Machine learning, a modern view

- Supervised training data $\mathcal{D}$
- Seen as a sampling from a probability distribution $P(\boldsymbol{x}, y)$
- Test data seen sampled from $P(\boldsymbol{x})$
- $\hat{y} = f(\boldsymbol{x})$, loss function $\mathcal{L}(y, \hat{y})$
- We are trying to minimize the expectation of loss $\mathbb{E}_{x,y}(L)$
- How to do it? Minimize **empirical risk**
  - Minimize the loss on the training data, hope to work at test/deployment time

The shape of $f(\boldsymbol{x})$

- Parameterized family of functions $f(\boldsymbol{x}; \boldsymbol{\theta})$
- Neural network, fully connected layers (aka multilayer perceptron MLB)
- Non-linearities

Training neural networks

- Training neural networks == Optimizing the loss function over the training data

$$\boldsymbol{\theta}^* = argmin_\theta \ \mathcal{L}(\boldsymbol{\theta})$$

- Stochastic gradient descent in the loss function space
- Why it works?
  - Gradient descent works on convex functions
  - Loss surface is **not** convex
  - But it has many identical and or similar minima, and you will likely end up in one of them

Convolutional neural networks

The image domain

- The image domain has special properties
  - Input data arranged as a map, with channels (eg. 3000 x 4000, with red, blue, green as channels)
  - Number of features is very large (36M in this case)
  - Conveniently represented as a multidimensional array, aka **tensor** in machine learning (not the same as tensors in physics)
  - Local relationships between features matter
    - If we flatten the data into a vector, important information is lost.
- Fully connected layers are unpractical in the image domain (too large!)

Problems for the visual domain

- The visual domain creates new type of machine learning problems
- Image classification: output discrete value (cat / dog)
- Image detection: output bounding box of image (x, y, h, w)
- Image segmentation: output a map identifying the different objects.

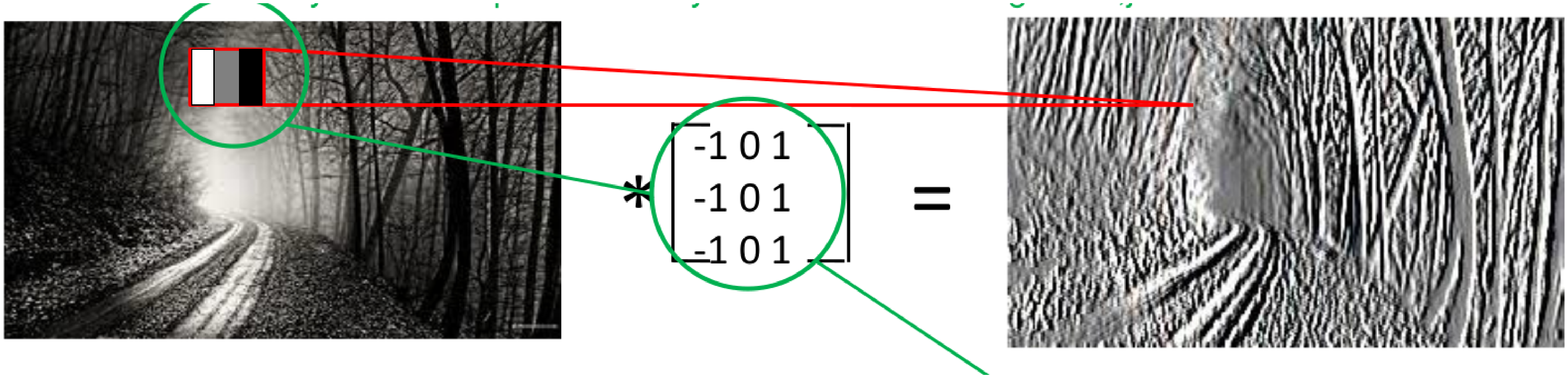
Convolution

- **Discrete convolution**: mathematical operation between two matrices:

$$h(x,y) = (f * g)(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i,j)$$

- In practice:
  - f is the **signal** (large)
  - g is the **convolution kernel** (usually small 3x3, 5x5 etc)
- Convolutions can be used to implement several simple image processing operations
  - Commonly called **filters**

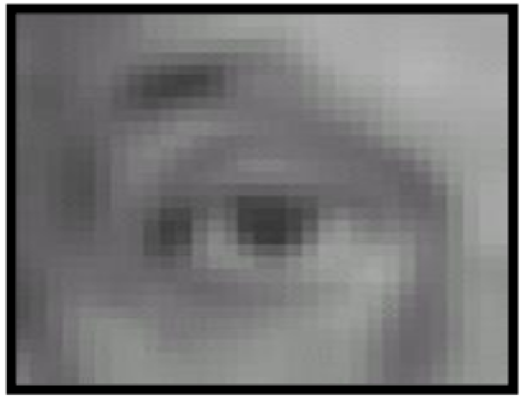
Vertical edge detection with a convolution



Blur with a convolution

Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} =$$



Blur (with a mean filter)

Shift with a convolution

0	0	0
0	0	1
0	0	0

Convolutional Filter Weights

Original

0	0	0
1	0	0
0	0	0

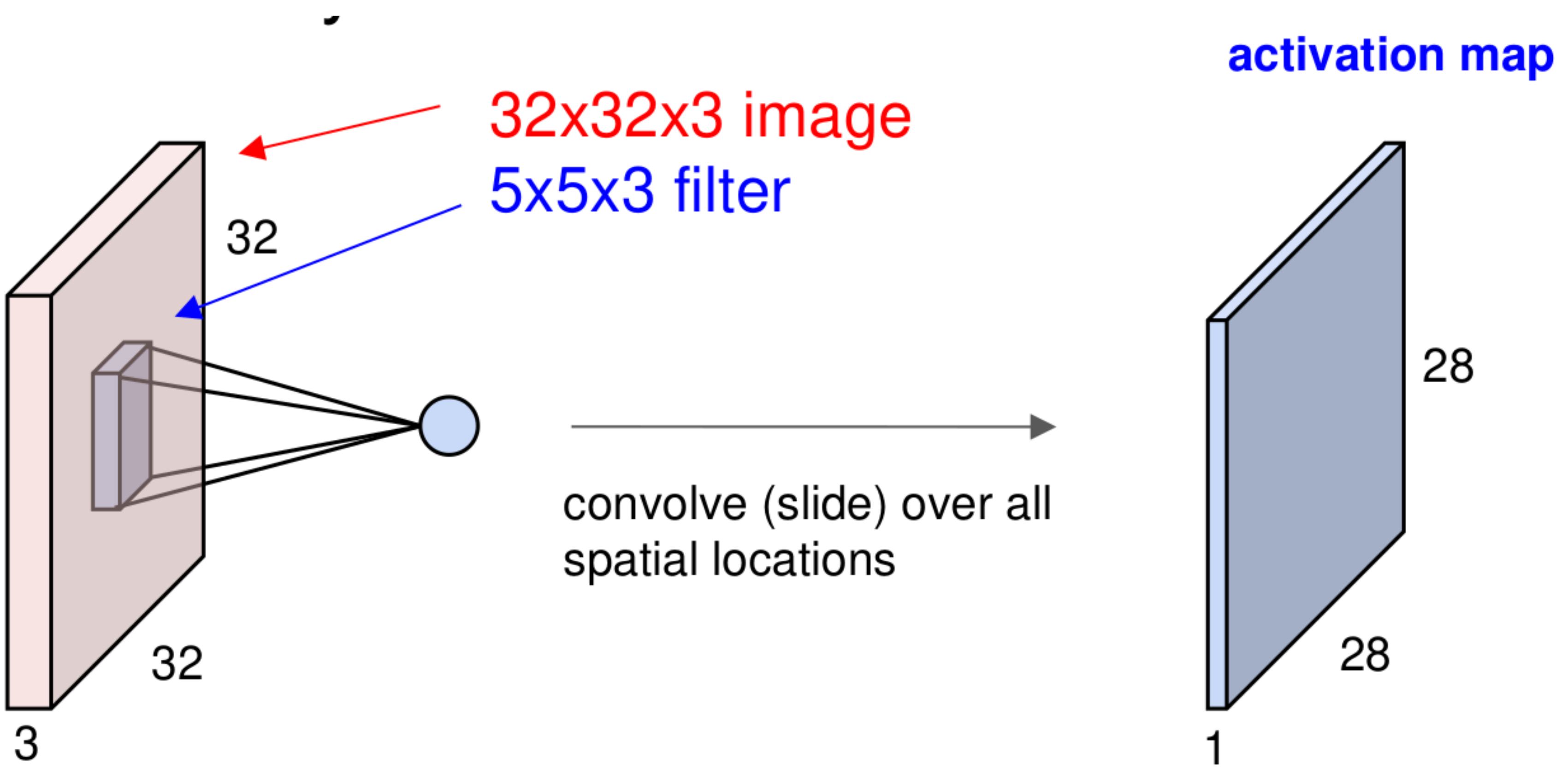
Sliding Mask (Correlation) Weights

Shifted right
By 1 pixel

Convolutional layers in neural networks

- If we flatten the input and the output matrices, we can view a convolution as:
    - a matrix multiplication
    - … with a matrix of a particular sparcity pattern (only non-zero in a neighborhood)
    - … where non-zero weights repeat the same way for each neighborhood
- So a convolution is a peculiar type of fully connected network
    - We could try to learn this…
    - But in practice we just enforce this pattern and use a parameterized convolution as a layer followed by a non-linearity
- As an individual convolution is very specific, we will do a collection of several convolutions

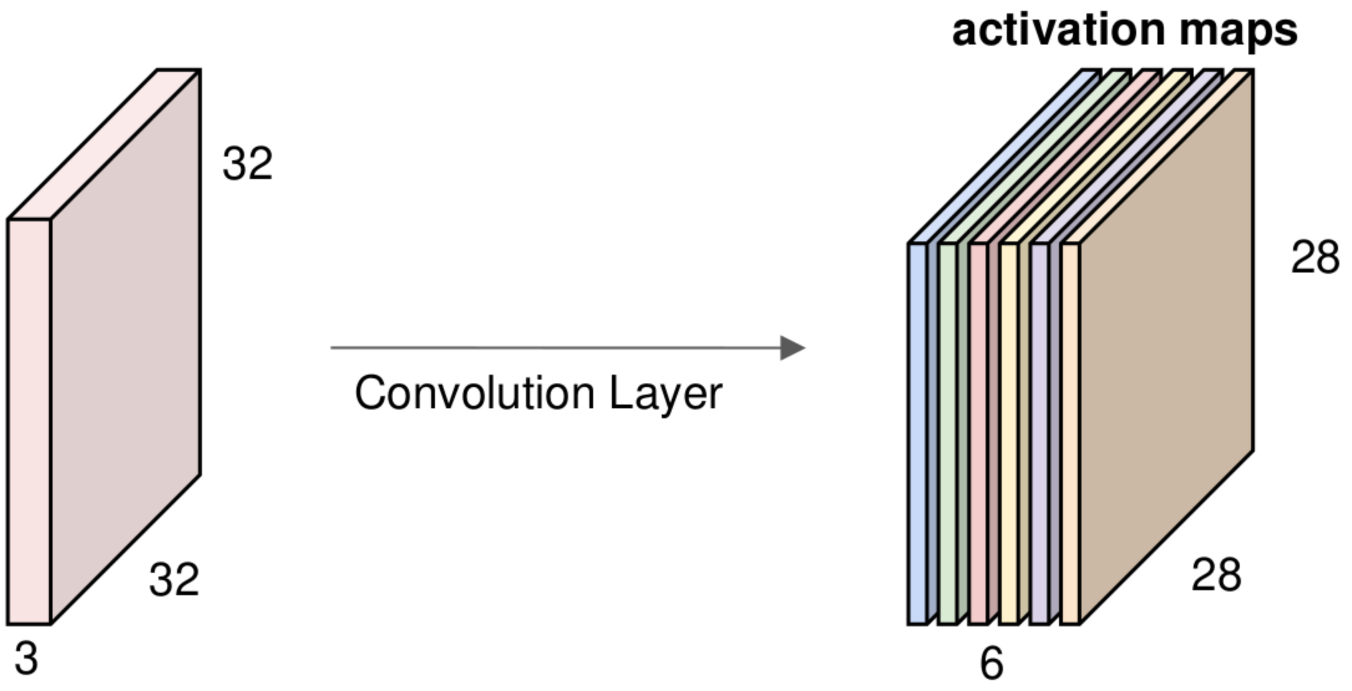
Convolutional layer



Slide based on cs231n by Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolutional layer with multiple filters / activation maps

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

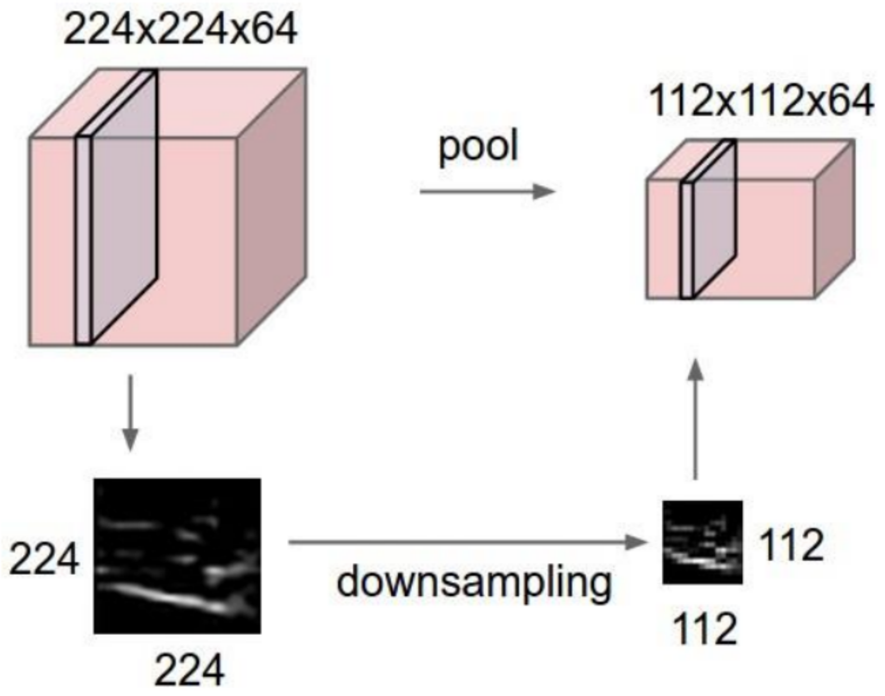


We stack these up to get a "new image" of size 28x28x6!

Slide based on cs231n by Fei-Fei Li & Andrej Karpathy & Justin Johnson

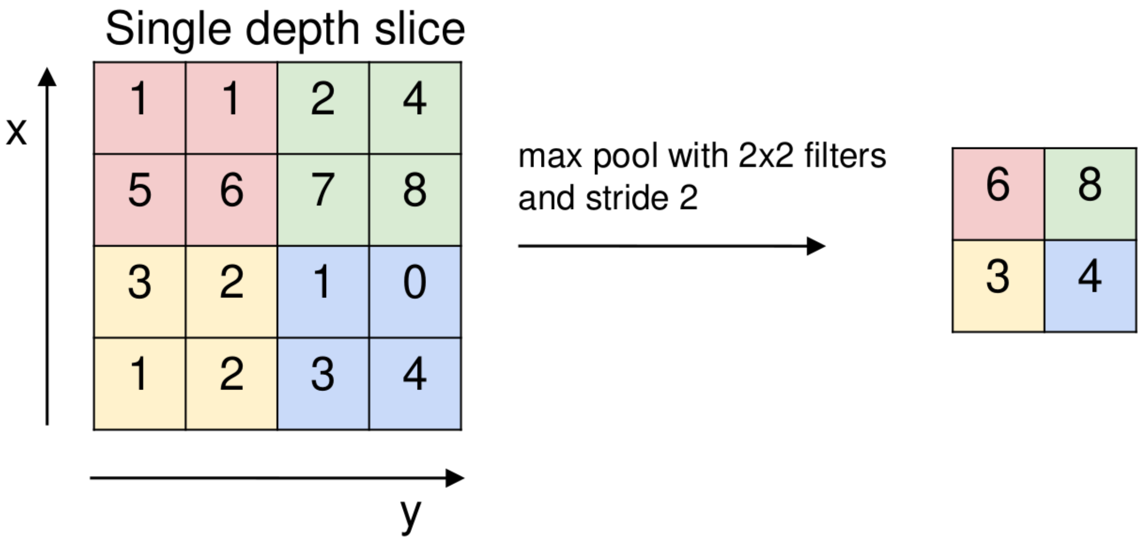
Pooling

- Makes the representations smaller and more manageable
- Operates over each activation map independently



Max Pooling

- Early CNN used averaging pooling
- In later work, it was considered that the output of convolutions show yes/no type presence of certain features - this would be diluted by averaging
- $\rightarrow$ max pooling!



Convolutional neural network

- Standard architecture of a modern convolutional neural network:
  - (Convolution layer + Nonlinearity + Max Pool) repeated several times
  - one, two or three fully connected layer
  - softmax output
- Trained with a cross-entropy or softmax loss

Losses

- Cross-entropy loss (two way classification)

$$\hat{y} = sigmoid(z) = \frac{1}{1 + e^{-z}}$$

$$H(p, q) = -\sum_i p_i \log(q_i) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- Softmax loss (n-way)

$$\hat{y}_i = softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

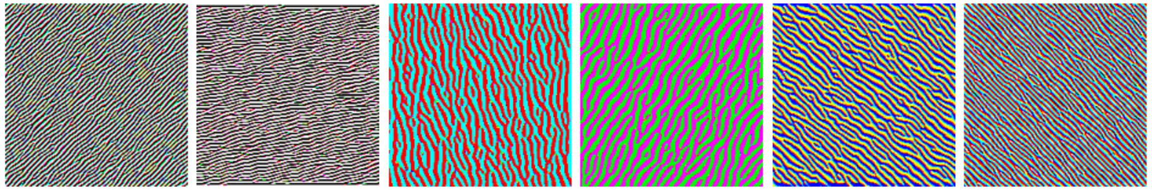
$$\mathcal{L} = -\sum_i y_i \log(\hat{y}_i)$$

What is being learned at the intermediary levels?

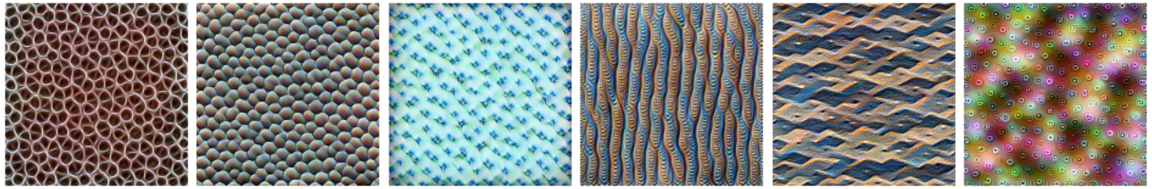
- Features
- These are things that we previously hand-engineered
- Examples from Chris Olah

Feature Visualization

How neural networks build up their understanding of images



Edges (layer conv2d0)



Textures (layer mixed3a)

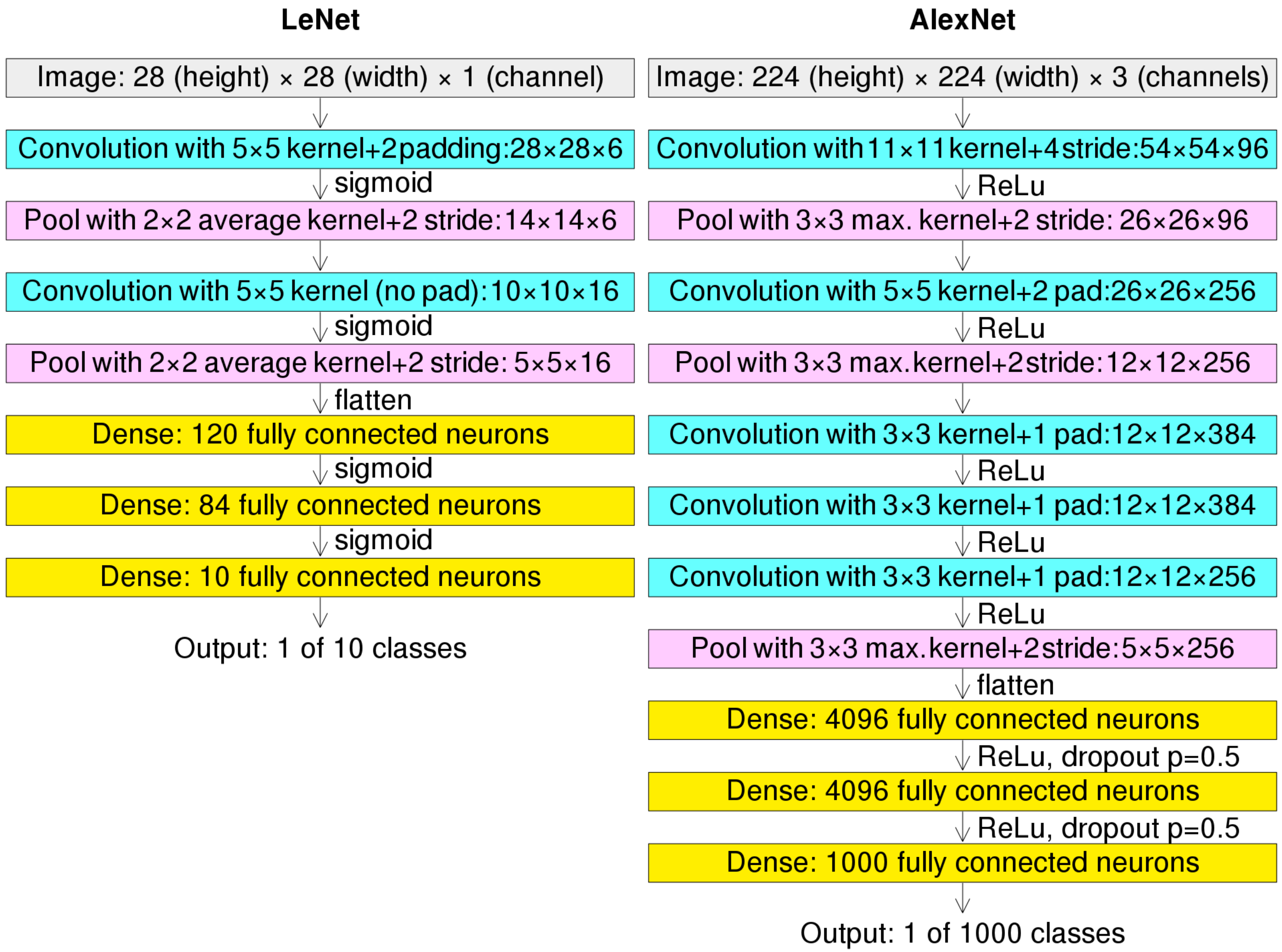
Patterns (layer mixed4a)

Parts (layers mixed4b & mixed4c)

Convnet architectures

- Typical convnets (LeNet, AlexNet)
  - Several blocks of (Convolution + Nonlinearity + Pooling)
  - One fully connected layer at the end with the number of outputs equal to the number of classes n
  - Cross-entropy loss (if $n = 2$) or softmax loss (if $n > 2$)

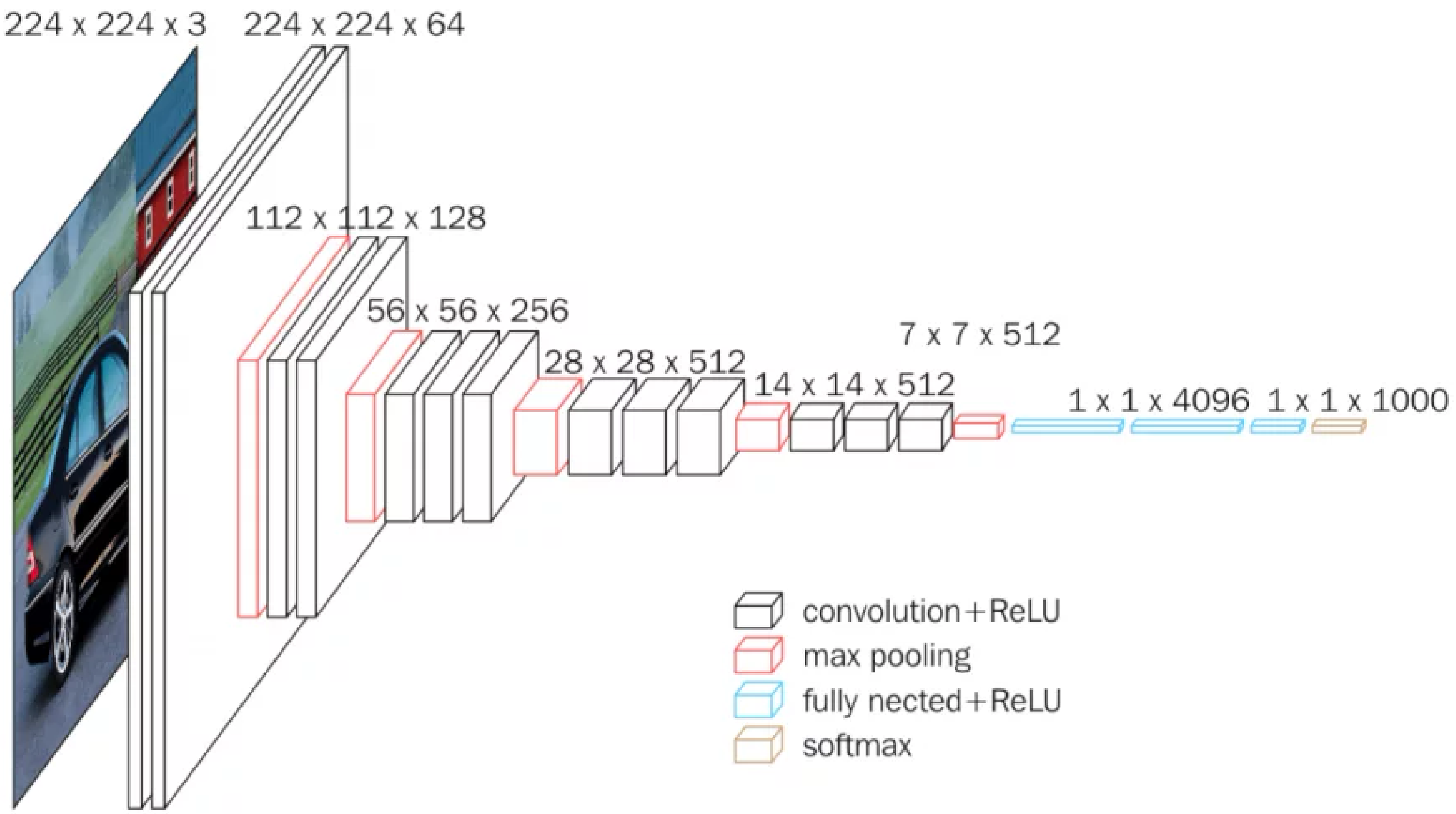
LeNet (1998) vs AlexNet (2012)



VGG series

- VGG - a series of architectures developed at the Visual Geometry Group at University of Oxford.
    - Pretrained versions of 11, 16, 19 layers available

VGG-16


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

Convnet architecture (ResNet)

- A modification of the architecture, where the input is added to the output:

$$y = F(x) + x$$

- Why would you do this?
  - Avoid loosing the input in deep network
- Allows very deep layers networks with 50, 101 and 152 layers

ResNet architecture