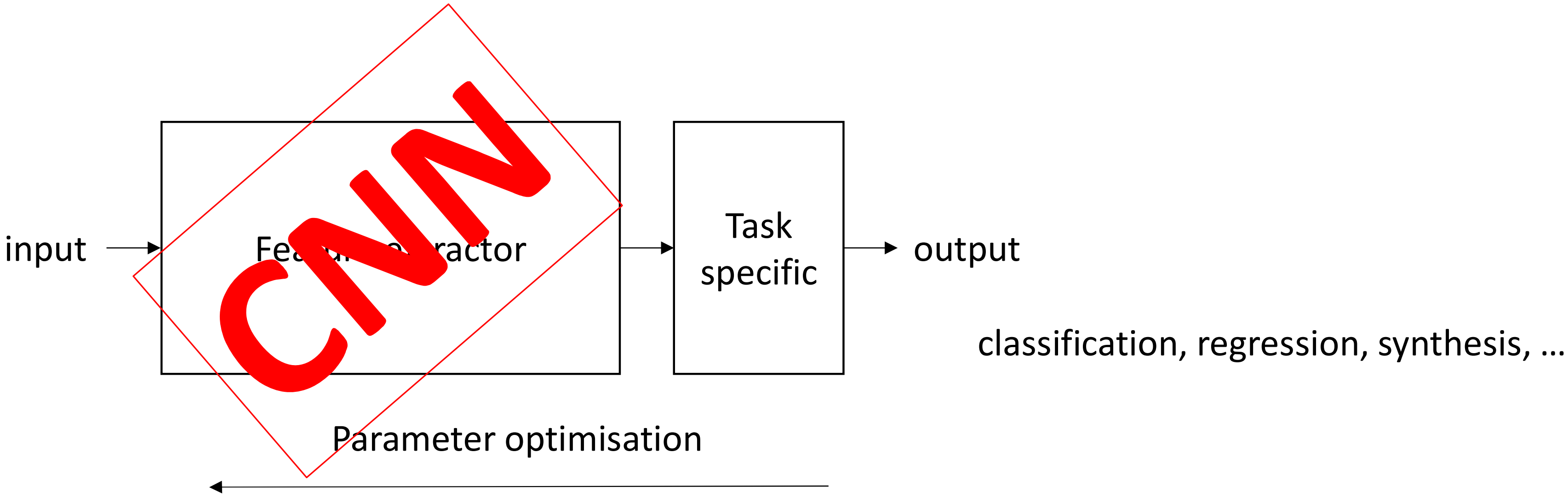
Deep Learning

Bernhard Kainz

Motivation

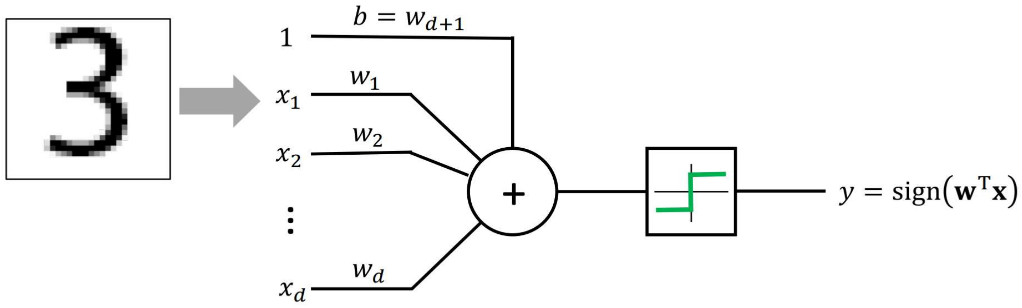
- Deep learning is **popular** because it works (often).
  - Big promise: just collect enough data and label it, then you get a magic black-box predictor that can predict any correlations at the click of a button. (only supervised setting really works well)
- Deep learning and Big data = **big money** = highly competitive and sometimes poisonous working environment.
- Deep learning can be **dangerous**, e.g. deep fakes, adversarial attacks, etc.

Fundamental learning system



*CNN = convolutional neural network

Why did neural networks fail in image analysis?



Stack a 32x32x3 RGB image into a 3072x1 vector

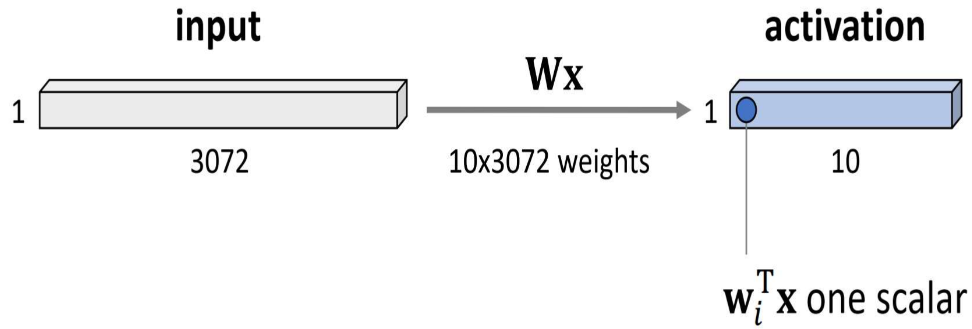


Figure: adapted from Fei Fei et al.

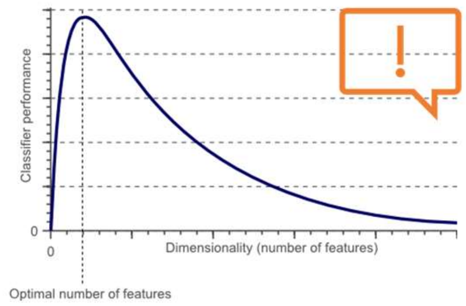
Curse of dimensionality

As the number of features or dimensions grows,
the amount of data we need to generalise accurately grows exponentially!

To approximate a (Lipschitz) continuous function $f\colon \mathbb{R}^d \to \mathbb{R}$
with ϵ accuracy one needs $O(\epsilon^{-d})$ samples

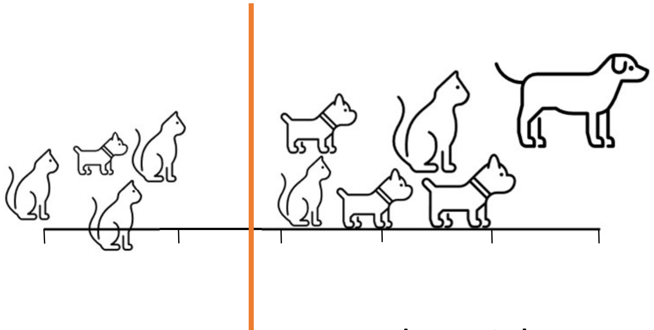
https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

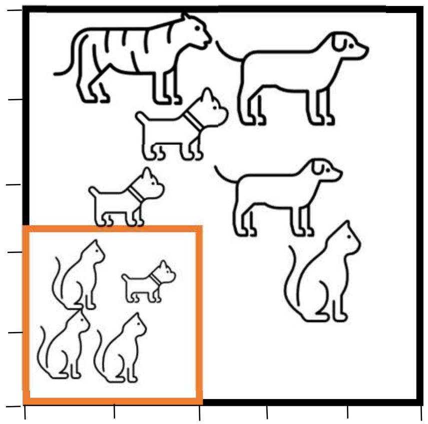
https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

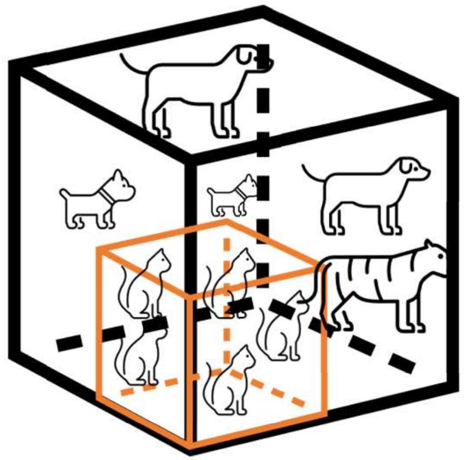


Curse of dimensionality

As the number of features or dimensions grows,
the amount of data we need to generalise accurately grows exponentially!







One parameter: Body weight or Body size, …

Two parameters: Body weight and Body size, …

Three parameters: Body weight and Body size and has a leech …

10 samples:

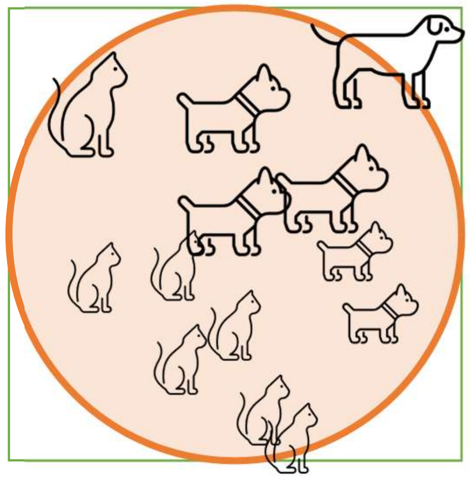
20% samples = 0.2
5 unit intervals
10/5=2 samples/interval

20% samples = 0.45^2
5×5=25 unit squares
10/25 = 0.4 samples/interval

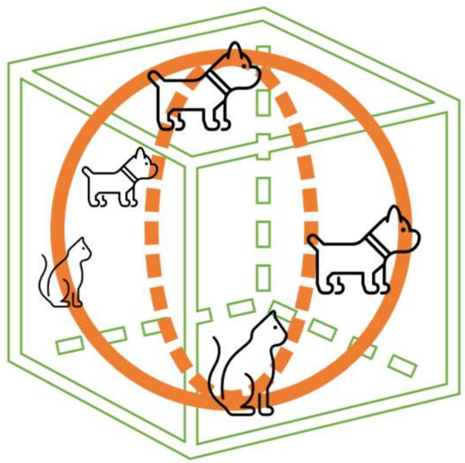
20% samples = 0.58^3
5x5x5 unit intervals
10/125 = 0.08 samples/interval

Curse of dimensionality

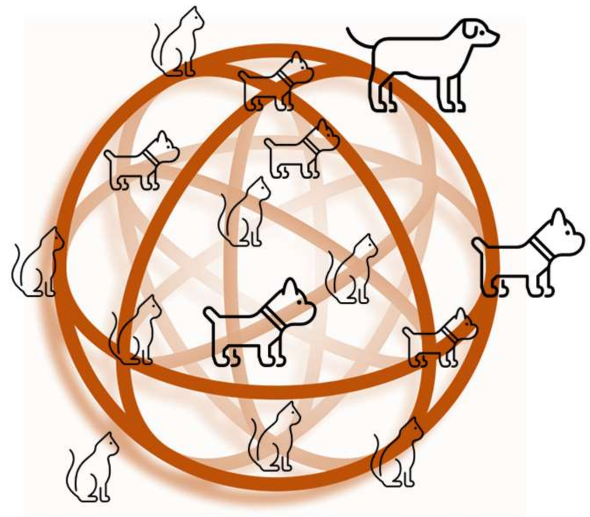
Ratio between red and green



$\pi 0.5^2 \approx 0.785$



$\frac{4}{3}\pi 0.5^3 \approx 0.52$

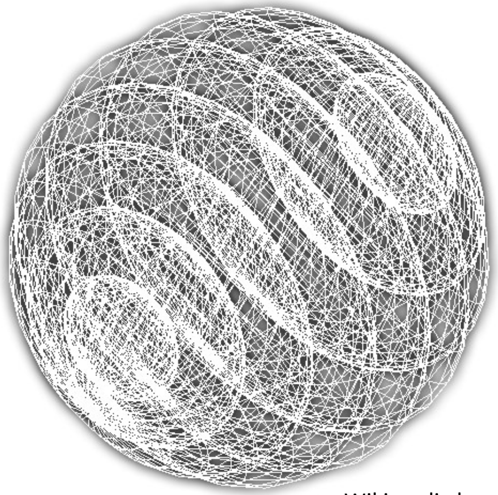


Wikimedia hypersphere

10 dimensions ≈ 0.0159

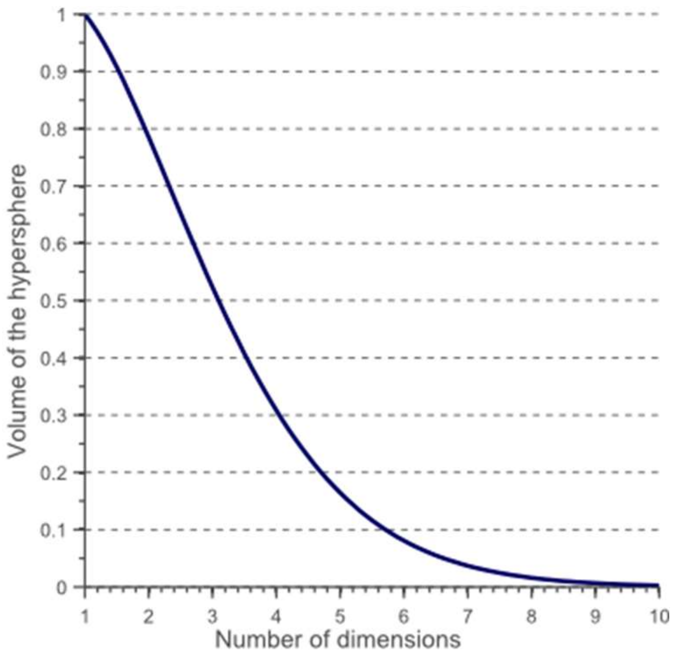
The higher dimensional the feature space the more training samples will be in the corners of the hypercube, thus generalisation suffers.

Curse of dimensionality



Wikimedia hypersphere

$$V_{sphere}(d) = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)2^d} \backsim O(c^{-d})$$



https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

The higher dimensional the feature space the more training samples will be in the corners of the hypercube, thus generalisation suffers.

Curse of dimensionality

To approximate a (Lipschitz) continuous function $f\colon \mathbb{R}^d \to \mathbb{R}$
with ϵ accuracy one needs $O(\epsilon^{-d})$ samples

Input image resolution = 12 Mpixel * 3 channels = 36M elements
With $\epsilon \sim 0.1$, we need $10^{36000000}$ samples (10^{78} to 10^{82} atoms in the known, observable universe)

what do we learn from that?

- a) feature selection is important to build good classifiers. As we will see, the key of deep learning is to learn this feature selection instead of doing it manually.
- b) finding the right amount of features is key. Too few or too many will have a severe impact on the generalization abilities of your predictor model. Too few is easy too understand but too many requires an intuition about sample sparsity in high-dimensional spaces.
- c) the more features we choose as input the sparser our training samples will be distributed in the feature space. This means that decision boundaries become really tight around the used training samples because they all live close to each other at the boundaries of the space and our model will overfit the training data.

Deep Learning

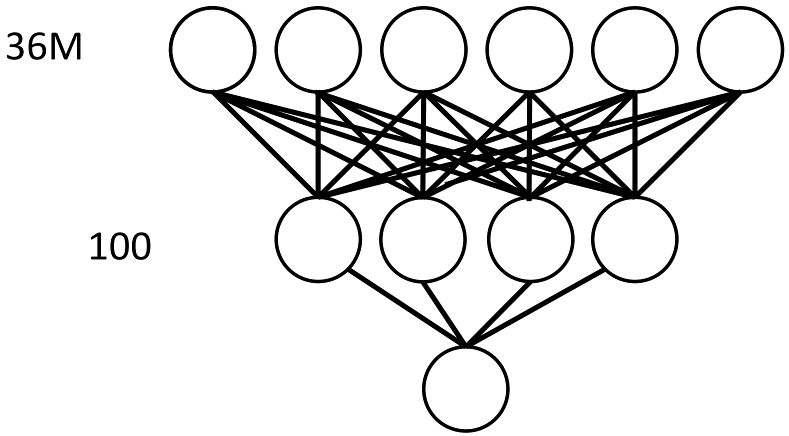
Bernhard Kainz

Curse of dimensionality

To approximate a (Lipschitz) continuous function $f\colon \mathbb{R}^d \rightarrow \mathbb{R}$ with ϵ accuracy one needs $O(\epsilon^{-d})$ samples

Input image resolution = 12 Mpixel * 3 channels = 36M elements
With $\epsilon \sim 0.1$, we need $10^{36000000}$ samples (10^{78} to 10^{82} atoms in the known, observable universe)

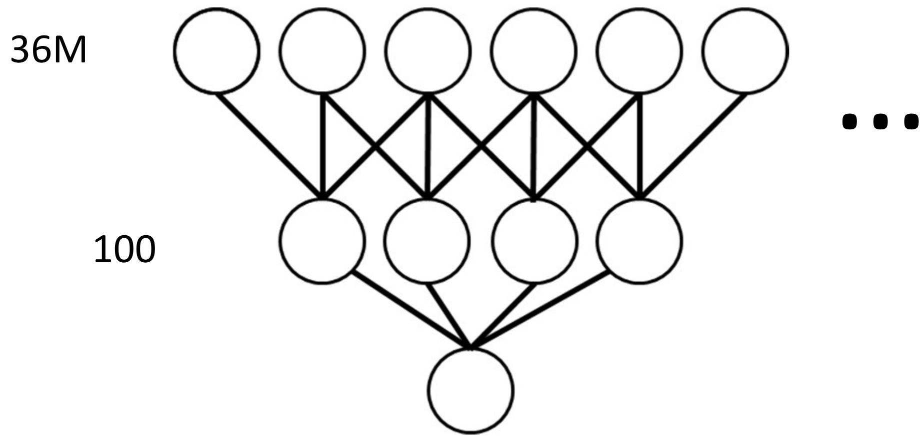
Curse of dimensionality



… n^2 parameters!

- Input image resolution = 12 Mpixel * 3 channels = 36M elements
- MLP with one hidden layer of width=100 has **3.6B parameters**
- That’s more than the number of cats and dogs on earth!

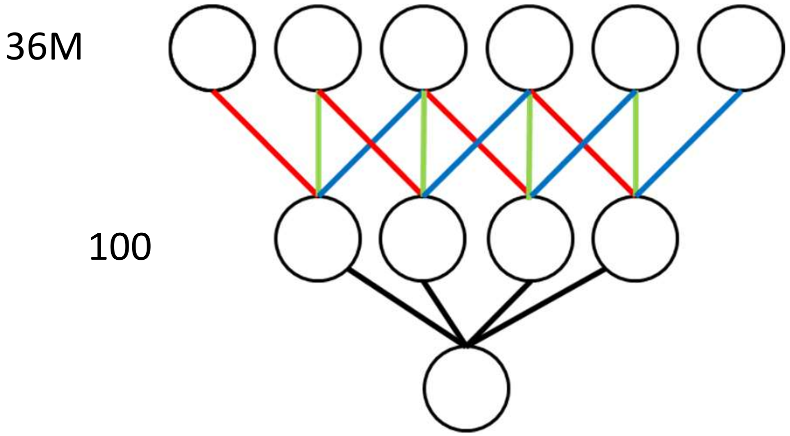
Curse of dimensionality



k*n parameters!

- Input image resolution = 12 Mpixel * 3 channels = 36M elements
- MLP with one hidden layer of width=100 has **3.6B parameters**
- That’s more than the number of cats and dogs on earth!

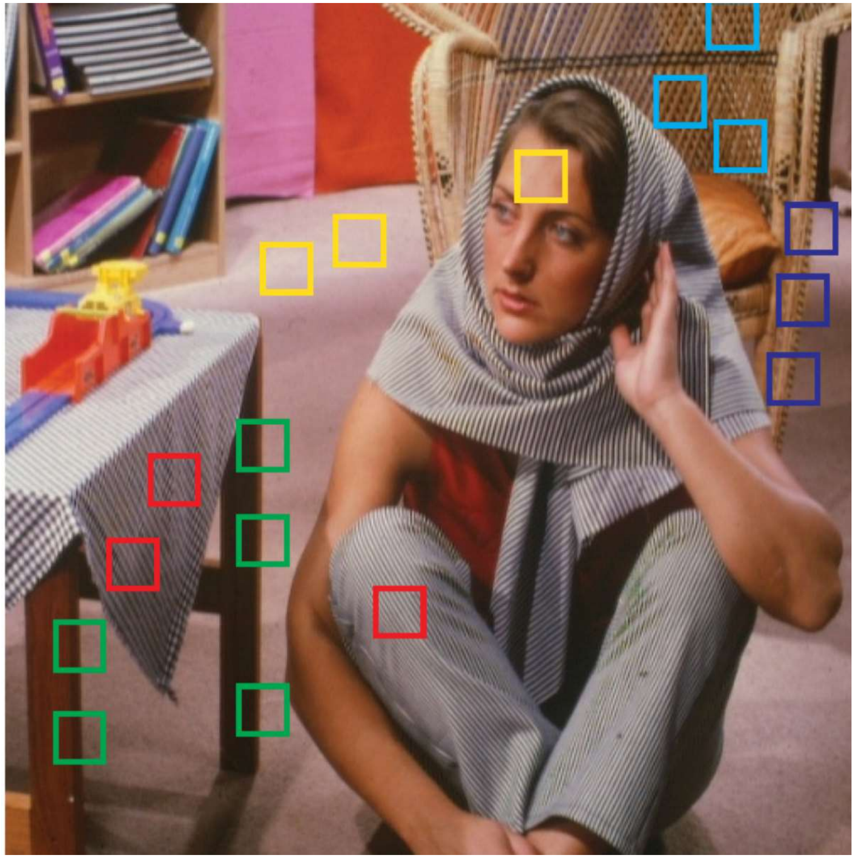
Curse of dimensionality

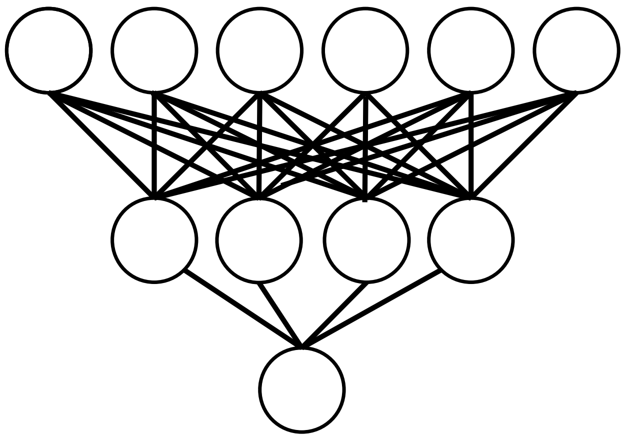


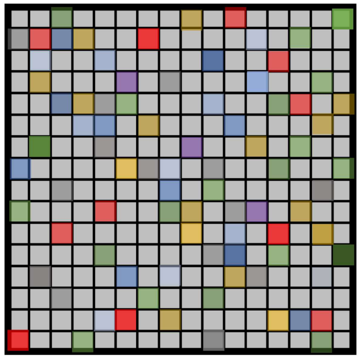
... k parameters!

- Input image resolution = 12 Mpixel * 3 channels = 36M elements
- MLP with one hidden layer of width=100 has **3.6B parameters**
- That's more than the number of cats and dogs on earth!

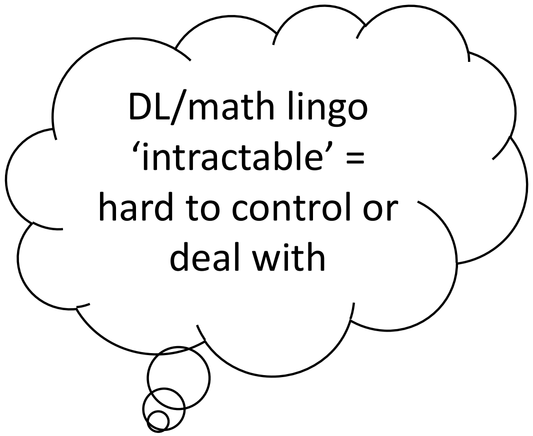
Self similarity



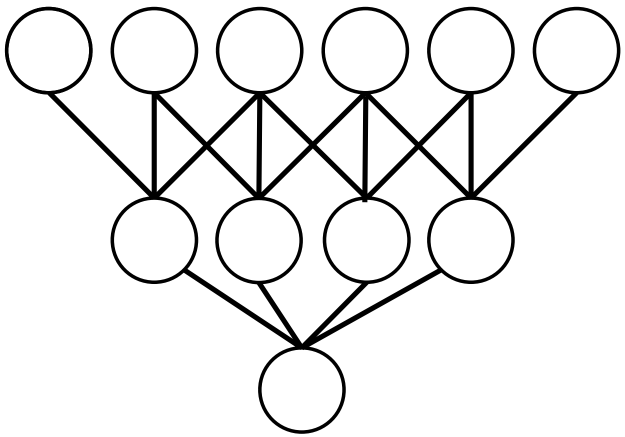


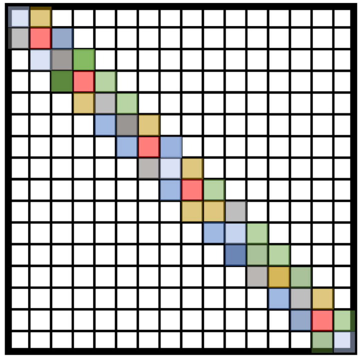


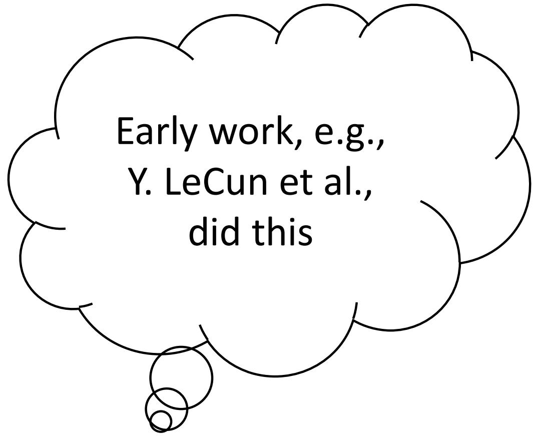
$$y_j = w_{j,1}x_1 + \cdots + w_{j,n}x_n$$



n^2 parameters, $36M^2$ parameters!

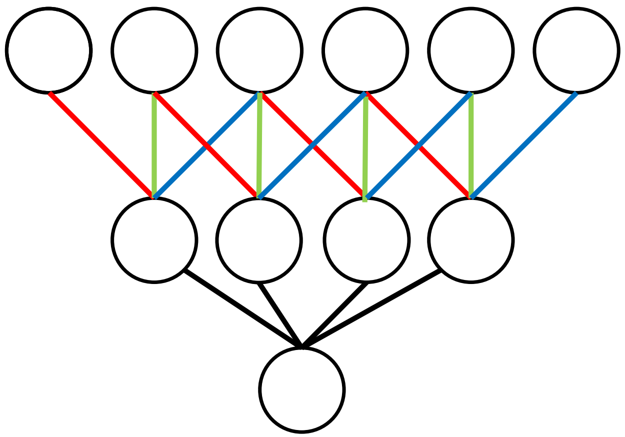


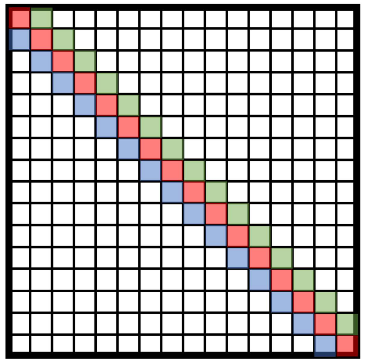




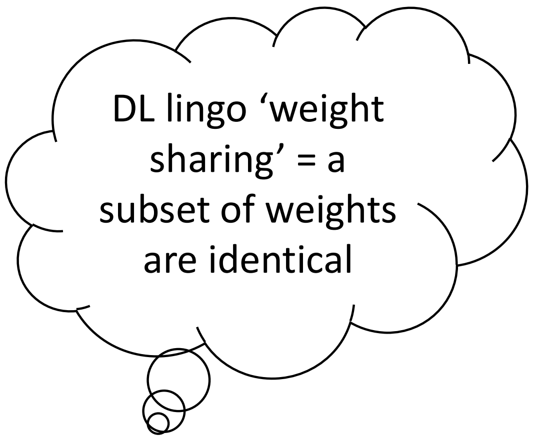
$$y_j = w_{j,i-1}x_{i-1} + w_{j,i}x_i + w_{j,i+1}x_{i+1}$$

Each input neuron is connected to a small number k of hidden neurons.
Sparse connections: k*n parameters, e.g., 3*36M parameters!

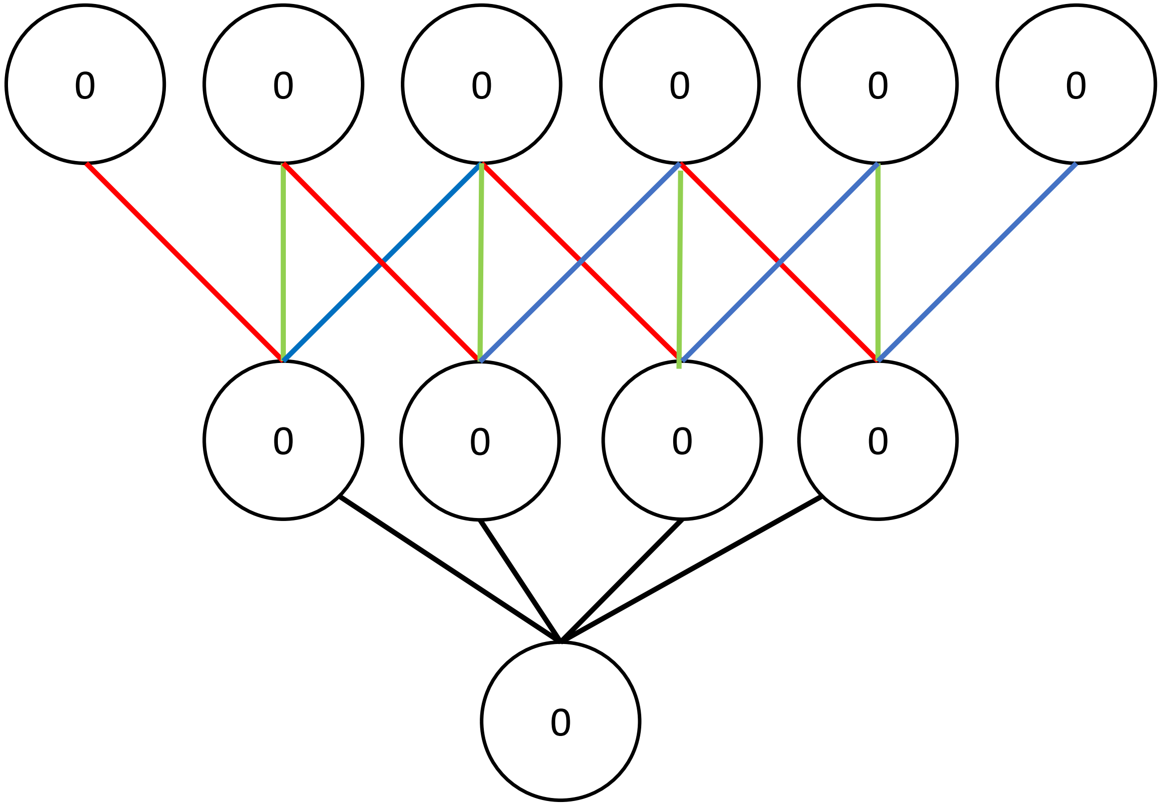




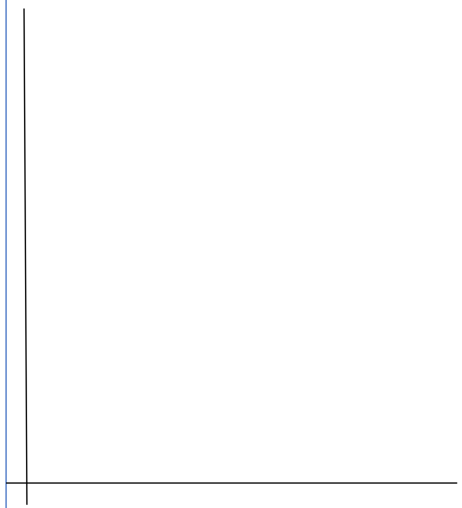
$$y_j = w_{-1}x_{i-1} + w_0 x_i + w_{+1}x_{i+1}$$



Each input neuron is connected to a small number k of hidden neurons and weights are shared
Shared weights (position independent): k parameters, e.g. 3 parameters!

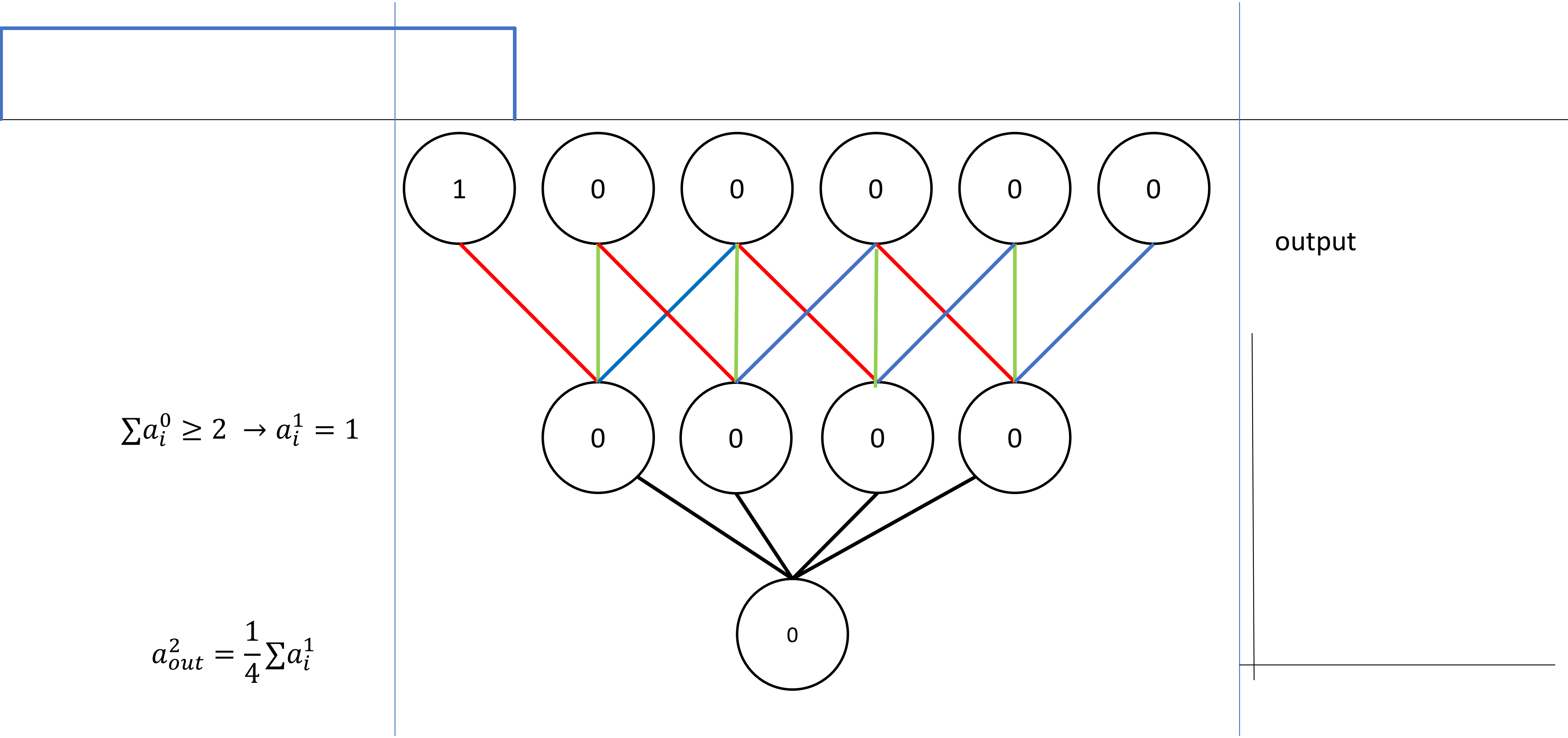


output



$$\sum a_i^0 \geq 2 \ \rightarrow a_i^1 = 1$$

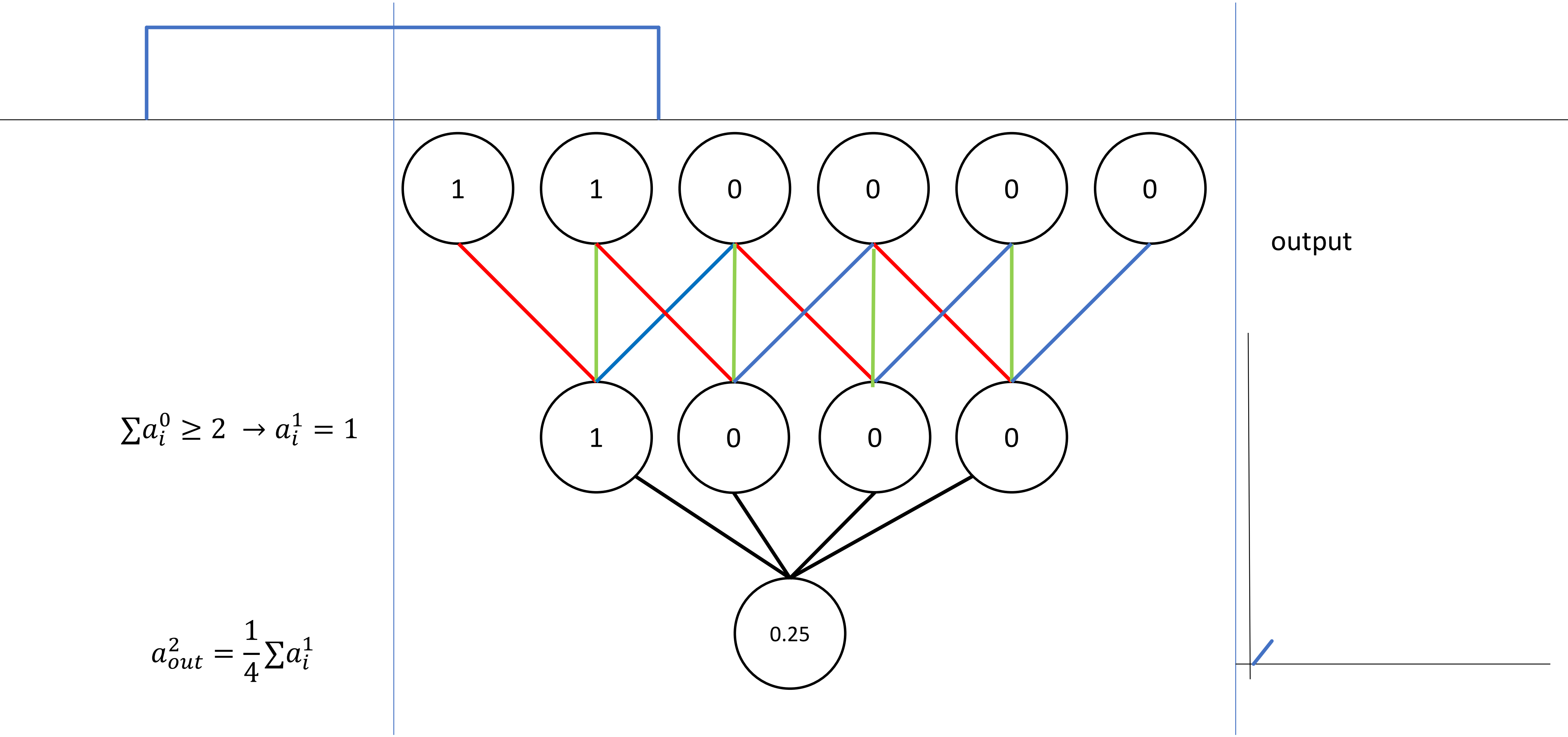
$$a_{out}^2 = \frac{1}{4} \sum a_i^1$$



output

$$\sum a_i^0 \geq 2 \rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

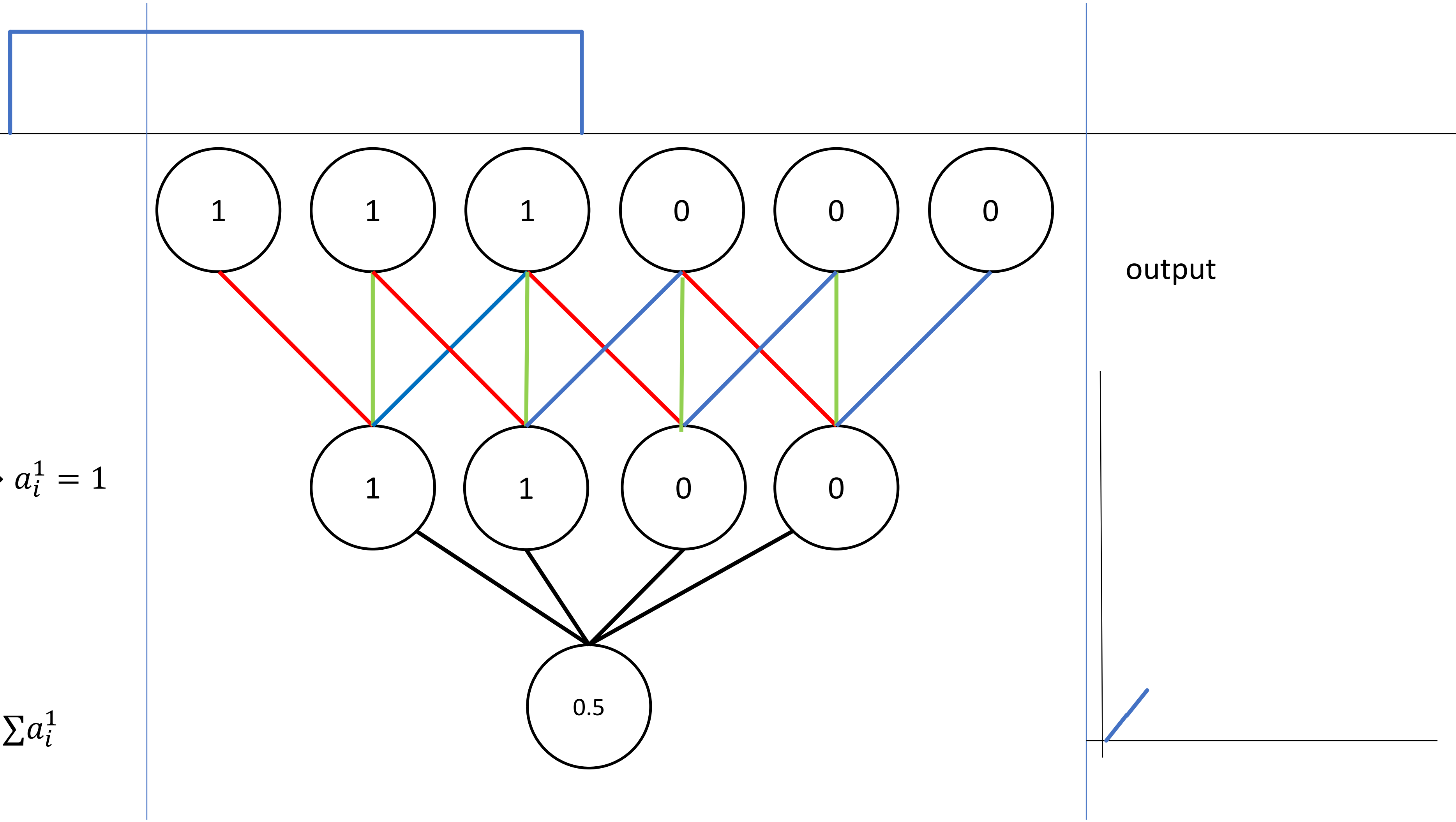


$$\sum a_i^0 \geq 2 \; \rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

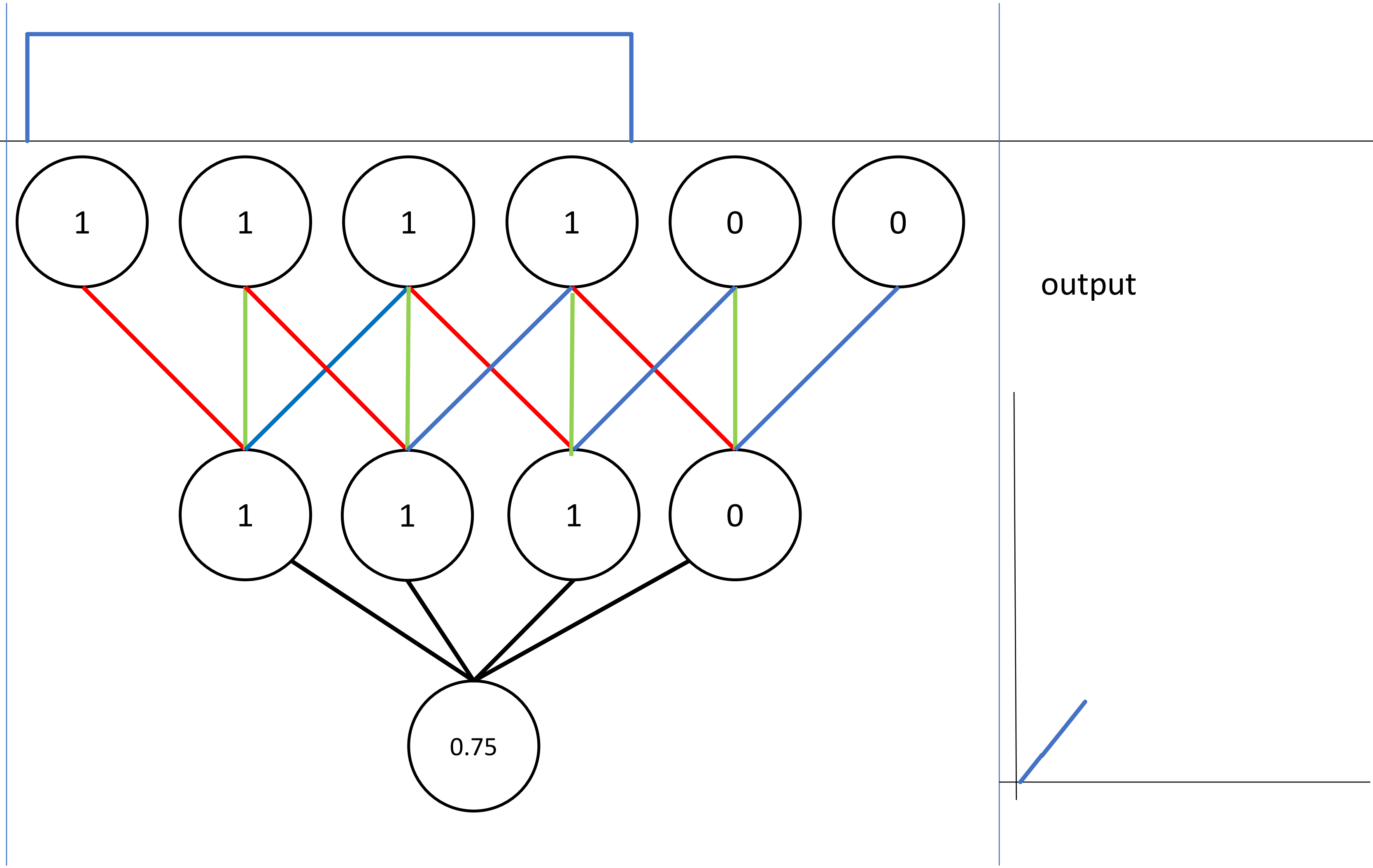
$$\sum a_i^0 \geq 2 \ \rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

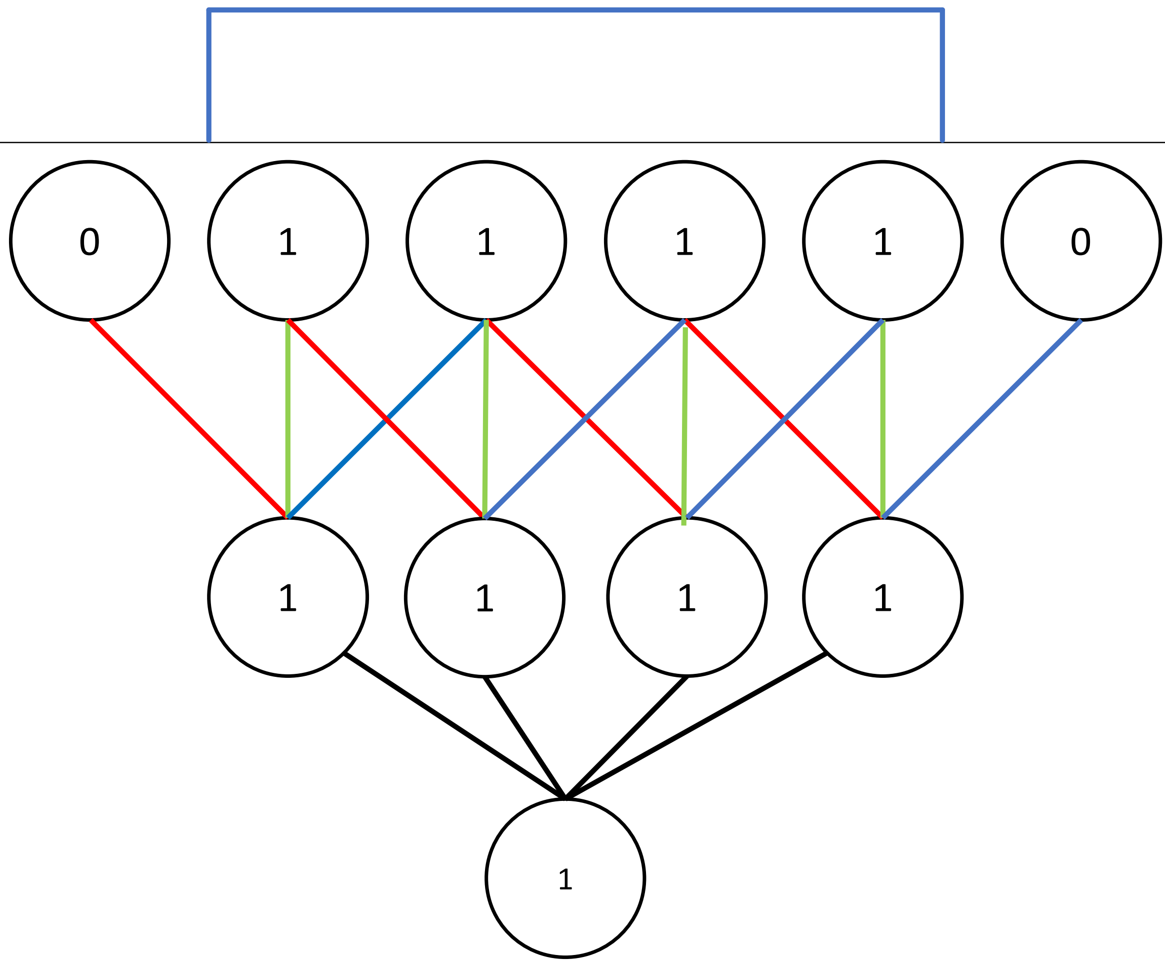


$$\sum a_i^0 \geq 2 \;\rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

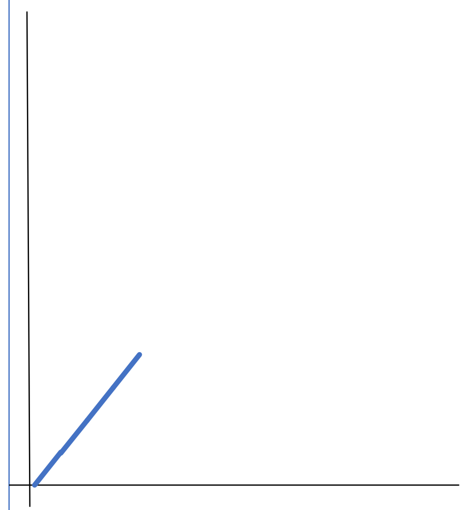


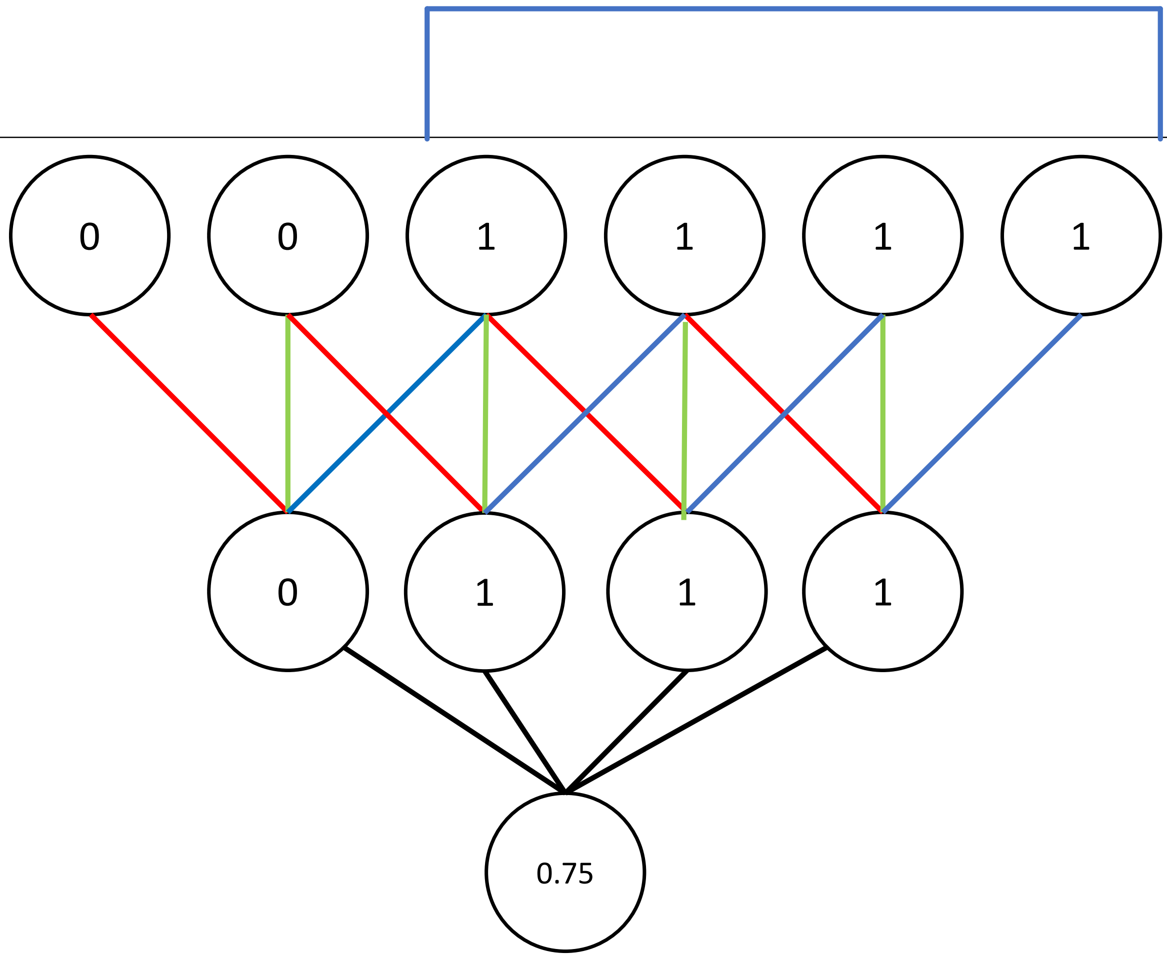
$$\sum a_i^0 \geq 2 \;\rightarrow a_i^1 = 1$$



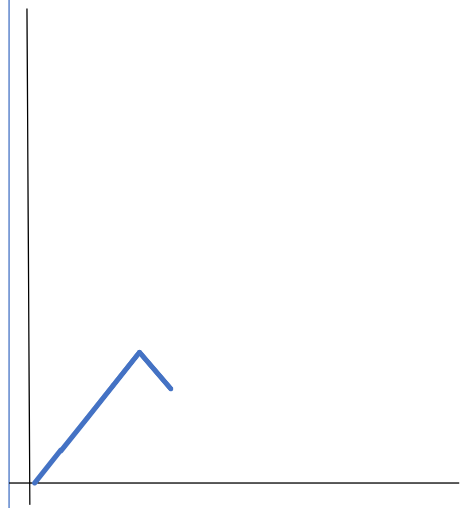
$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

output





output

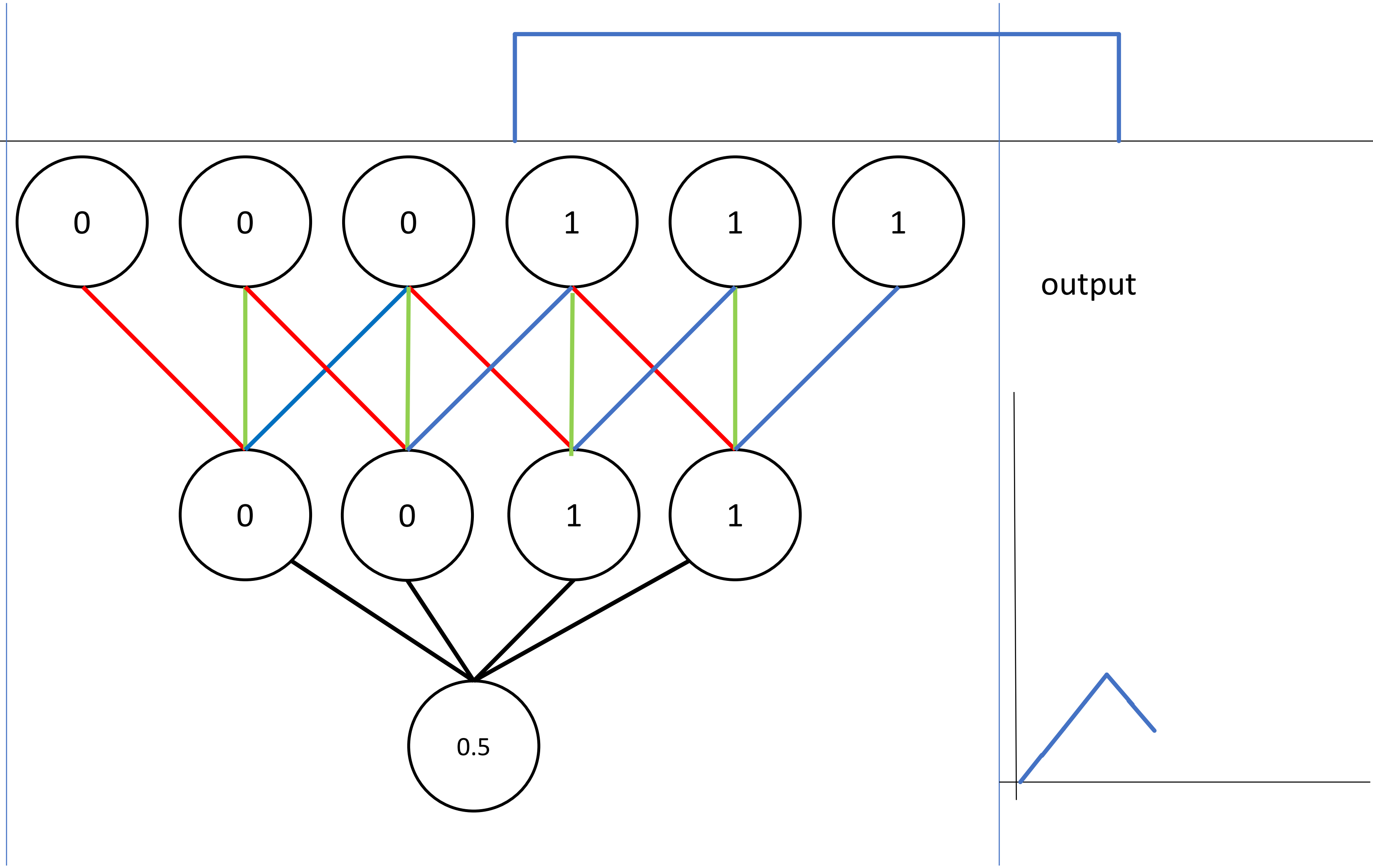


$$\sum a_i^0 \geq 2 \rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

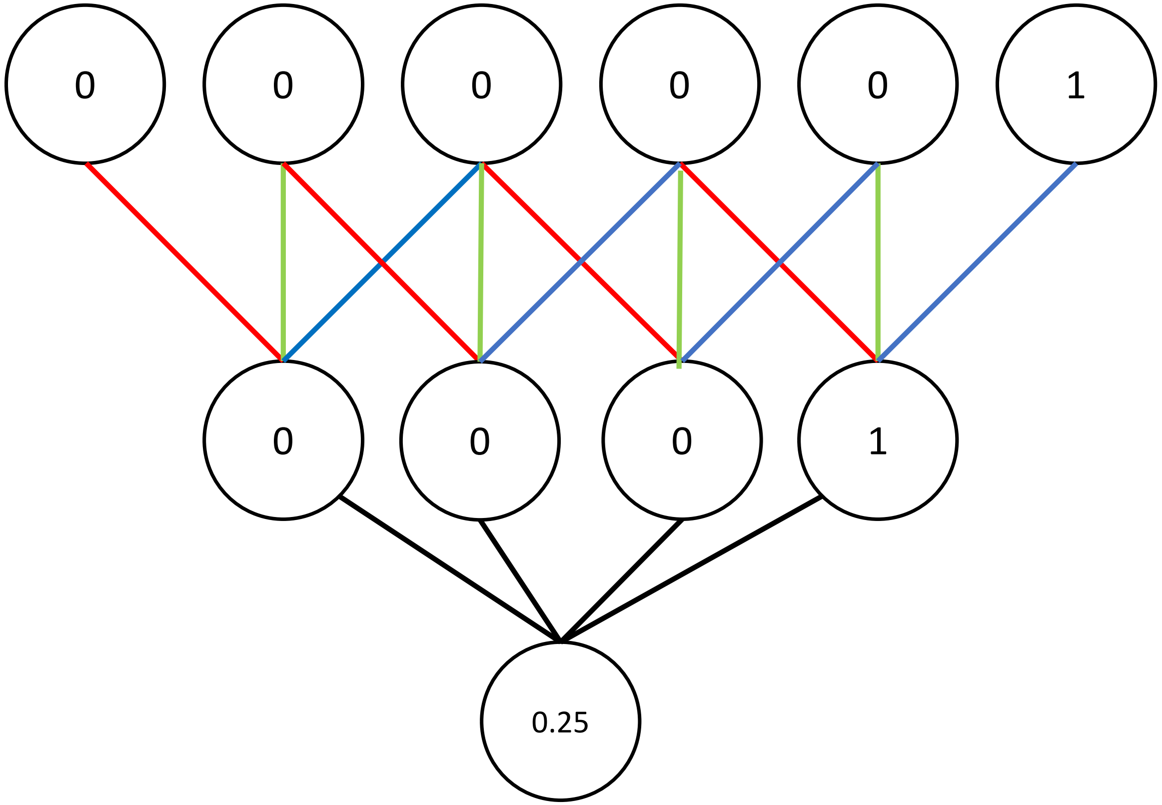
$$\sum a_i^0 \geq 2 \rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

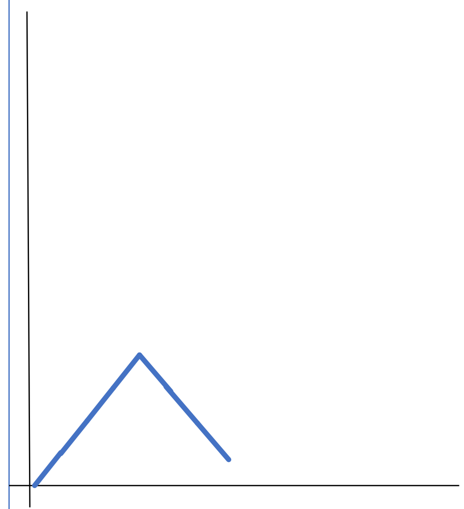


$$\sum a_i^0 \geq 2 \ \rightarrow a_i^1 = 1$$

$$a_{out}^2 = \frac{1}{4}\sum a_i^1$$

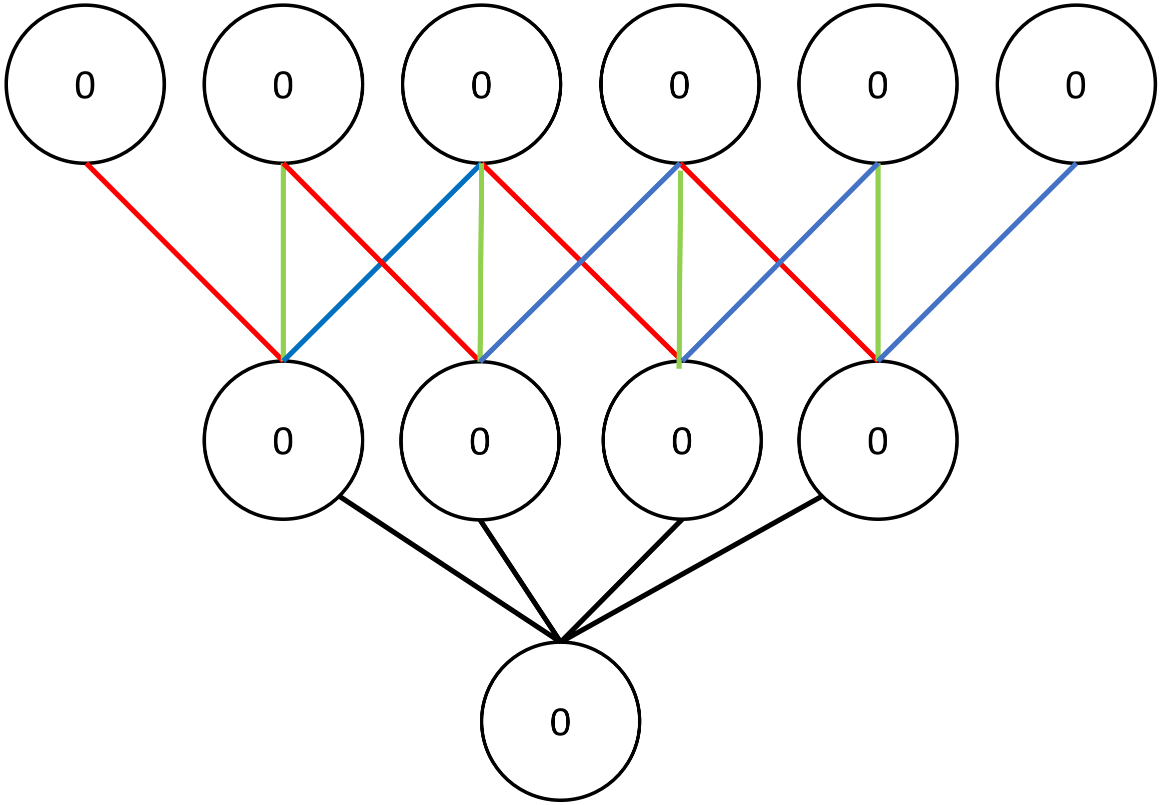


output

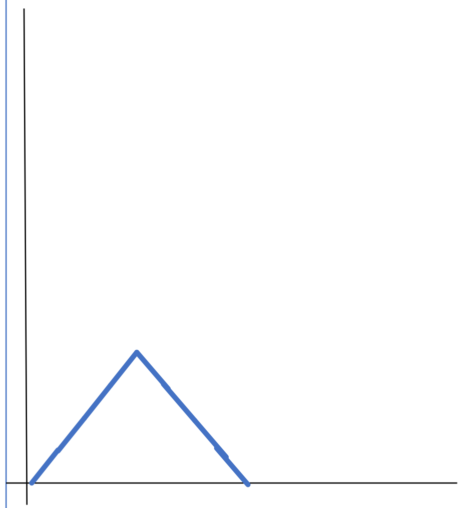


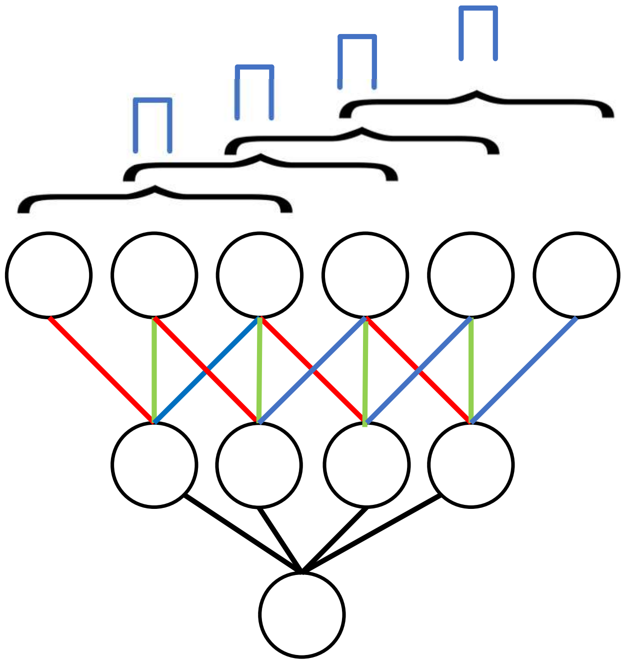
$\sum a_i^0 \geq 2 \rightarrow a_i^1 = 1$

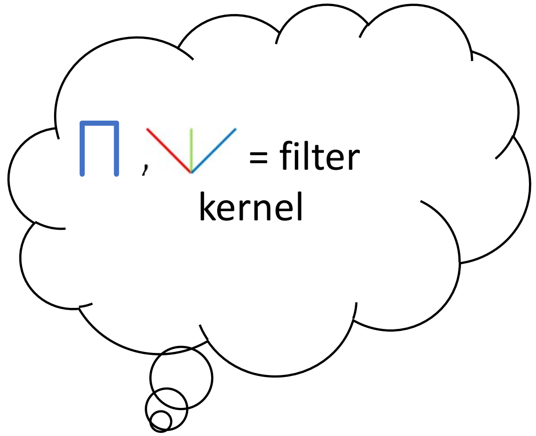
$a_{out}^2 = \frac{1}{4}\sum a_i^1$

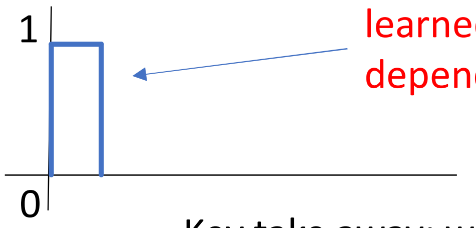


output







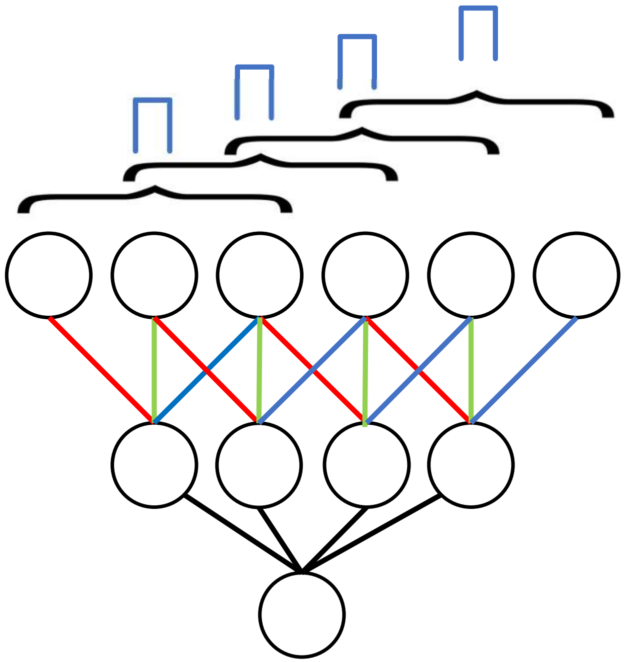


learned through backpropagation,
dependant on the task! Up to hundreds per layer.

Key take away: weight sharing! (a limited number of learnable “filter” parameters for a fixed but overlapping input range);
Think filtering with sliding window!

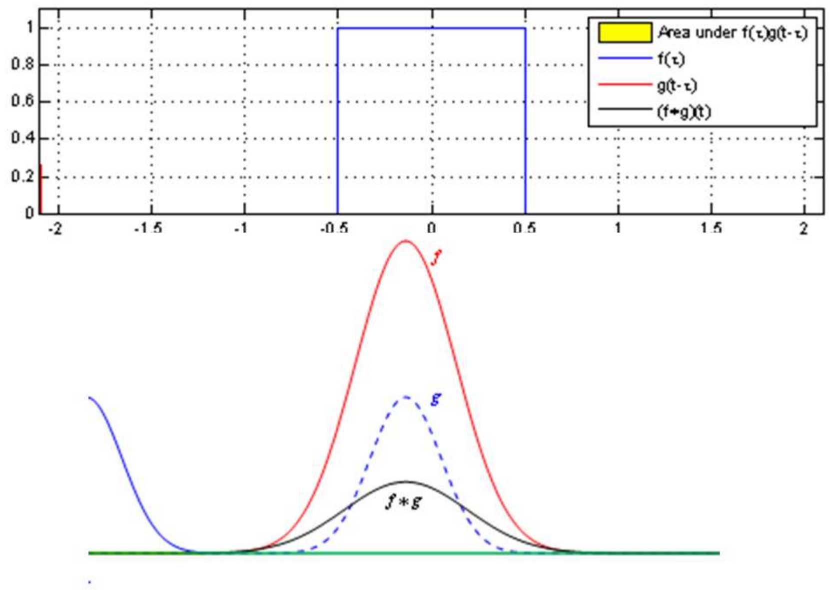
e.g., 3 parameters!

Why "convolution"?



$$Convolution: \ (x * w)_i = \sum_k x_k w_{i-k} = \sum_k x_{i-k} w_k$$

$$cp. cross\ correlation: \ (x * w)_i = \sum_k x_k w_{i+k}$$



wikipedia.org

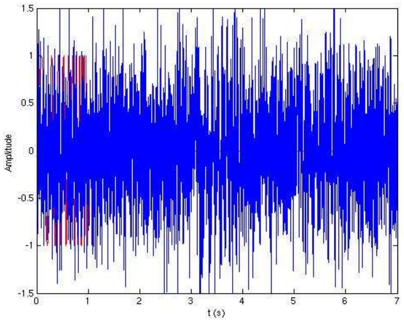
Network output (continuous):

$$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)d\tau \ \ for\ f, g : [0, \infty) \to \mathbb{R}$$

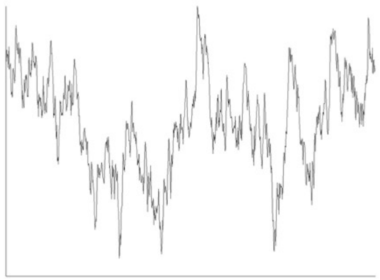
Some features of convolution are similar to <u>cross-correlation</u>: for real-valued functions, of a continuous or discrete variable, it differs from cross-correlation only in that either *f*(*x*) or *g*(*x*) is reflected about the y-axis; thus it is a cross-correlation of *f*(*x*) and *g*(−*x*), or *f*(−*x*) and *g*(*x*).

Why not simply input = output for this feature detector?

Signals in the wild:



Features in the wild:



Watch:
https://www.youtube.com/watch?v=N-zd-T17uiE

Properties of convolutions

- Commutativity, $f * g = g * f$
- Associativity, $f * (g * h) = (f * g) * h$
- Distributivity, $f * (g + h) = (f * g) + (g * h)$
- Associativity with scalar multiplication, $a(f * g) = (af) * g$

What do we learn from this

- a) weight sharing reduces the number of parameters from n^2 in a multi-layer perception to a small number, for example 3 as in our experiment or 3 by 3 image filter kernels or similar
- b) these filter kernels can be learned through back propagation exactly in the same way as you would train a multi-layer perception. Each layer may have many filter-kernels, so it will produce many filtered versions of the input with different filter functions.
- c) for real-valued functions, of a continuous or discrete variable, convolution differs from cross-correlation only in that either $f(x)$ or $g(x)$ is reflected about the y-axis; so it is a cross-correlation of $f(x)$ and $g(-x)$, or $f(-x)$ and $g(x)$.

Second problem: no spatial structure preservation, fully connected layer

Stack a 32x32x3 RGB image into a 3072x1 vector

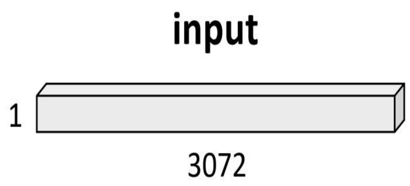


Figure: adapted from Fei Fei et al.

- we need priors about the data!

spatial structure preservation, convolutional layer

- Keep spatial structure of the 32x32x3 RGB image

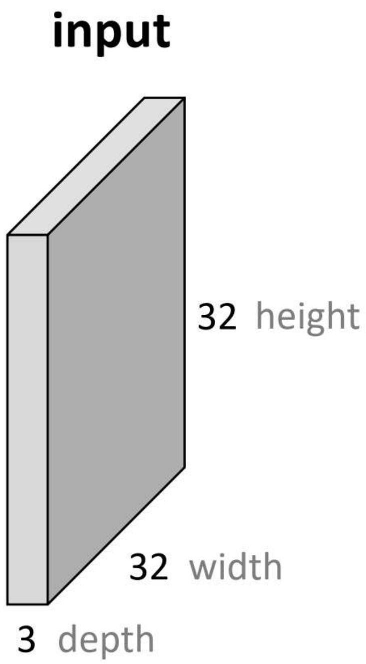
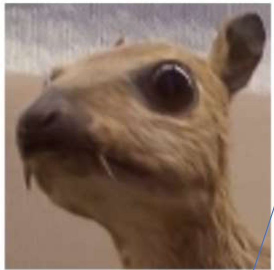


Figure: adapted from Fei Fei et al.

Examples of 2D image filters



(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

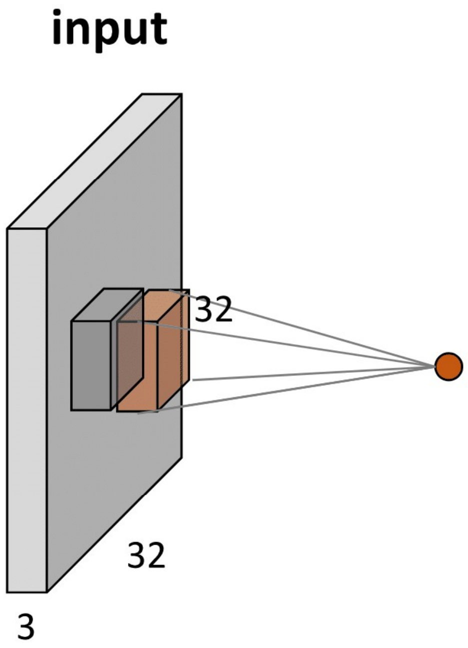
Gaussian Blur

Remember: all learned through backpropagation, dependant on the task!

Slide credit: Smola, Li 2019

Convolutional layer

- Keep spatial structure of the 32x32x3 RGB image



Dot product between the filter and a 5x5x3 image patch

Figure: adapted from Fei Fei et al.

Convolutional layer

- Keep spatial structure of the 32x32x3 RGB image

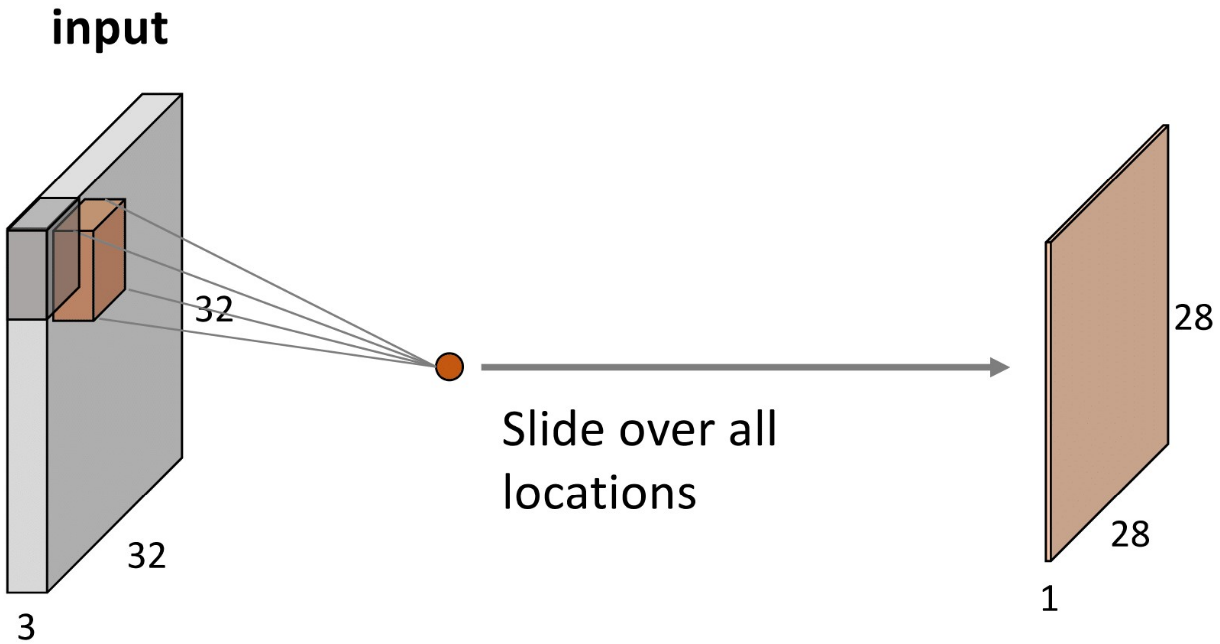


Figure: adapted from Fei Fei et al.

Convolutional layer

- Keep spatial structure of the 32x32x3 RGB image

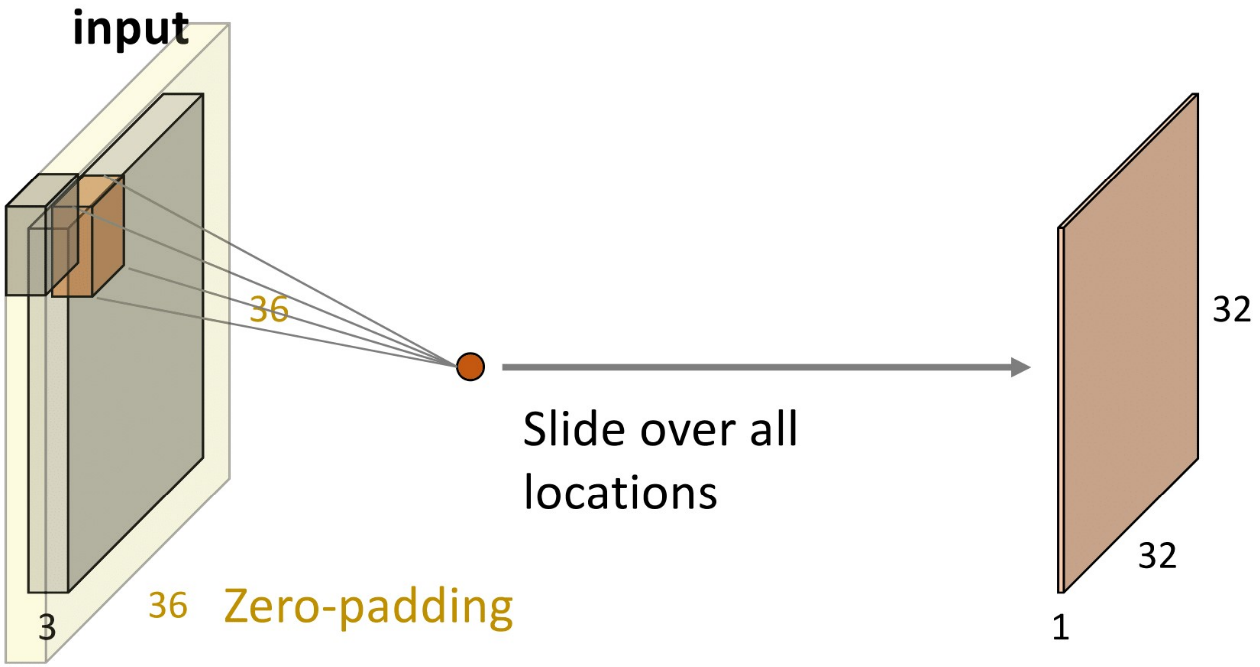


Figure: adapted from Fei Fei et al.

Convolutional layer

- Keep spatial structure of the 32x32x3 RGB image

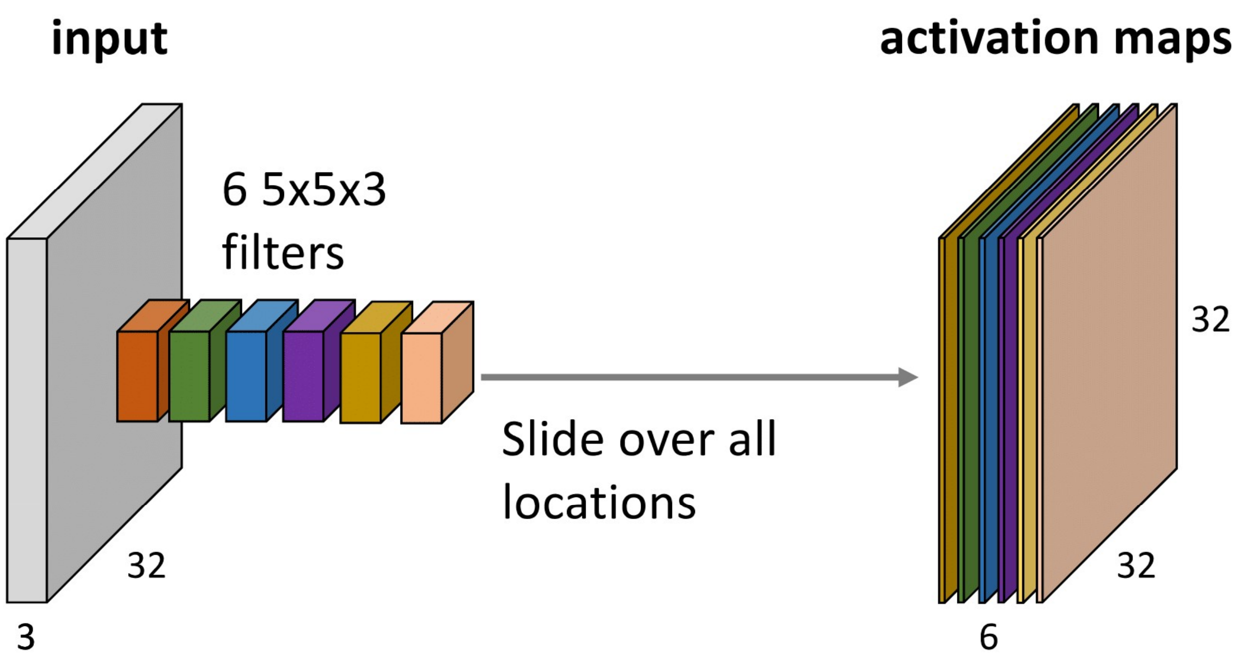


Figure: adapted from Fei Fei et al.

Convolutional layer

- CNN = sequence of convolutional layers interleaved with ReLUs

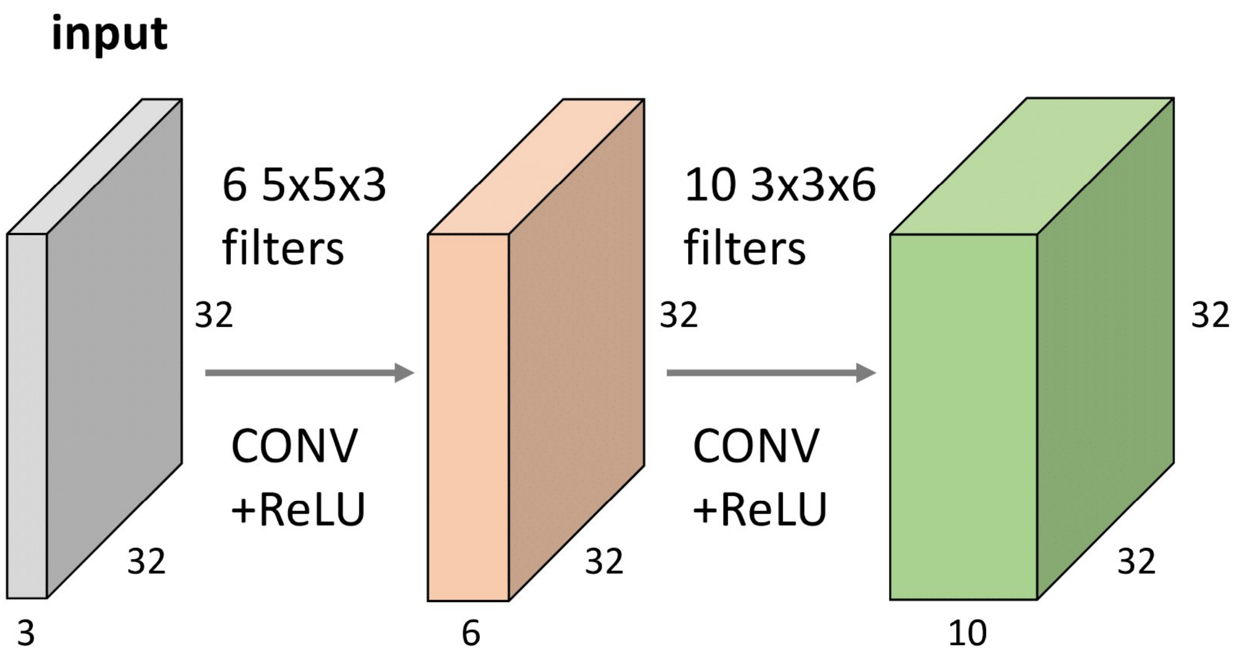


Figure: adapted from Fei Fei et al.

Number of parameters

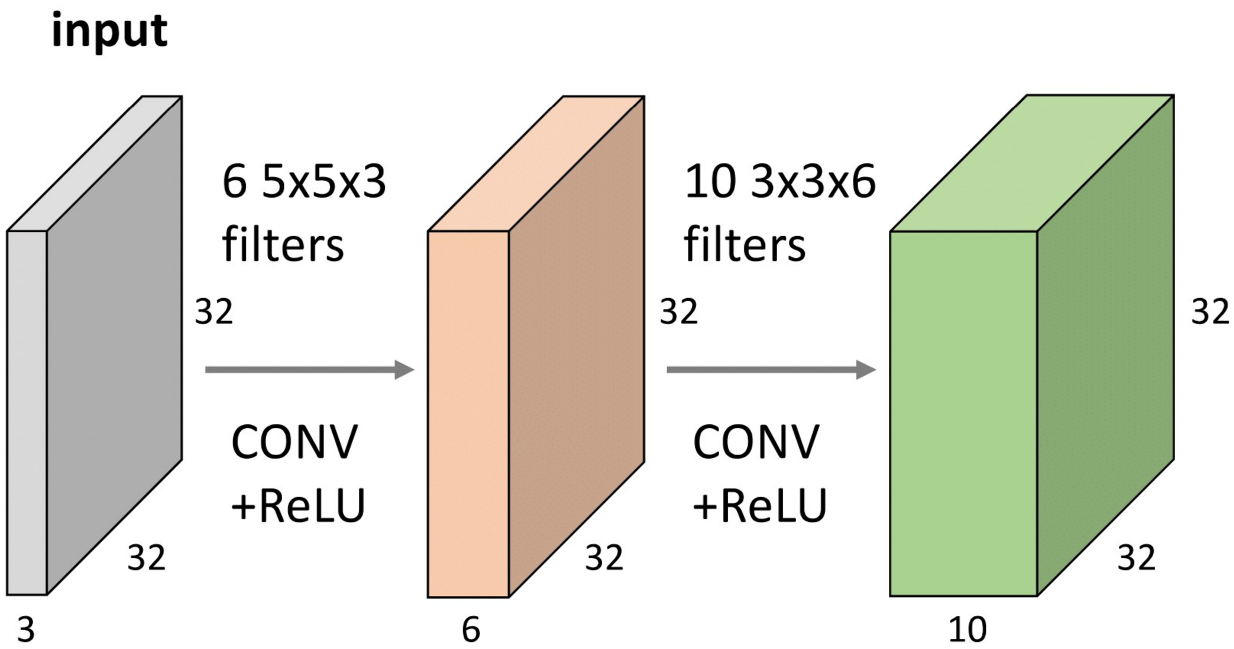
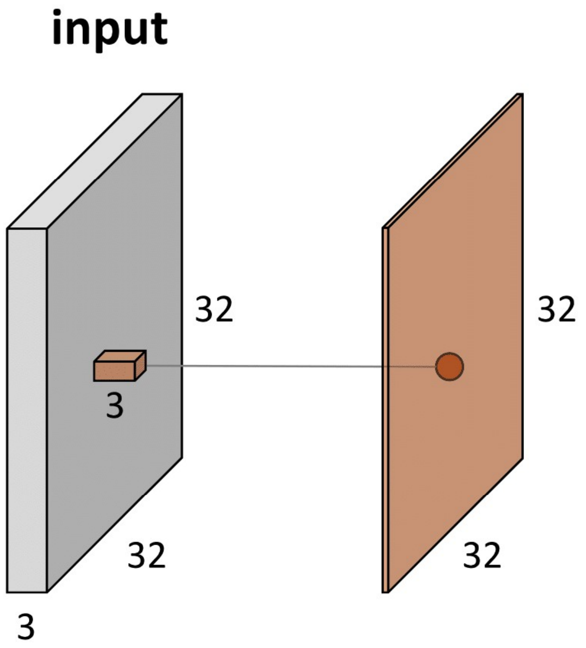


Figure: adapted from Fei Fei et al.

1x1 convolution

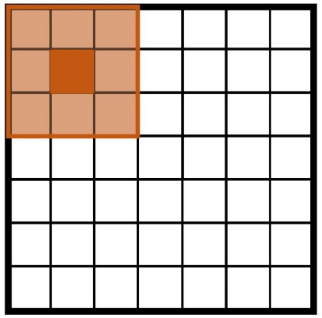
- Each 1x1x3 filter performs a 3-dimensional dot product



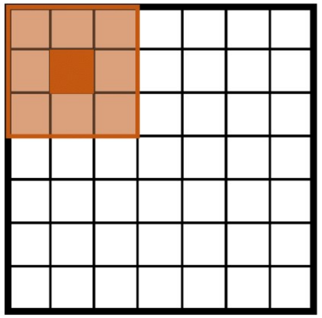
Dot product between the filter and a 1x1x3 image pixel feature vector

Figure: adapted from Fei Fei et al.

Padding and strides



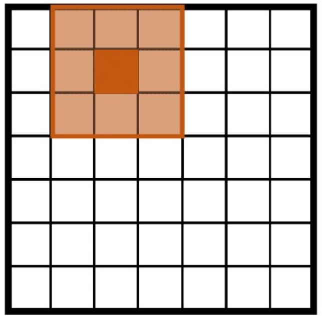
7x7 input
3x3 filter
stride 1
no padding



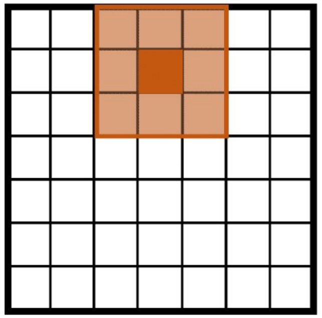
7x7 input
3x3 filter
stride 2
no padding

Figure: adapted from Fei Fei et al.

Padding and strides



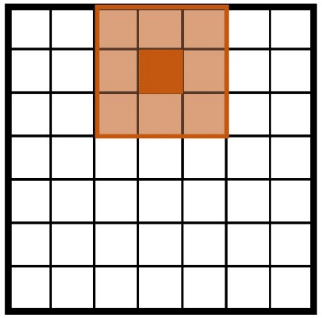
7x7 input
3x3 filter
stride 1
no padding



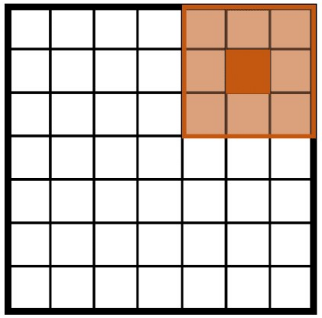
7x7 input
3x3 filter
stride 2
no padding

Figure: adapted from Fei Fei et al.

Padding and strides



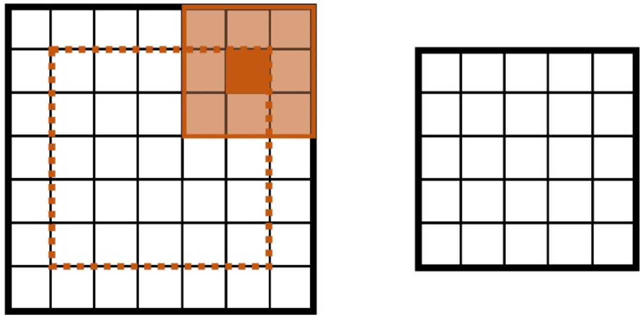
7x7 input
3x3 filter
stride 1
no padding



7x7 input
3x3 filter
stride 2
no padding

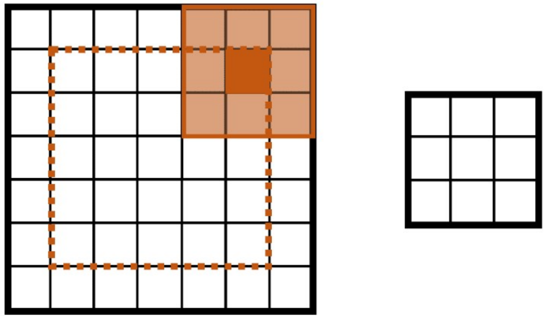
Figure: adapted from Fei Fei et al.

Padding and strides



7x7 input
3x3 filter
stride 1
no padding

5x5 output

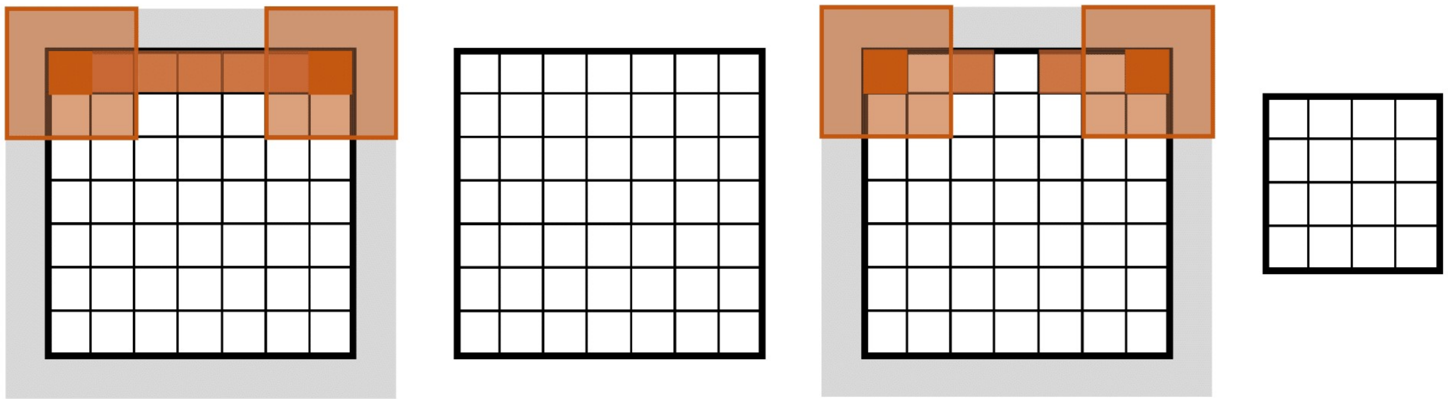


7x7 input
3x3 filter
stride 2
no padding

3x3 output

Figure: adapted from Fei Fei et al.

Padding and strides



7x7 input
3x3 filter
stride 1
zero padding

7x7 output

7x7 input
3x3 filter
stride 2
zero padding

4x4 output

Figure: adapted from Fei Fei et al.

Computational complexity

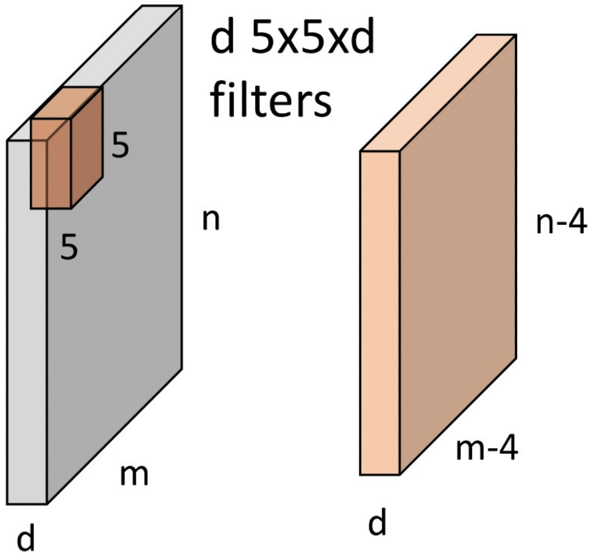


Figure: adapted from Fei Fei et al.

Factorized convolution

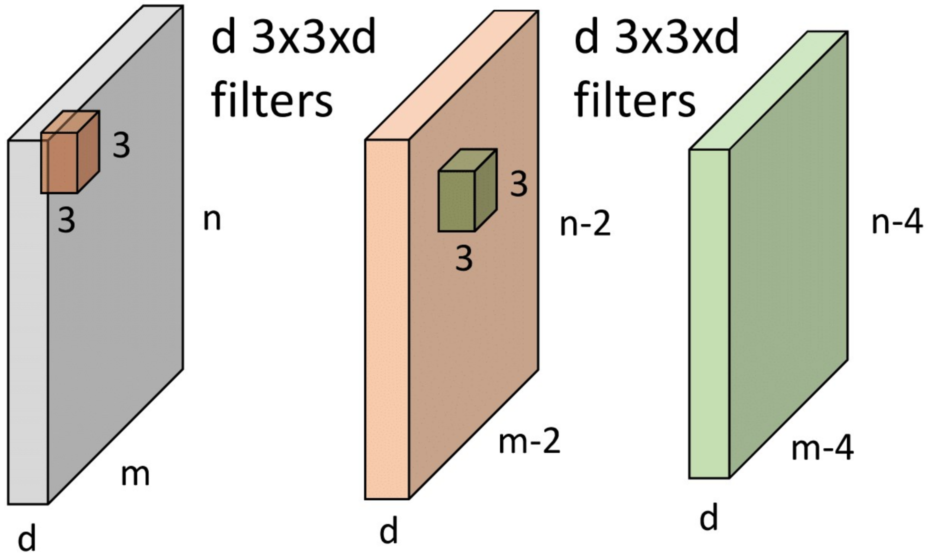


Figure: adapted from Fei Fei et al.

Separable convolution

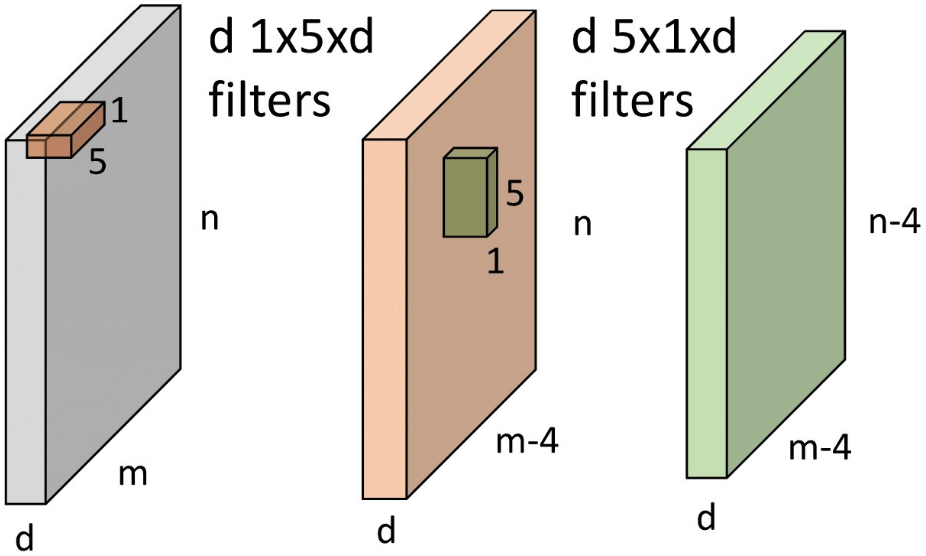


Figure: adapted from Fei Fei et al.

Pooling

- Permutation-invariant aggregation+downsampling (typically max or avg)
- Reduces resolution
- Hierarchical features
- Contributes to approximate shift/deformation invariance

6	1	2	4
1	6	7	8
3	5	2	0
1	2	3	4

Figure: adapted from Fei Fei et al.

Pooling

- Permutation-invariant aggregation+downsampling (typically max or avg)
- Reduces resolution
- Hierarchical features
- Contributes to approximate shift/deformation invariance

6	1	2	4
1	6	7	8
3	5	2	0
1	2	3	4

6	

Figure: adapted from Fei Fei et al.

Pooling

- Permutation-invariant aggregation+downsampling (typically max or avg)
- Reduces resolution
- Hierarchical features
- Contributes to approximate shift/deformation invariance

6	1	2	4
1	6	7	8
3	5	2	0
1	2	3	4

6	8

Figure: adapted from Fei Fei et al.

Pooling

- Permutation-invariant aggregation+downsampling (typically max or avg)
- Reduces resolution
- Hierarchical features
- Contributes to approximate shift/deformation invariance

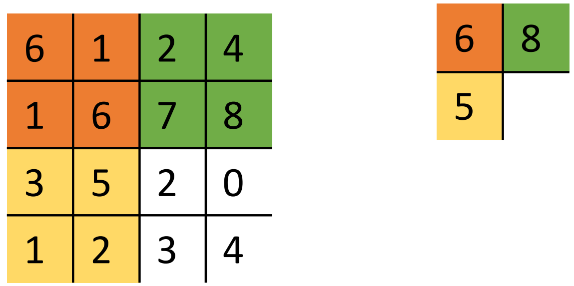


Figure: adapted from Fei Fei et al.

Pooling

- Permutation-invariant aggregation+downsampling (typically max or avg)
- Reduces resolution
- Hierarchical features
- Contributes to approximate shift/deformation invariance

6	1	2	4
1	6	7	8
3	5	2	0
1	2	3	4

6	8
5	4

Figure: adapted from Fei Fei et al.

Pooling

- Applied to each channel separately

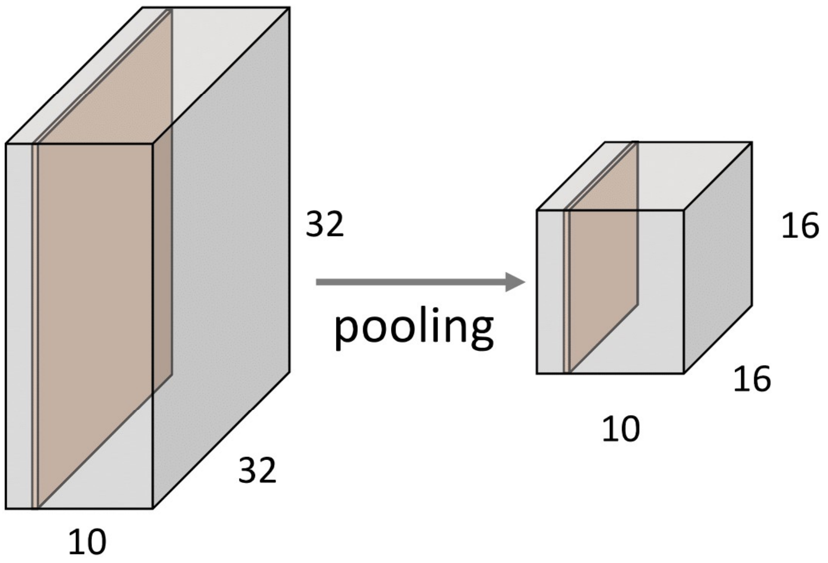
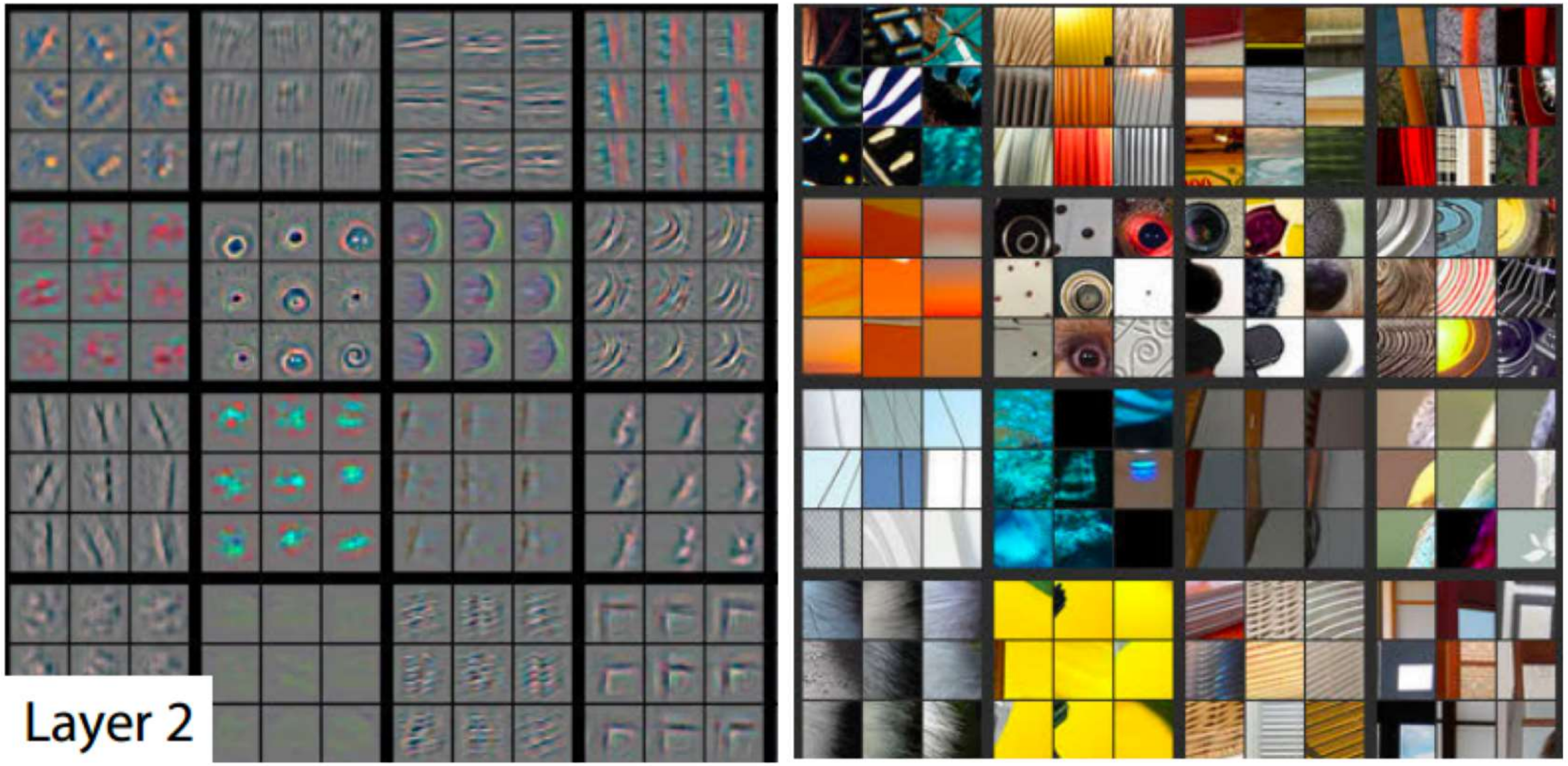
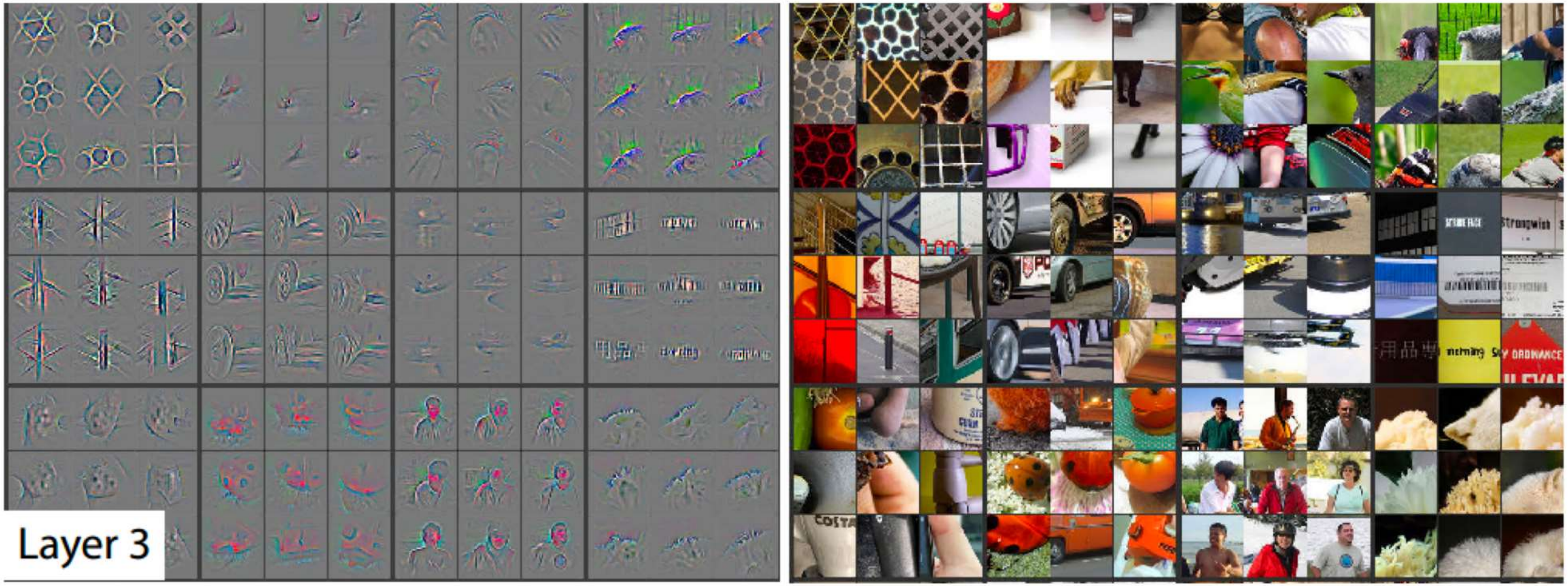


Figure: adapted from Fei Fei et al.

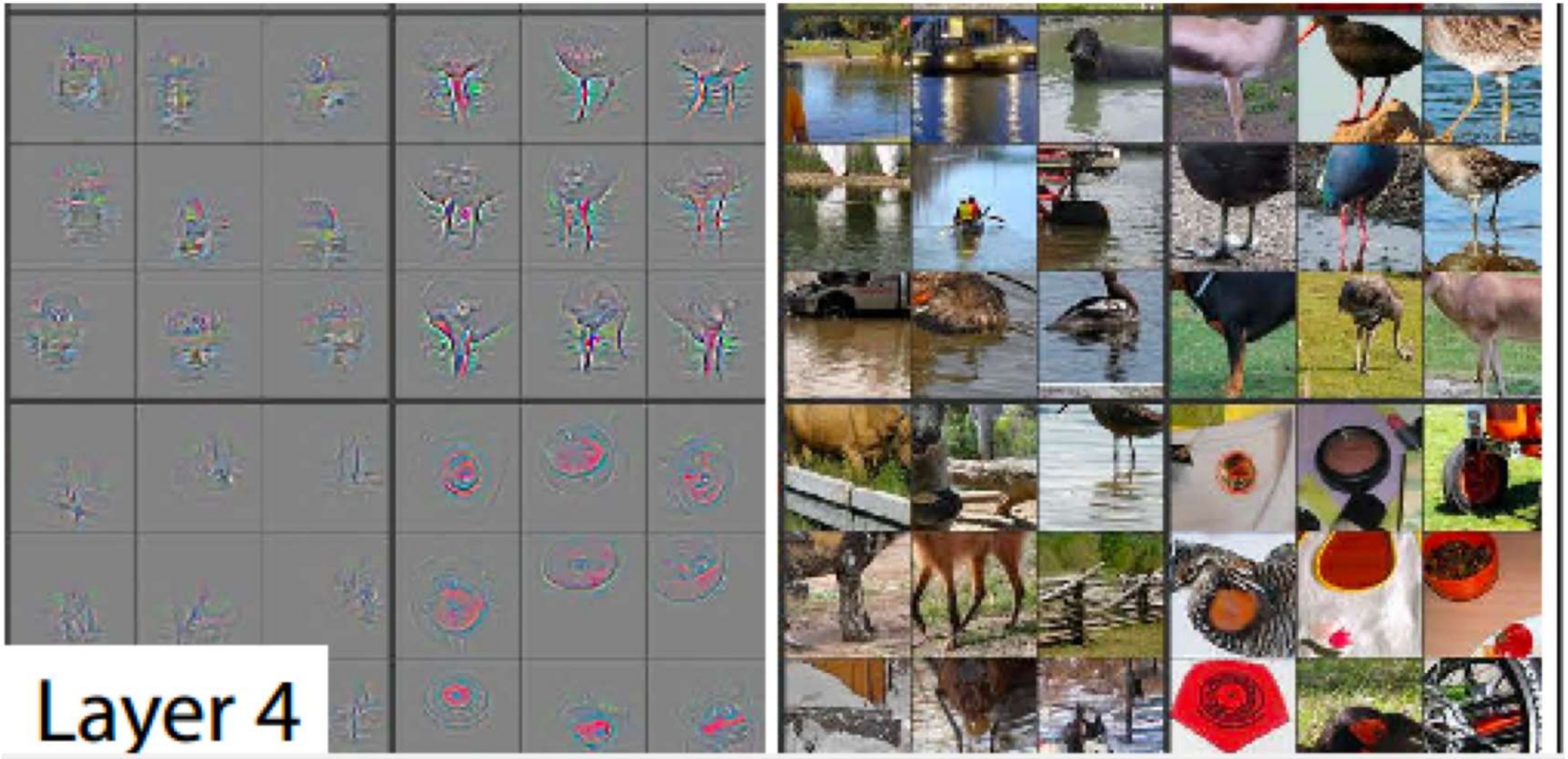
What CNNs learn?



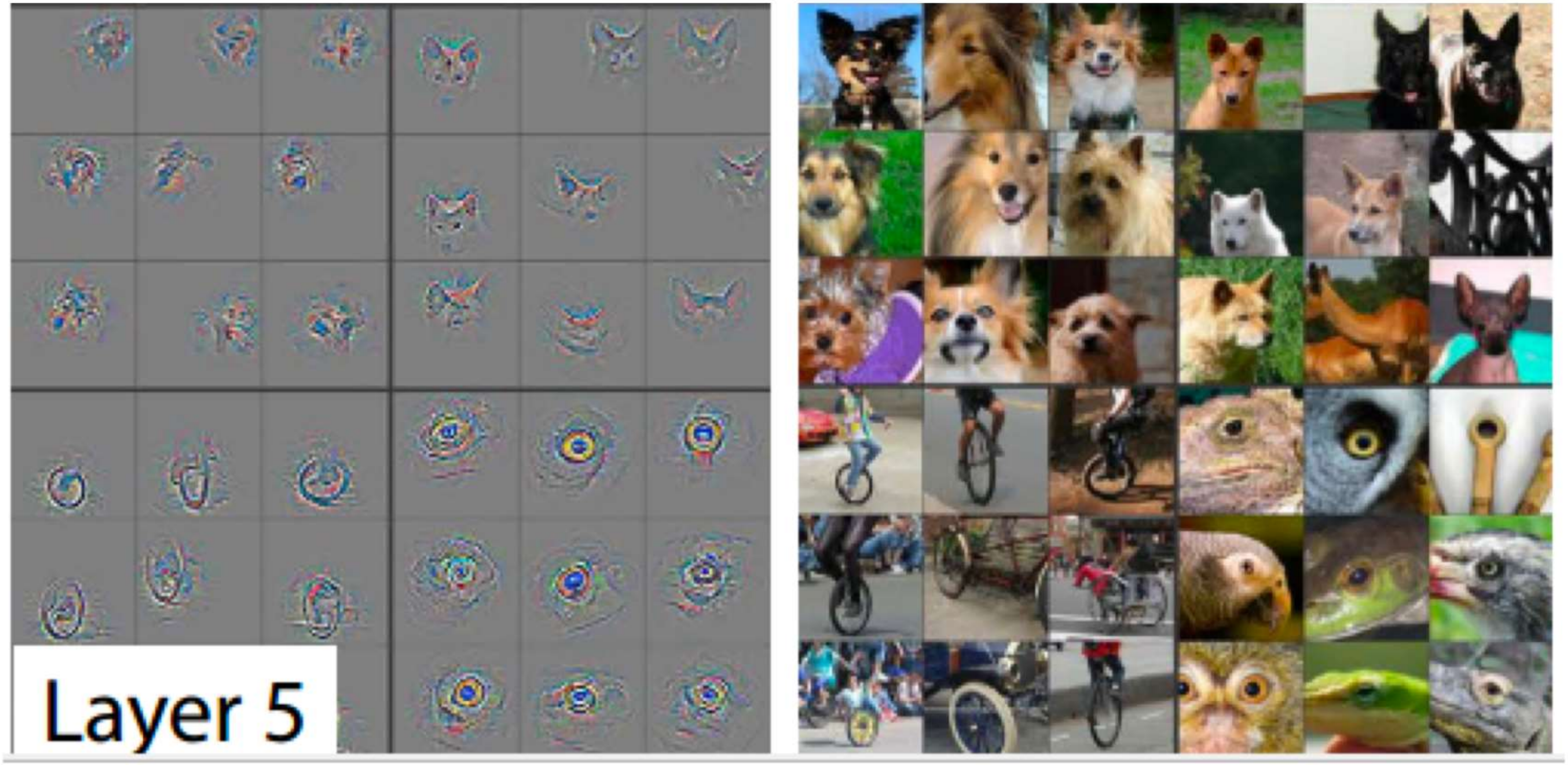
What CNNs learn?



What CNNs learn?

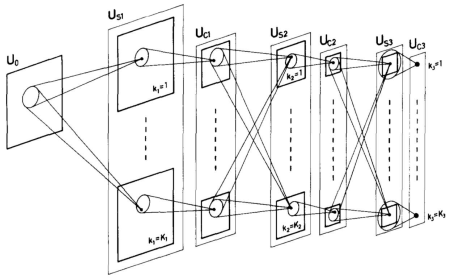


What CNNs learn?



Neocognitron

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position



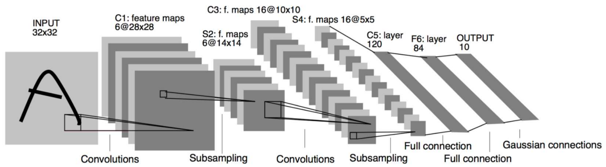
K. Fukushima

Fukushima 1980

Lacks backprop

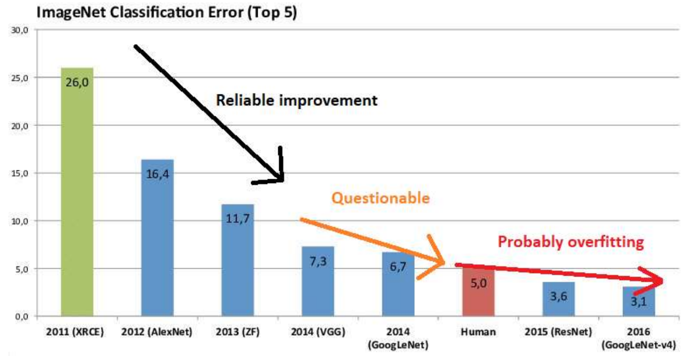
Gradient-Based Learning Applied to Document Recognition

YANN LECUN, MEMBER, IEEE, LÉON BOTTOU, YOSHUA BENGIO, AND PATRICK HAFFNER



LeCun et al. 1998

Adds backprop



Success in 2012!

No GPUs, no success for larger problems…

Most of this initially proposed in 1980 and the 1990s

So what do we learn from this?

- a) convolutions can massively reduce the computational complexity of neural networks but the real power of CNNs is revealed when priors are implemented and for example spatial structure is preserved. This is also one of the reasons why CNNs have been so successful in Computer Vision
- b) CNNs are pipeline of learnable filters interleaved with nonlinear activation functions producing d-dimensional feature maps at every stage. Training works like a common neural network: initialise randomly, present exampled from the training database, update the filter weights through backpropagation by propagating the error back through the network.
- c) convolution and pooling can be used to reduce the dimensionality of the input data until it forms a small enough representation space for either traditional machine learning methods for classification or regression or to steer other networks to for example generate a semantic interpretation like a mask of a particular object in the input.