Deep neural learning

Devika Subramanian

Rice University

devika@rice.edu

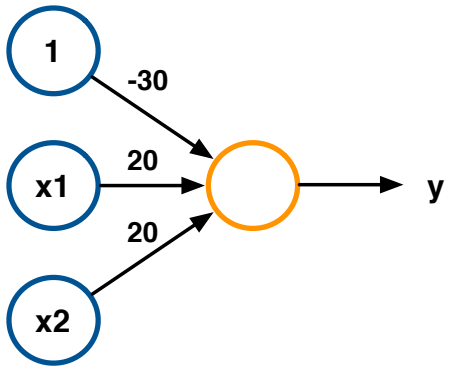
August 15 and 16, 2019

Outline

take a logistic reg mod, another, etc.

Stacked models: feedforward neural networks
Forward propagation in a feedforward neural networks
Parameter learning by gradient descent
Back propagation in a feedforward neural networks
Training deep neural networks
Deep neural network architectures: convolutional networks

Prelude to neural networks: logistic unit



$$y = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} = g(-30 + 20x_1 + 20x_2)$$

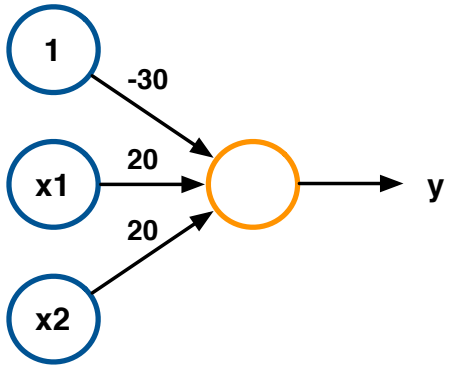
x_1	x_2	y
0	0	g(-30) $\approx$ 0
0	1	g(-10) $\approx$ 0
1	0	g(-10) $\approx$ 0
1	1	g(10) $\approx$ 1

~AND function

A logistic unit can represent any linearly separable function.

Logistoc Unit aka Logistic Regression

Prelude to neural networks: perceptron



$$y = sign(\theta^T x) = sign(-30 + 20x_1 + 20x_2)$$

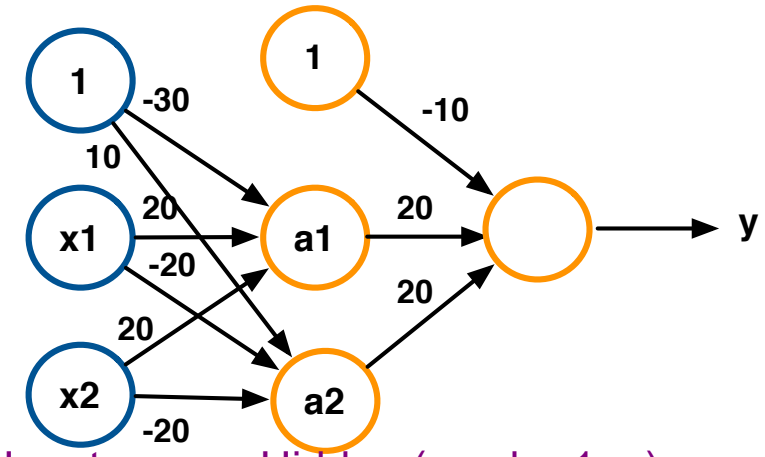
x_1	x_2	y
-1	-1	sign(-70) = -1
-1	1	sign(-30) = -1
1	-1	sign(-30) = -1
1	1	sign(10) = 1

A perceptron can represent any linearly separable function.

Neural networks: stacked models (logistic units)

Can't represent XORs?
https://archive.org/details/Perceptrons/

use multiple logistic units!



x_1	x_2	a_1	a_2	y
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Rosenblatt;
this can compute any logistic unit

$$y = a_1 \; OR \; a_2 = (x_1 \; AND \; x_2) \; OR \; (x_1 \; NAND \; x_2) = x_1 \; XNOR \; x_2$$

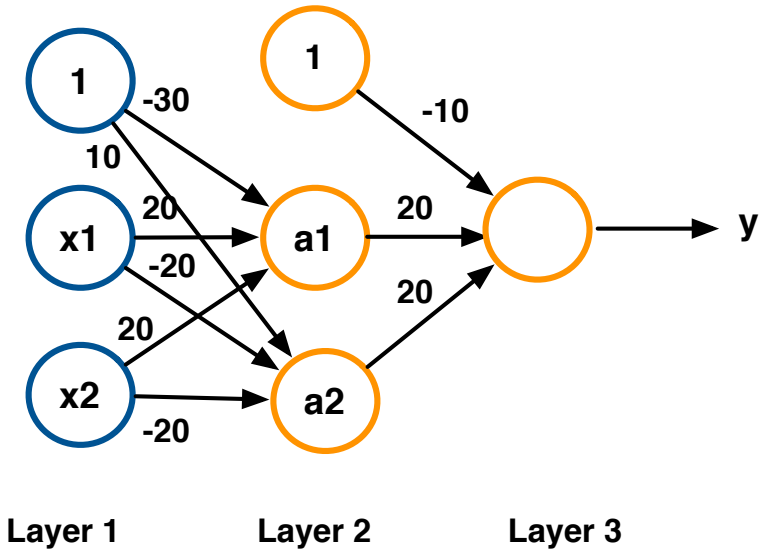
Representational power of stacked logistic models (neural networks)

- Any boolean function can be represented by a three layer network of logistic units.
- Any continuous function on the reals can be represented by a three layer network.
- Any function on the reals (including discontinuous functions) can be represented by a four layer network.
- Any non-linearity works, not just the sigmoid function, e.g., rectified linear unit (ReLU).

Rectified Linear Unit

Parameters of a neural network

All units in Layer 1 is connected to all non-biased unit in Layer 2...



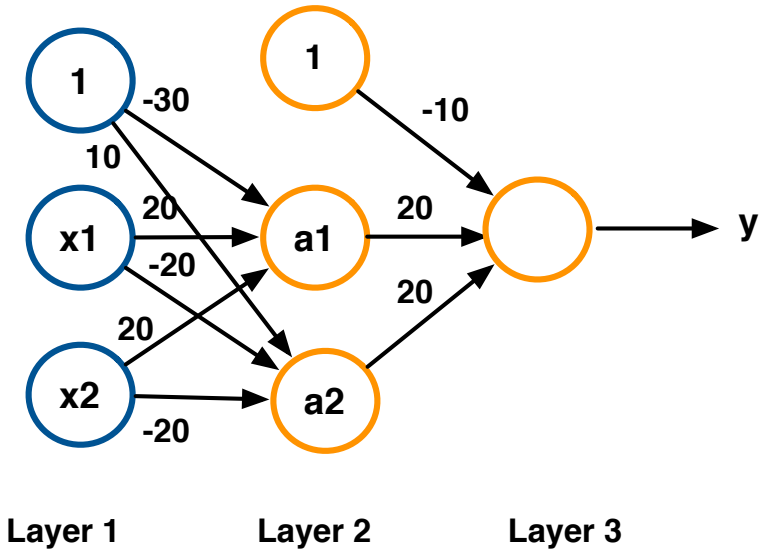
- $a_j^{(l)}$ = activation of j^{th} unit in layer l
- $\Theta^{(l)}$ = parameters/weights mapping units in layer l to layer $l+1$

2 rows x 3 cols to Layer 2 — 1 row x 1 col to Layer 3

$$\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix}; \Theta^{(2)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix}$$

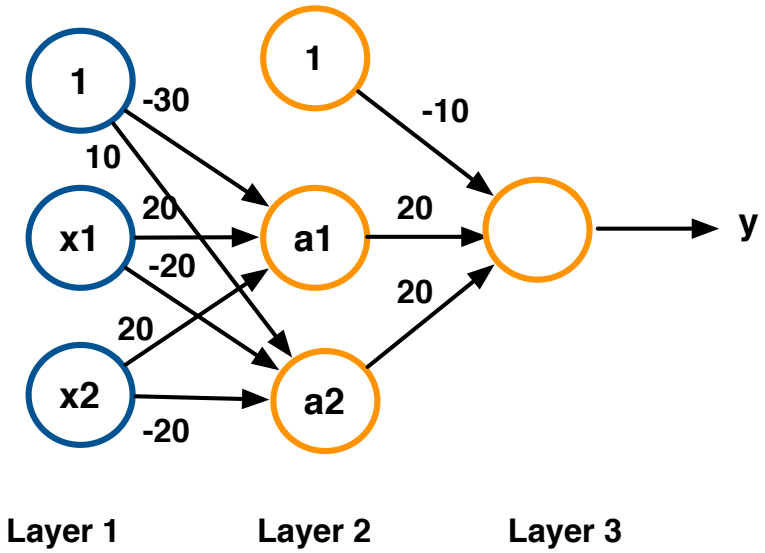
fully connected, feed forward

Forward propagation



$$
\begin{aligned}
a^{(1)} &= x \\
z^{(l+1)} &= \Theta^{(l)} * [1; a^{(l)}],\ l = 1 \ldots L \\
a^{(l+1)} &= g(z^{(l+1)}),\ l = 1 \ldots L
\end{aligned}
$$

Backpropagation: intuition



- error at output layer: $\delta^{(L)} = y - y_{actual}$
- errors made at layer l: $\delta^{(l)}$ determined by errors at layer $l+1$

$$\delta^{(l)} = \left(\Theta^{(l)}\right)^T \delta^{(l+1)}$$

from y take error ~almost a linear allocation of blame backwards

- stochastic gradient descent to adjust parameters Θ based on errors.

Backpropagation: examples

- Le Net demo: `http://yann.lecun.com/exdb/lenet/`
- Nettalk demo: `https://www.youtube.com/watch?v=gakJlr3GecE`

Choices for deep feedforward networks

- Select architecture: how many hidden layers, how many units in each hidden layer.
- Select non-linearity for each layer (ReLU, tanh) (tanh hyperbolic)
- Select loss function (cross-entropy, mean squared error, etc.) (class, regre)
- Select regularization approach (L1 or L2 penalty for each layer)

Choices for deep feedforward networks

- Select parameter initialization approach
  - Initialize bias weights to zero
  - All other parameters, use Xavier initialization; sample $\theta_{ij}^{(l)}$ from Uniform(-b,b) where $b = \frac{\sqrt{6}}{\sqrt{S_l + S_{l-1}}}$ most popular (e.g., keras)
- Select learning rate α
  - Global learning rate for all parameters not recommended
  - Per-parameter learning rate (Adam) best optimizer for now
- Choose parameter update scheme: vanilla gradient descent, gradient descent with momentum

Choices for deep feedforward networks

- Manipulating training data sets
  - Normalize/preprocess your training data: standardization, mean subtraction, whitening, PCA
  - Data augmentation: create new (x, y) pairs from given (x, y) pairs by standard operations of rotation, translation (domain-specific)
- Deciding when to stop: using a validation set
- Minibatch gradient estimation: size of minibatch

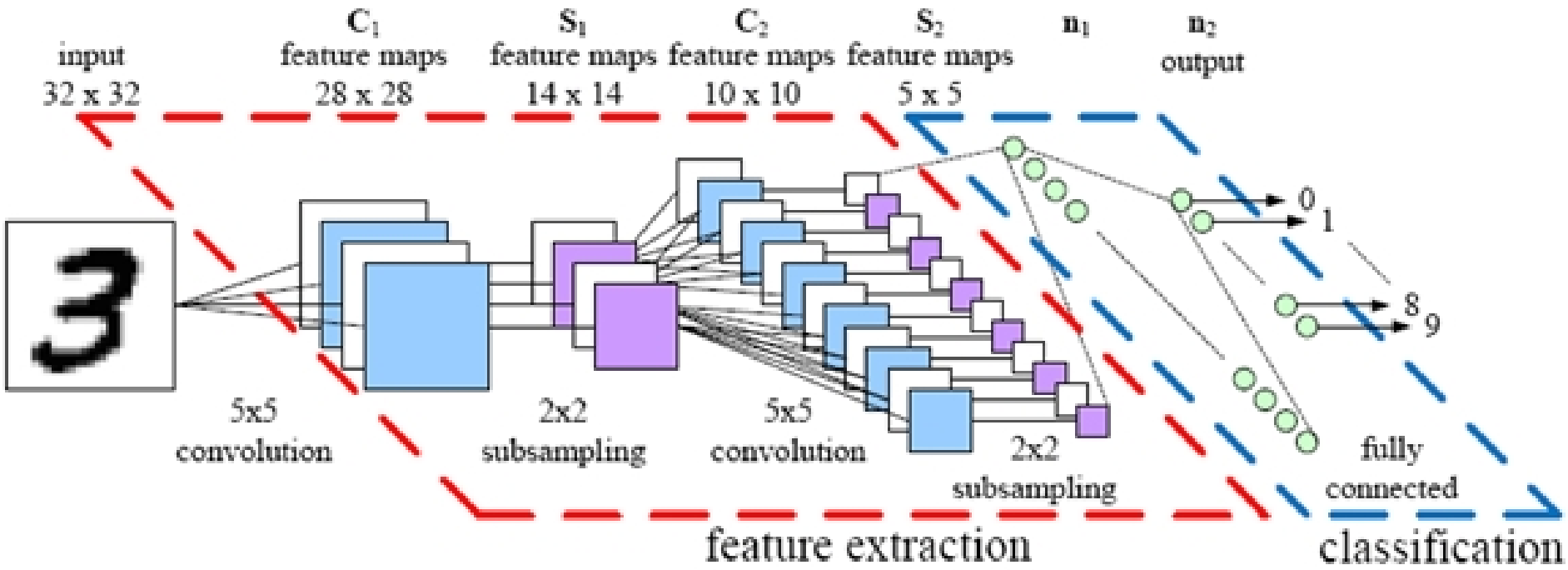
https://playground.tensorflow.org

Life is good in *Deepland!*

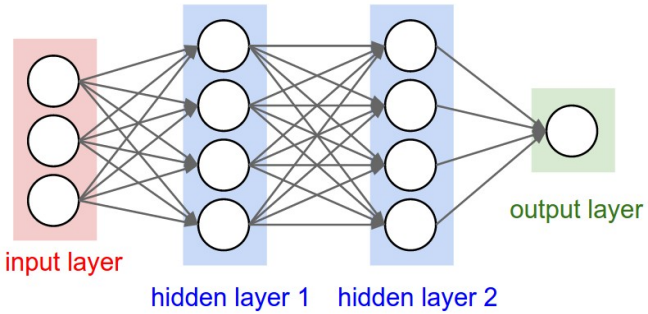
- "No need to design algorithms anymore"
- For any given problem, just:
  1. **label some training data**
  2. **define objective function**
  3. train neural network

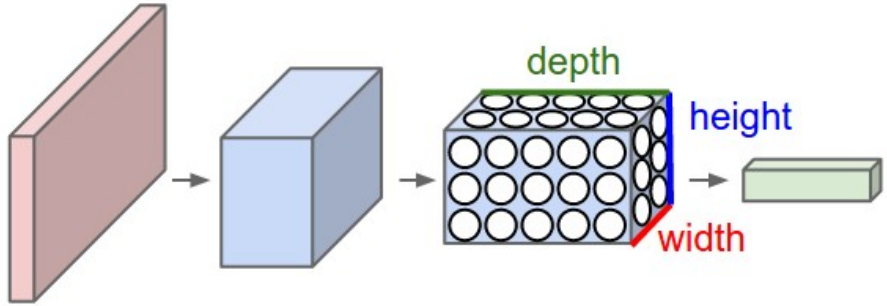
  4. sell your startup for millions!

Deep learning: the Le Net architecture



Deep learning architectures



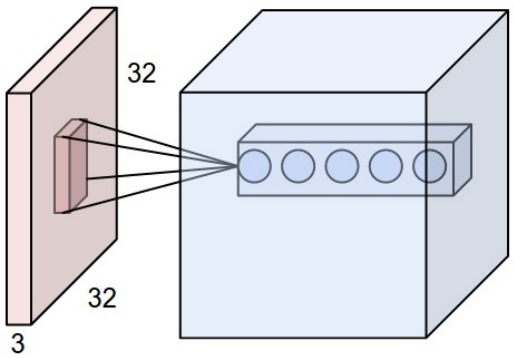


- A deep convolutional neural network is made up of layers. Every layer has a simple API: It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.
- There are three types of layers: convolutional layers, pooling layers, dense or fully-connected layers. Fully connected layers correspond to traditional neural networks.

Source: cs231n at Stanford University

Convolutional layer

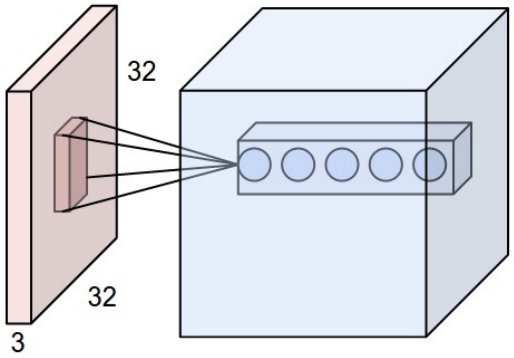
example
4 x 4
convolved with a 2 x 2
results in a 3 x 3



- An example input volume in red (e.g. a 32x32x3 CIFAR-10 image), and an example volume of neurons in a convolutional layer.
- Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input.
- Each neuron computes a weighted sum of its inputs followed by a non-linear activation function, such as the sigmoid function.

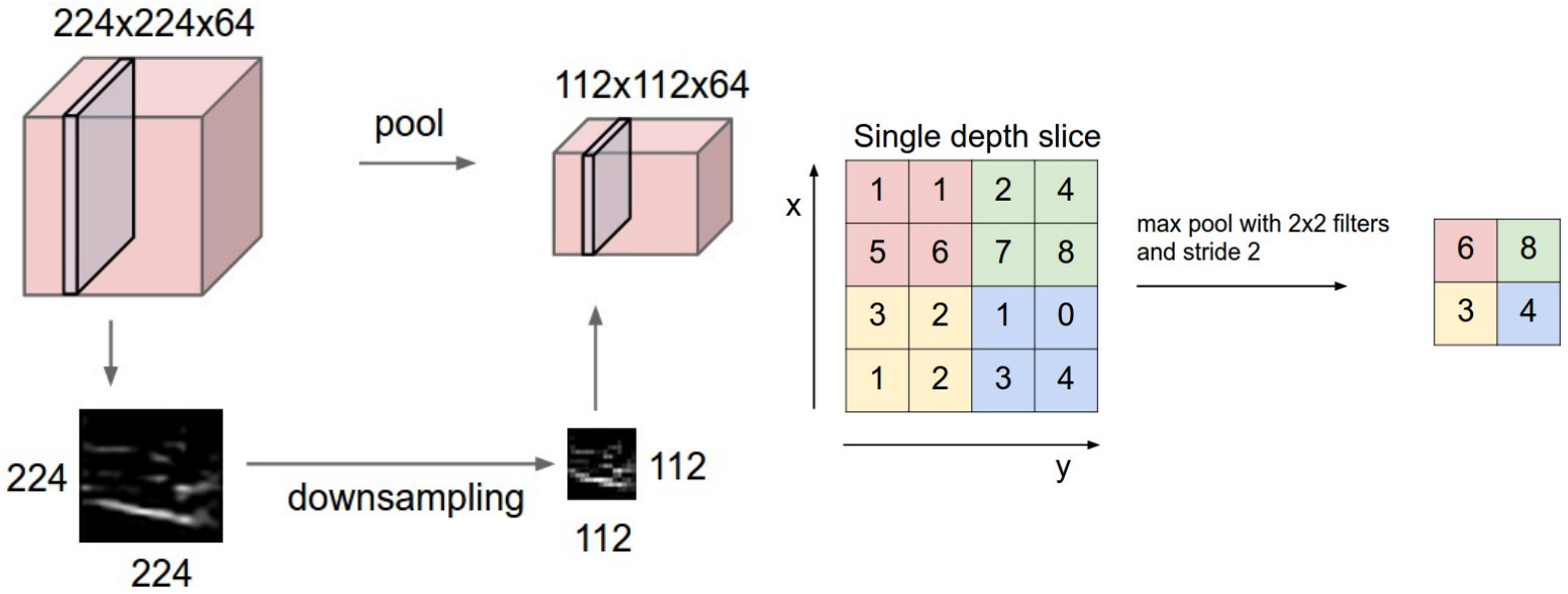
Source: cs231n at Stanford University

Parameter sharing in a convolutional layer



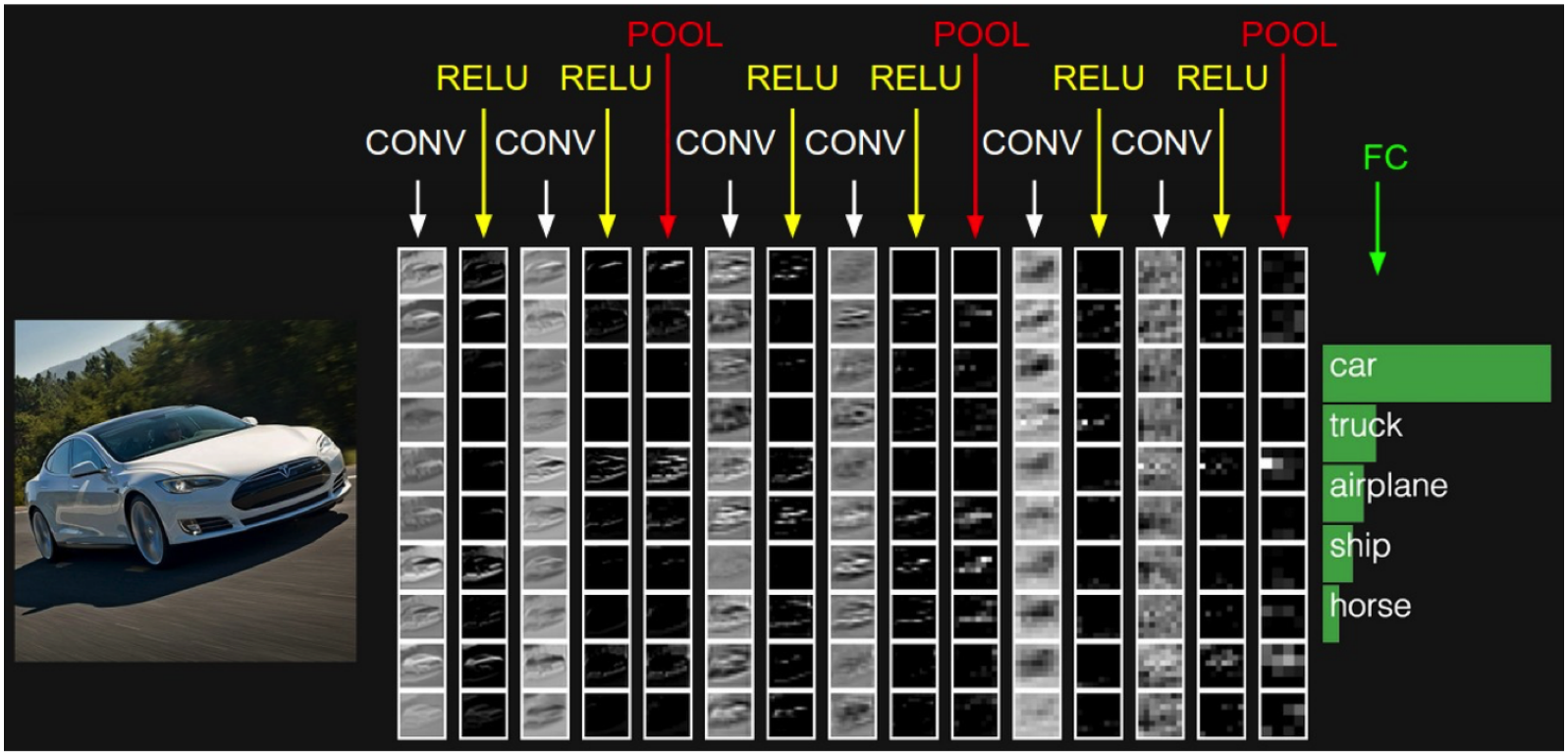
- Each unit in the blue output layer has 5x5x3 connections with the 5x5x3 patch it is connected to.
- If the output volume is H x W x D, then a dense layer of connections would require (H x W x D x 32 x 32 x 3) parameters.
- A convolutional layer only needs Dx5x5x3 parameters! All parameters in a given depth layer are shared. The output at given depth slice is computed as a convolution of these parameters (filter) with the input volume.

Pooling layer



It is common to periodically insert a pooling layer in-between successive convolutional layers in a deep network architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.

Convolutional neural network for object recognition



Convolutional neural networks: recent results

- Object recognition: CIFAR-10 data (10 object types)
  - AlexNet: 8 layers, 2012 (15.4% error rate)
  - VGGNet, 19 layers, 2014 (7.3% rate)
  - ResNet, 152 layers, 2015 (3.6% error rate)
- AlphaGo: tree search guided by a learned value function implemented as a deep neural network.