Machine Learning

CSE 142

Xin (Eric) Wang

Friday, December 3, 2021

Today

- Neural networks and deep learning (cont.)

High-level final exam review

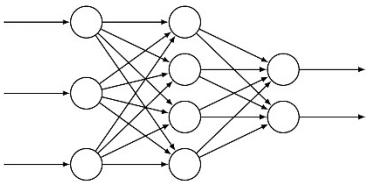
- What is machine learning? What's the basic ML task?
  - To build models that achieve tasks using available features
- Different ML problems
  - Supervised, unsupervised, semi-supervised, reinforcement learning
  - Classification, clustering, regression, dimensionality reduction
- Generalization and over/underfitting
- Geometric, probabilistic, and logical models
- Dimensionality reduction and feature construction
- Training and testing, statistics (FP, N, accuracy, etc.)
  - Contingency table, coverage plot, ROC curve
- Linear regression
- Concept learning, hypothesis space
- Decision tree learning

High-level final exam review (cont.)

- The perceptron model (and dual model)
- Support vector machine (SVM) model
  - Soft margin SVM
- The kernel trick and kernel classifiers
- Clustering, distance metrics
- Naïve Bayes classifier
- Ensemble methods
  - Bagging and boosting
- ML experiments and measuring performance
- Artificial neural networks

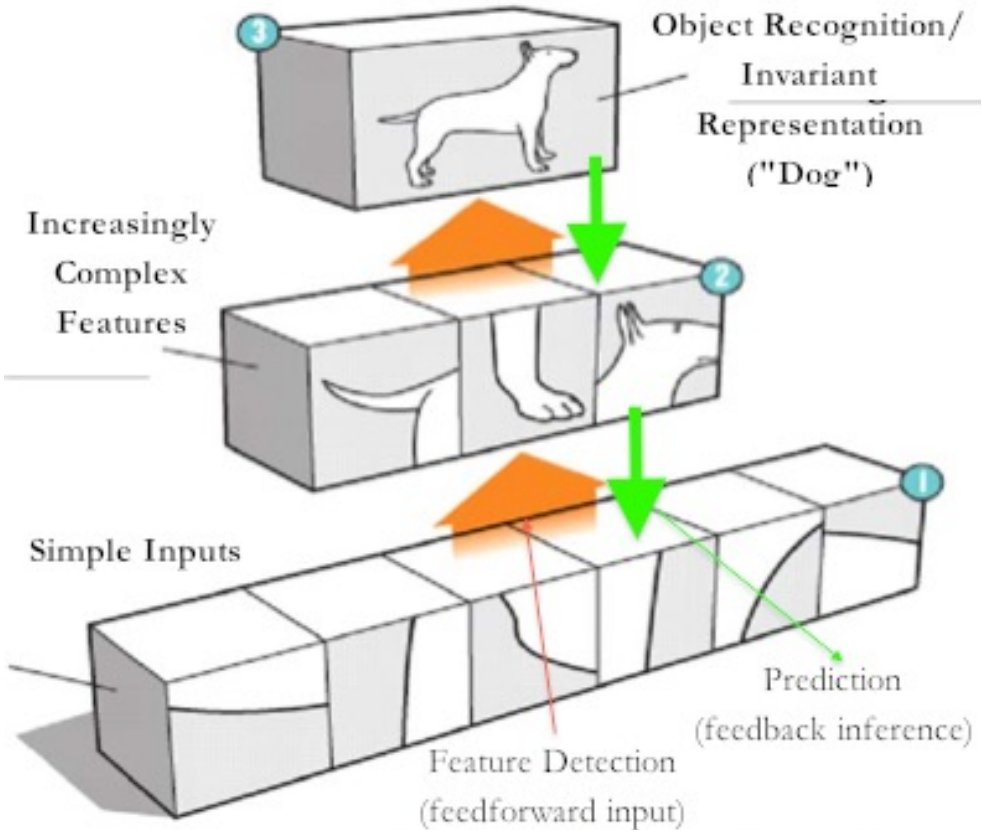
Deep learning

- Deep learning is about learning deep (many-layered) neural networks – multiple non-linear transformations from input to output
- Biological motivation: The human brain is a deep neural network, with many layers of neurons that act as feature detectors, detecting more and more abstract (high-level) features in deeper levels
- E.g., to classify or detect a cat in an image:
  - Bottom layers: Edge detectors, curves, corners straight lines
  - Middle layers: Fur patterns, eyes, ears
  - Higher layers: Body, head, legs
  - Top layer: Cat

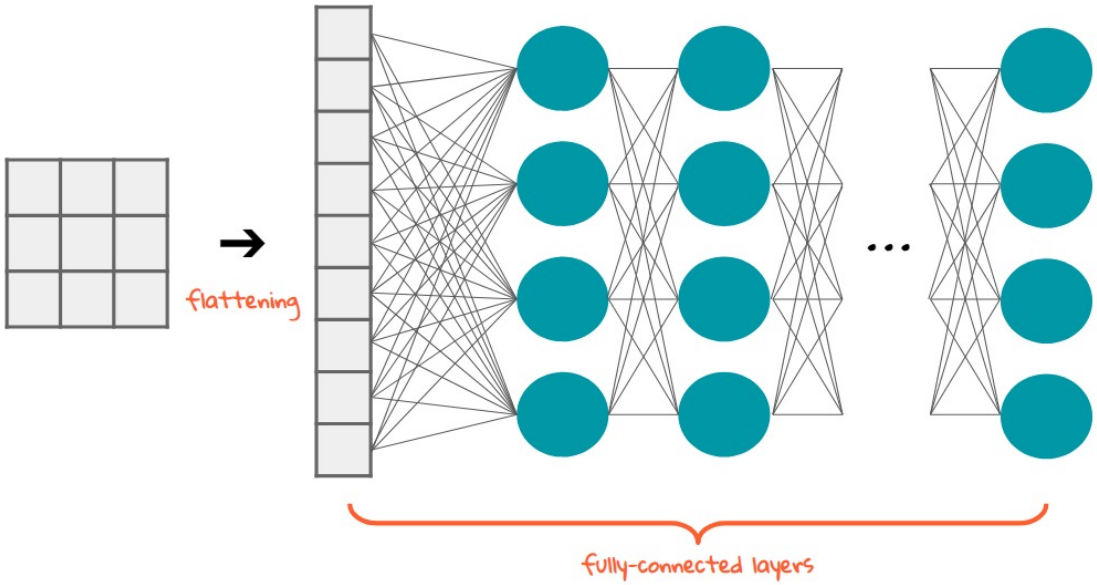


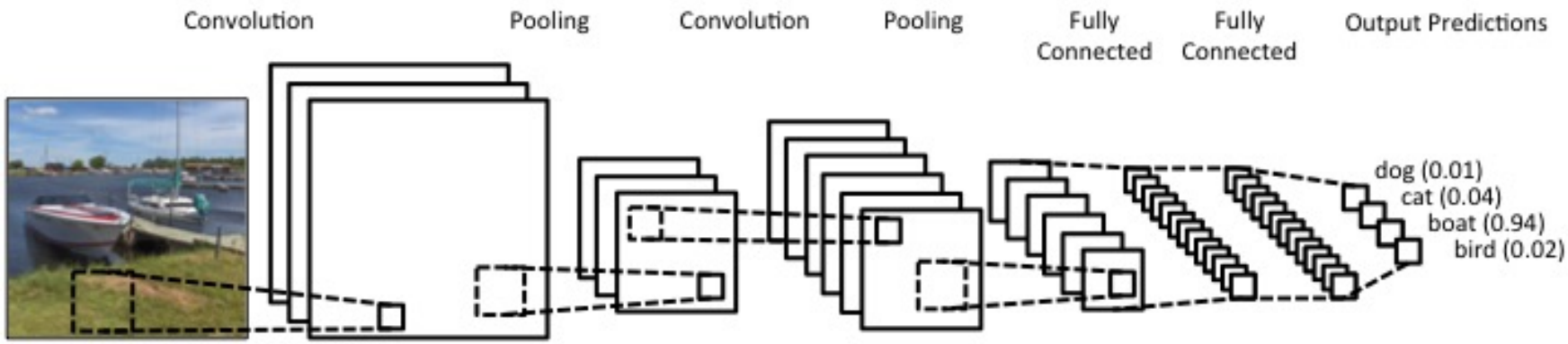
Deep learning

- Each level learns new (more complex or abstract) features from combinations of features from the level below

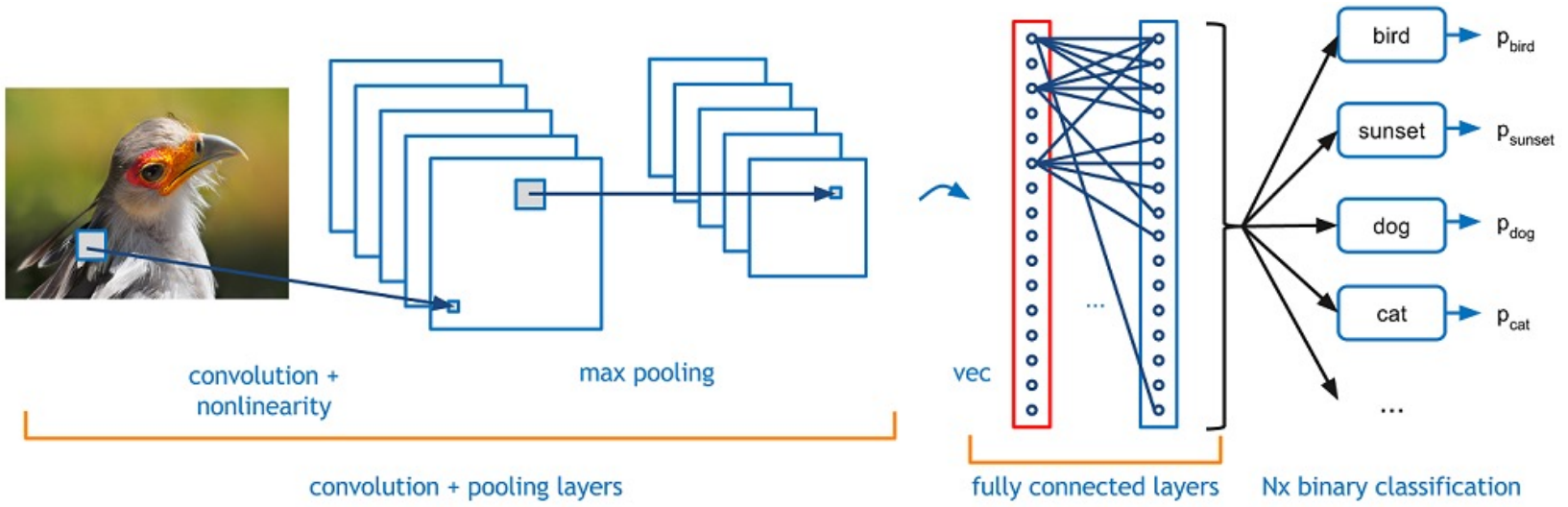


Convolutional neural networks (CNNs)





Convolutional neural networks (CNNs)



A partially-connected feedforward NN with initial convolutional filtering layers

- Biologically inspired (visual cortex)
- Good for high-dimensionality input (e.g., images)

Typical structure:

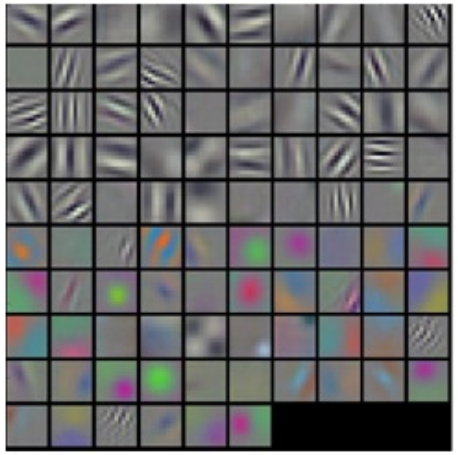
- Convolution + pooling layers (one to several)
- Fully connected layers (one or two)
- Output layer

1. Architecture
2. Hyperparameters:
   - Number, size, shape of conv. filters/kernels
   - Max-pooling size
3. Training
4. Implementation (GPUs?)

Convolution layers

- To classify an image, a fully-connected deep network would have a huge set of weights to learn
  - E.g., 1M (1000*1000) pixels as input, 10,000 units at layer 1, 5,000 units at layer 2, 2000 units at layer 3, 1000 output units
    - $1M \times 10{,}000 + 10{,}000 \times 5000 + 5000 \times 2000 + 2000 \times 1000$
      $= 10B$ weights
    - The vast majority in the first layer

- Instead, let's use a convolution layer
  - Learn 96 filters of size 11×11×3 for R/G/B
    - That's 96×11×11×3 = 34,848 parameters
  - Each filter is passed over the image and produces an output (filtered) image
  - The filter defines the CNN layer's *receptive field*

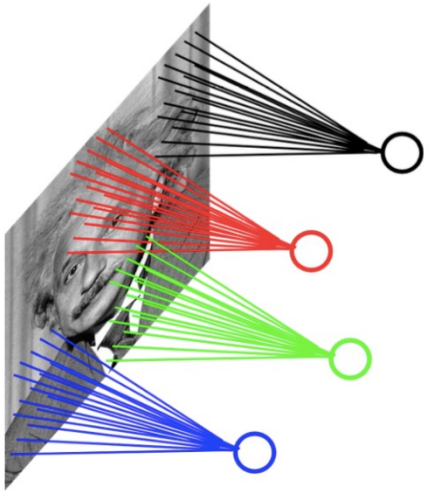


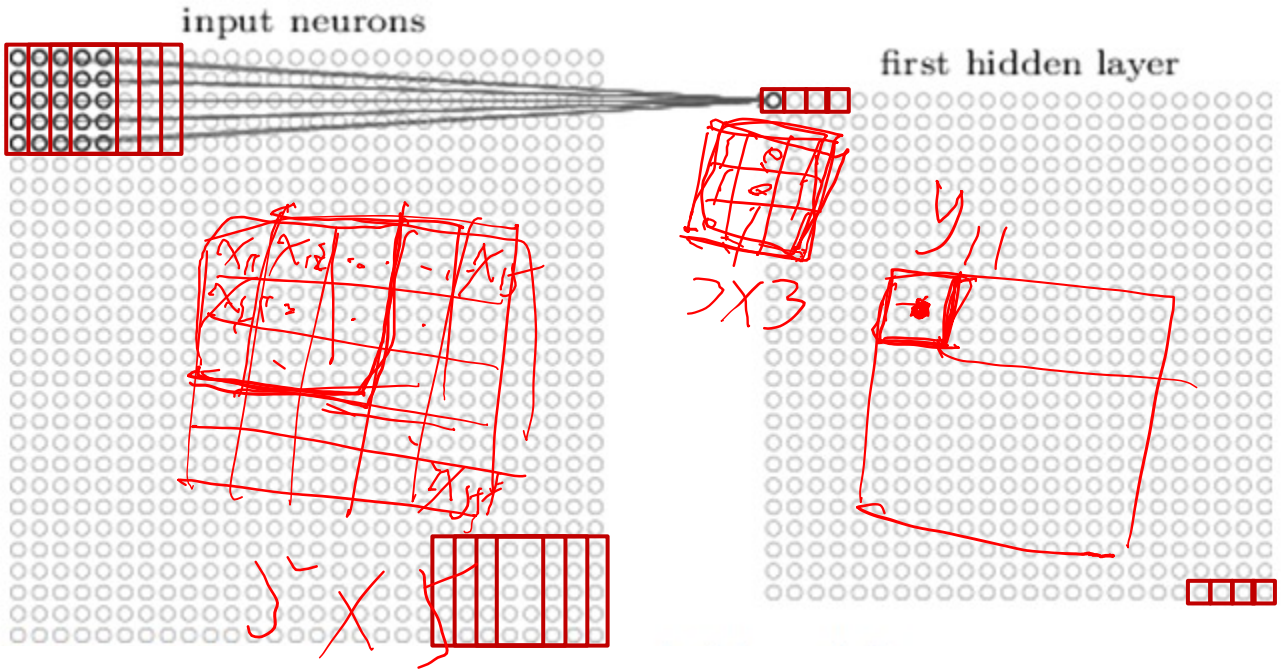
Convolution layers

With 7 filters, we have
$7 \times (5 \times 5 \times 3 + 1) = 532$
weights to learn
(including bias term)

5×5×3 filter

Apply to R, G, and B images

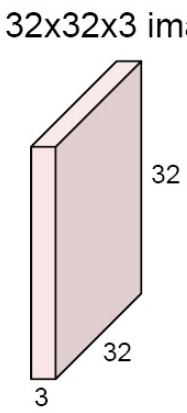


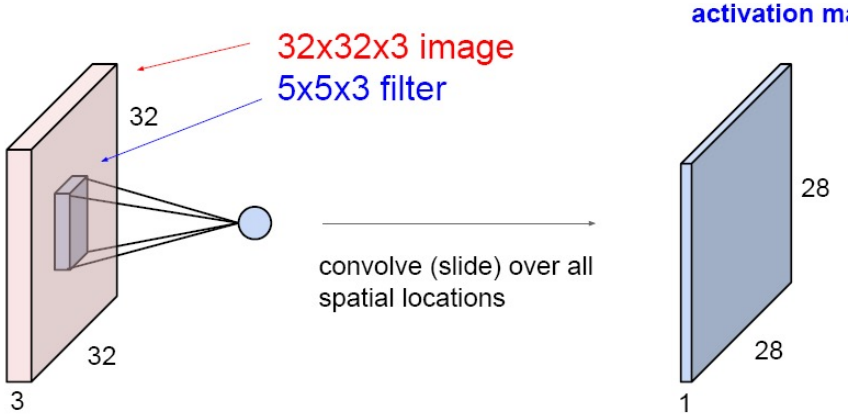


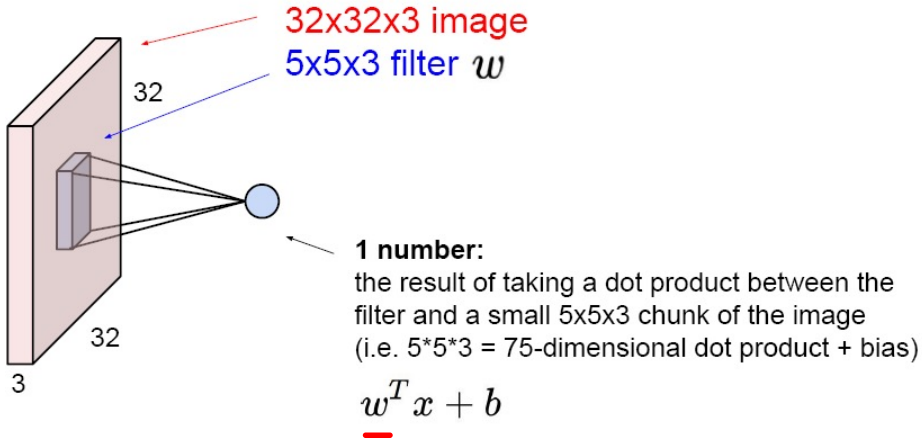
$N \times M$ image

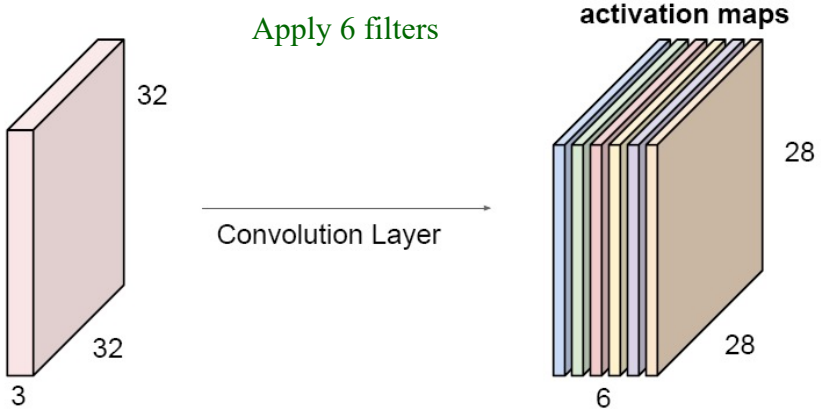
$(N-4) \times (M-4)$ image

Convolution layers





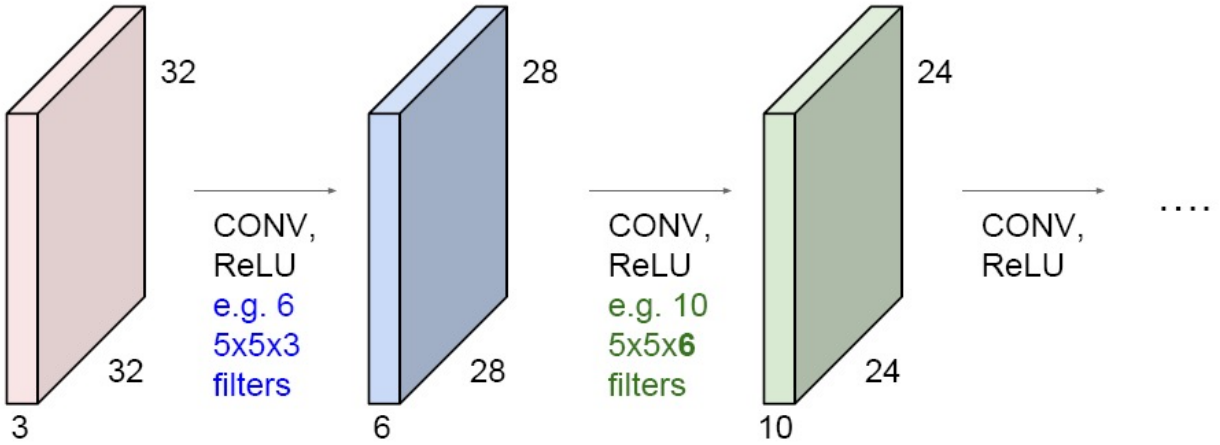




Convolution layers

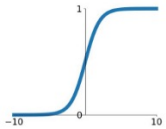
Example: 2 filters

Multiple convolution layers

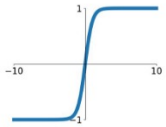


A convolution layer typically includes a nonlinear activation function and may include a pooling layer

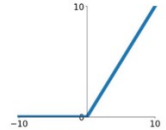
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



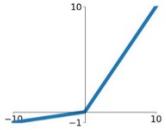
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

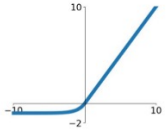


Leaky ReLU
$\max(0.1x, x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

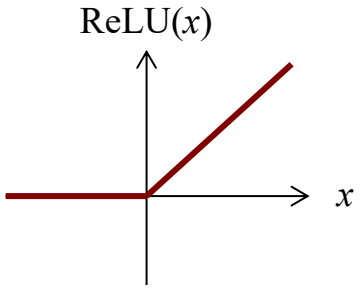
ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

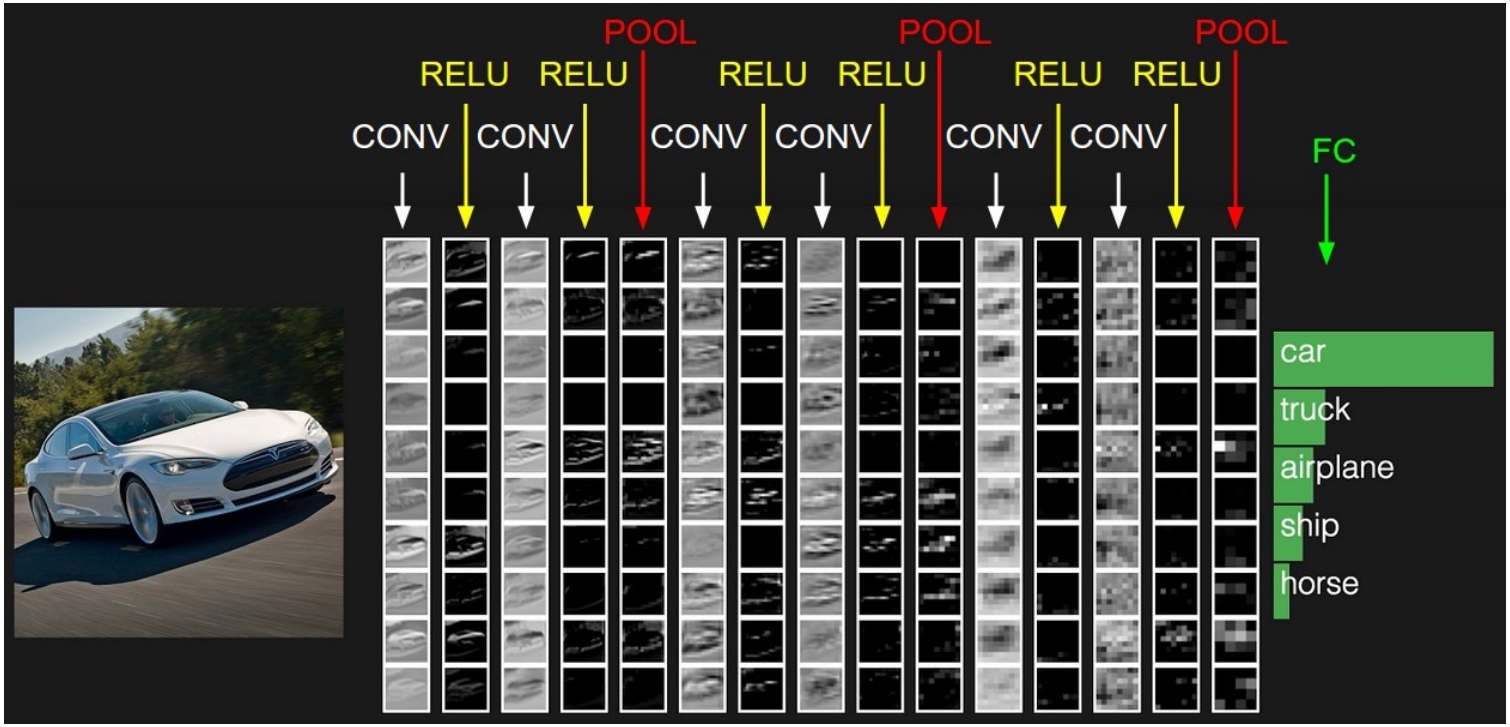


Rectified linear units (ReLUs)

- An ReLU layer applies a nonlinear activation function to the convnet output:

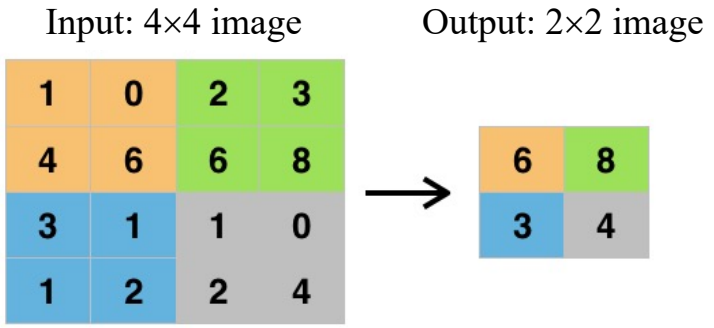
$$f(x) = \max(0, x)$$





Pooling layers

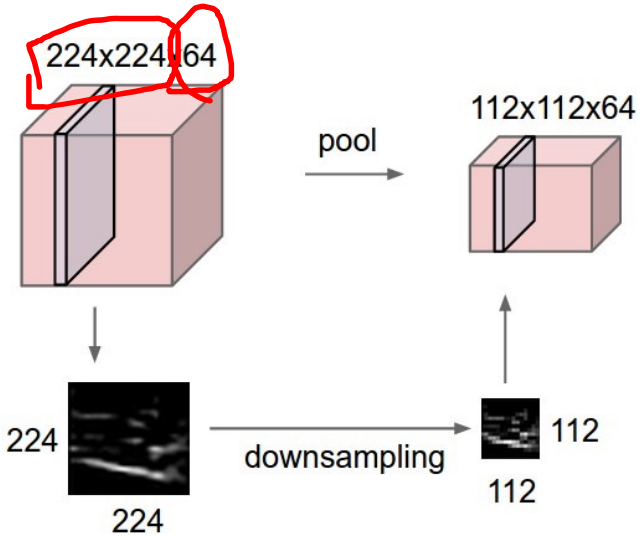
- Immediately after the convolution layer comes a *pooling* layer
  - Subsampling, data reduction, summarizing
- Main purposes:
  - Reduce the size of the representation (and thus the number of parameters in the network); i.e., dimensionality reduction
  - Control overfitting
- No learnable parameters:
  - Filter size (M×N)
  - Type of pooling
    - Max, mean, L2-norm

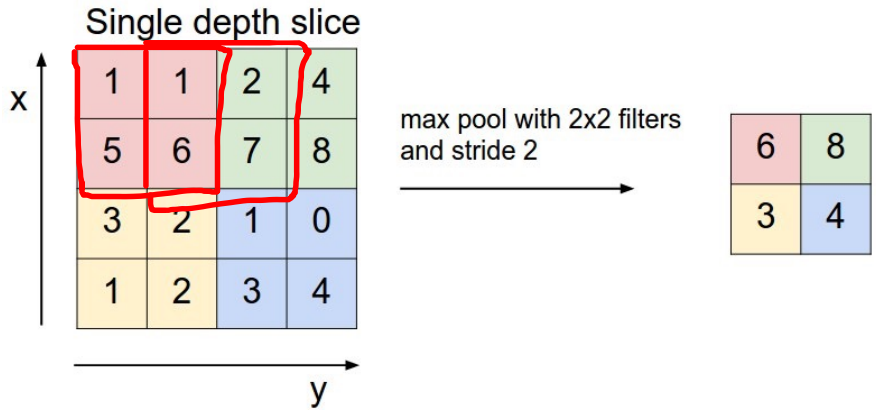


Max pooling example:
Filter size = 2×2

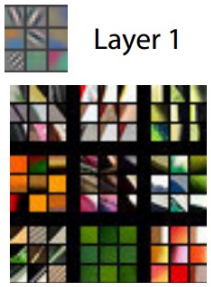
Max pooling (MP)

- Max pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value of the features in that region

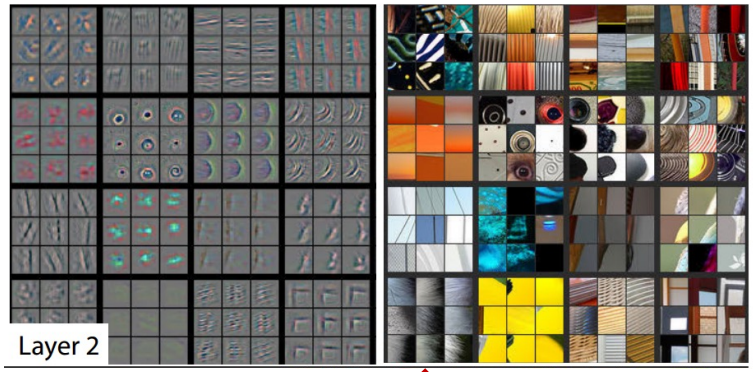




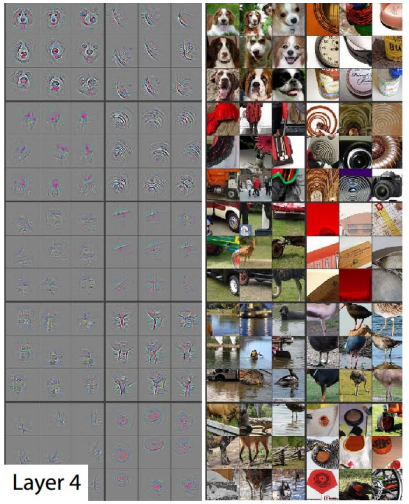
Example: 5 layers of CNN-MP



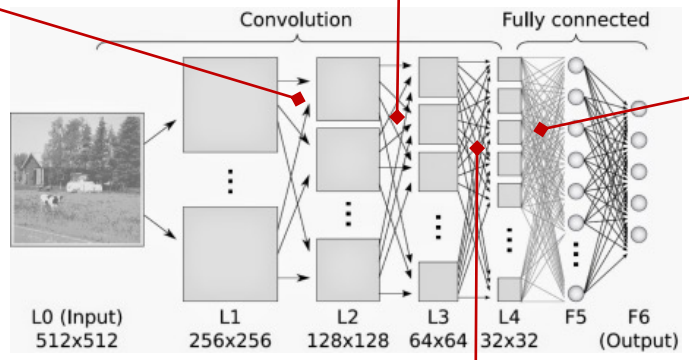
Layer 1

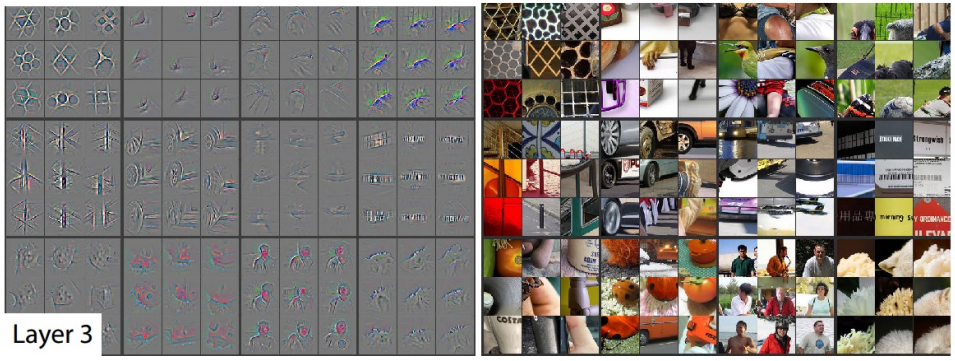


Layer 2



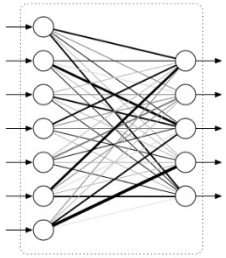
Layer 4



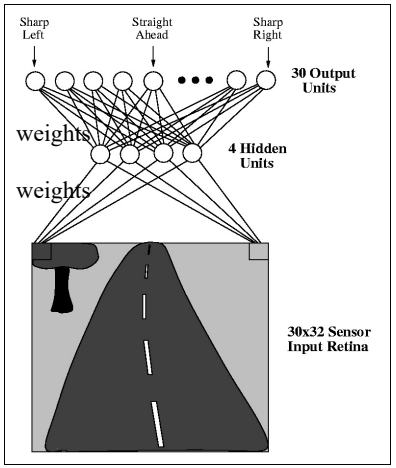


Layer 3

Fully-connected (FC) layers



- After (several) convolutional/pooling layers, the high level representations are typically fed into 1 or 2 perceptron-like fully-connected layers
  - The "meta pixels" of the last convolutional layer are inputs to a fully-connected layer
  - The 2D image structure is now lost – e.g., a 16x16 image becomes 256 individual inputs to the fully-connected layer
    - Just like ALVINN



  - The last fully-connect layer is called the *output layer*, outputting class scores
  - *N* output nodes for *N* categories
  - The output nodes implement the Softmax function

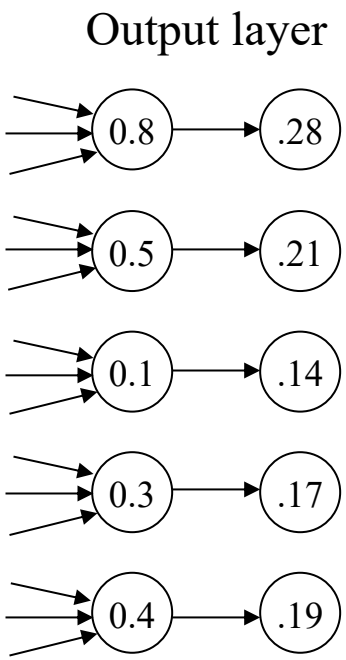
Softmax function

- For k output nodes with values $\{x_1, x_2, \ldots, x_k\}$, the softmax function produces values that sum to 1.0

$$s(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

x_i	$f(x_i)$
0.8	0.57
0.5	0.43

x_i	$f(x_i)$
0.2	0.22
0.3	0.24
0.7	0.35
0.1	0.19



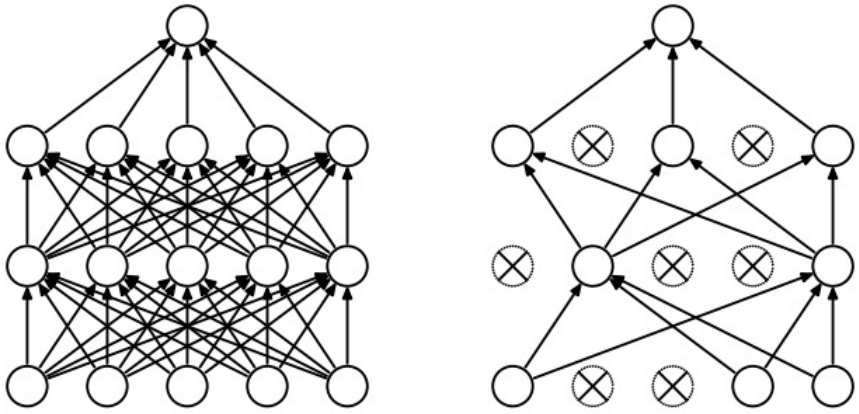
- May be interpreted as *P*(class)

Training a convolutional neural network

- Choose the architecture and hyperparameters
  - Layers, number of filters, filter sizes, activation functions, etc.
- Initialize all filters and parameters / weights with random values
- Iterate over training data:
  - Forward propagate to output unit values
    - Output unit o_i can be considered a probability for class i
  - Calculate error from training label
  - Backpropagate the error via gradient descent on the weights
  - Update the weights
- When "acceptable error" is reached, you're done!

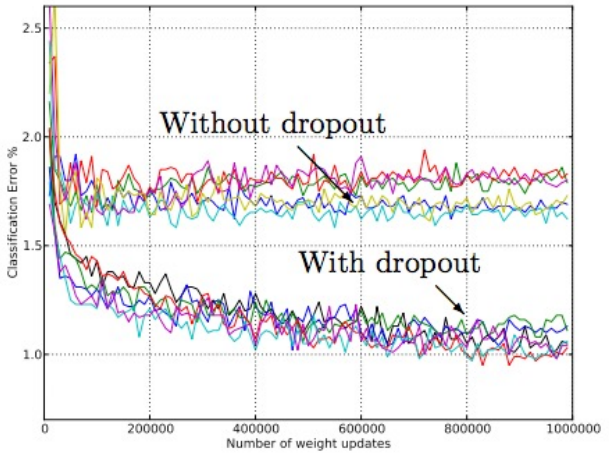
Dropout

- Ensemble models are too expensive for deep networks that take days to train each model
- Dropout is an efficient version of model combination that costs only a factor of two during training
- For each training iteration, set the output of each hidden neuron to zero, with probability 0.5
  - These neurons do not contribute to forward or backward propagation
  - Doubles the number of iterations needed to converge
  - Significantly reduces overfitting



(a) Standard Neural Net

(b) After applying dropout.

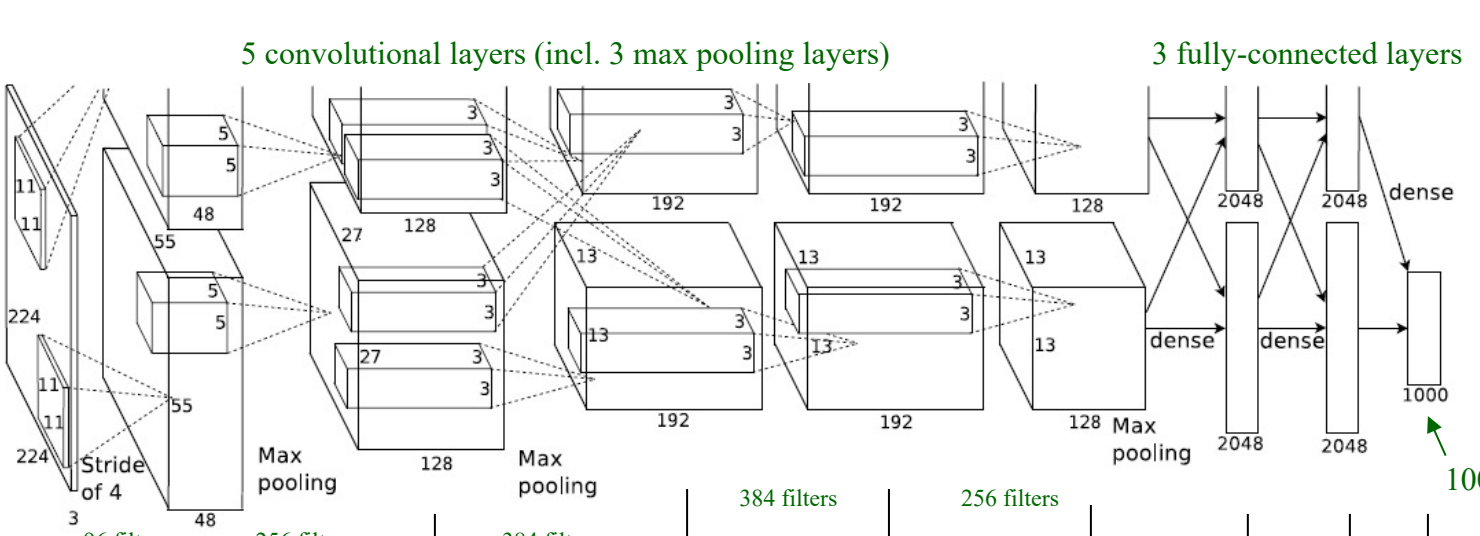


AlexNet (2012)

[PDF] **Imagenet classification** with **deep convolutional neural** networks
A Krizhevsky, I Sutskever, GE Hinton - Advances in **neural** information …, 2012 - kr.nvidia.com
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is …
☆ 99 Cited by 77743 Related articles All 121 versions »

Best performance on ImageNet competitions in 2012

Implemented on two GPUs



809,800 Neurons

96 filters | 256 filters | 384 filters | 384 filters | 256 filters

224×224×3 = 150,528

55×55×96 = 290,400

27×27×256 = 186,624

13×13×384 = 64,896

13×13×384 = 64,896

13×13×256 = 43,264

4096

4096

1000

60,954,656 Weights

96×11×11×3 = 34,848

256×5×5×48 = 307,200

384×3×3×256 = 884,736

384×3×3×192 = 663,552

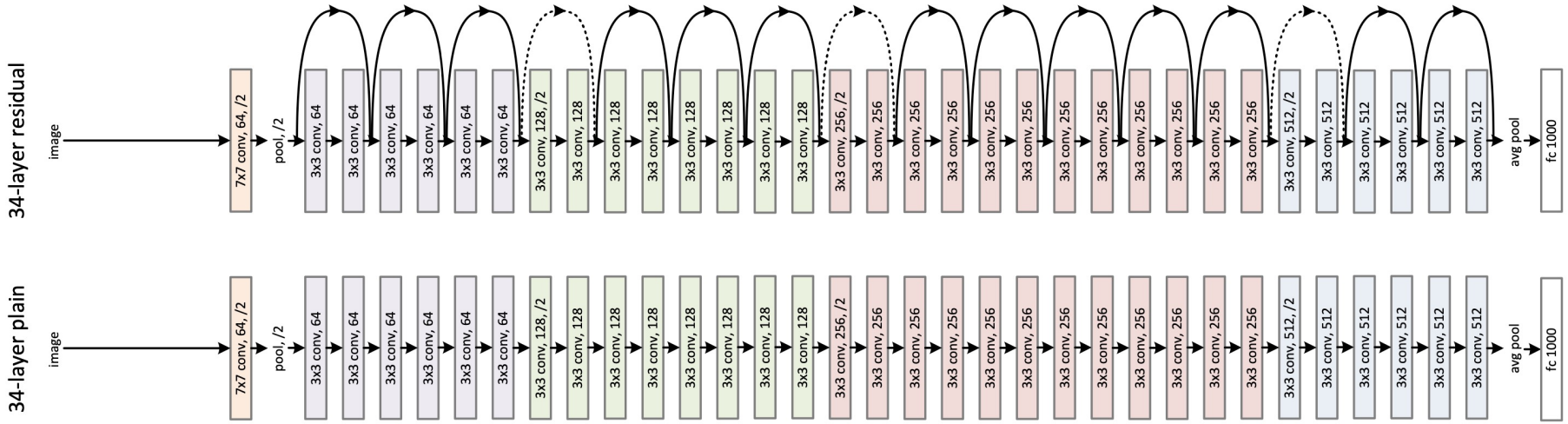
256×3×3×192 = 442,368

6×6×256×4096 = 37,748,736

4096×4096 = 16,777,216

4096×1000 = 4,096,000

ResNet (2016)



Deep residual learning for **image recognition**

K He, X Zhang, S Ren, J Sun - … and pattern **recognition**, 2016 - openaccess.thecvf.com

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer …

☆ 99 Cited by 71768 Related articles All 55 versions »

Recommended reading: some fundamental NNs

- Recurrent neural network
  - E.g., LSTM: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Transformer (self-attention network)
  - http://peterbloem.nl/blog/transformers
- Generative adversarial networks (GAN)
- Deep Q-learning
- …

Some disadvantages of deep learning

- The models are very complex, with lots of parameters to choose and optimize:
  - Number of layers, size of layers, node functions
  - From "feature engineering" to "DL architecture engineering"
- Very slow to train
- Computationally expensive
- Some problems more amenable to deep learning than other applications
  - Simpler models may be sufficient for many domains
- The learned models can be very hard to explain (e.g., compared with decision trees)
  - What does neuron 524 do?

Success of deep learning

- State-of-the-art performance in a wide range of different areas:
  - Language modeling (GPT-3)
  - Image recognition (AlexNet, ResNet, ViT, etc.)
  - Sentiment classification and other NLU tasks (BERT)
  - Speech recognition (2010, Dahl et al.)
  - AlphaGO, AlphaZero, AlphaFold, etc.
- What do these problems have in common?
  - Each are non-linear classification problems where the information is highly hierarchal in nature
  - Problems that humans excel at and machines do very poorly
- Andrew Ng (ML Professor at Stanford / Cofounder of Coursera):

  "I've worked all my life in Machine Learning, and I've never seen one algorithm knock over benchmarks like Deep Learning."

End of the course!