Introduction to Deep Learning

Pabitra Mitra

Indian Institute of Technology Kharagpur
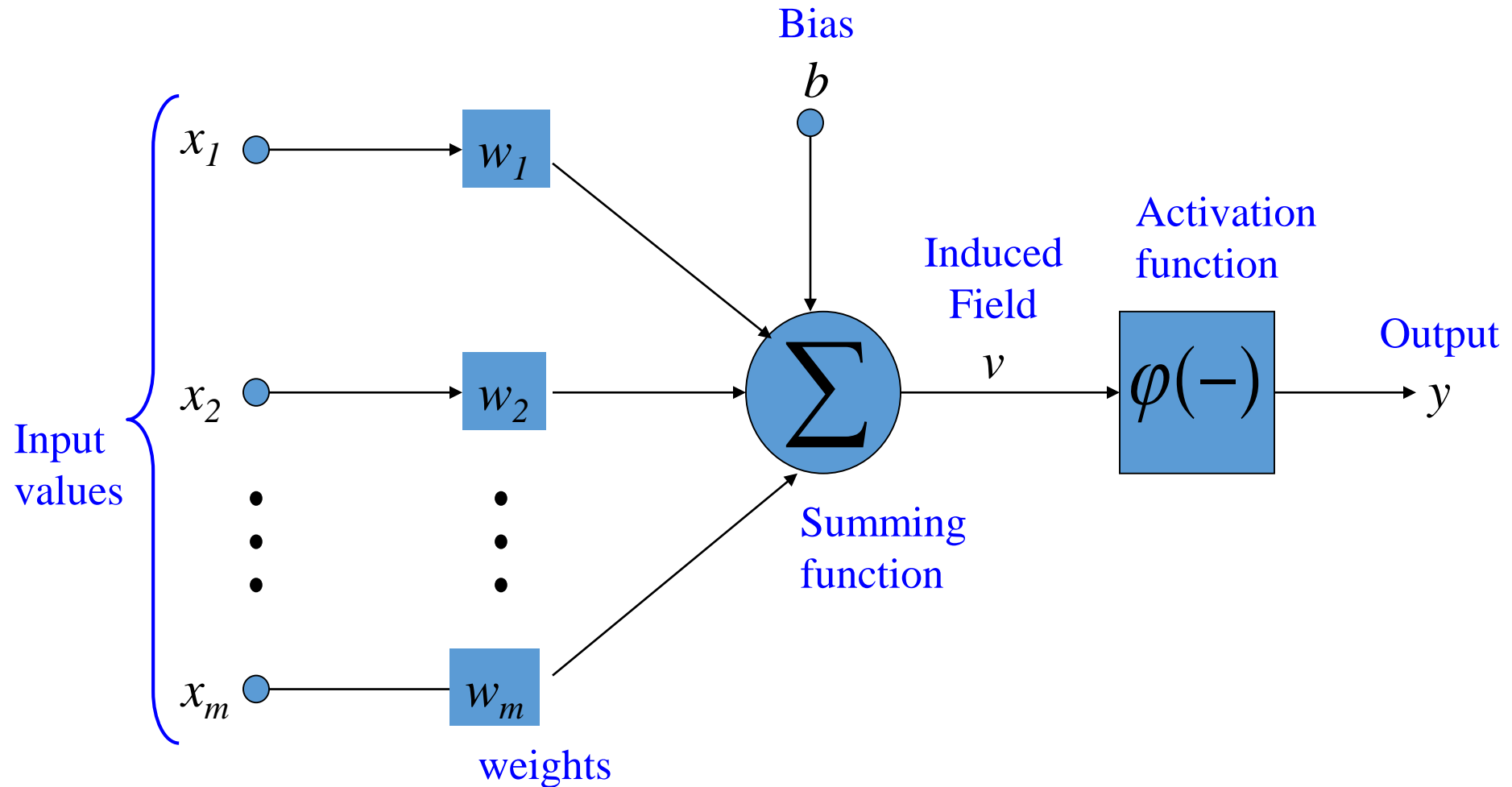
pabitra@cse.iitkgp.ac.in

NSM Workshop on Accelerated Data Science

Deep Learning

- Based on neural networks

- Uses deep architectures

- Very successful in many applications

Perceptron

Neuron Models

- The choice of activation function φ determines the neuron model.

Examples:

- step function:
$$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > c \end{cases}$$

- ramp function:
$$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > d \\ a + ((v-c)(b-a)/(d-c)) & \text{otherwise} \end{cases}$$
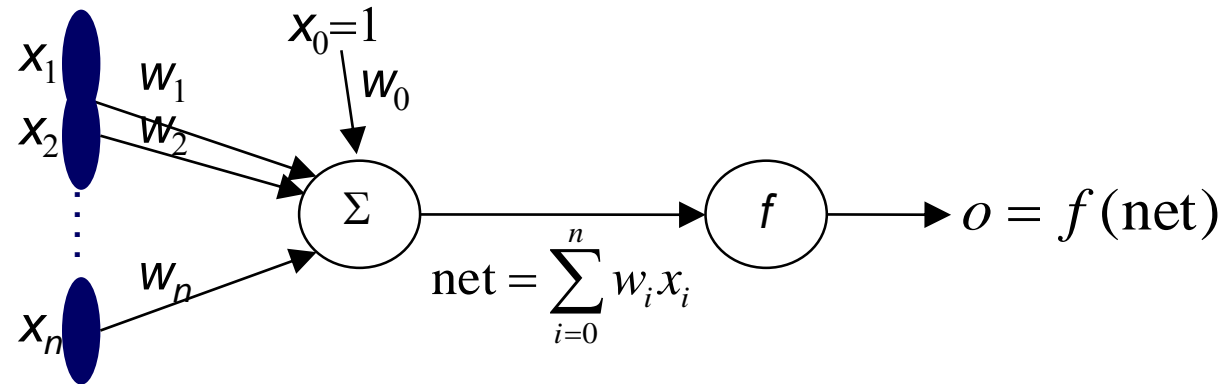
- sigmoid function with z,x,y parameters
$$\varphi(v) = z + \frac{1}{1 + \exp(-xv + y)}$$

- Gaussian function:
$$\varphi(v) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{v-\mu}{\sigma} \right)^2 \right)$$

Sigmoid unit

- f is the sigmoid function

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

- Derivative can be easily computed:

- Logistic equation
  - used in many applications
  - other functions possible (tanh)

$$\frac{df(x)}{dx} = f(x)\big(1 - f(x)\big)$$

- Single unit:
  - apply gradient descent rule
- Multilayer networks: backpropagation

Multi layer feed-forward NN (FFNN)

- FFNN is a more general network architecture, where there are hidden layers between input and output layers.

- Hidden nodes do not directly receive inputs nor send outputs to the external environment.

- FFNNs overcome the limitation of single-layer NN.

- They can handle non-linearly separable learning tasks.

Input layer

Output layer

Hidden Layer
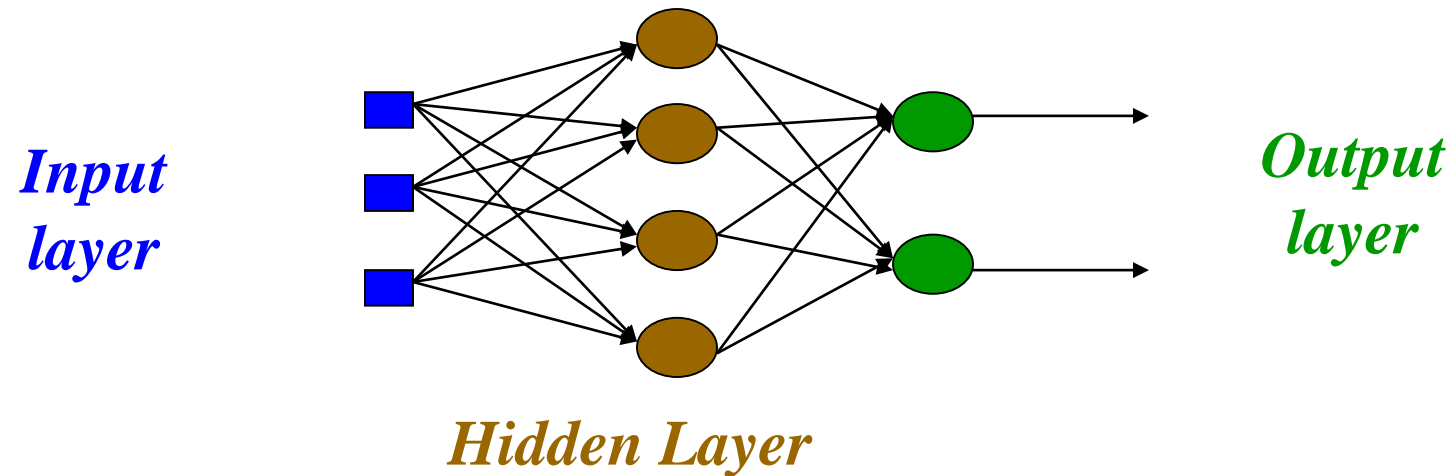
3-4-2 Network

Backpropagation

- Initialize all weights to small random numbers

- Repeat

  For each training example

  1. Input the training example to the network and compute the network outputs
  2. For each output unit k

  $$\delta_k \leftarrow o_k\,(1 - o_k)\,(t_k - o_k)$$

  3. For each hidden unit h

  $$\delta_h \leftarrow o_h\,(1 - o_h)\,\Sigma_{k \in \text{outputs}}\,w_{k,h}\delta_k$$

  4. Update each network weight $w_{j,i}$

  $$w_{j,i} \leftarrow w_{j,i} + \Delta w_{j,i}$$

  where $\Delta w_{j,i} = \eta\,\delta_j x_{j,i}$

NN DESIGN ISSUES

- Data representation
- Network Topology
- Network Parameters
- Training
- Validation

Expressiveness

- Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer (Cybenko et al '89)
  - Hidden layer of sigmoid functions
  - Output layer of linear functions
- Any function can be approximated to arbitrary accuracy by a network with two hidden layers (Cybenko '88)
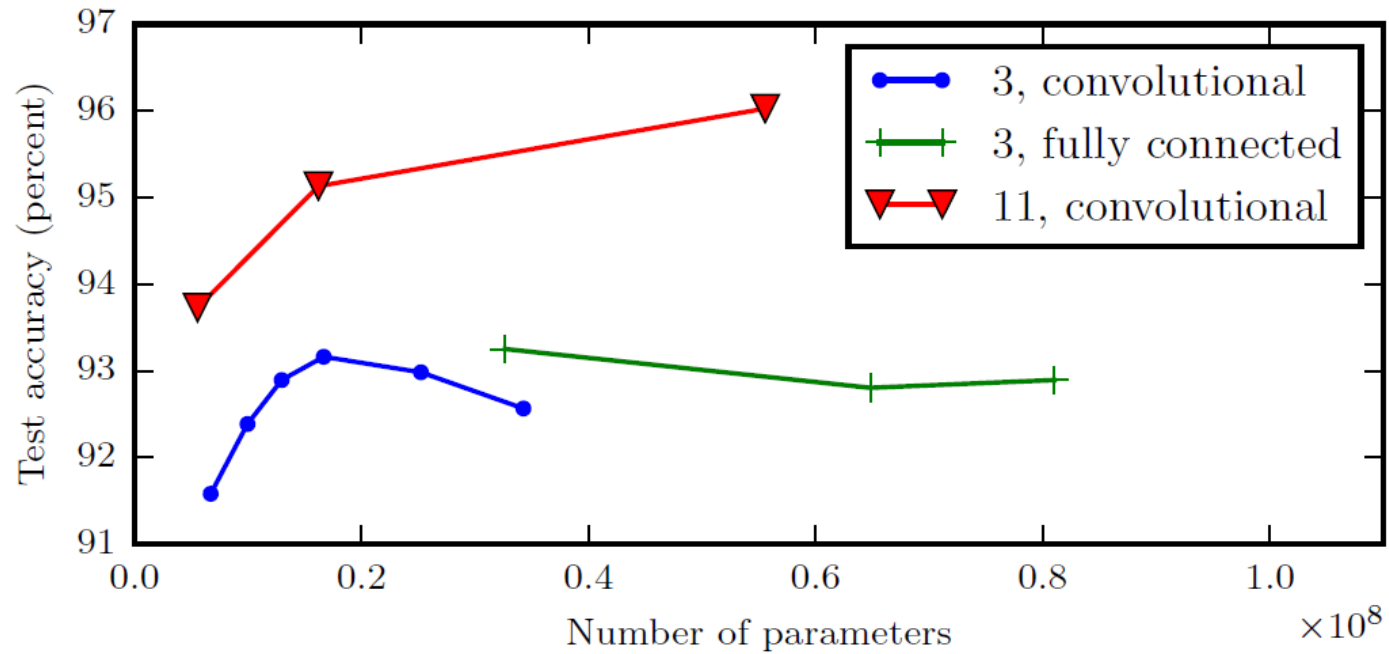 - Sigmoid units in both hidden layers
  - Output layer of linear functions

Choice of Architecture Neural Networks
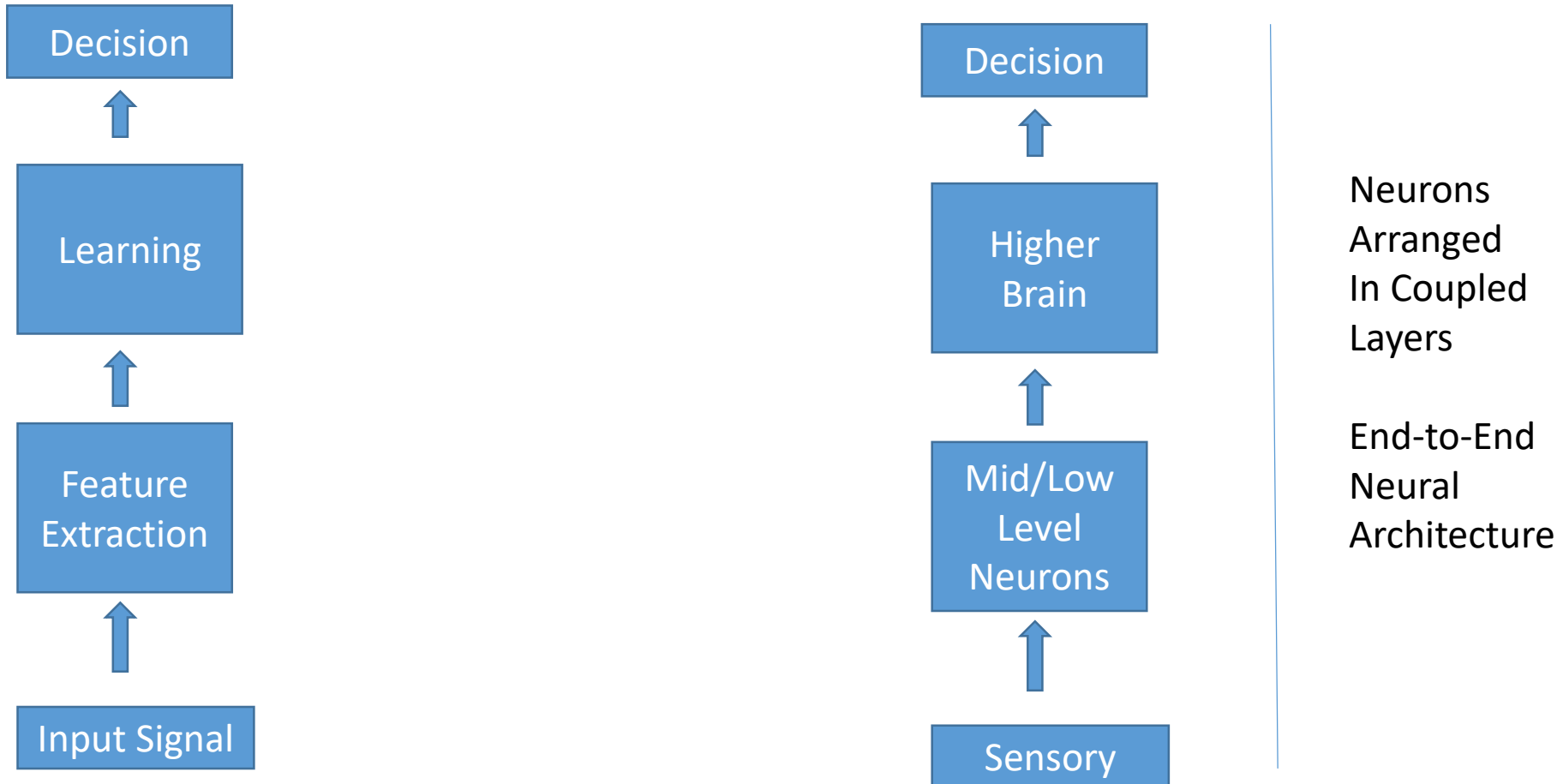
- Training Set vs Generalization error
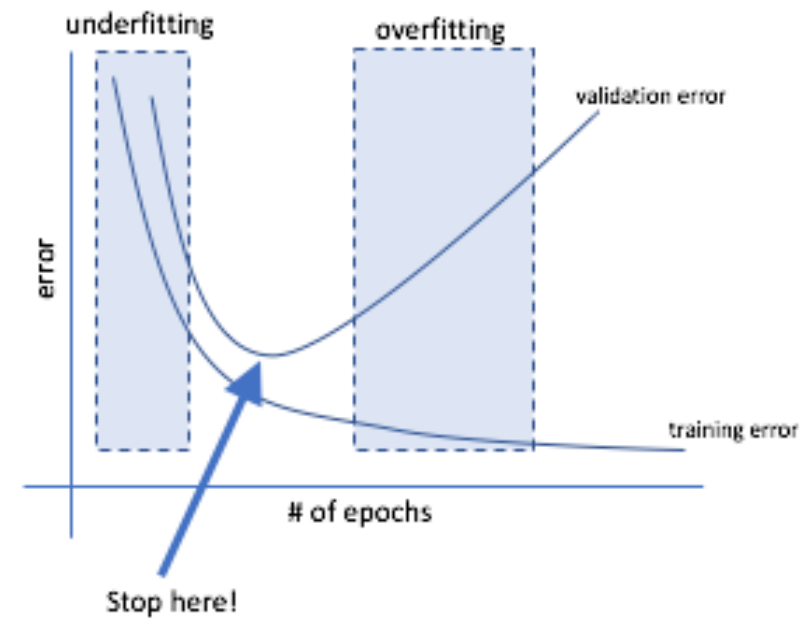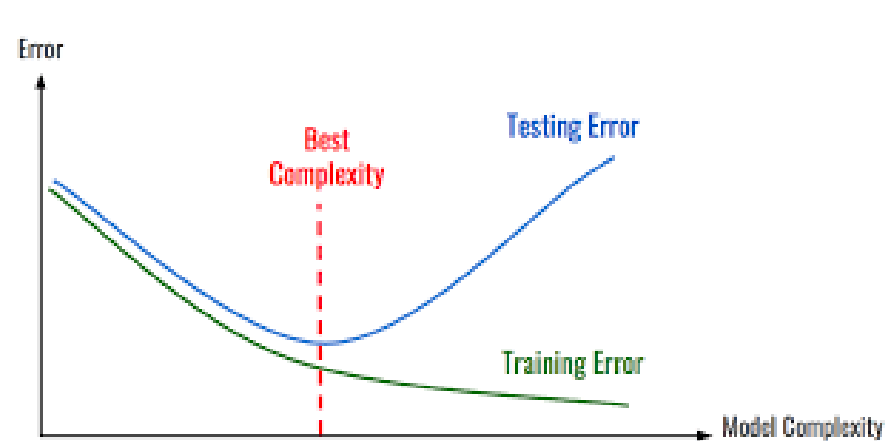
Motivation for Depth

Large, Shallow Models Overfit More

Motivation: Mimic the Brain Structure

Motivation

- Practical success in computer vision, signal processing, text mining

- Increase in volume and complexity of data

- Availability of GPUs

Convolutional Neural Network: Motivation

Hierarchical organization

Retinal ganglion cell receptive fields
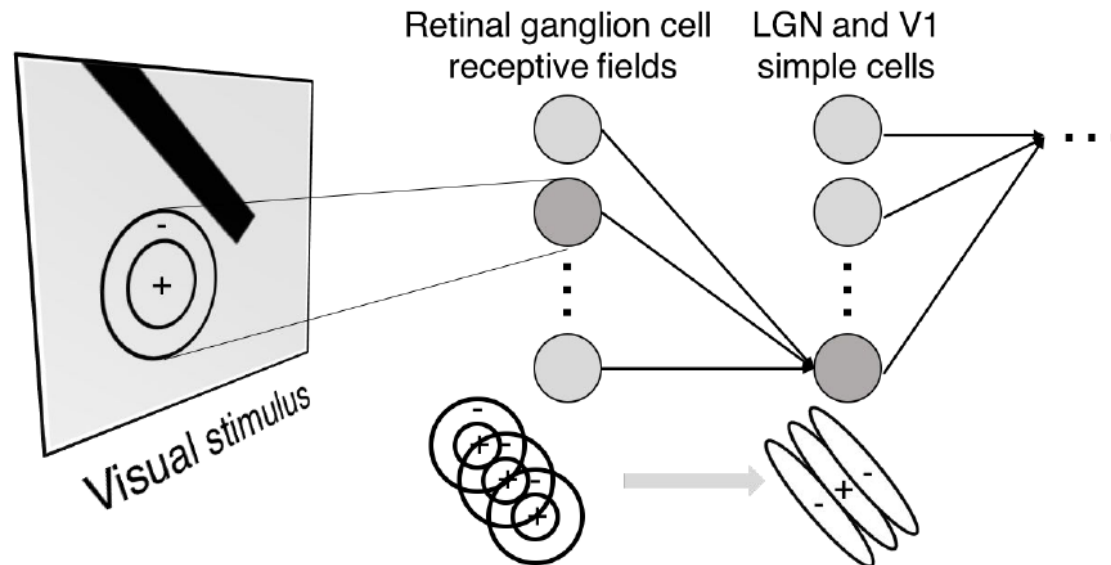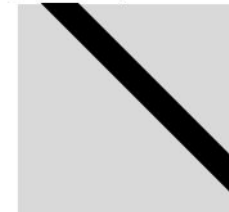
LGN and V1 simple cells

Visual stimulus

Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

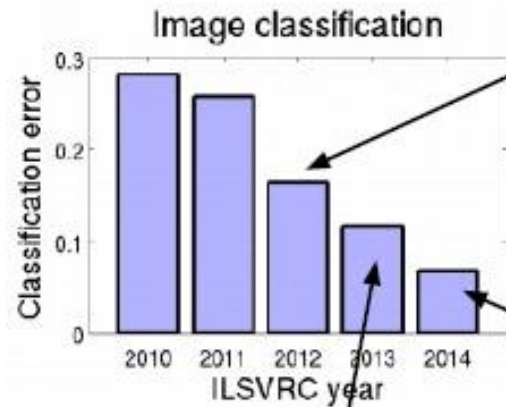**Complex cells:**
Response to light orientation and movement

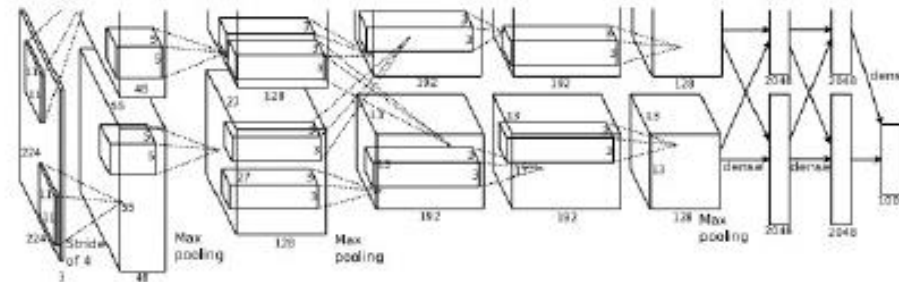**Hypercomplex cells:**
response to movement with an end point

No response

Response (end point)

Image classification

[Krizhevsky, Sutskever, Hinton. 2012] **16.4% error**

[Szegedy et al., 2014] **6.6% error**
[Simonyan and Zisserman, 2014] **7.3% error**

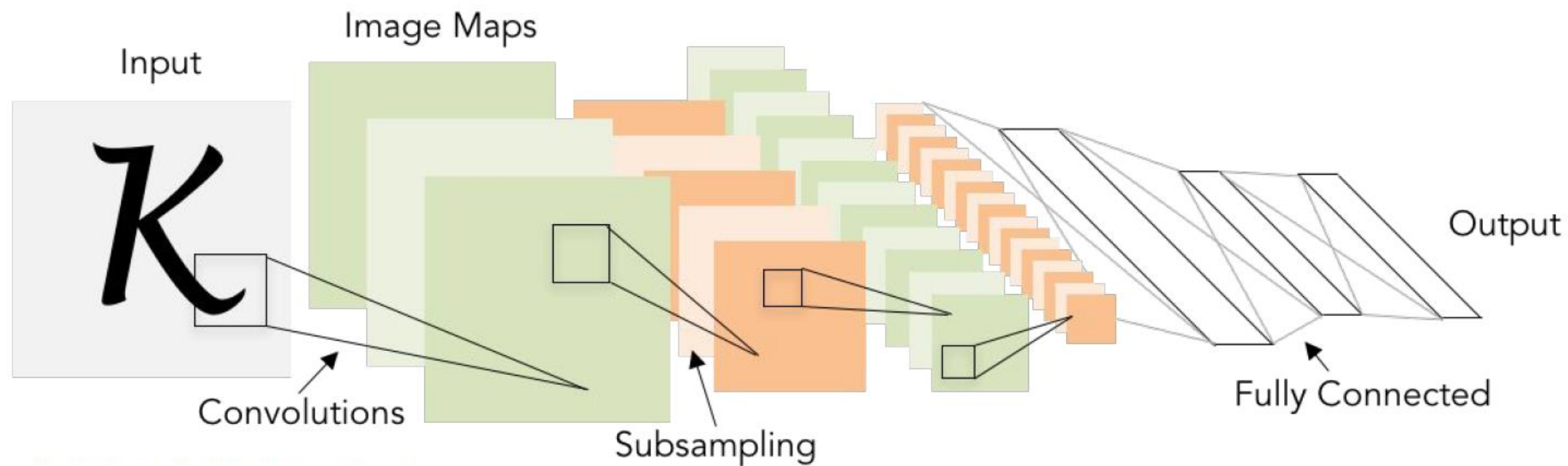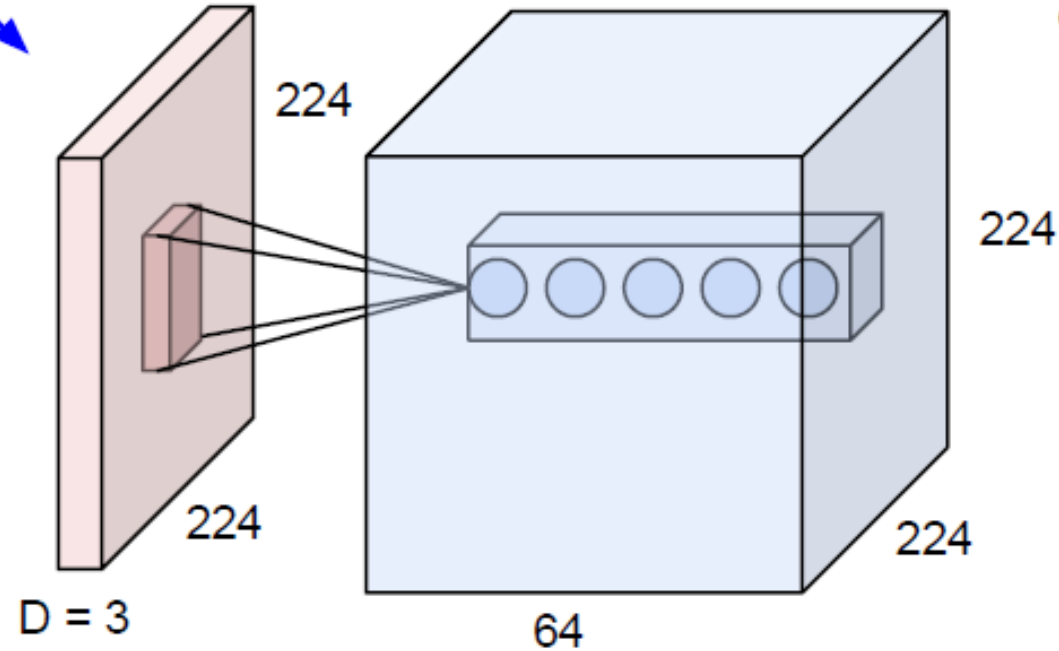[Zeiler and Fergus, 2013] **11.1% error**

CNN

Input

Image Maps

Convolutions

Subsampling

Output

Fully Connected

Convolutional Layer

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

Can be implemented efficiently with convolutions

224

224

224

224

$D = 3$

64

Every blue neuron is connected to a **3x3x3** array of inputs

Pooling Layer

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

224

224

downsampling

112

112

Max Pooling Layer

Single depth slice

Fully Connected Layer

image
conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096
FC-1000
softmax

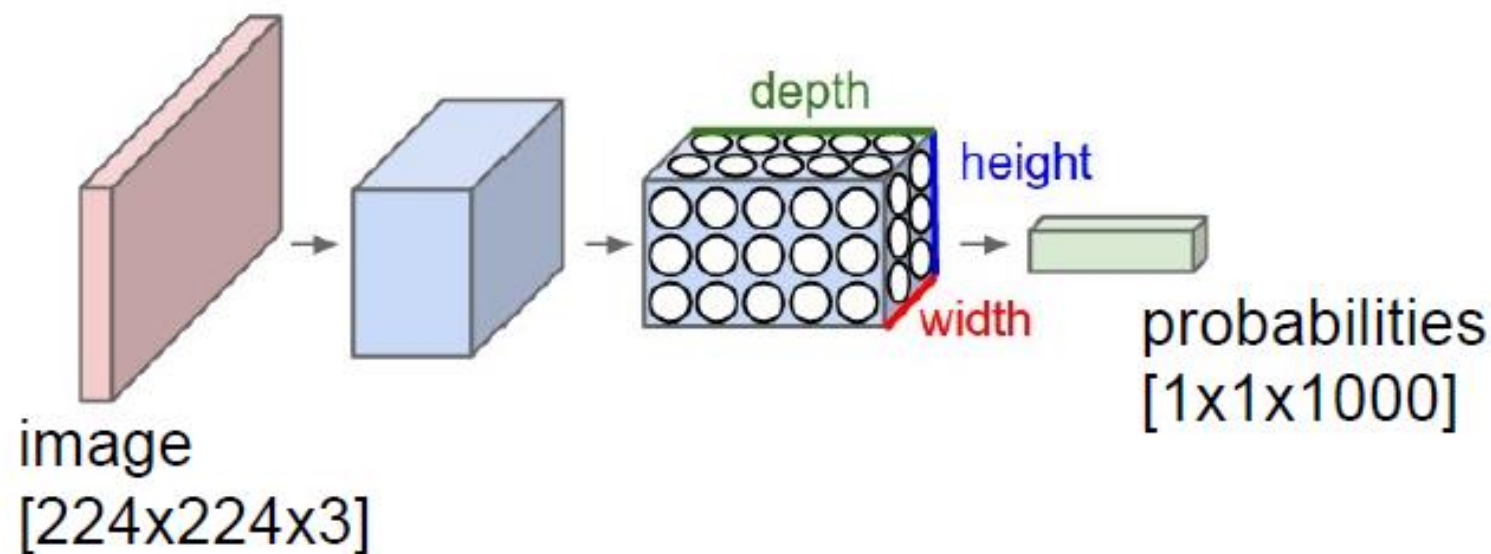[7x7x512]

[1x1x4096] "neurons"
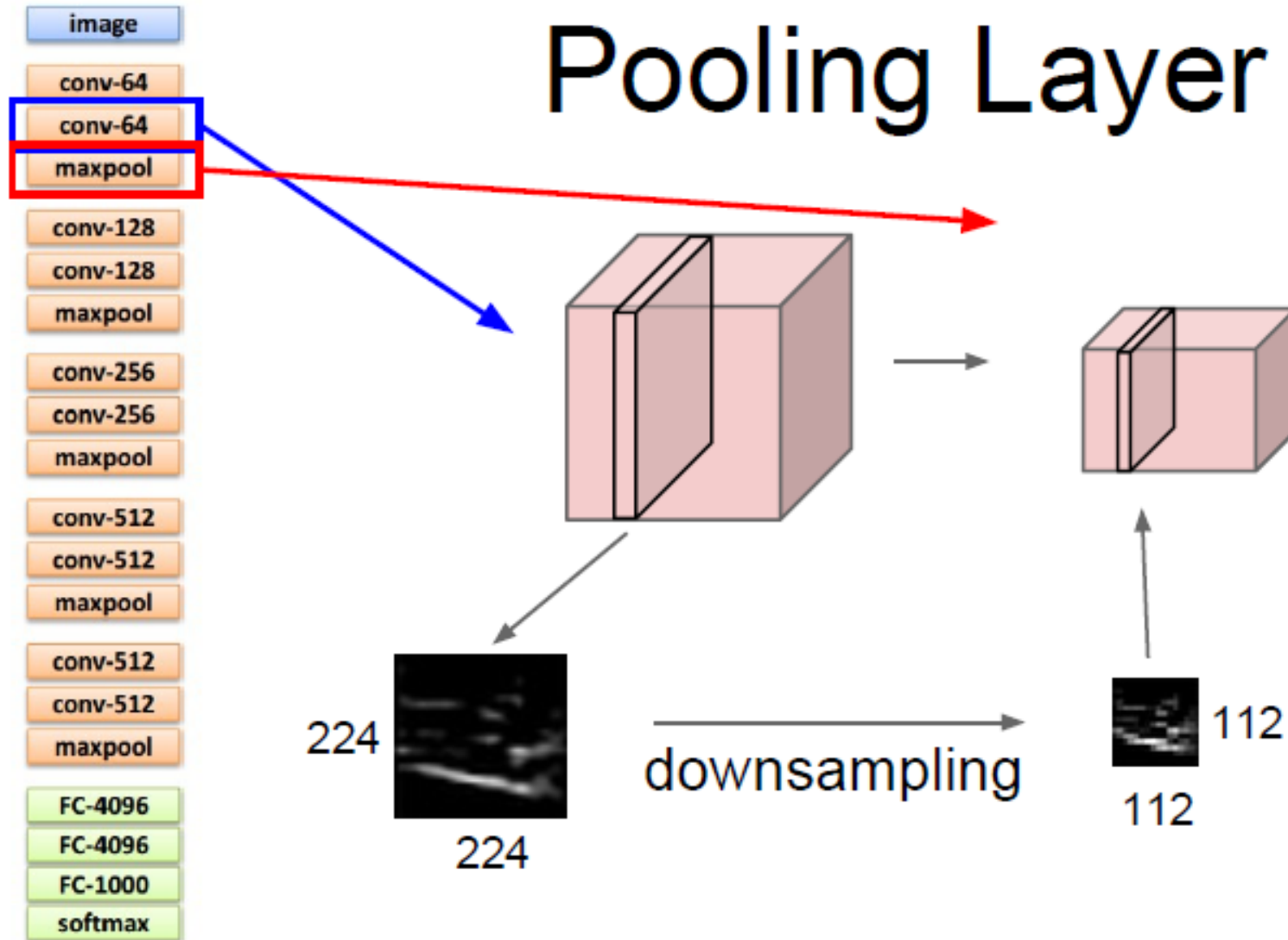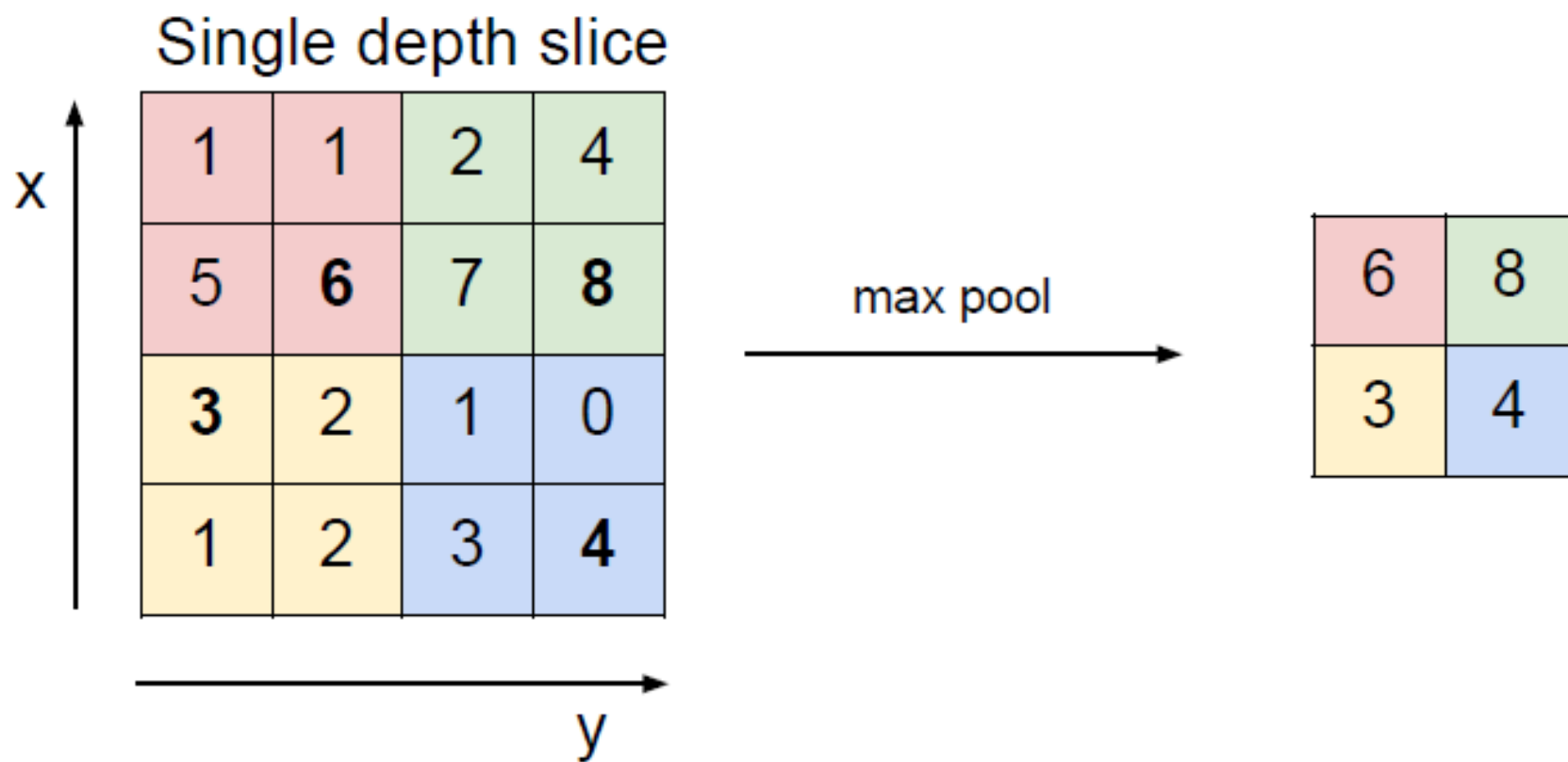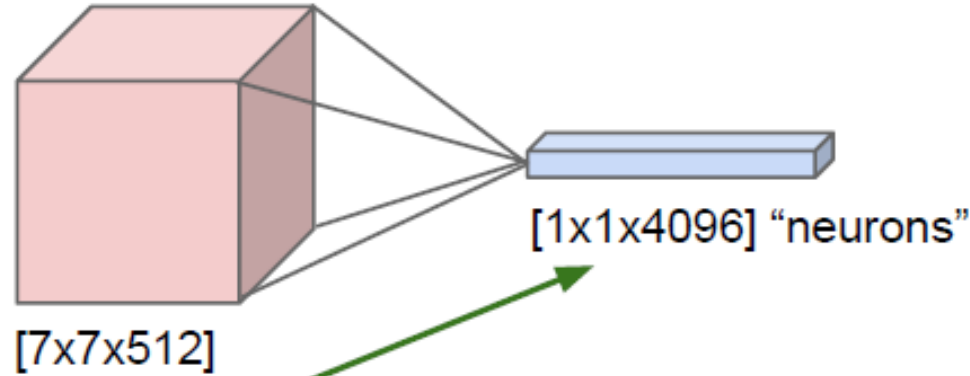
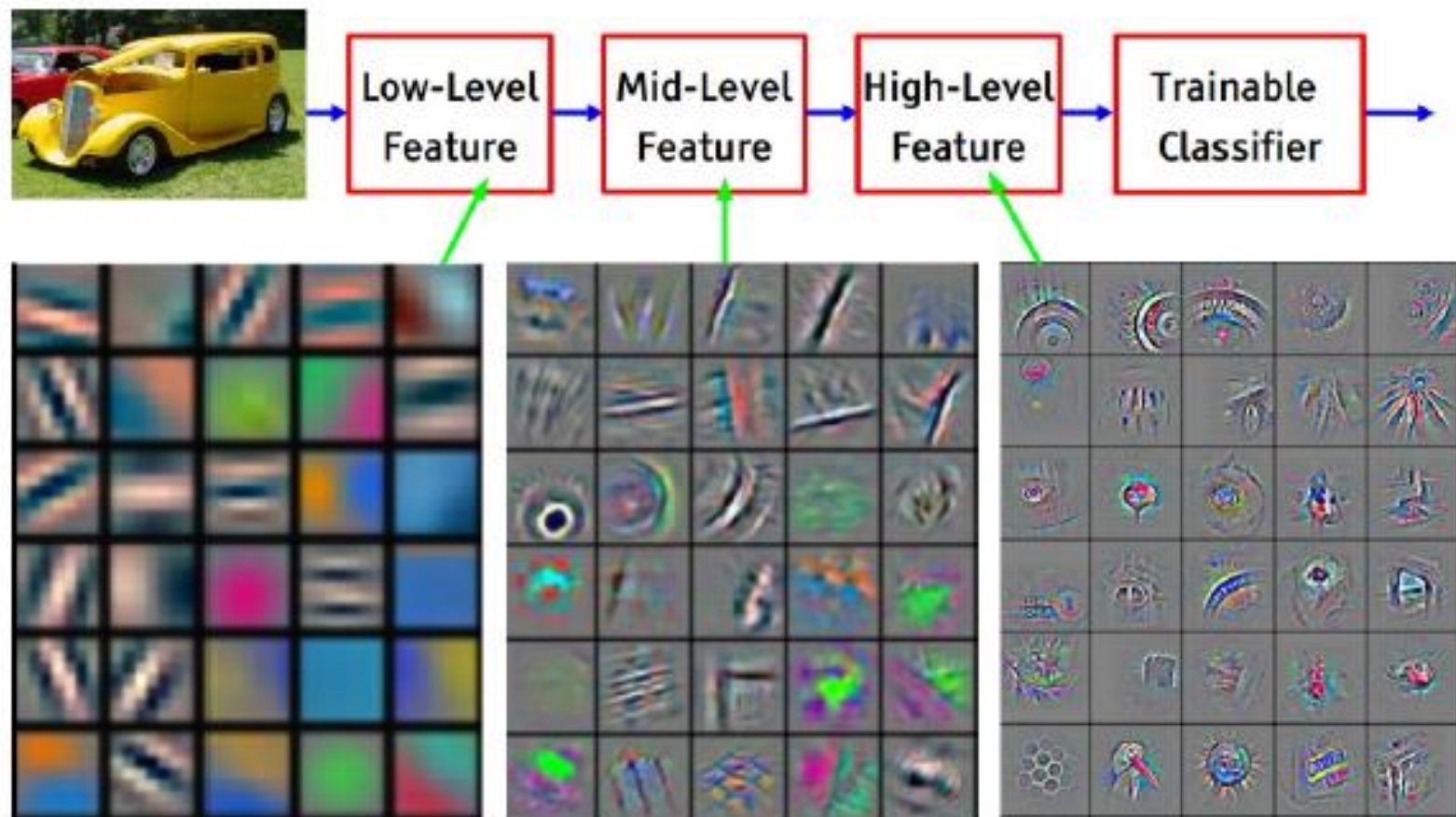**Every "neuron" in the output:**

1. computes a **dot product** between the input and its weights

$$f = w^T x + b$$

2. **thresholds** it at zero

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

Every layer of a ConvNet has the same API:
- Takes a 3D volume of numbers
- Outputs a 3D volume of numbers
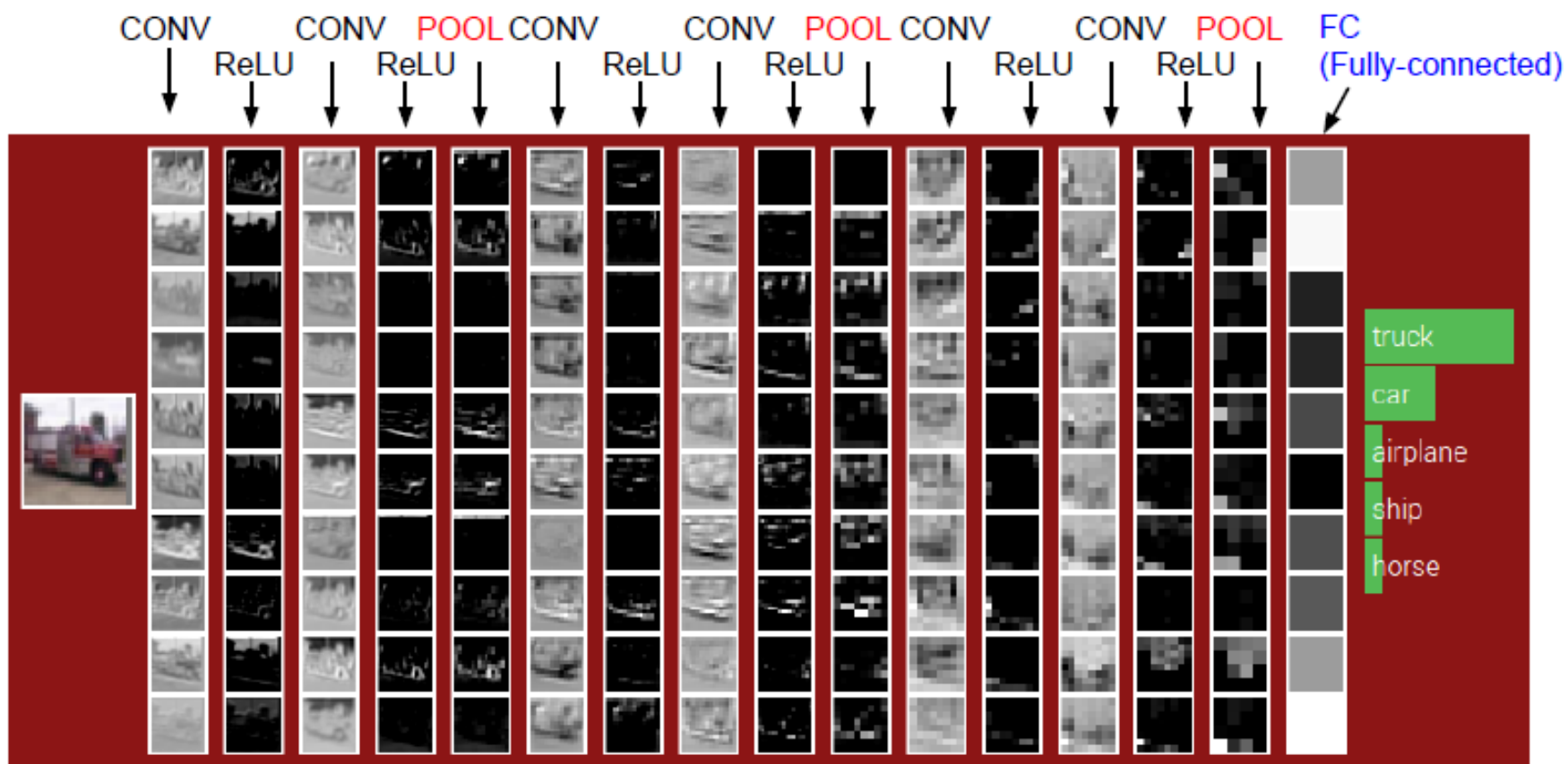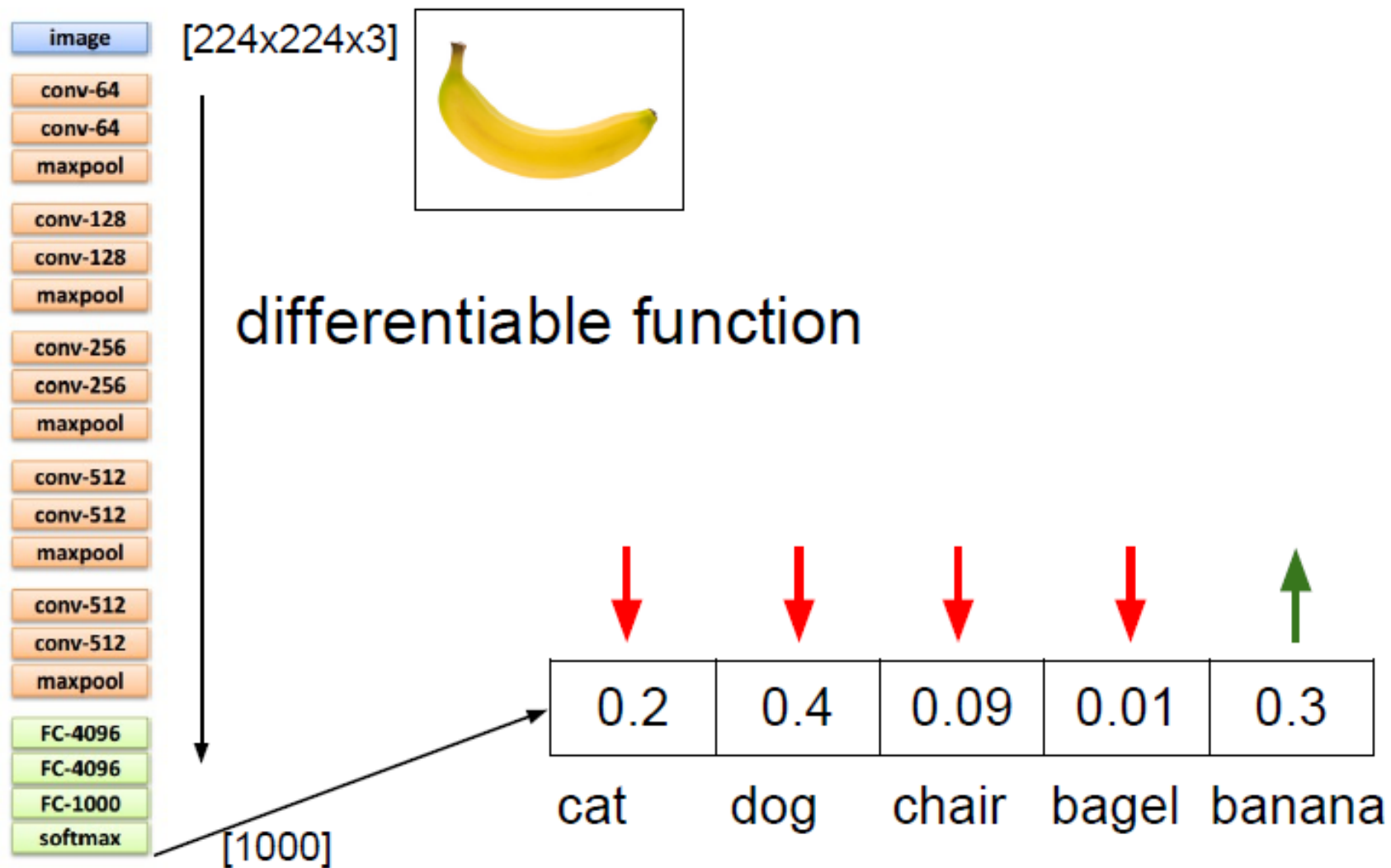- Constraint: function must be **differentiable**

image
[224x224x3]

probabilities
[1x1x1000]

What do the neurons learn?

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Example activation maps

image

[224x224x3]

| conv-64 |
| conv-64 |
| maxpool |

| conv-128 |
| conv-128 |
| maxpool |

differentiable function

| conv-256 |
| conv-256 |
| maxpool |

| conv-512 |
| conv-512 |
| maxpool |

| conv-512 |
| conv-512 |
| maxpool |

| FC-4096 |
| FC-4096 |
| FC-1000 |
| softmax |

[1000]

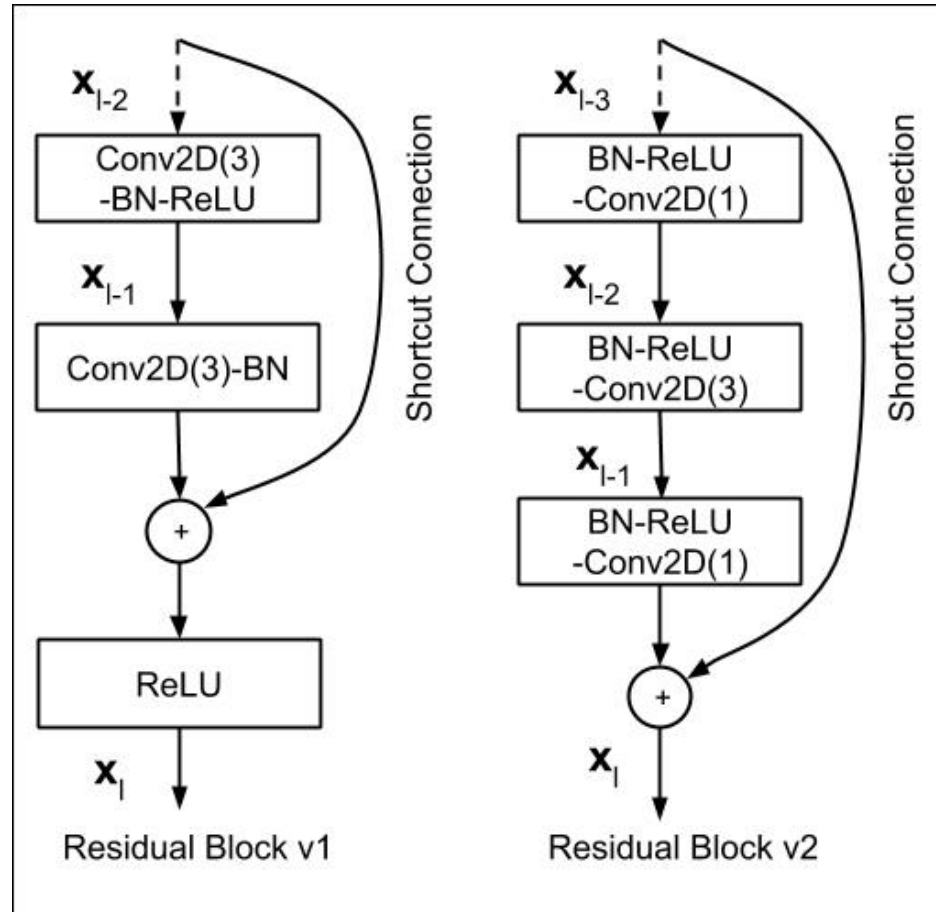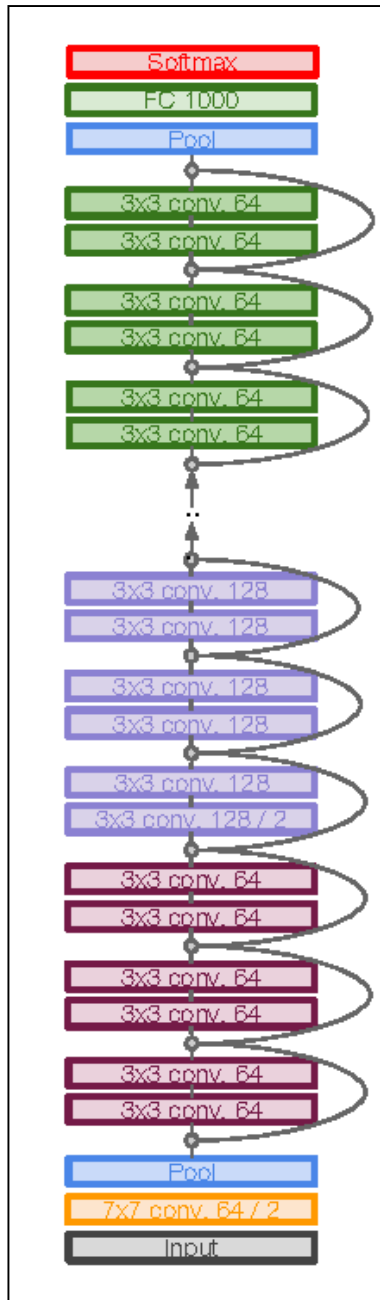| 0.2	0.4	0.09	0.01	0.3
cat	dog	chair	bagel	banana

Training

Loop until tired:
1. **Sample** a batch of data
2. **Forward** it through the network to get predictions
3. **Backprop** the errors
4. **Update** the weights

ResNet

CNN + Skip Connections

Pyramidal cells in cortex

Full ResNet architecture:

- Stack residual blocks

- Every residual block has two 3x3 conv layers

- Periodically, double # of filters and downsample spatially using stride 2 (in each dimension)

- Additional conv layer at the beginning

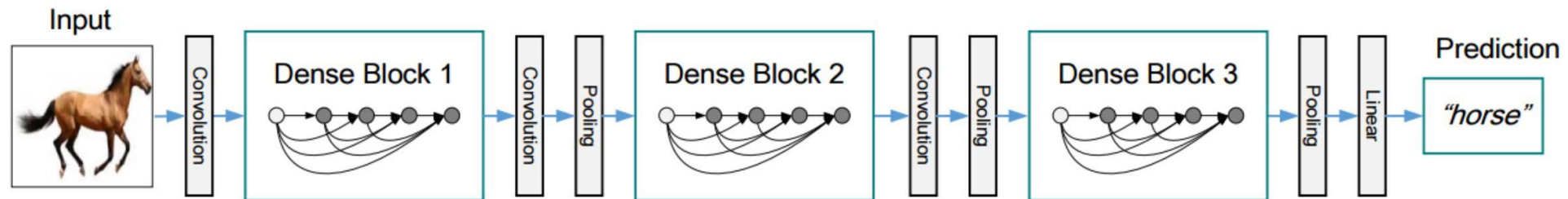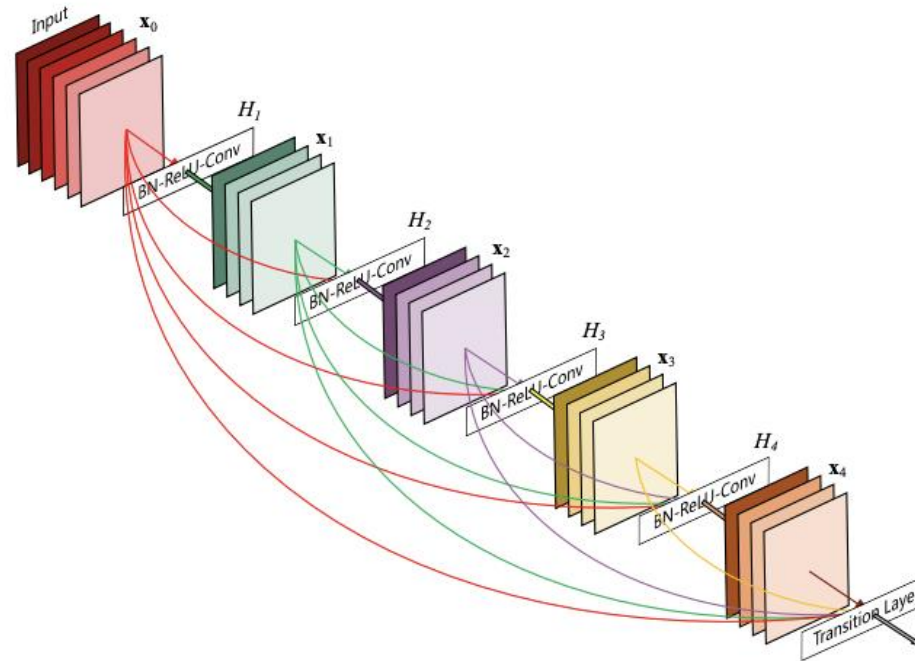- No FC layers at the end (only FC 1000 to output classes)

Densenet

Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.

Challenges of Depth

- Overfitting – dropout

- Vanishing gradient – ReLU activation

- Accelerating training – batch normalization

- Hyperparameter tuning

Computational Complexity

Comparing complexity...
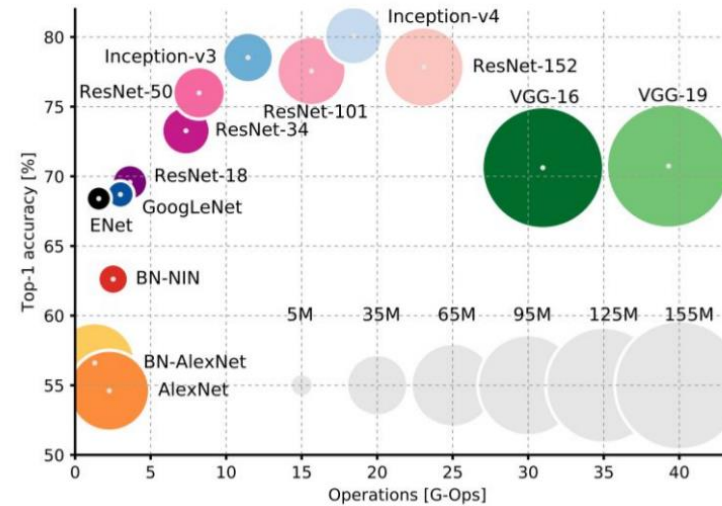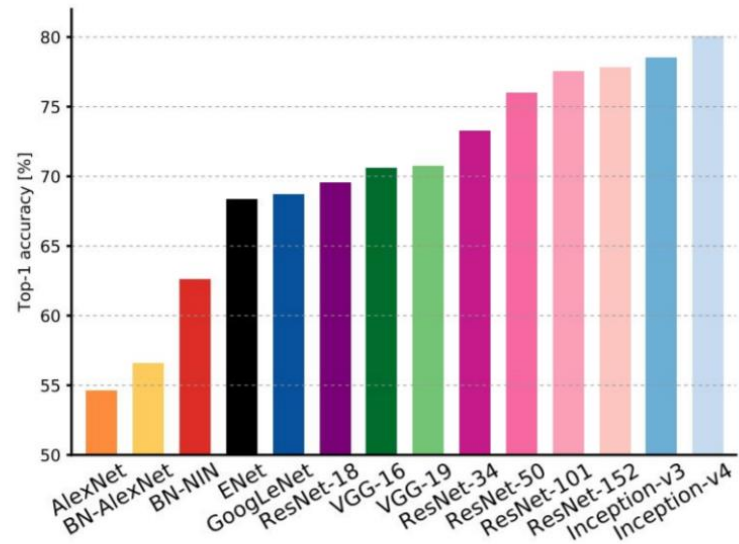
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Types of Deep Architectures

- RNN, LSTM (sequence learning)

- Stacked Autoencoders (representation learning)

- GAN (classification, distribution learning)

- Combining architectures – unified backprop if all layers differentiable
  - Tensorflow, PyTorch

References

- Introduction to Deep Learning – Ian Goodfellow

- Stanford Deep Learning course