Convolutional neural networks

The visual domain

- In the examples up to this point we were talking about datasets with **features**
- Where the individual features were significant numbers in the domain:
  - area of the house
  - ranking of tennis player
- The number of features were managable (eg 1, 2, or 20)
- Visual domain is different!
  - Image: 4000 x 3000 x 3 color channels - 36M features
  - The blue color channel at pixel (3287, 2711) is unlikely to make a lot of difference in classification

Properties of the visual domain

- Map structure
  - Inputs are arranged on map (usually rectangular)
  - Multiple channels (at input: red, blue, green, maybe depth)
  - After the initial processing, there may be other channels
  - Typical representation: a multi-dimensional array (**tensor**)

Invariances

- Location invariance
  - An object has to be detected anywhere in the image
  - Convnets largely solved this
- Rotation invariance
  - An object has to be detected even if rotated
  - Not easy to find a consistent solution.
- Invariance to lighting, obstructions etc.

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.

Fully connected network for visual data

- In principle, a fully connected network would work for visual data as well!
- But the number of parameters is very large.
  - Fully connected layer between a 12MP input and the same size second layer would be: $3 \times 12 \times 10^6 \times 3 \times 12 \times 10^6 = 1.296 \cdot 10^{15}$
- Two difference challenges
  - Memory and computational complexity
  - Difficulty learning that many parameters from so little data

Inductive bias

- The term "inductive bias" is often used in machine learning to denote *choosing architectures in such a way as to fit the problem*
- Assumptions or preconceptions that a model or algorithm makes about the underlying distribution of data.
    - If correct, this might make learning easier
- It is not the same thing as fully engineering the system, this is still a learning system.

Convolutions

- Insert slides about convolutions here

Convolutional layer

- Two ways to think about it:
  - Sparse layer with tied values
    - Sparse layer: parameters are zero except in a neighborhood
    - Tied values: parameters repeat exactly
  - Just forget the neural network story, and think about it as parameterized convolution followed by a nonlinearity

Architecture of convolutional layer

- Insert slides about the convolutions here

Pooling layer

- Max pooling

What happens at the end?

- Add one (or at most two) fully connected layers

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)

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

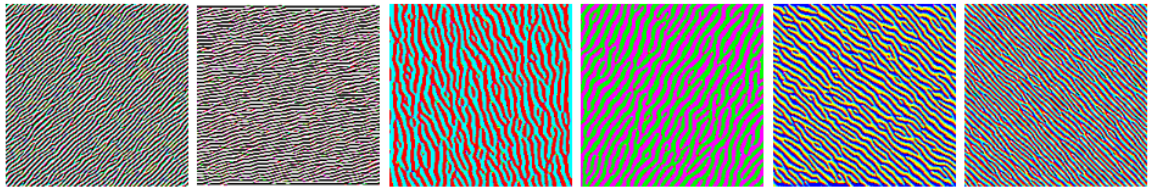
$$\mathcal{L} = -\sum_i y_i \log(\hat{y}_i)$$

What is being learned at the intermediary levels?

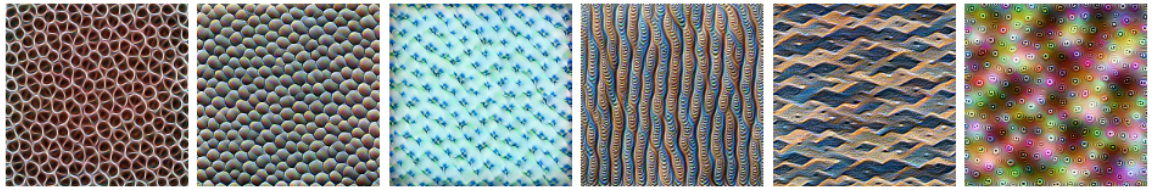
- Features
- These are things that we previously hand-engineered
- Examples from Chris Olah

Feature Visualization

How neural networks build up their understanding of images



Edges (layer conv2d0)



Textures (layer mixed3a)

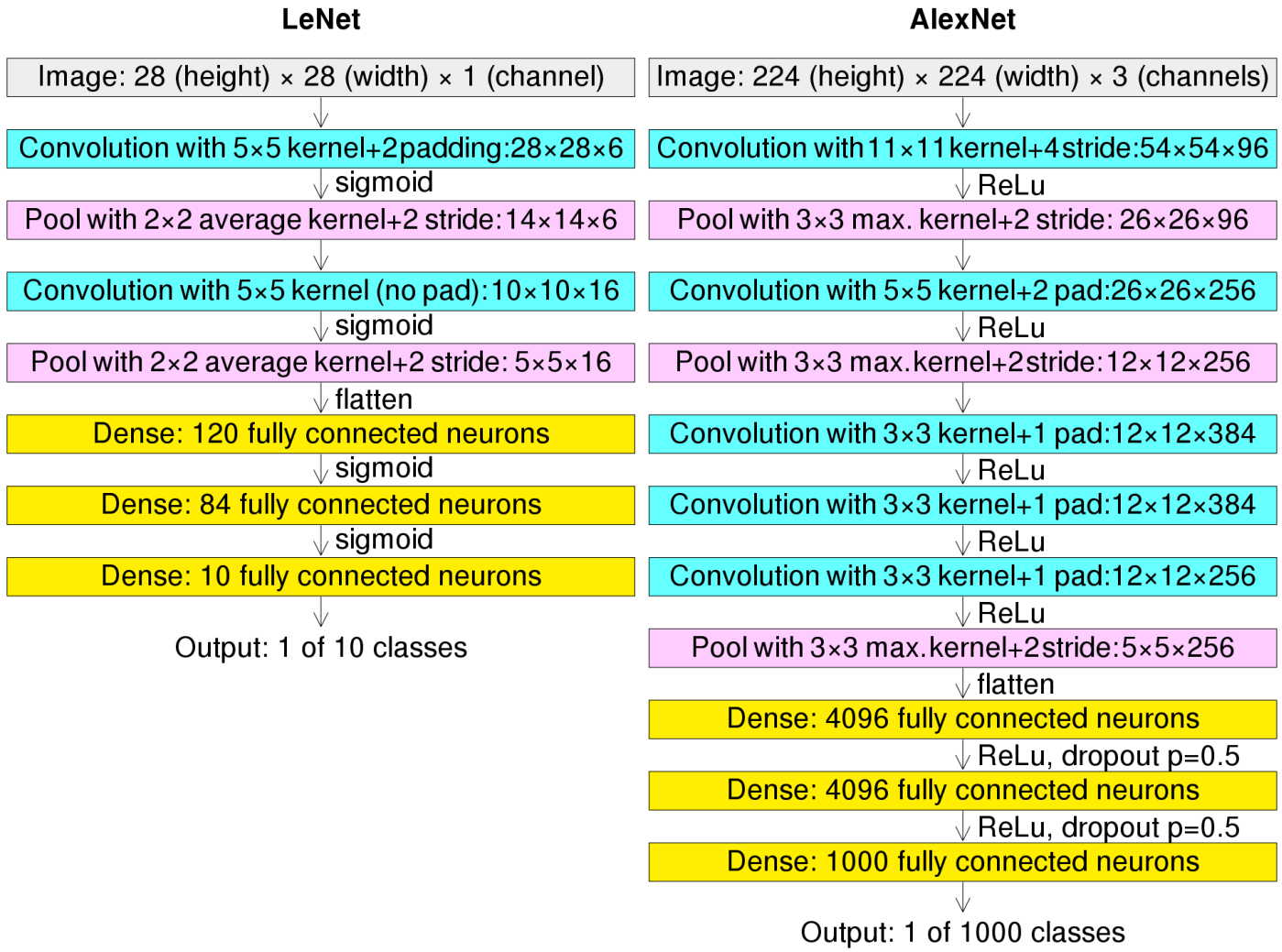
Patterns (layer mixed4a)

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 $n > 2$)

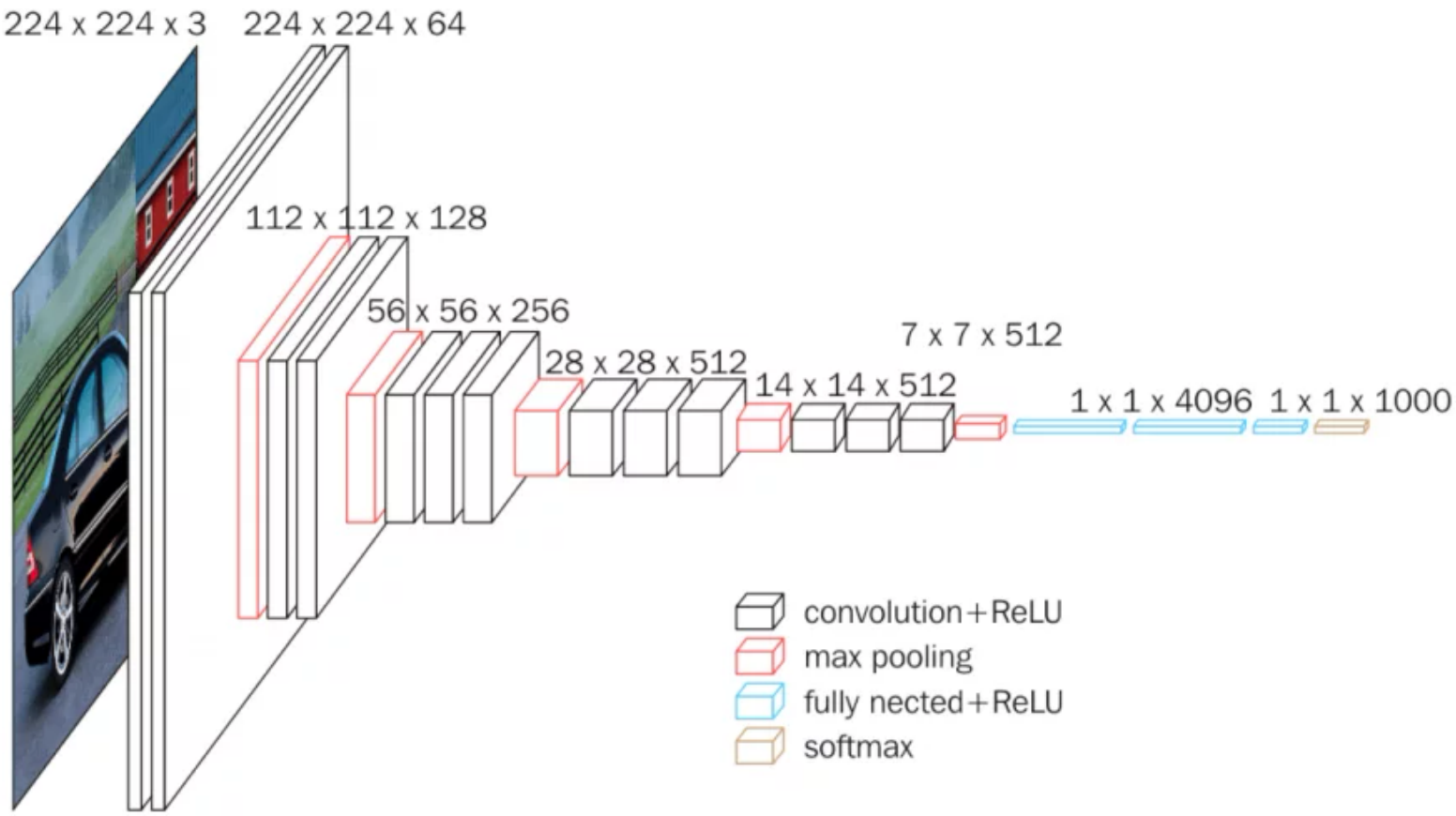
LeNet (1998) vs AlexNet (2012)



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