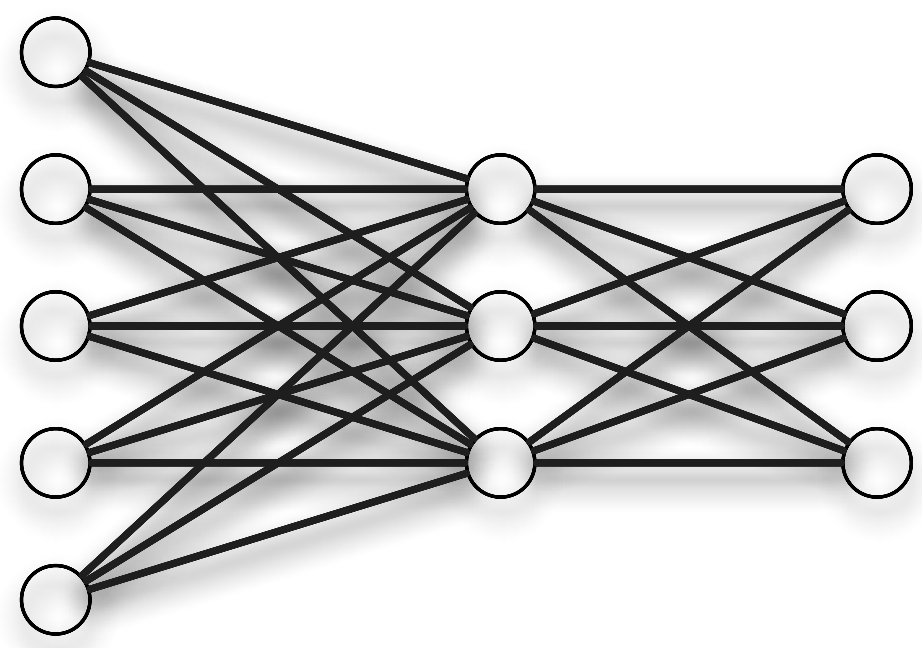
$$y = s(Wx + B)$$

- Elementwise smooth thresholding – s



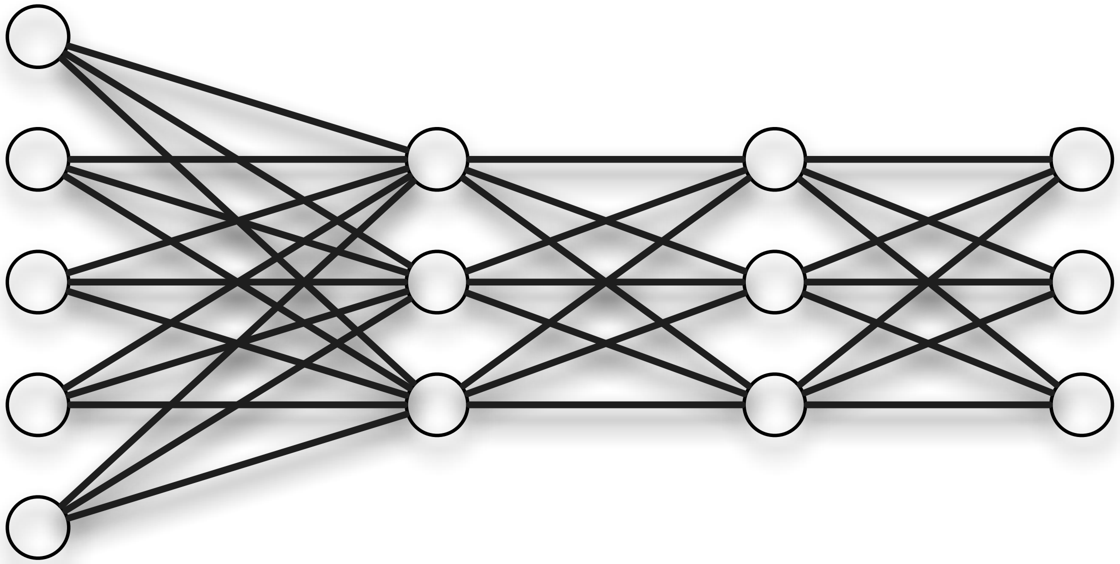
Artificial Neural Networks One hidden layer

- Multi-class classification
- One hidden layer
- Trained by back-propagation
- Popular since the 1990s

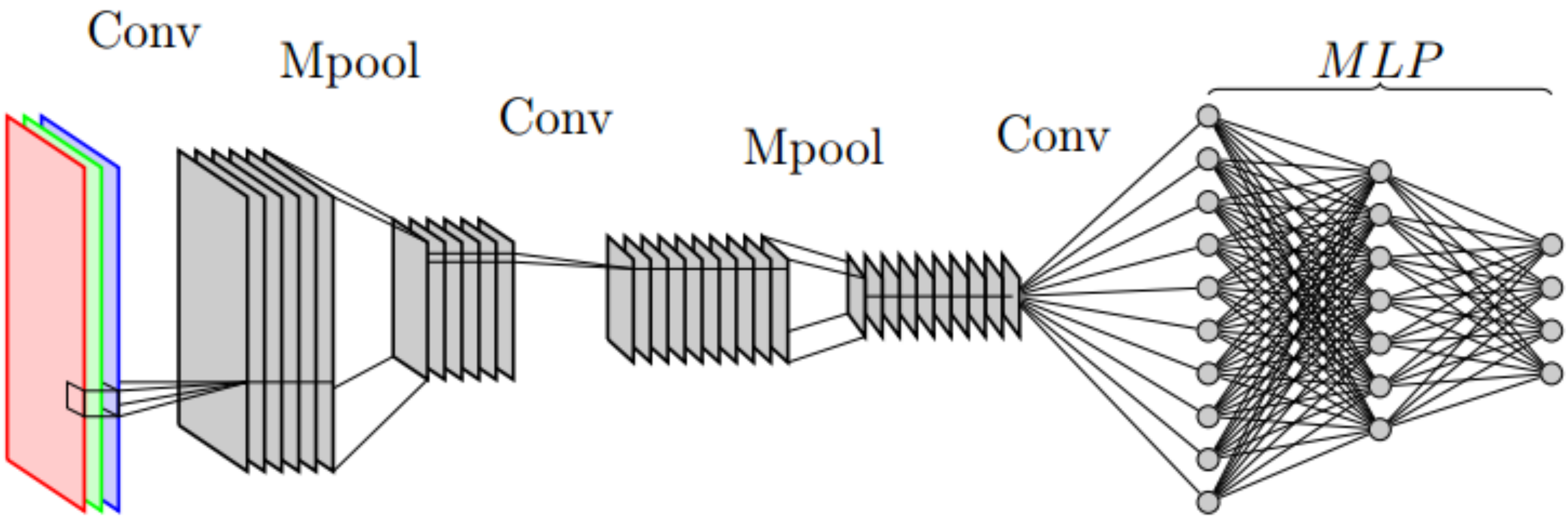


Deep Neural Networks
Many layers

- However
- Naively implemented would give to many parameters
- Example
- 1M pixel image
- 1M hidden layers
- 10^{12} parameters between each pairs of layers

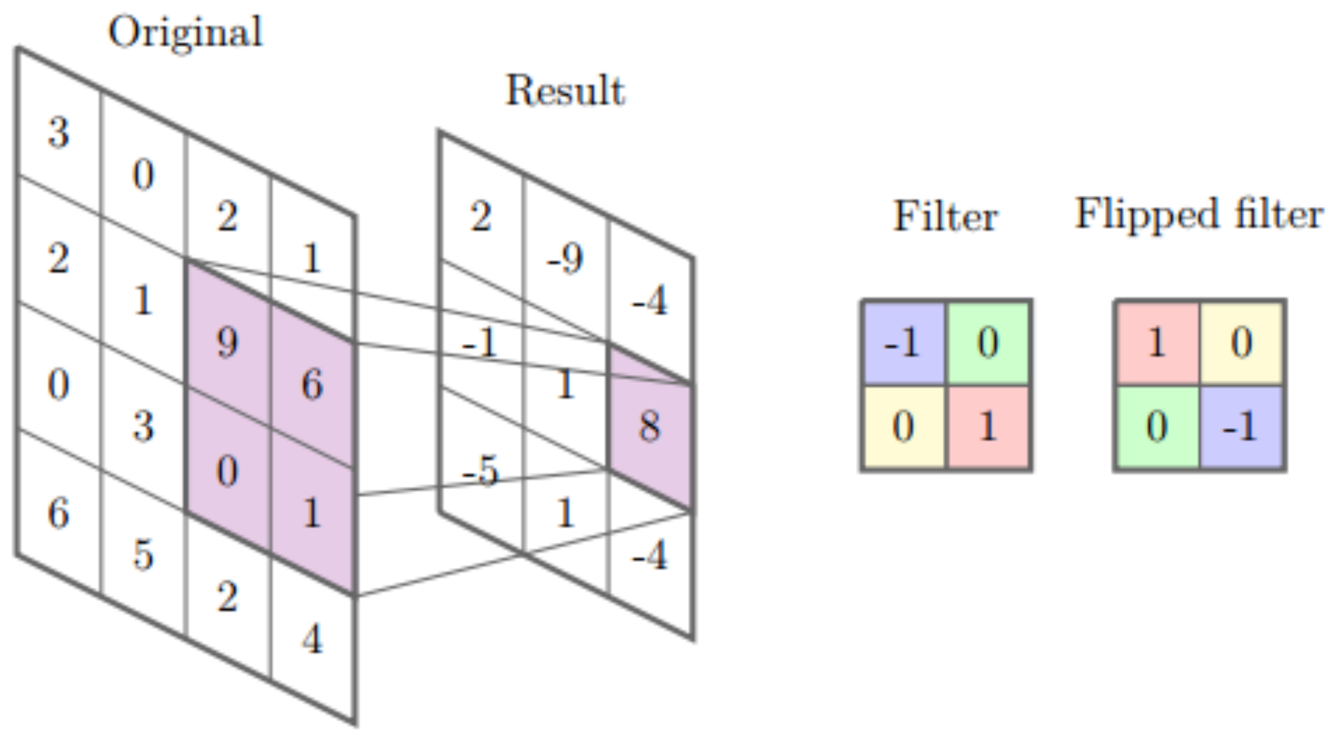


Convolutional neural network, CNN

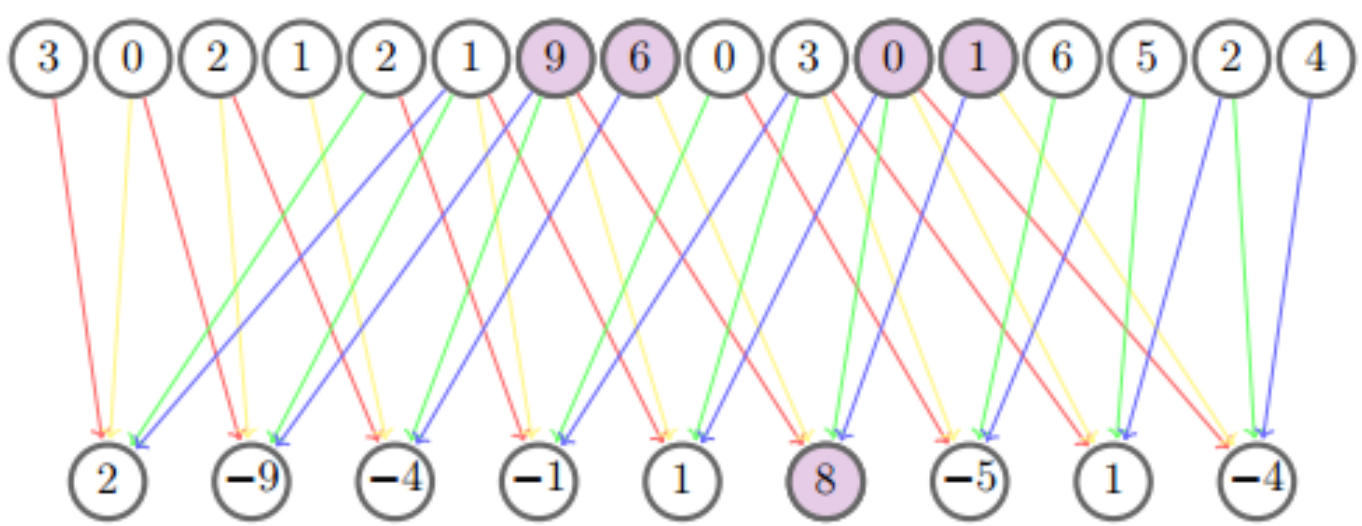


CNN-Blocks - Convolutional layer

Convolution of an image as a filter-operation.



Convolution of an image represented as a sparsely connected ANN.



CNN-Blocks - Convolutional layer

- Input: Data block x of size

$$m \times n \times k_1$$

- Output: Data block y of size

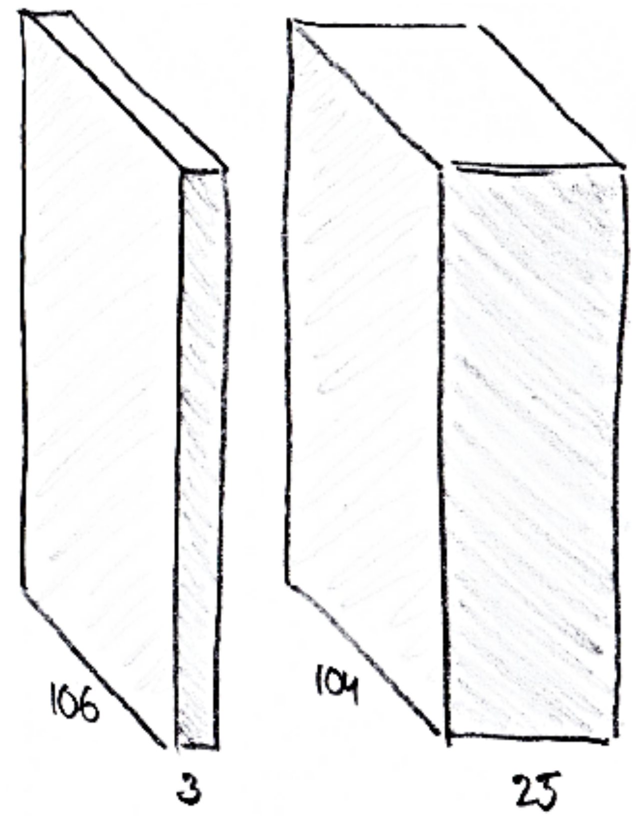
$$m \times n \times k_2$$

- Filter: Filter kernel block w of size

$$m_w \times n_w \times k_1 \times k_2$$

- Offsets: Vector w_o of length

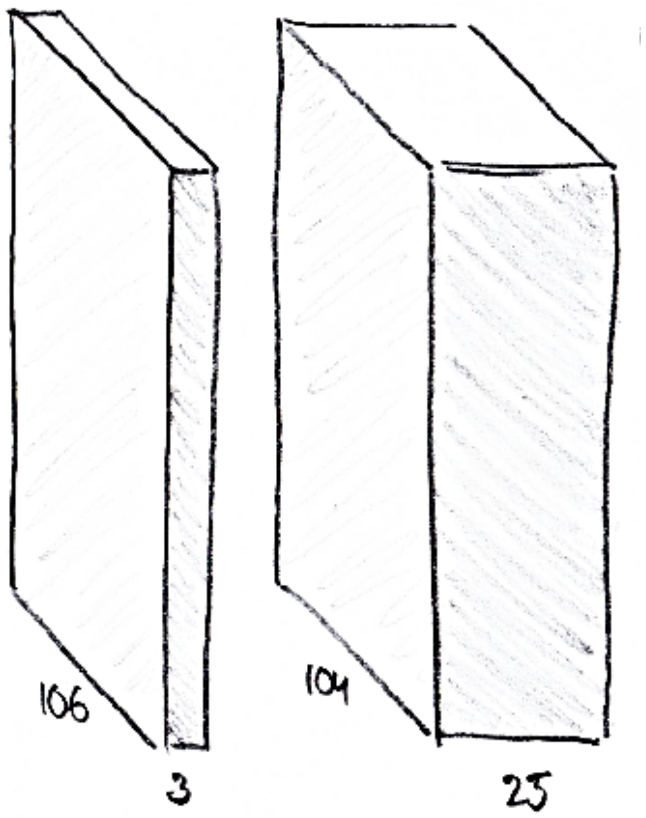
$$k_2$$



$$y(i,j,k) = w_o(k) + \sum_u \sum_v \sum_l x(i-u, j-v, l) w(u,v,l,k)$$

CNN-Blocks - Convolutional layer

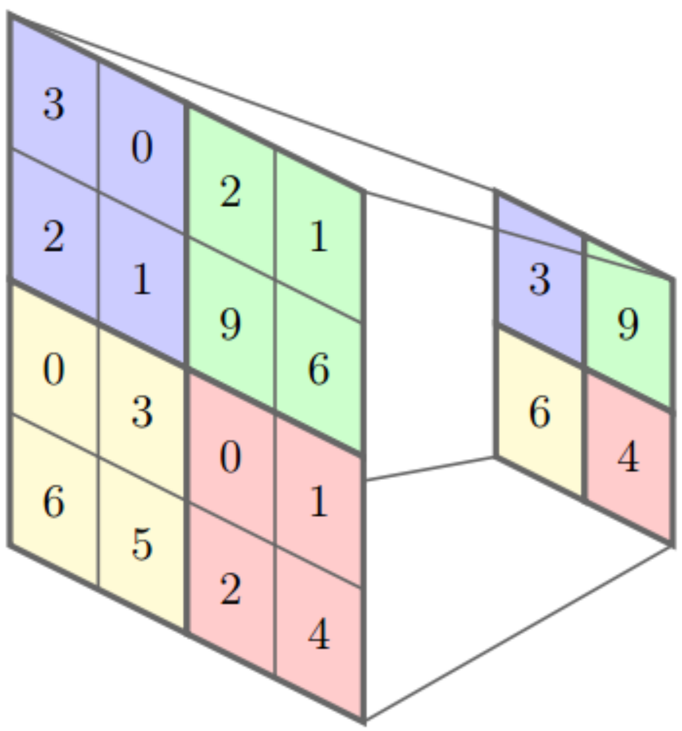
$$y(i,j) = \sum_u \sum_v x(i-u, j-v) w(u,v)$$

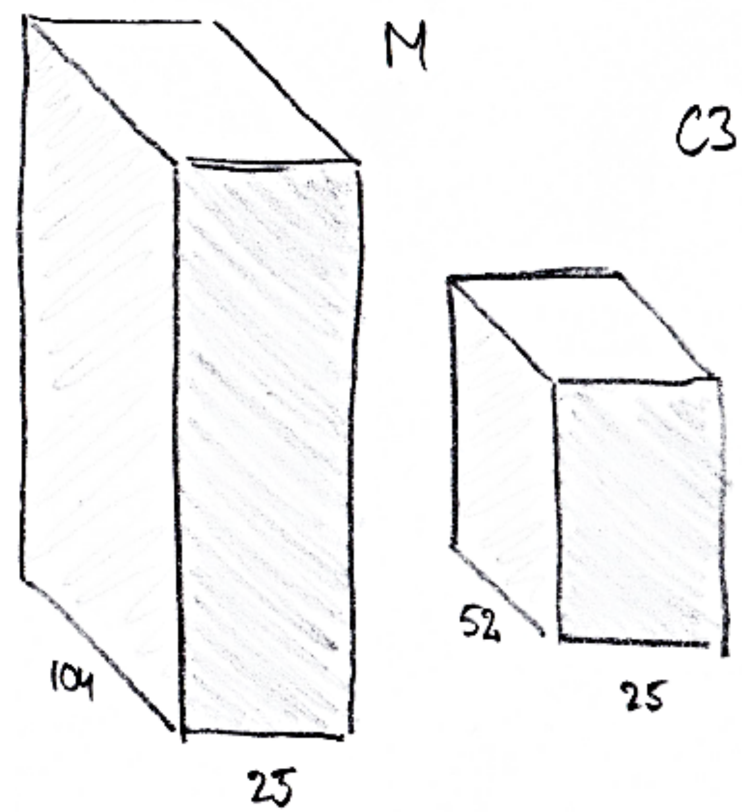


$$y(i,j) = w_o + \sum_l \left(\sum_u \sum_v x(i-u, j-v, l) w(u,v,l) \right)$$

$$y(i,j,k) = w_o(k) + \sum_u \sum_v \sum_l x(i-u, j-v, l) w(u,v,l,k)$$

CNN-Blocks - Max-pooling

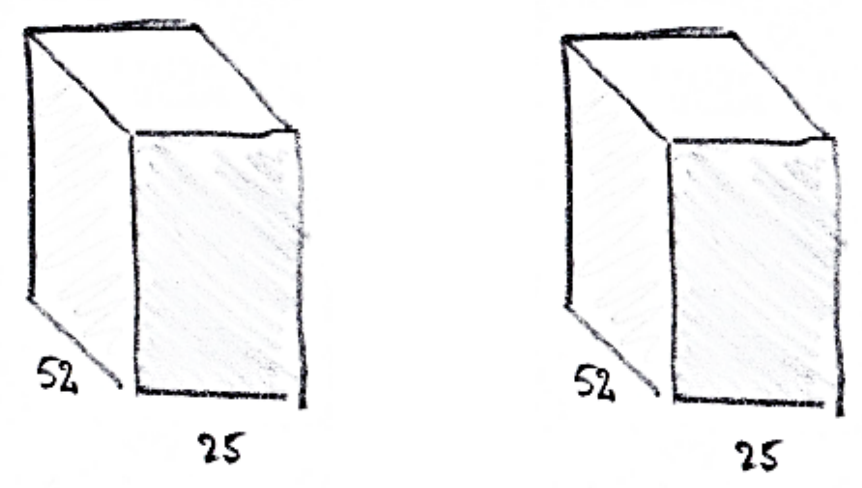




CNN-Blocks - RELU

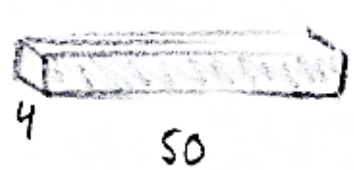
$$f(x) = max(0, x)$$

$$y(i, j, k) = max(x(i, j, k), 0)$$

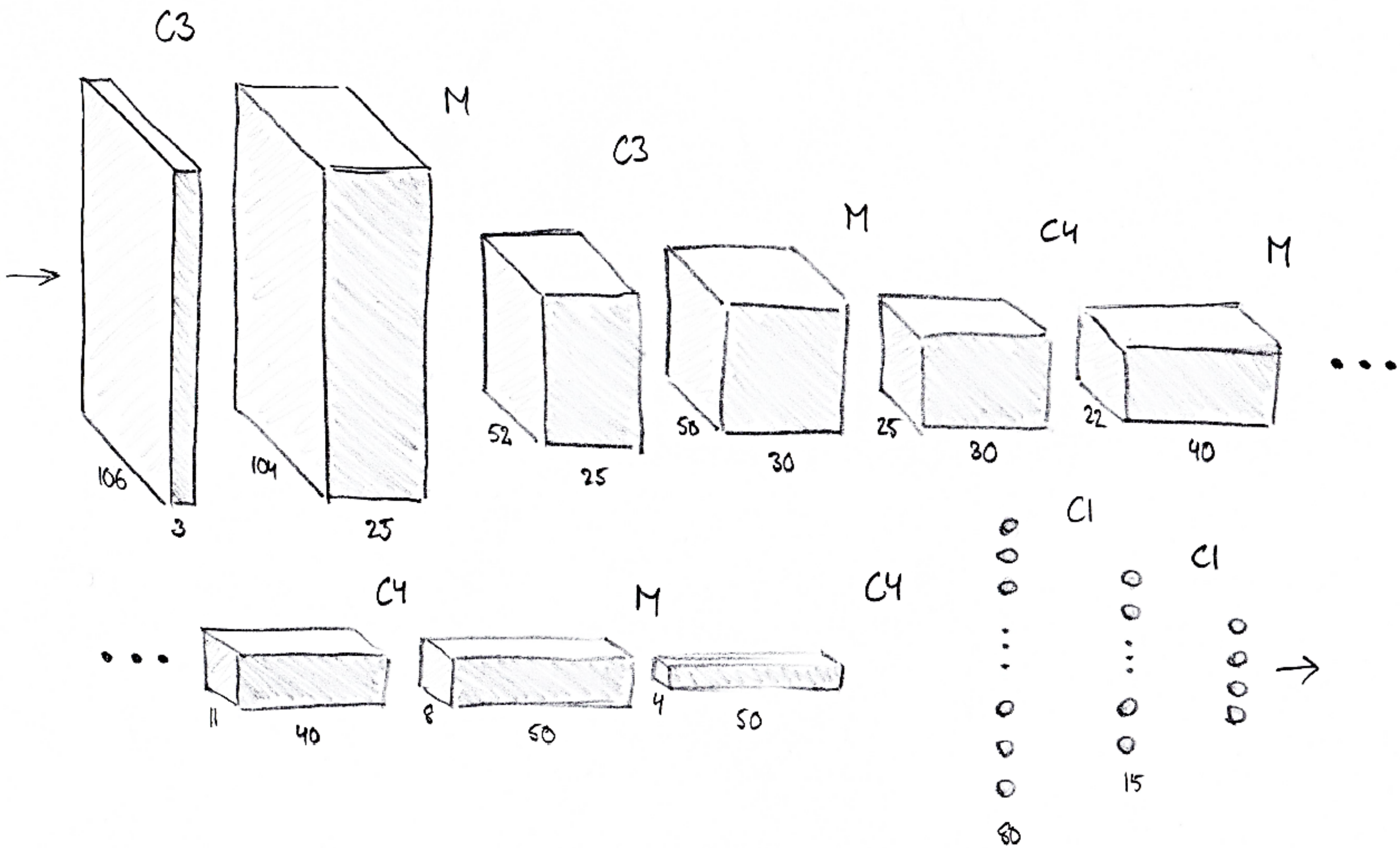


CNN-Blocks – Softmax (convert from ’log probabilites’ d_j to ’probabilites’ that sum to 1)

$$p_j = \frac{e^{d_j}}{\sum_{k=1}^{m} e^{d_k}}$$



Result, Network design



CNN-Blocks – input – output (Kalle: Run demo in browser)

$$y = f(x, w)$$

Input: image x of size m x n x k, typically k=1 (gray-scale) or k=3 (colour)
Output: vector y of size 1 x 1 x N, which we interpret as N probabilities y_j
The probability that the image x is of class j

Training data (x_i, c_i)

$$y = f(x, w)$$

Input: image x of size m x n x k, typically k=1 (gray-scale) or k=3 (colour)
Output: vector y of size 1 x 1 x N, which we interpret as N probabilities y_j
The probability that the image x is of class j

$$y_{c_i}$$

$$-\log y_{c_i}$$

$$T = \{(x_1, c_1), \ldots (x_N, c_N)\}$$

Training data

- Classification network $$y = f(x, w)$$

- Evaluate one example (x_k, c_k) (like adding another layer)

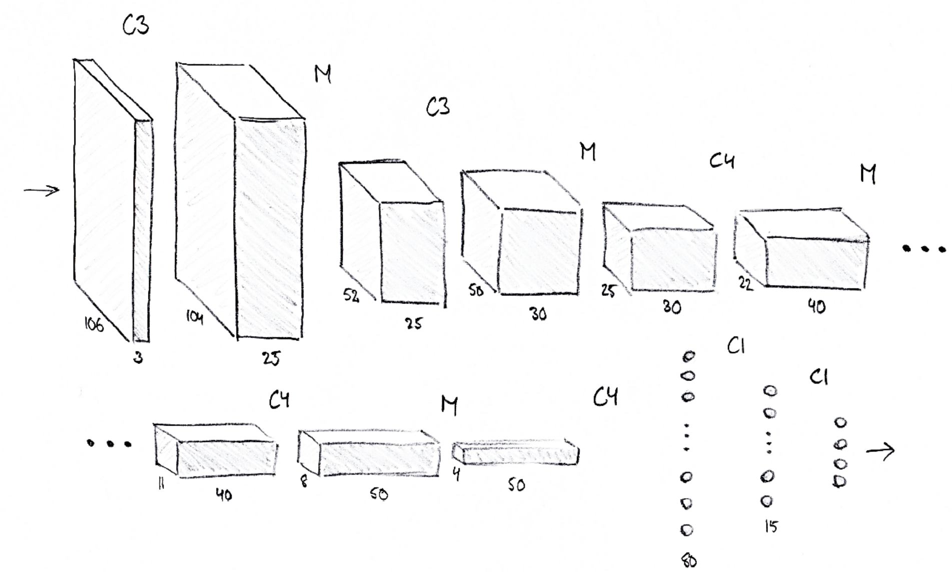
$$\sum_{k=1}^{N} -\log y(x_k, w)_{c_k}$$

- Evaluation function: $$g(T, w) = \sum_{k=1}^{N} -\log y(x_k, w)_{c_k}$$

- Solve

$$\min_w g(T, w)$$

Example: OCR, classify images as a-z, Network design

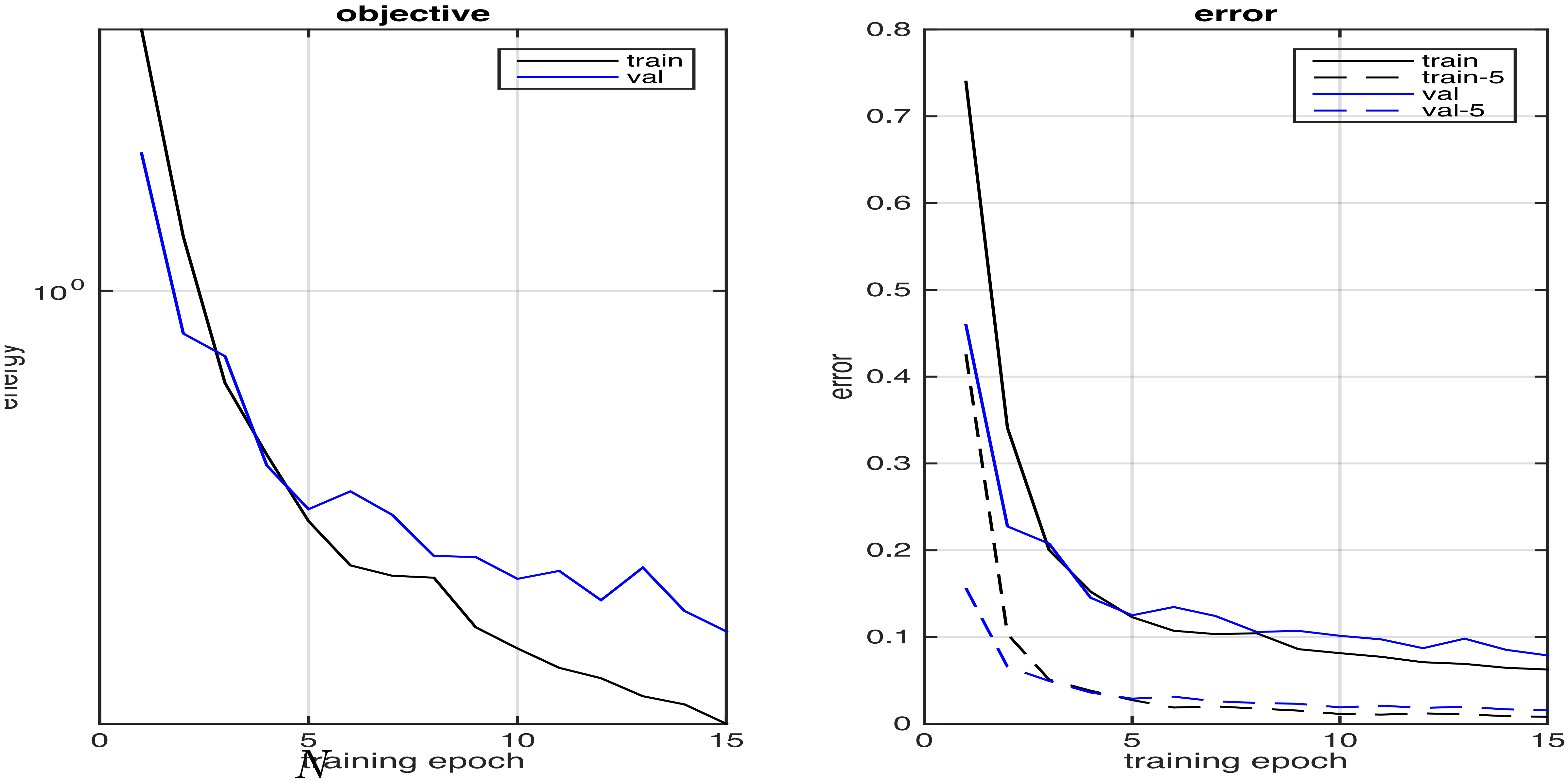



```
>> net

net =

       layers: {1x7 cell}
    imageMean: 0.9176775

>> vl_simplenn_display(net)
      layer|     1|     2|     3|     4|     5|     6|     7|
       type|   cnv| mpool|   cnv| mpool|   cnv|  relu|   cnv|
    support|   5x5|   2x2|   5x5|   2x2|   4x4|   1x1|   2x2|
     stride|     1|     2|     1|     2|     1|     1|     1|
        pad|     0|     0|     0|     0|     0|     0|     0|
    out dim|    20|    20|    50|    50|   500|   500|    26|
   filt dim|     1|   n/a|    20|   n/a|    50|   n/a|   500|
rec. field|      5|     6|    14|    16|    28|    28|    32|
c/g net KB|    4/0|   0/0| 196/0|   0/0|3129/0|   0/0| 406/0|
total network CPU/GPU memory: 3.6/0 MB
>>
```

Example: OCR, classify images as a-z

Training data

Example: OCR, classify images as a-z, Training



$$g(T, w) = \sum_{k=1}^{N} -\log y(x_k, w)_{c_k}$$

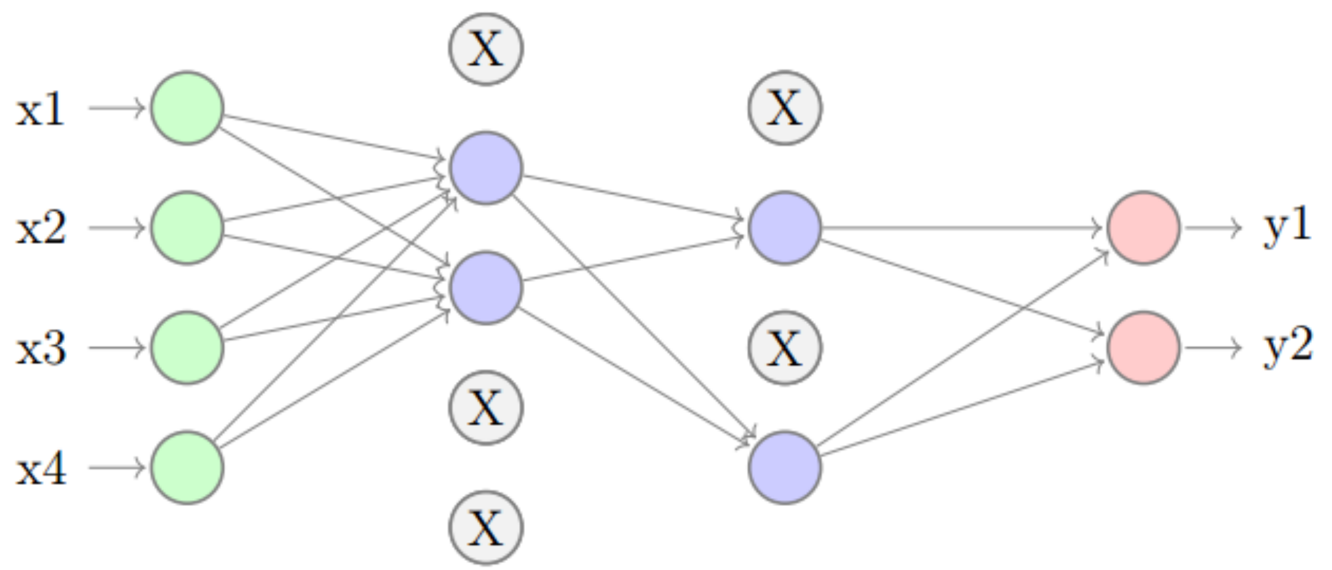
$$\#\{c_k = argmax_i \, y(x_k, w)_i\}$$

Tricks

- **Stochastic Gradient Descent**
  - Computation of $$g(T,w) = \sum_{k=1}^{N} -\log y(x_k,w)_{c_k}$$
  - Requires going through all examples (all N).
  - If N is large and/or if computing $y(x_k,w)$ is time-consuming, use stochastic gradient descent, i.e. update parameters using subsets of training data.
- **Jittering** - construct a larger training set by perturbing the examples, jittering, translating images, rotating images, warping, mirroring, adding noise, …'
- **Dropout** – in each computation of $y(x_k,w)$ let a random subset of the neurons die, i e set the output to zero.

Generalisation, Expand data set

Generalisation, Dropout



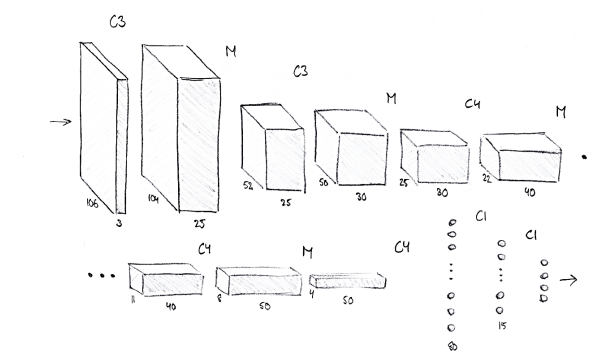
Generalisation, Weight decay (Prior on small weights)

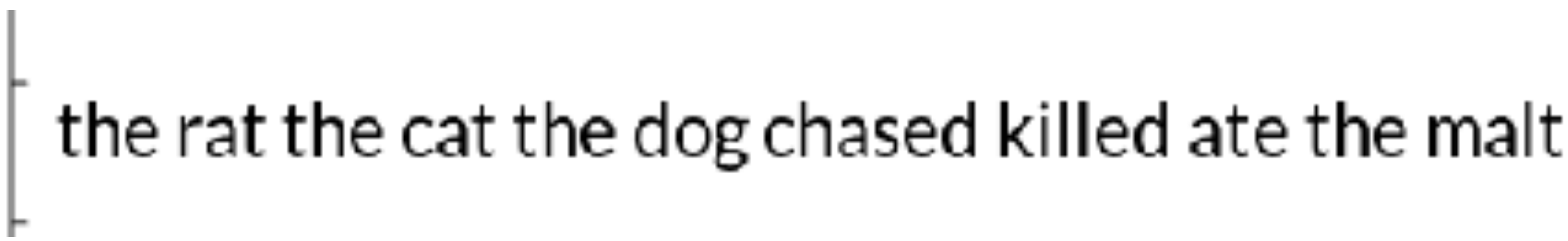
$$E(w) = -\sum_{n=1}^{N} \log\left(\frac{e^{y_{d(n)}(n)}}{\sum_{i=1}^{k} e^{y_i(n)}}\right) + \frac{\lambda}{2}\sum_{l} w_l^2$$

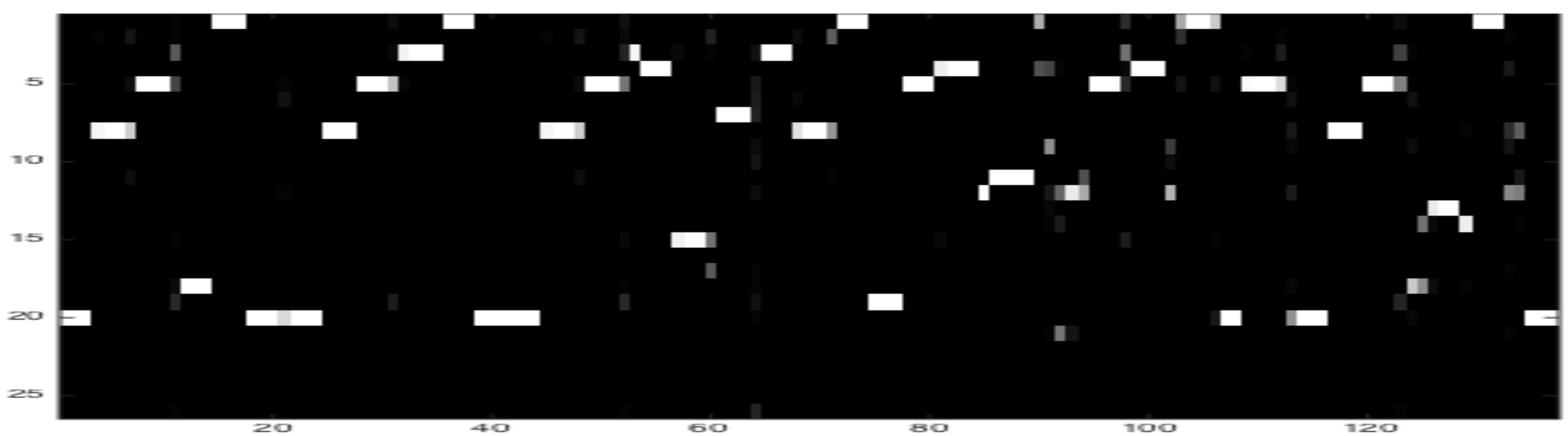
Thoughts

- Modelling. It takes time to
  - Figure out an appropriate network structure
  - Gather data and ground truth
- The optimization does not always work
  - Parameters explode
  - Nothing happens
- Visualization of the features is important for understanding.
- Feedback in networks

Example: OCR, classify images as a-z, Results

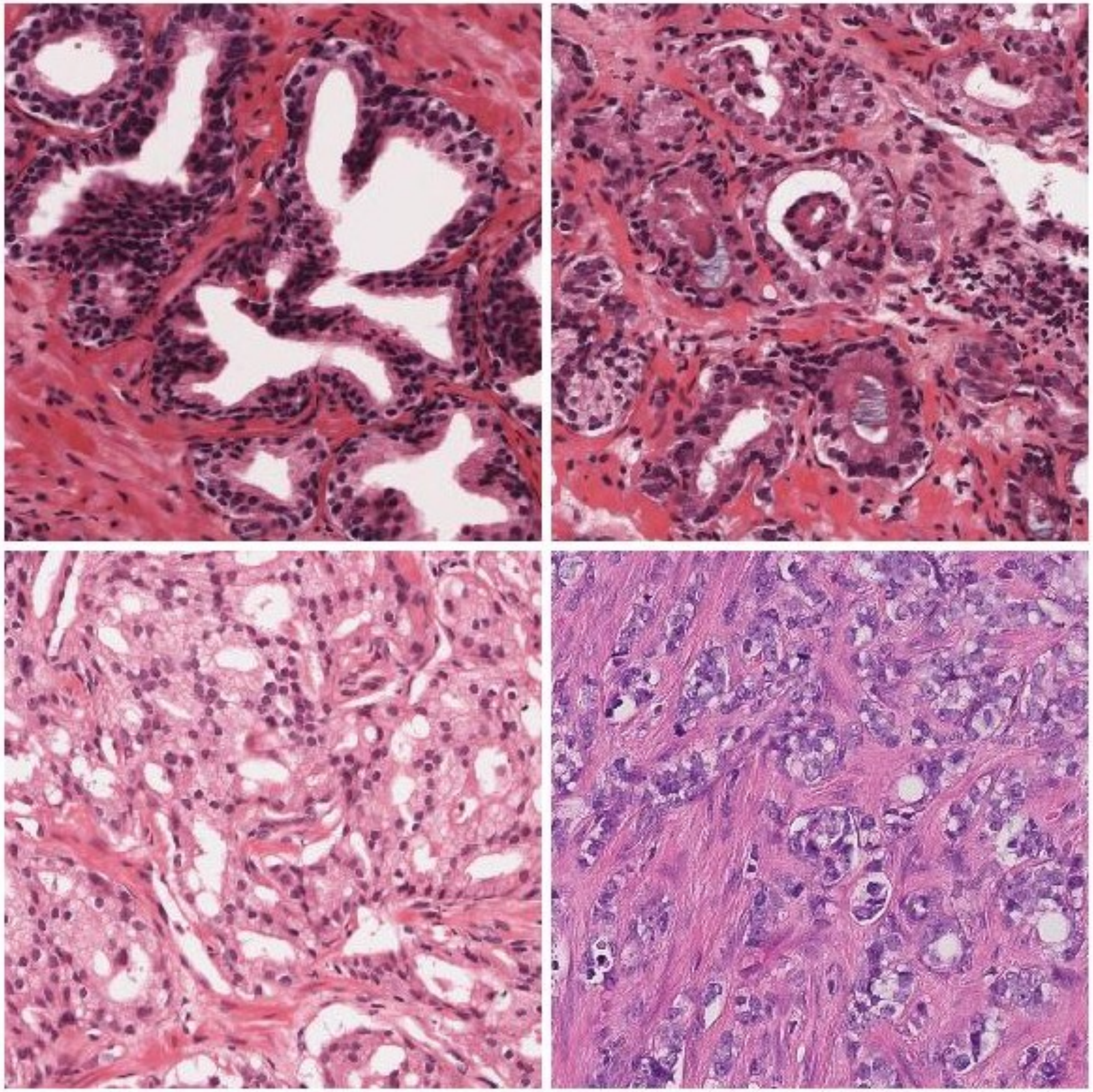




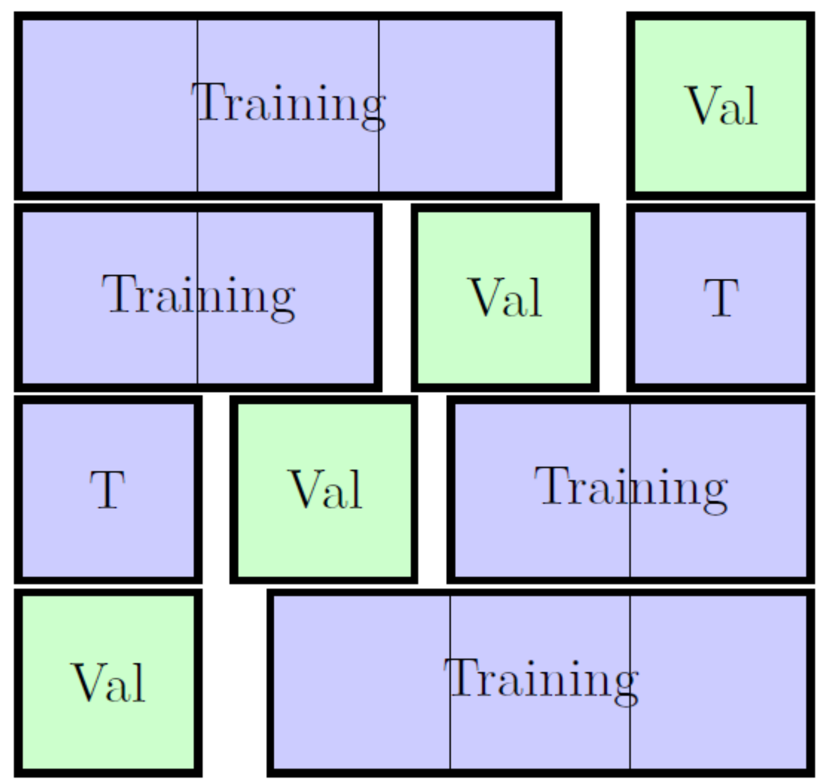


tttthhhheeecrrraaatttttthhheeeecccccaaat
tttttthhhheeeecdddooooggg
Zccchhhhaaassseeedddlkkkaiulleeecdddl
aaaatteeeetttthhheeeerrmmmnaaallttt

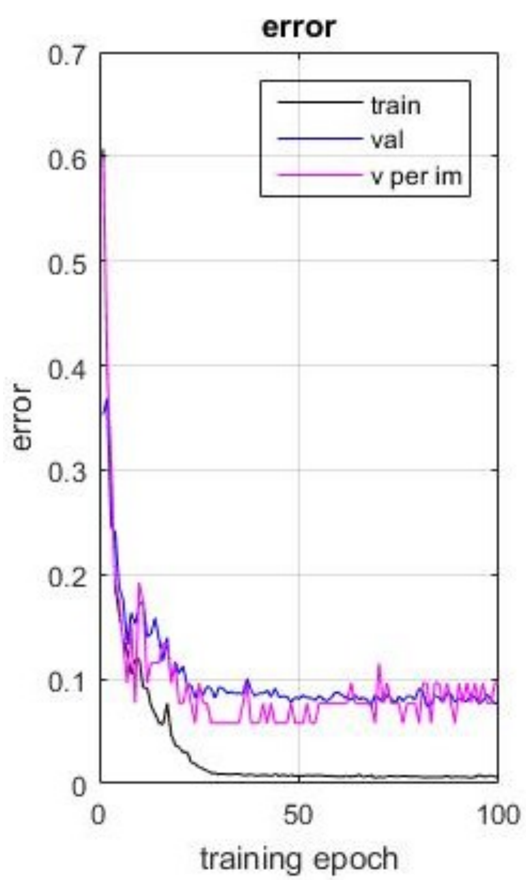
Example: Prostate cancer Data

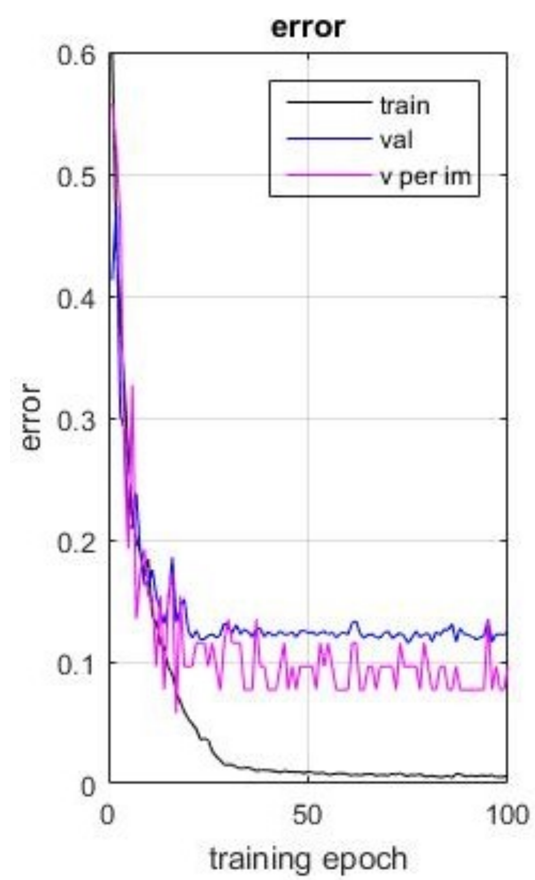


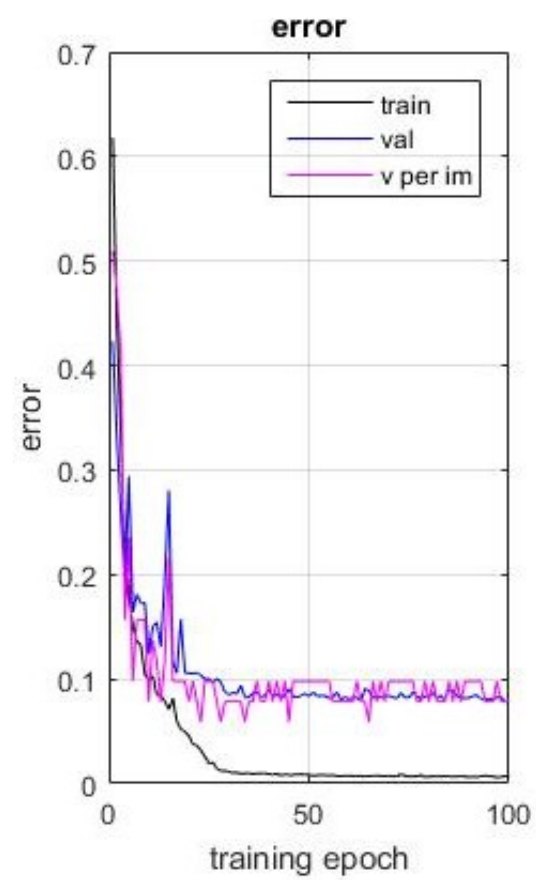
Result, Cross-validation

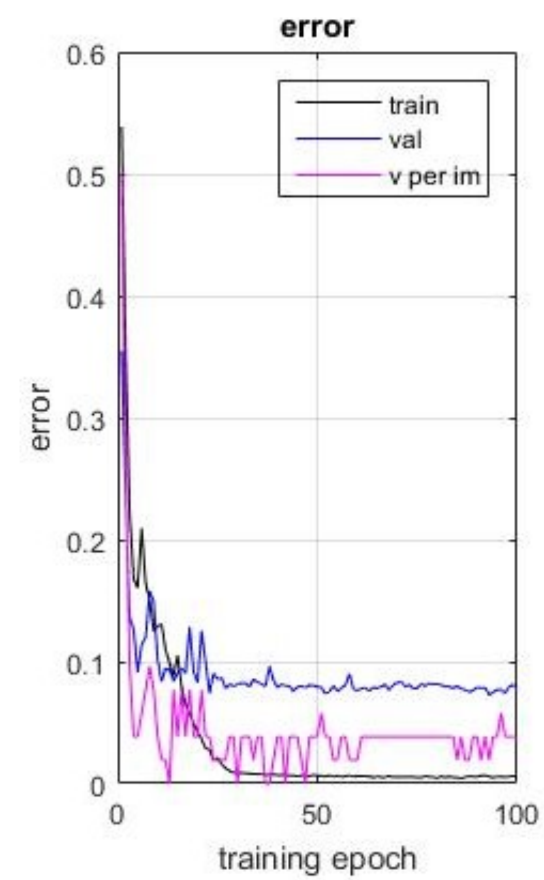


Example: Prostate cancer Training





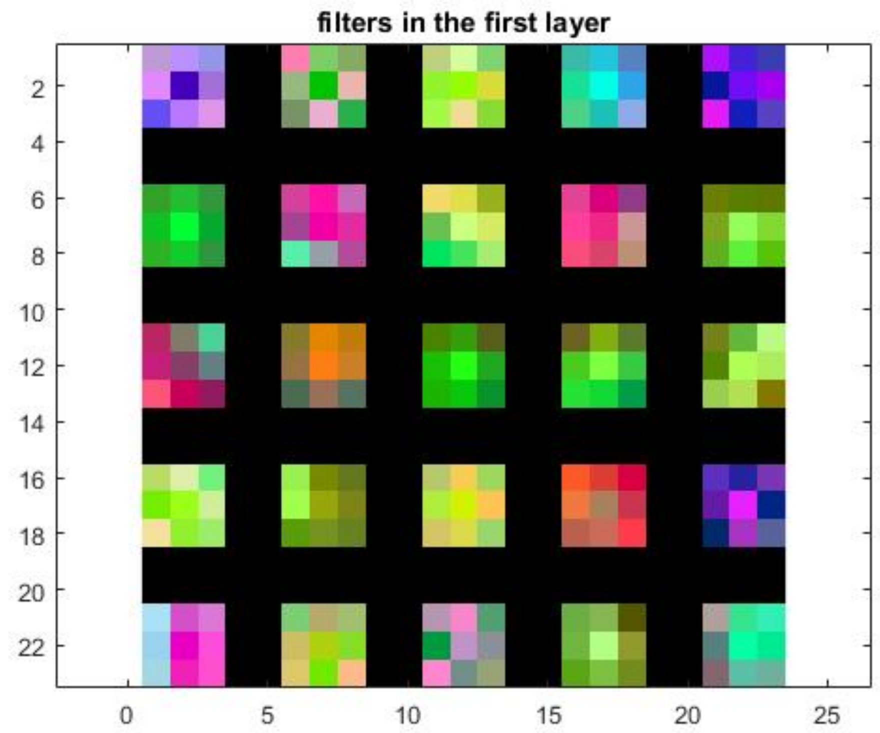


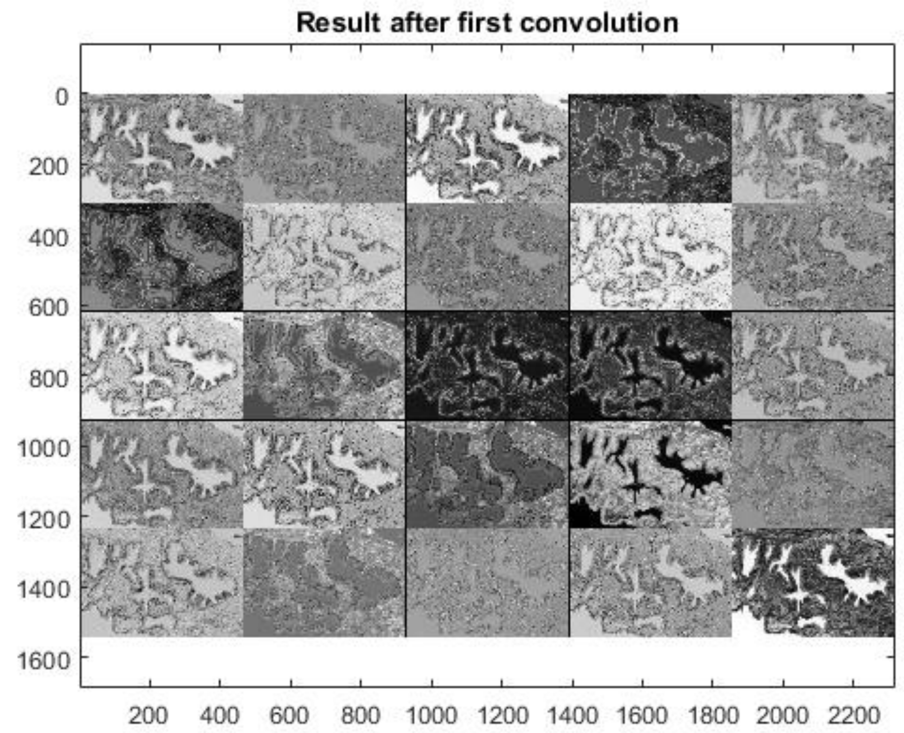


Example: Prostate cancer
Results: Confusion matrix

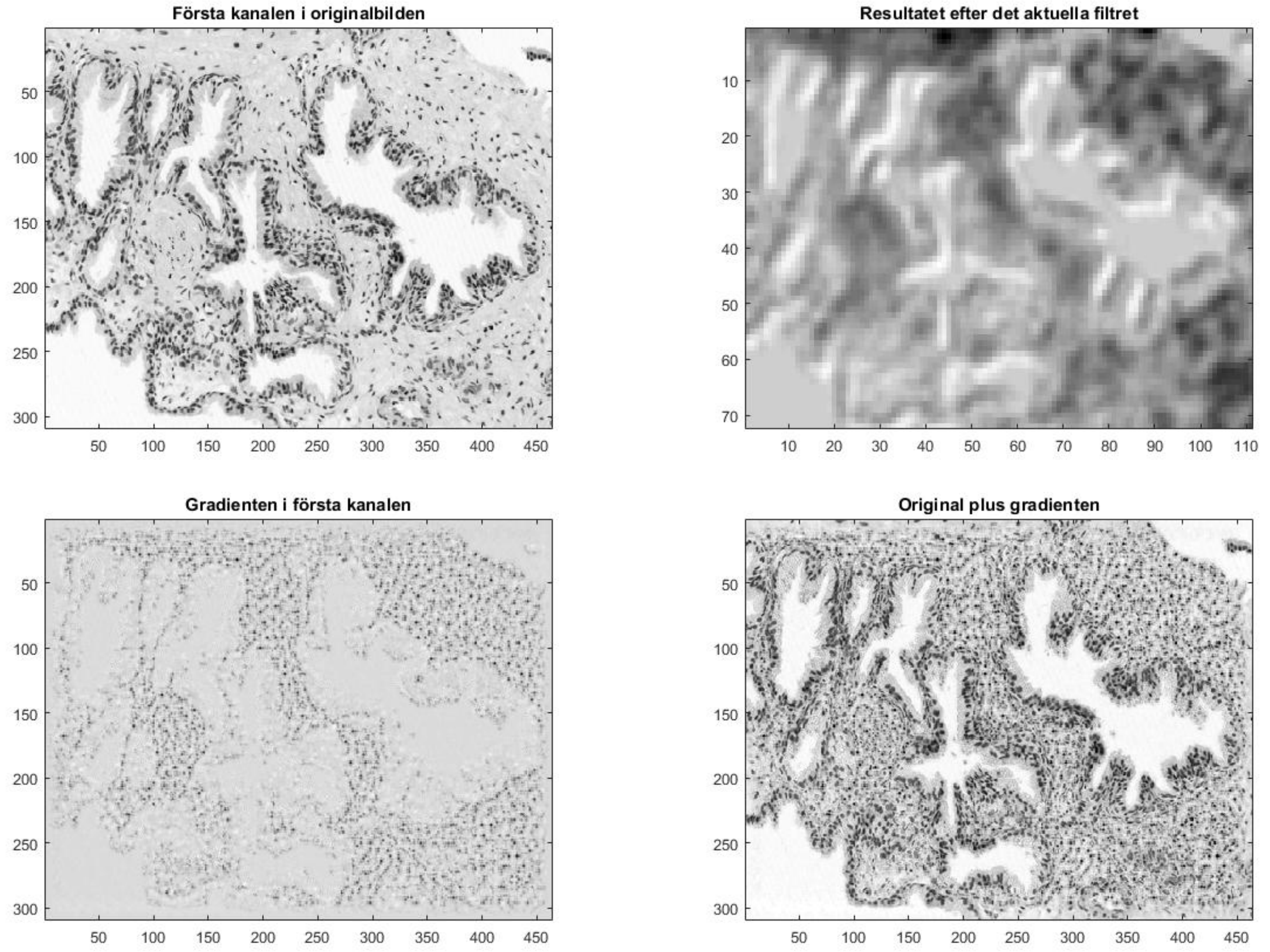
$$\begin{bmatrix} 51 & 0 & 0 & 0 \\ 3 & 46 & 3 & 1 \\ 0 & 6 & 43 & 0 \\ 0 & 3 & 0 & 52 \end{bmatrix}$$

Visualisation





Visualisation



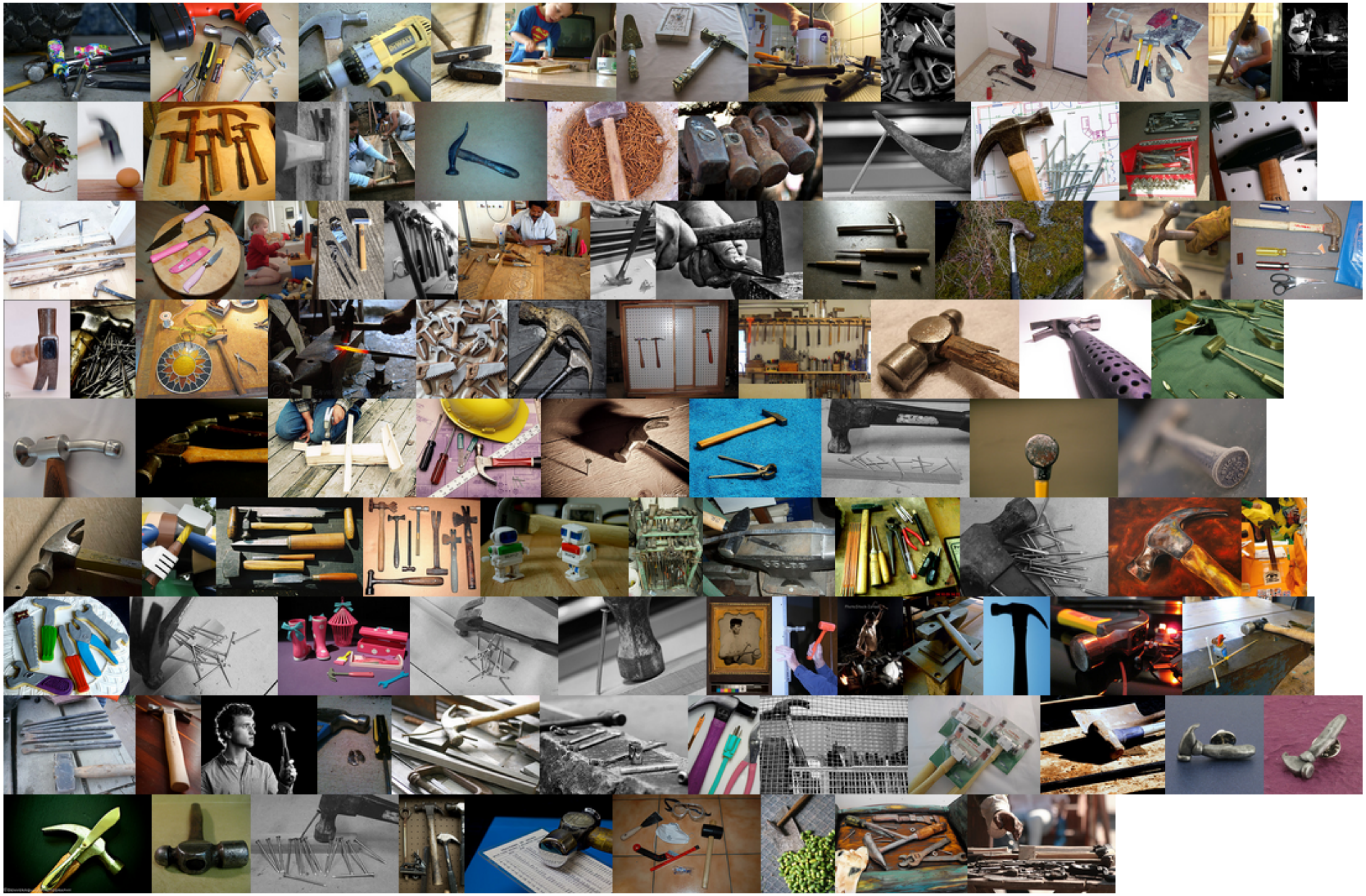
Examples: Image net, Data

- ImageNet Large Scale Visual Recognition Challenge
- Yearly challenge since 2010
- 2011 - 25% error
- 2012 - 16% error (using CNN). This kicked off the deep learning hype
- 2015 – 4% error
- By 2015, researchers reported that current software exceeds human ability at the narrow ILSVCR tasks.
- However, as one of the challenge's organisers, Olga Russakovsky, pointed out in 2015, the programs only have to identify images as belonging to one of a thousand categories!

Examples: Image net, Data

- ImageNet Large Scale Visual Recognition Challenge
- Listen to Fei Fei Li's TED talk
- https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures
- 1000 classes
- 1200 images per class
- Subset of imagenet, builds on wordnet structure

Test images for “Hammer”



Ladles are hard

Chimes are hard

ImageNet Challenge 2012

Task 1: Classification

Car

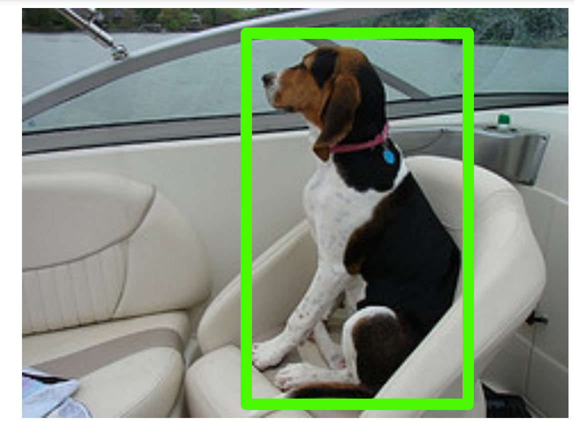
- Predict a class label
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 50% of training.

Task 2: Detection (Classification + Localization)

classification Car

- Predict a class label and a bounding box
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 40% of training.

Task 3: Fine-grained classification



classification Walker hound

- Predict a class label given a bounding box in test
- 1 prediction / image
- 120 dog classes (subset)
- ~200 images per class for training (subset)
- Bounding boxes for 100% of training

Examples: Image net, Network design
37 layers – classify 1000 image categories

layer	1	2	3	4	5	6	7	8	9	10	11	12
type	cnv	relu	cnv	relu	mpool	cnv	relu	cnv	relu	mpool	cnv	relu
support	3x3	1x1	3x3	1x1	2x2	3x3	1x1	3x3	1x1	2x2	3x3	1x1
stride	1	1	1	1	2	1	1	1	1	2	1	1
pad	1	0	1	0	0	1	0	1	0	0	1	0
out dim	64	64	64	64	64	128	128	128	128	128	256	256
filt dim	3	n/a	64	n/a	n/a	64	n/a	128	n/a	n/a	128	n/a
rec. field	3	3	5	5	6	10	10	14	14	16	24	24
c/g net KB	7/0	0/0	144/0	0/0	0/0	289/0	0/0	577/0	0/0	0/0	1153/0	0/0

total network CPU/GPU memory: 528.9/0 MB

13	14	15	16	17	18	19	20	21	22	23	24	25
cnv	relu	cnv	relu	mpool	cnv	relu	cnv	relu	cnv	relu	mpool	cnv
3x3	1x1	3x3	1x1	2x2	3x3	1x1	3x3	1x1	3x3	1x1	2x2	3x3
1	1	1	1	2	1	1	1	1	1	1	2	1
1	0	1	0	0	1	0	1	0	1	0	0	1
256	256	256	256	256	512	512	512	512	512	512	512	512
256	n/a	256	n/a	n/a	256	n/a	512	n/a	512	n/a	n/a	512
32	32	40	40	44	60	60	76	76	92	92	100	132
2305/0	0/0	2305/0	0/0	0/0	4610/0	0/0	9218/0	0/0	9218/0	0/0	0/0	9218/0

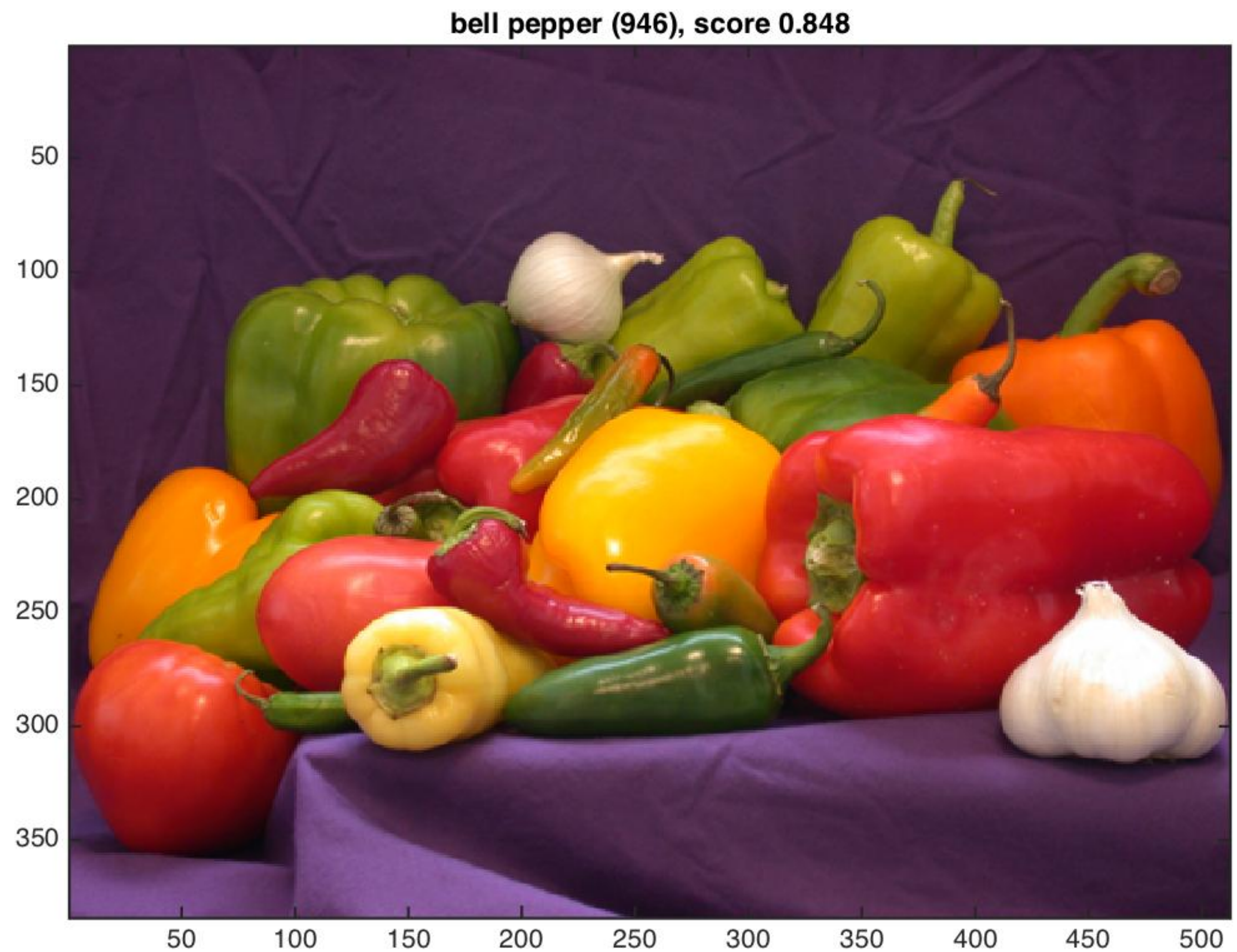
25	26	27	28	29	30	31	32	33	34	35	36	37
cnv	relu	cnv	relu	cnv	relu	mpool	cnv	relu	cnv	relu	cnv	sftm
3x3	1x1	3x3	1x1	3x3	1x1	2x2	7x7	1x1	1x1	1x1	1x1	1x1
1	1	1	1	1	1	2	1	1	1	1	1	1
1	0	1	0	1	0	0	0	0	0	0	0	0
512	512	512	512	512	512	512	4096	4096	4096	4096	1000	1000
512	n/a	512	n/a	512	n/a	n/a	512	n/a	4096	n/a	4096	n/a
132	132	164	164	196	196	212	404	404	404	404	404	404
9218/0	0/0	9218/0	0/0	9218/0	0/0	0/0	401424/0	0/0	65552/0	0/0	16004/0	0/0

Examples, Network design
37 layers – classify 1000 image categories

layer	1	2	3	rel
type	cnv	relu	cnv	rel
support	3x3	1x1	3x3	1x
stride	1	1	1	
pad	1	0	1	
out dim	64	64	64	6
filt dim	3	n/a	64	n/
rec. field	3	3	5	
c/g net KB	7/0	0/0	144/0	0/

total network CPU/GPU memory: 528.9/0 MB

35	36	37
relu	cnv	sftm
1x1	1x1	1x1
1	1	1
0	0	0
4096	1000	1000
n/a	4096	n/a
404	404	404
0/0	16004/0	0/0



Training Deep Learning

- Data
  - Obtain data,
  - cut-outs of right size,
  - jittering,
  - Data expansion (translation, rotation, scaling, mirroring, adding noise, …)
- Data
  - Obtain ground truth
  - How should the problem be coded

Training Deep Learning

- Hyperparameters
  - How many layers
  - Size of convolution kernels
  - Number of channels
  - Order of layers
- Training parameters
  - Initializing weights
  - Momentum
  - …

Non-linear function

- Different choices of non-linear functions.
- Faster learning
- Rectifier …

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

- … currently most popular, faster training
- Arguments

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

$$f(x) = \ln(1 + e^x)$$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

$$f(x) = (1 + e^{-x})^{-1}$$

$$f(x) = tanh(x)$$

Training Deep Learning

- Once all of these are in place, there are several good systems for optimizing the parameters
  - MatConvNet, TensorFlow, Caffe, Torch7, Theano
  - Can train on single CPU
  - Faster if compiled for GPU
  - Even faster on cluster of computers with multiple GPU (e g LUNARC, http://www.lunarc.lu.se)
- More links on home page for PhD course
- http://www.control.lth.se/Education/DoctorateProgram/deep-learning-study-circle.html

Deep learning - summary

- What is deep learning
- Supervised vs unsupervised learning
- Goal function, energy function E
- Choice of non-linearity ReLU
- Optimization Back-propagation, SGD
- Tricks dropout
- Examples from speech and vision
- Software
- References
- To learn more – take course on Machine Learning FMAN45

LUND UNIVERSITY