Lecture 10. Convolutional ANNs

COMP90051 Statistical Machine Learning

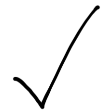
Semester 1, 2021
Trevor Cohn

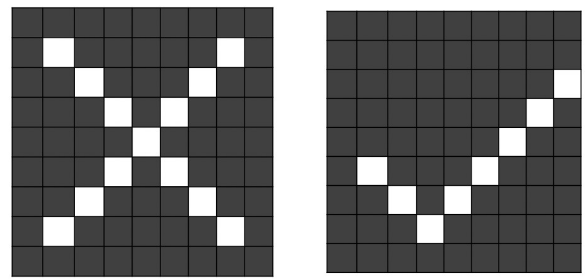
THE UNIVERSITY OF MELBOURNE

This lecture

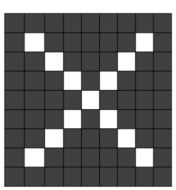
- Convolutional Neural Networks
  - Convolution operator
  - Elements of a convolution-based network
- CNNs in practice
  - LeNet, ResNet

Motivating example

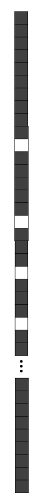
- Image classification  vs 
  - instance is matrix of pixels



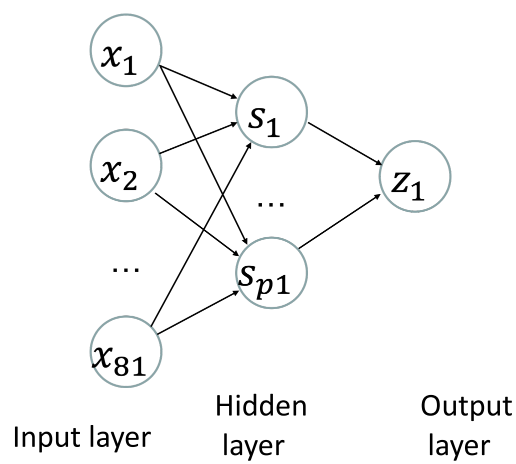
- How can we apply an ANN?
  - flatten into vector, then use fully connected network



9x9



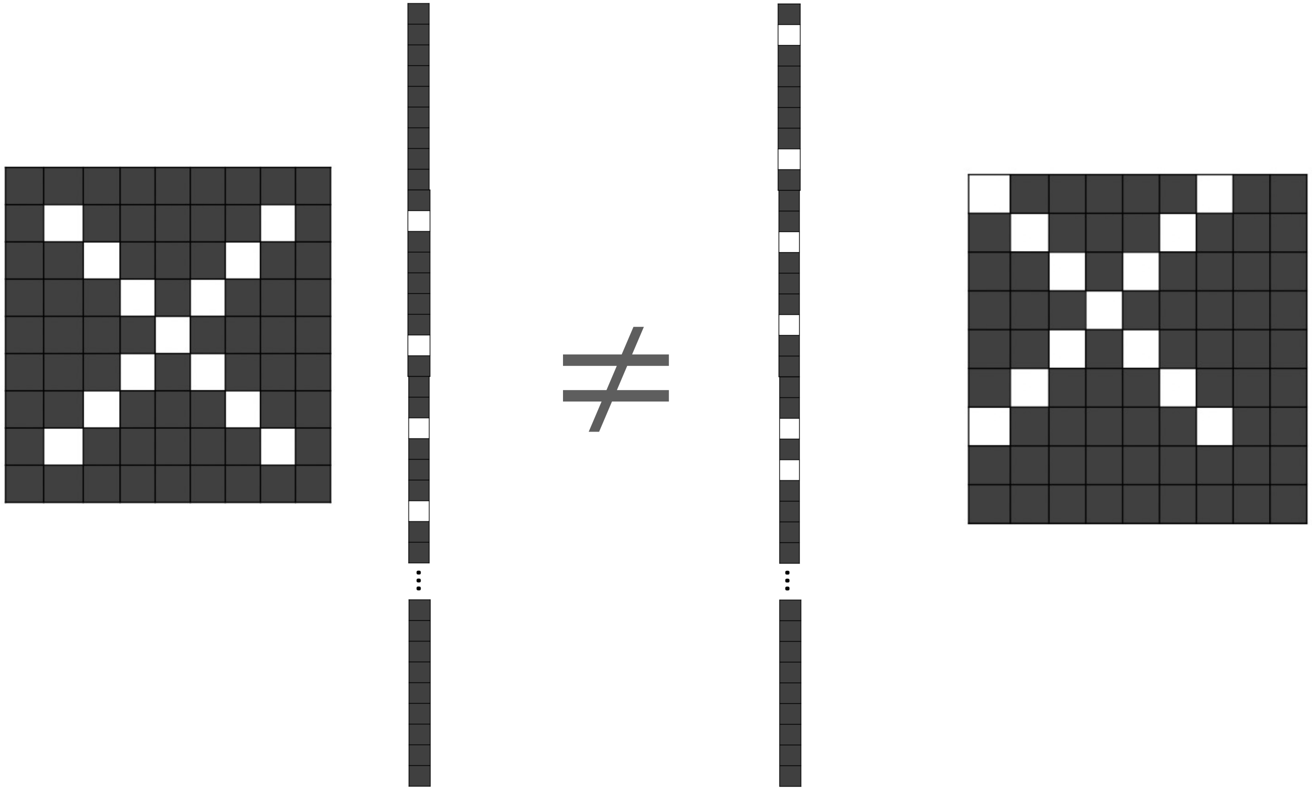
81x1



3

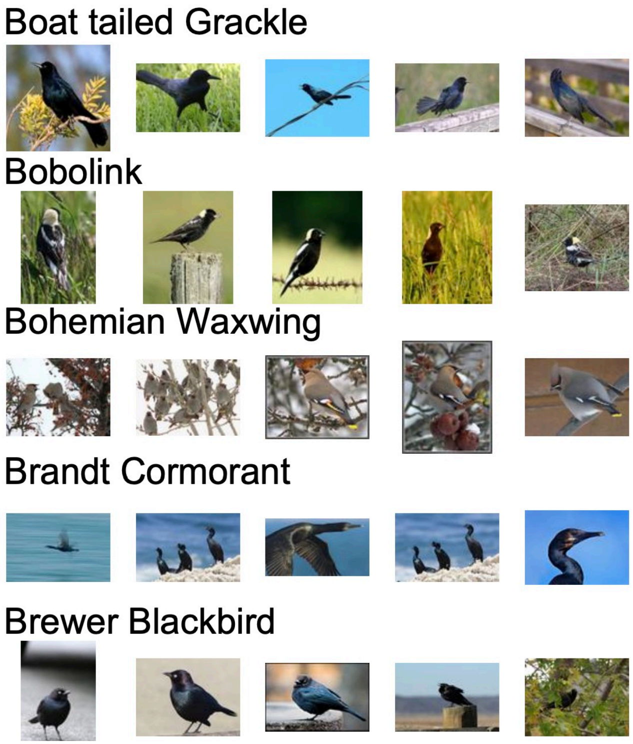
FC-ANN has no spatial invariance

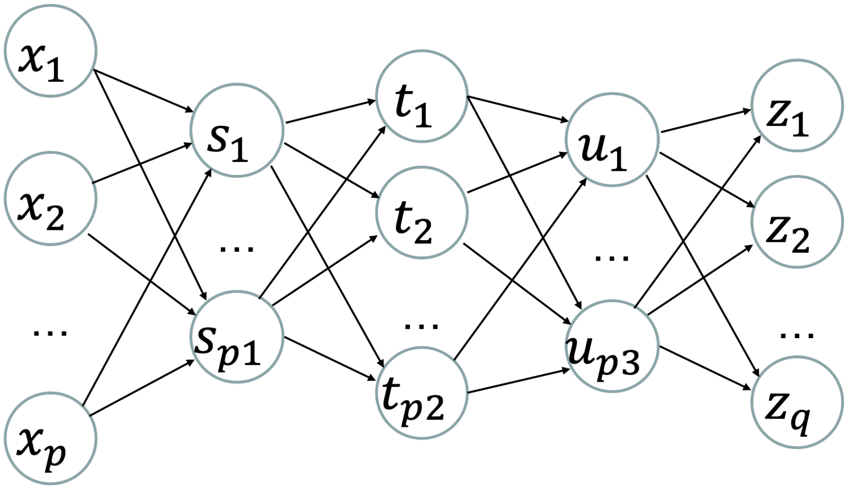
- Disadvantage: must learn the same concept again and again!



Use more depth?

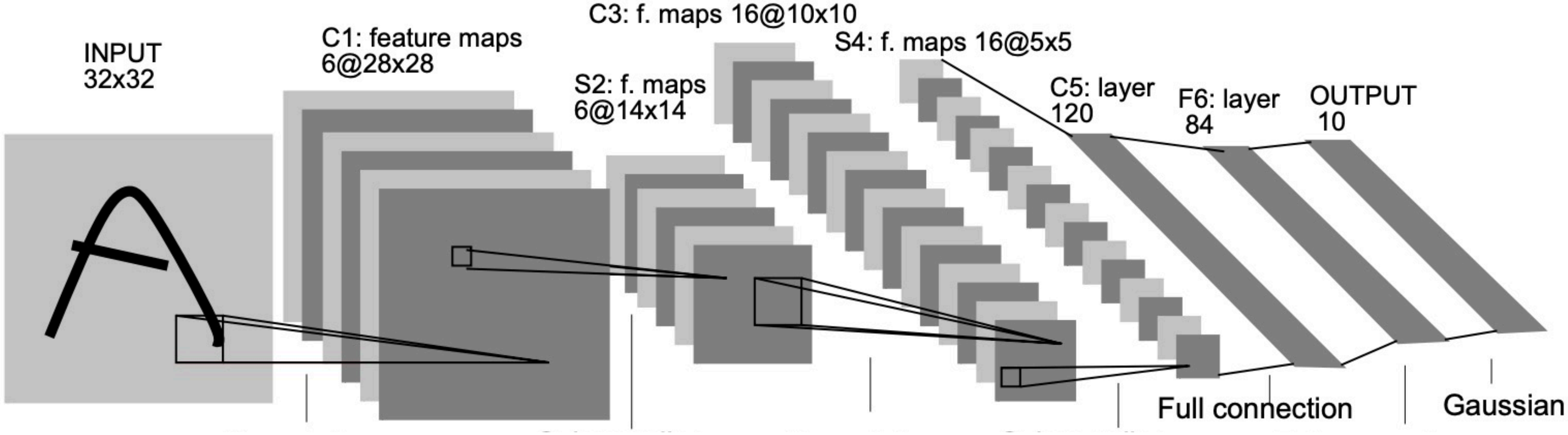
- Inefficient, requires huge numbers of parameters with more hidden layers





Convolutional Neural Network (CNN)

- Key idea is to learn translation invariant filters, a form of parameter sharing



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

Convolutional operators

Based on repeated application of small filters to patches of a 2D image or range of a 1D input

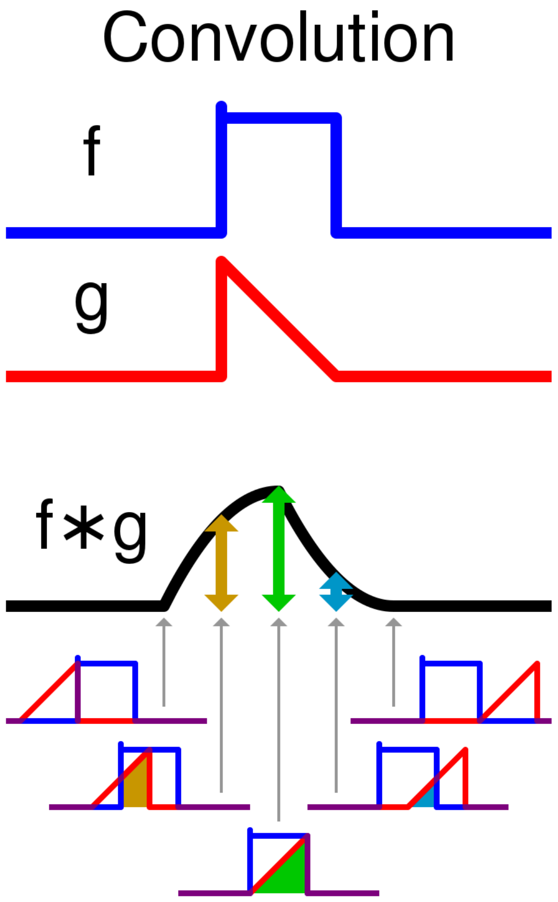
Convolution

- Concept from signal processing, with wide-spread application
  - Defined as

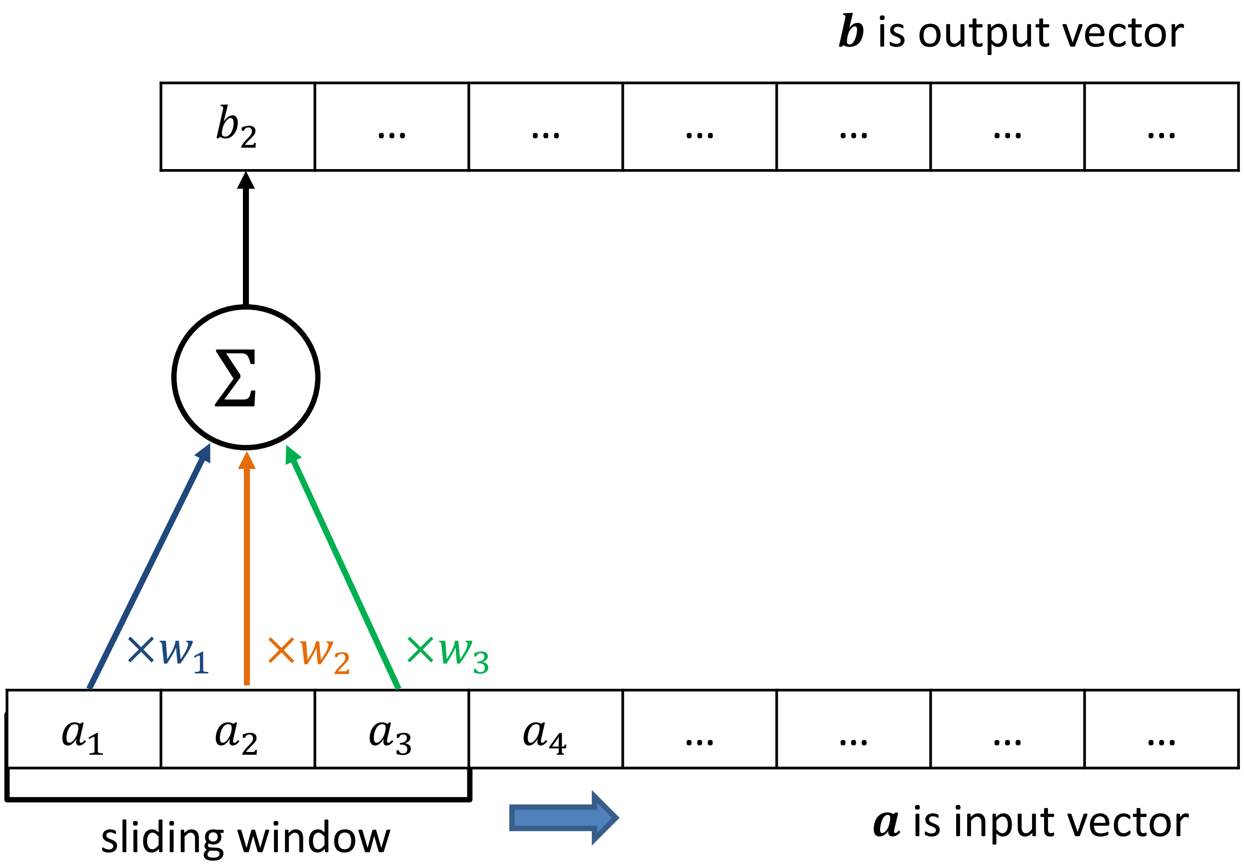
$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

  - Measures how the shape of one function is modified by another

- ConvNets use this idea applied to discrete inputs

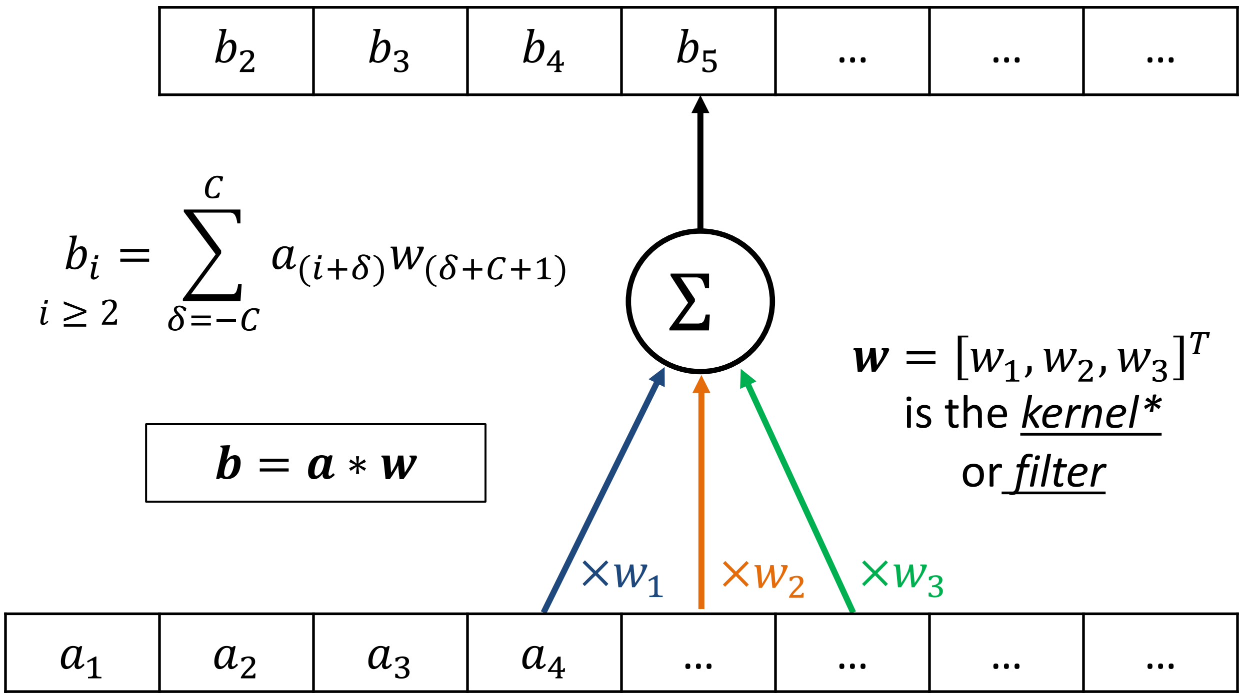


Convolution in 1D



Convolution in 1D

$\boldsymbol{b}$ is output vector



$$b_i = \sum_{\delta=-C}^{C} a_{(i+\delta)} w_{(\delta+C+1)}$$

$i \geq 2$

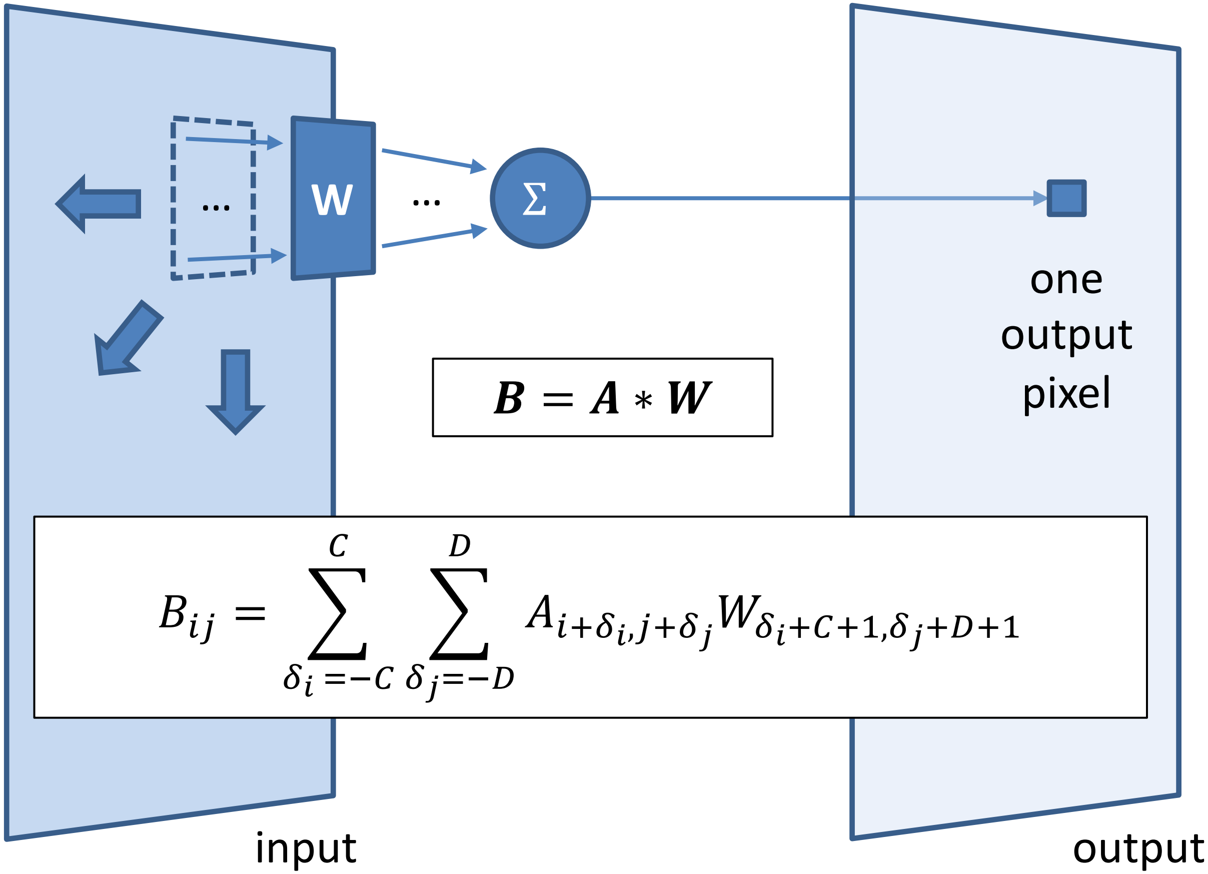
$$\boxed{\boldsymbol{b} = \boldsymbol{a} * \boldsymbol{w}}$$

$\boldsymbol{w} = [w_1, w_2, w_3]^T$
is the *kernel**
or *filter*

$\boldsymbol{a}$ is input vector

*Later in the subject, we will also use an unrelated definition of kernel as a function representing a dot product

Convolution on 2D images



$$\boldsymbol{B} = \boldsymbol{A} * \boldsymbol{W}$$

$$B_{ij} = \sum_{\delta_i = -C}^{C} \sum_{\delta_j = -D}^{D} A_{i+\delta_i, j+\delta_j} W_{\delta_i + C + 1, \delta_j + D + 1}$$

Convolution in 2D

- Use kernel to perform element-wise multiplication and sum for every local patch

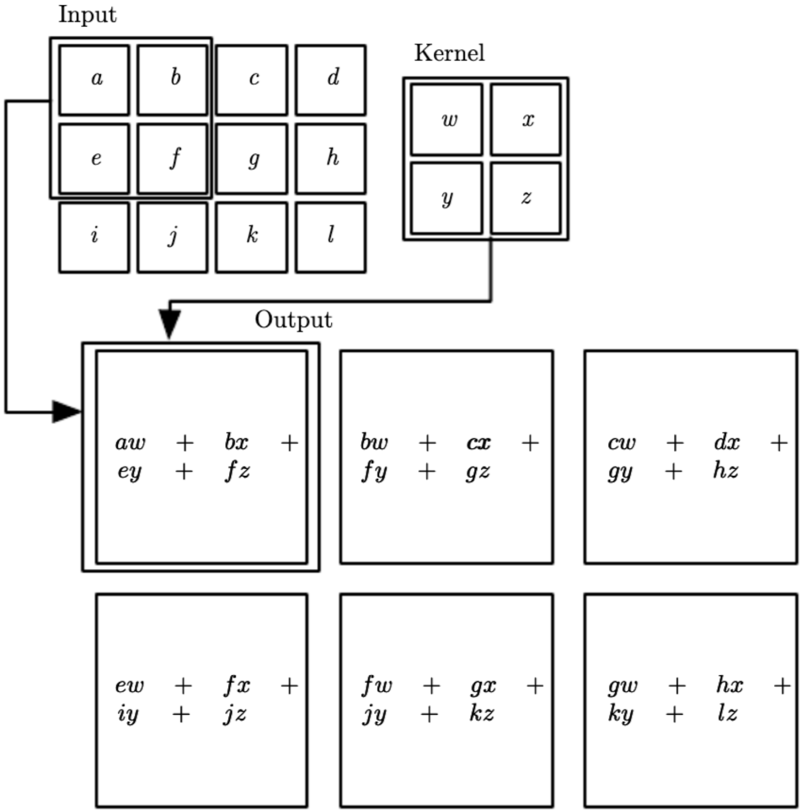
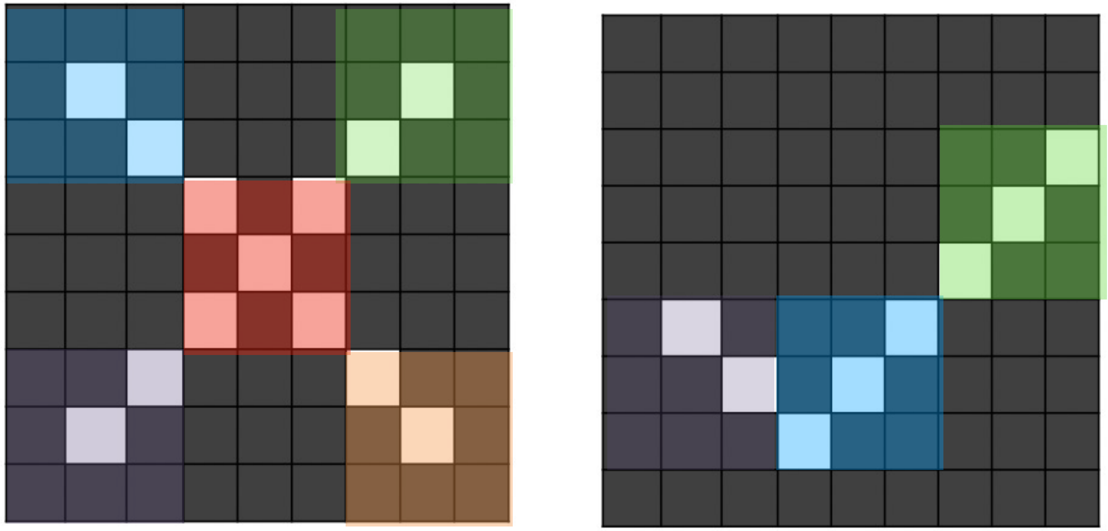


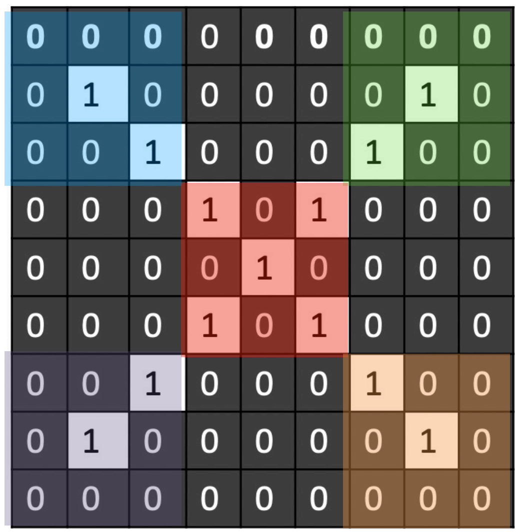
Image decomposes into local patches

- Different local patches include different patterns
  - we can first extract local features (local patterns) and then combine local features for classification



Convolutional filters (aka kernels)

- Filters/kernels can identify different patterns



Element-wise multiplication

$$\text{Sum}\left(\begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix}\odot\begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)=2$$

$$\text{Sum}\left(\begin{bmatrix}0&0&0\\0&1&0\\1&0&0\end{bmatrix}\odot\begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix}\right)=1$$

- When input and kernel have the same pattern: high response

Different kernels identify different patterns

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

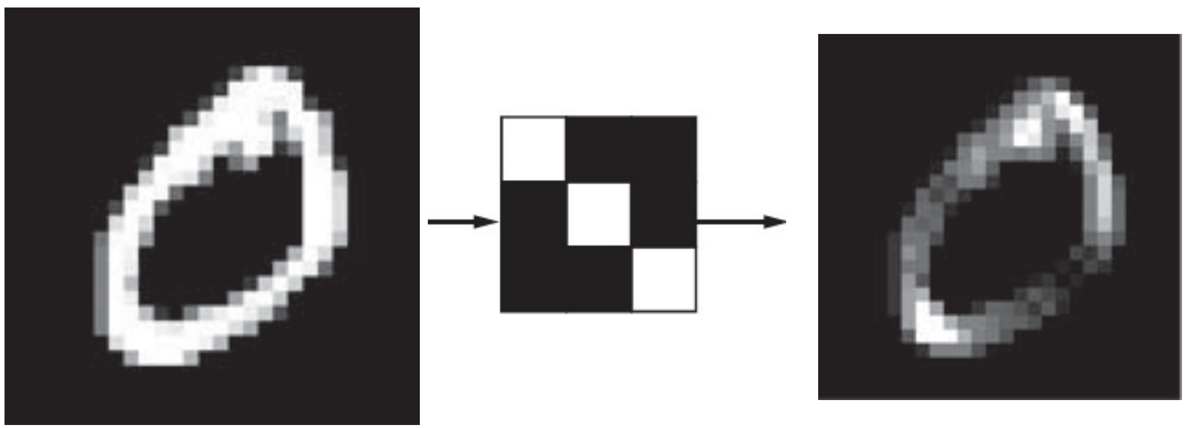
Element-wise multiplication

$$\text{Sum}\left(\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \odot \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} \right) = 2$$

$$\text{Sum}\left(\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} \odot \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} \right) = 5$$

Convolution in 2D example

- Response map (Feature map) for single kernel



Input

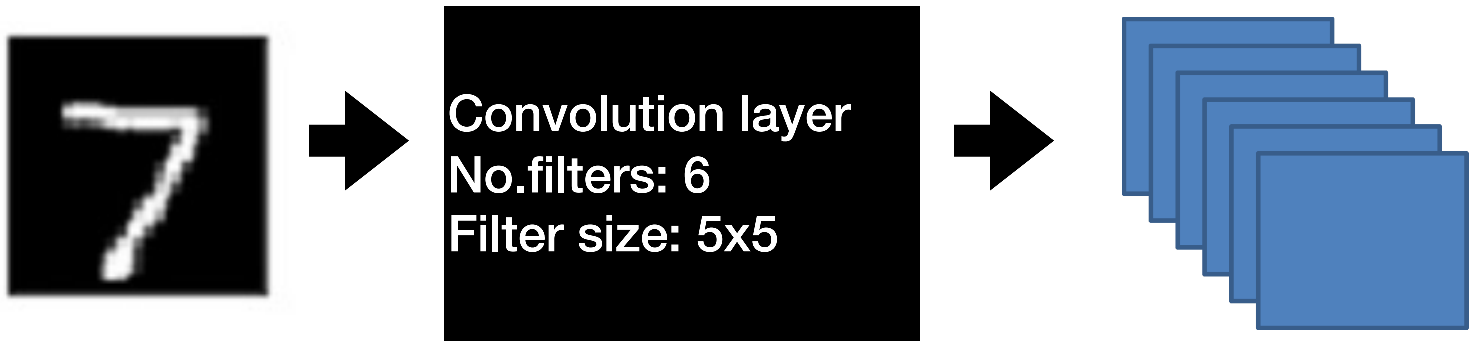
kernel

Feature map: 2D map of the presence of a pattern at different locations in an input

- Different kernels identify different patterns: use several filters in each layer of network

Convolution parameters

- Key parameters in convolution
  - Kernel size: size of the patches
  - Number of filters: depth (channel) of the output
  - Stride: how far to “slide” patch across input
  - Padding of input boundaries



32x32x1

Input: 1 channel

output: 6 channel

Convolution on Multiple-channel input

R

G

B

Kernel: applied to each channel

$* K(Channel\ 1)$

$* K(Channel\ 2)$

$* K(Channel\ 3)$

Element-wise sum

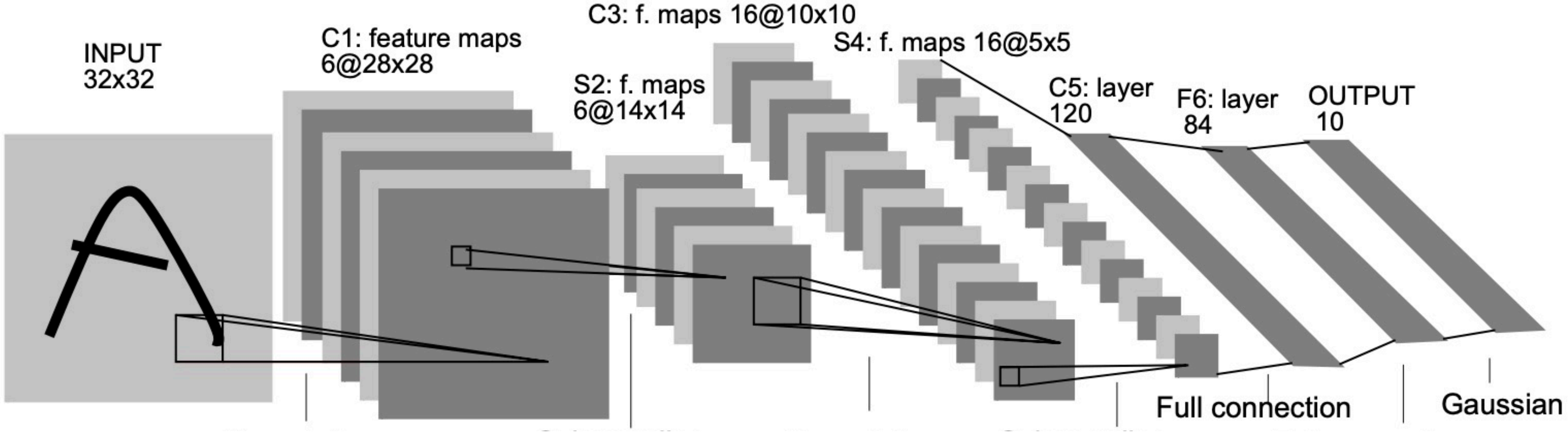
Single channel

Convolutional Neural Networks (CNN)

Deep networks combining convolutional filters, pooling and other techniques

CNN for computer vision

- LeNet-5 sparked modern deep models of vision
  - “C” = convolution, “S” = down-sampling, “F” = fully connected



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

Components of a CNN

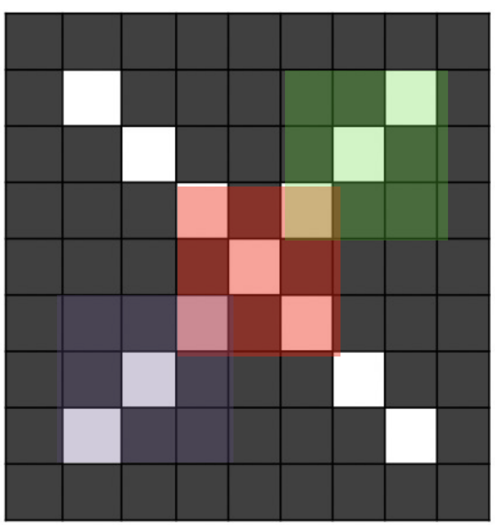
- Convolutional layers
  - Complex input representations based on convolution operation
  - Filter weights are learned from training data
- Downsampling, usually via Max Pooling
  - Re-scales to smaller resolution, limits parameter explosion
- Fully connected parts and output layer
  - Merges representations together

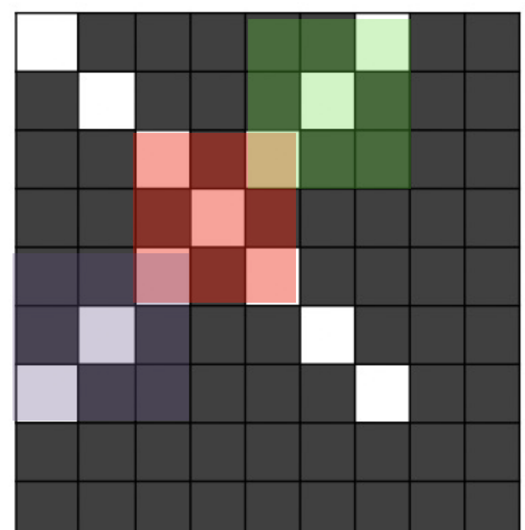
Downsampling via max pooling

- Special type of processing layer. For an $m \times m$ patch
$$v = \max(u_{11}, u_{12}, \dots, u_{mm})$$
- Strictly speaking, not everywhere differentiable. Instead, gradient is defined according to "sub-gradient"
  * Tiny changes in values of u_{ij} that is not max do not change v
  * If u_{ij} is max value, tiny changes in that value change v linearly
  * Use $\frac{\partial v}{\partial u_{ij}} = 1$ if $u_{ij} = v$, and $\frac{\partial v}{\partial u_{ij}} = 0$ otherwise
- Forward pass records maximising element, which is then used in the backward pass during back-propagation

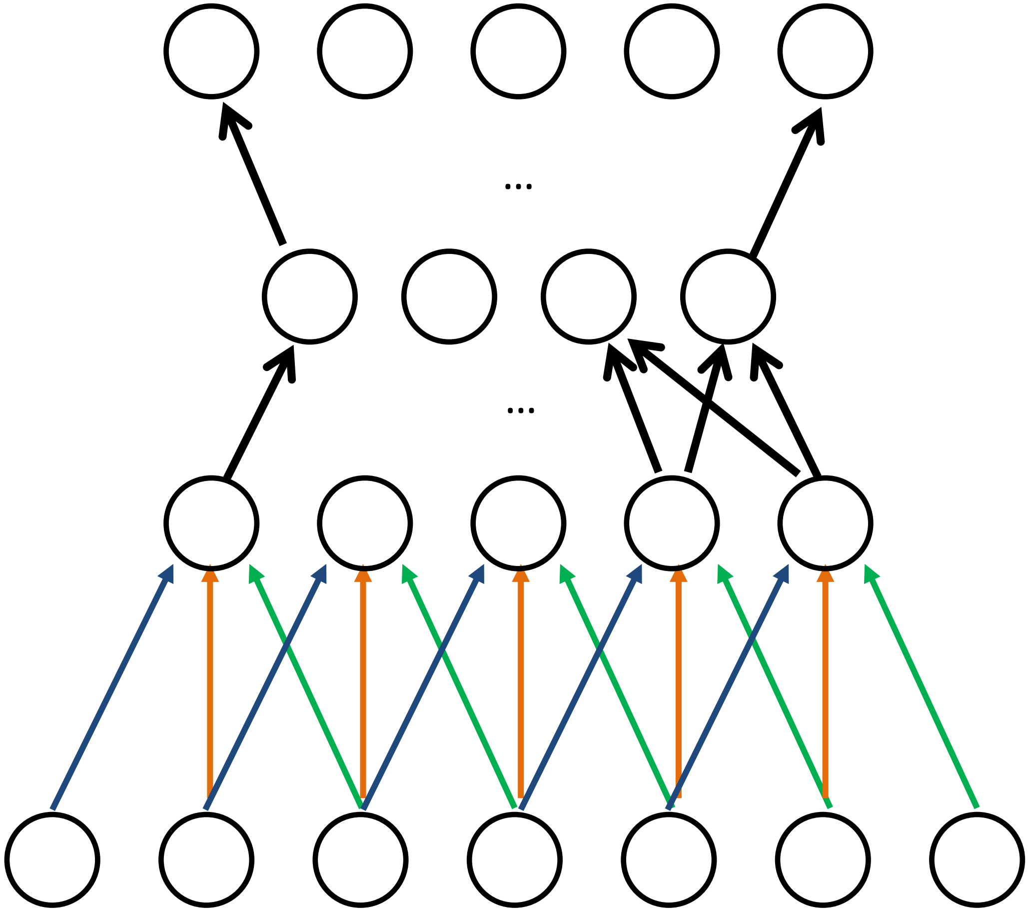
Convolution + Max Pooling → Translation invariance

- Consider shift input image
  - exact same kernels will activate, with same responses
  - max-pooling over the kernel outputs gives same output
  - size of max-pooling patch limits the extent of invariance
- Can include padding around input boundaries





Convolution as a regulariser



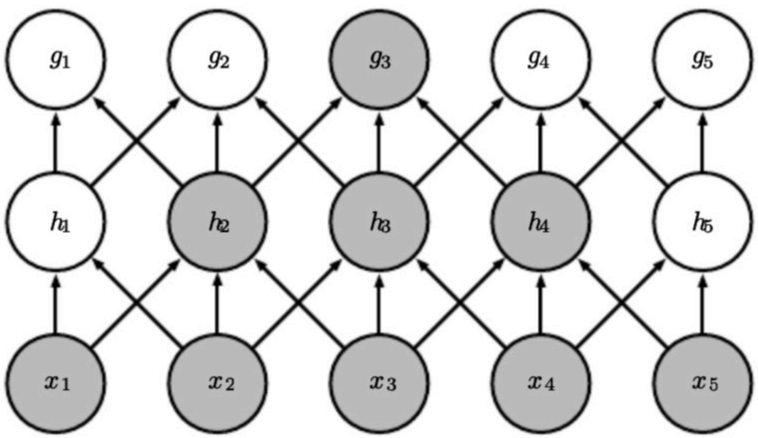
Fully connected, unrestricted

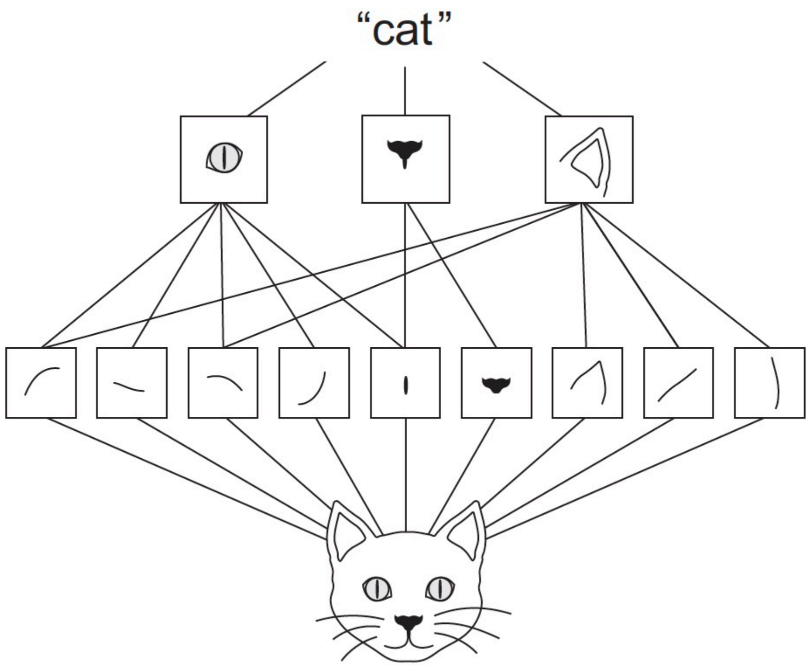
Fully connected, unrestricted

Restriction: same color – same weight

Conv Nets learn hierarchical patterns

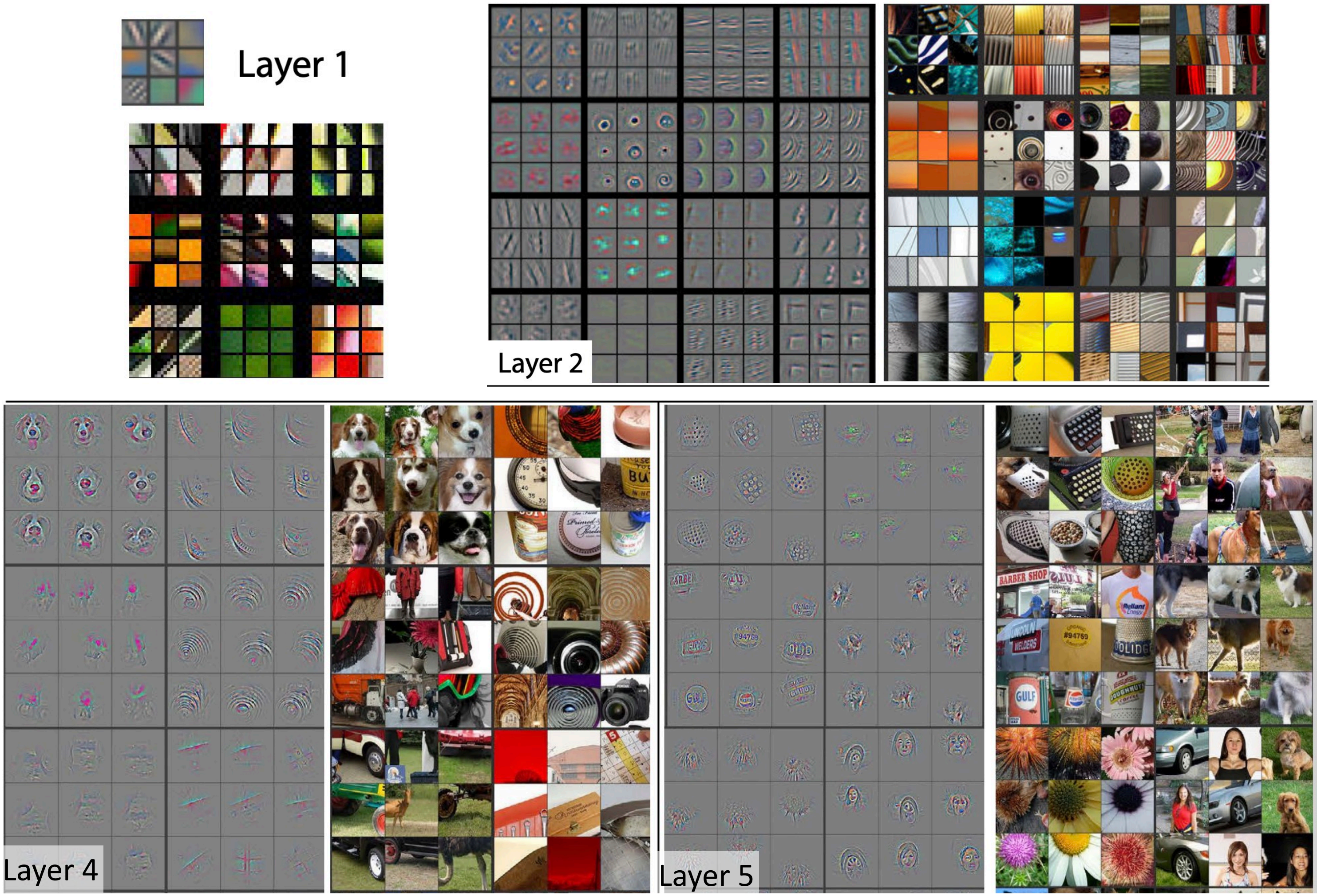
- Stacking several layers of convolution:
  larger size of receptive field (more of input is seen)





Inspecting learned kernels

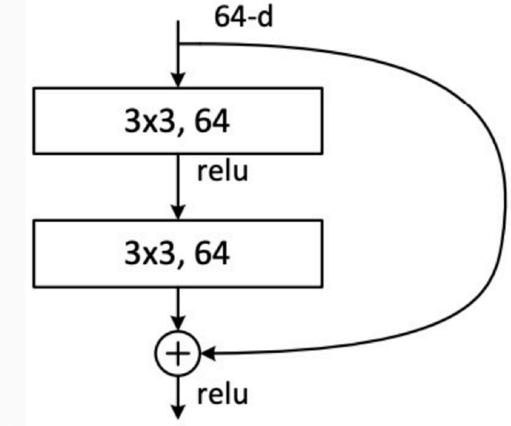
Kernels (grey) and some images that strongly activate each kernel

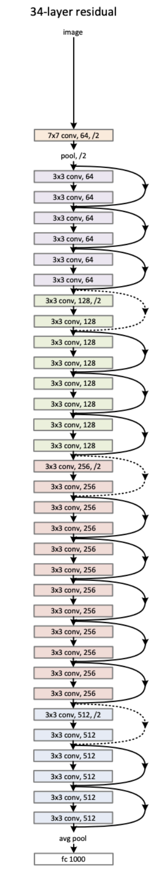


Zeiler, M., and R. Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. 2014

ConvNets in computer vision

- ResNet represents modern state-of-the-art
  - Up to 151 layers (!)
  - mixture of convolutions, pooling, fully connected layers
- Critical innovation is the "residual connection"
  - linear copy of input to output
  - easier to optimise despite depth, solving gradient vanishing problem



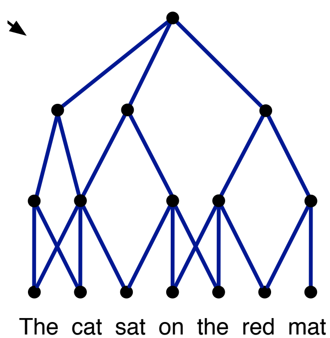


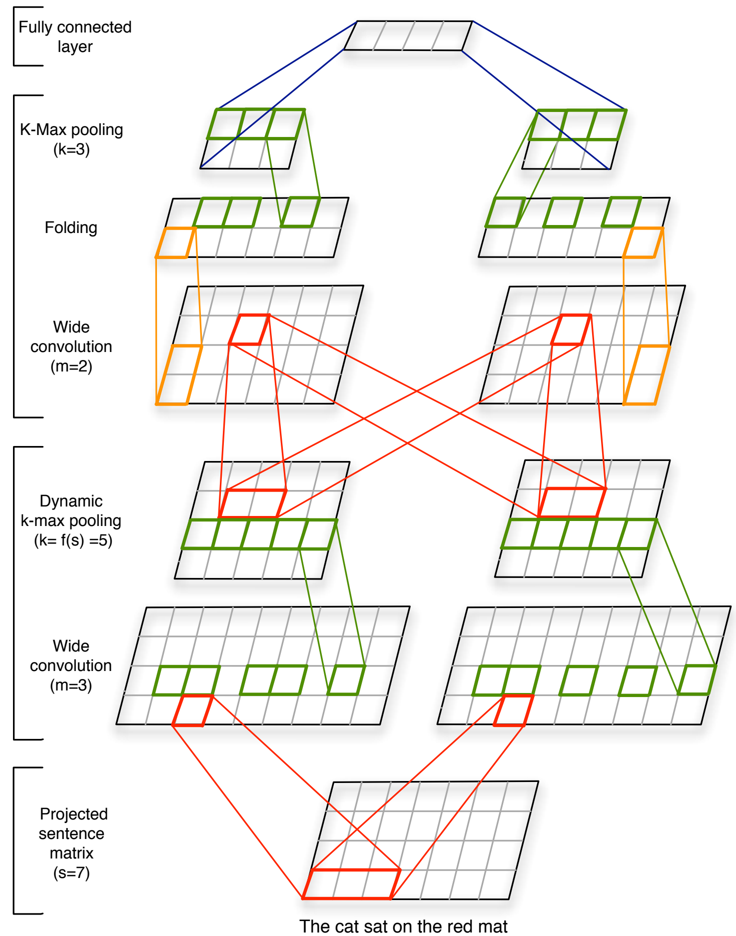
- Standard practise to *pretrain* big model on large dataset, then *fine-tune* (continue training) on small target task

He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. CVPR 2016.

ConvNets for Language

- Structure of text important for classifying documents
  - capture patterns of nearby words using 1d convolutions





Kalchbrenner, N., Grefenstette, E., Blunsom, P. A Convolutional Neural Network for Modelling Sentences. In *ACL 2014* (pp. 212-217).

Tools

- Tensorflow, Torch
  - python / lua toolkits for deep learning
  - symbolic or automatic differentiation
  - GPU support for fast compilation
- Various others
  - Caffe
  - CNTK
  - deeplearning4j ...
- Keras: high-level Python API. Can run on top of TensorFlow, CNTK

This lecture

- Convolutional Neural Networks
  - Convolution operator
  - 1d vs 2d convolutions
  - Elements of a convolution-based network
  - ConvNets in practise for vision & language
- Next lectures: Recurrent Neural Networks (RNNs)