# Convolutional Neural Networks

**Danna Gurari**
University of Texas at Austin
Spring 2021

https://www.ischool.utexas.edu/~dannag/Courses/IntroToMachineLearning/CourseContent.html

# Review

- Last class:
  - History of Neural Networks
  - Neural Network Architecture – Hidden Layers and Solving XOR Problem
  - Neural Network Architecture – Output Units
  - Training a Neural Network – Optimization
  - Training a Neural Network – Activation Functions & Loss Functions
- Assignments (Canvas):
  - Lab assignment 3 due tonight
  - Project proposal due next week
- Questions?

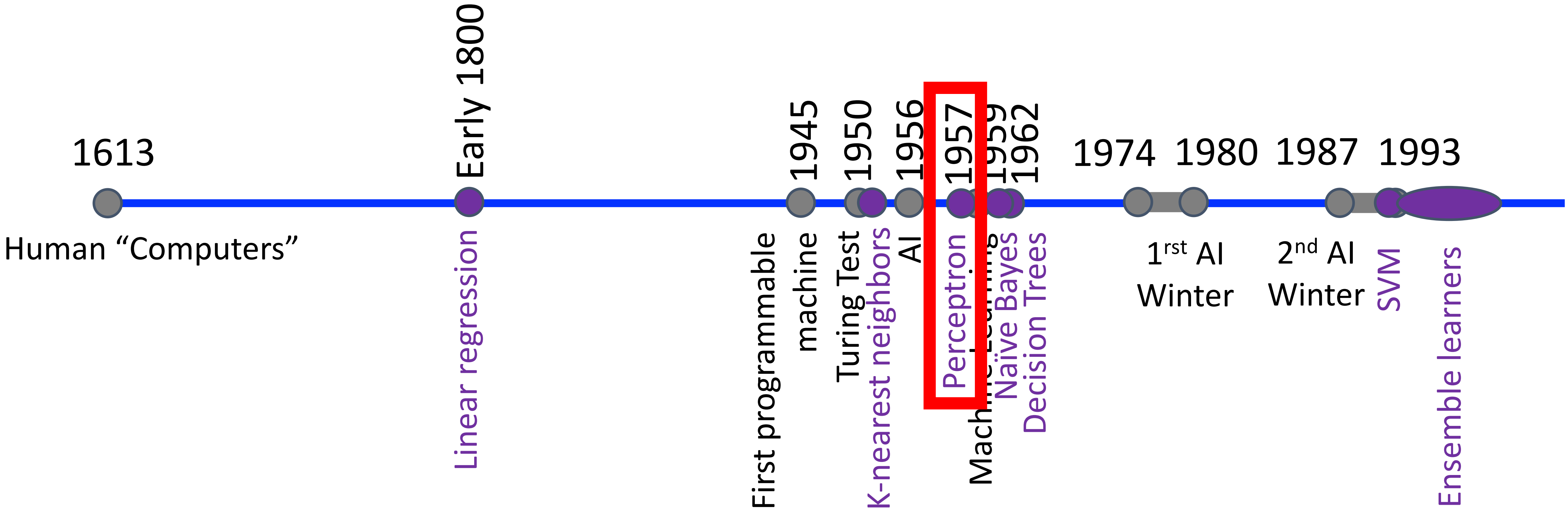
# Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Deep Features
- Guest Speaker: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI

# Today’s Topics

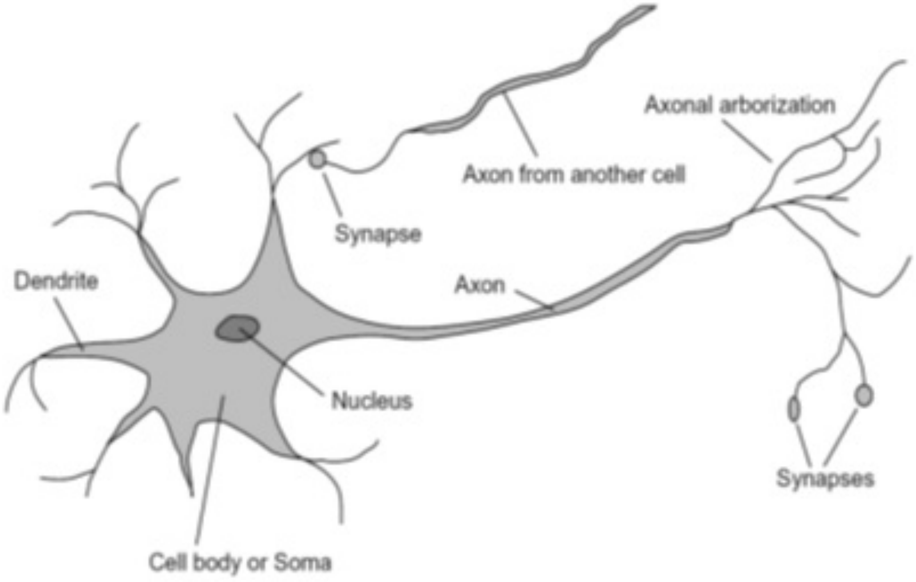
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Deep Features
- Guest Speaker: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI

# Recall:

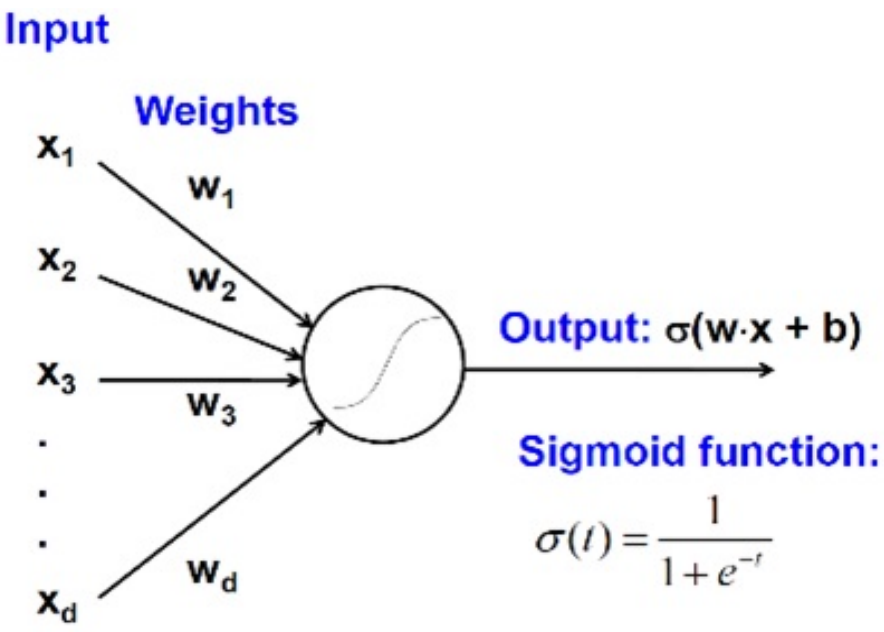


1613
Human “Computers”
Early 1800
Linear regression
1945
First programmable machine
1950
Turing Test
K-nearest neighbors
1956
AI
1957
Perceptron
Naive Bayes
1962
Decision Trees
1974
1980
1rst AI Winter
1987
1993
2nd AI Winter
SVM
Ensemble learners

# Recall:



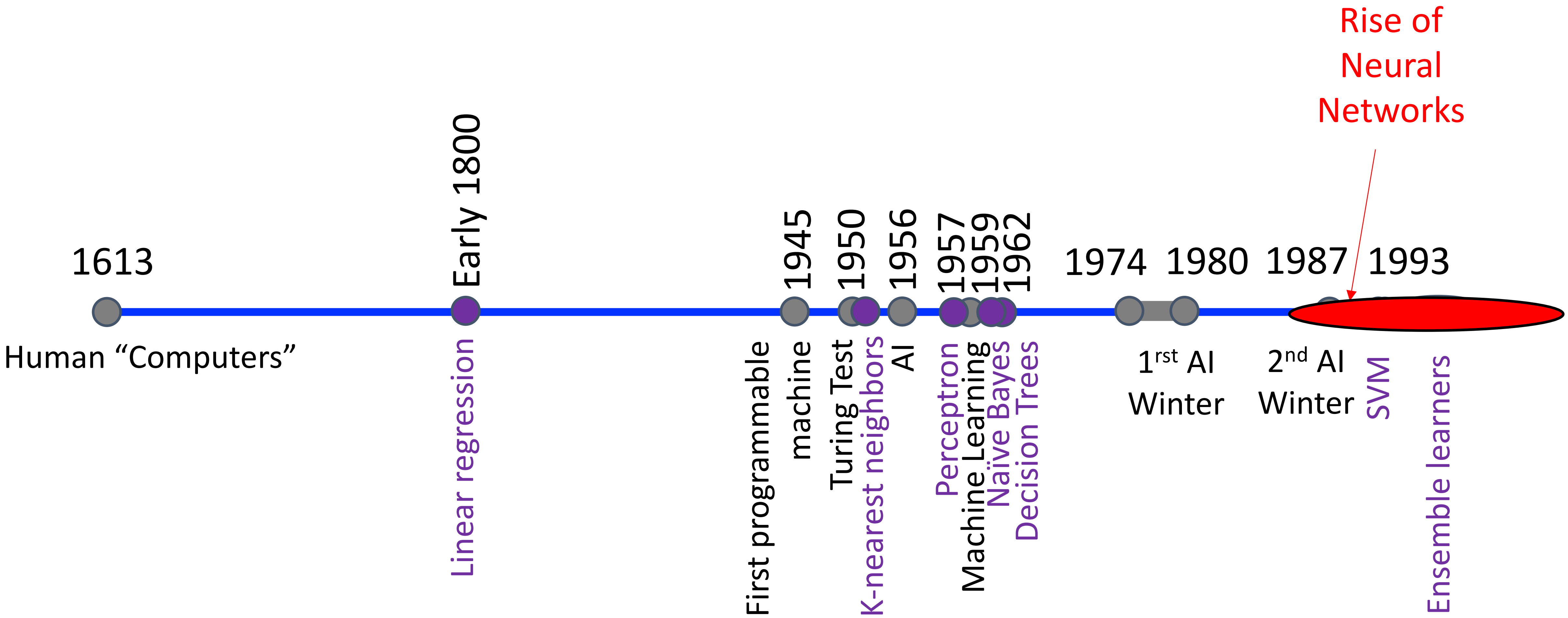

A biological neuron




An artificial neuron (Perceptron)
- a linear classifier

Slide credit: Jia-Bin Huang

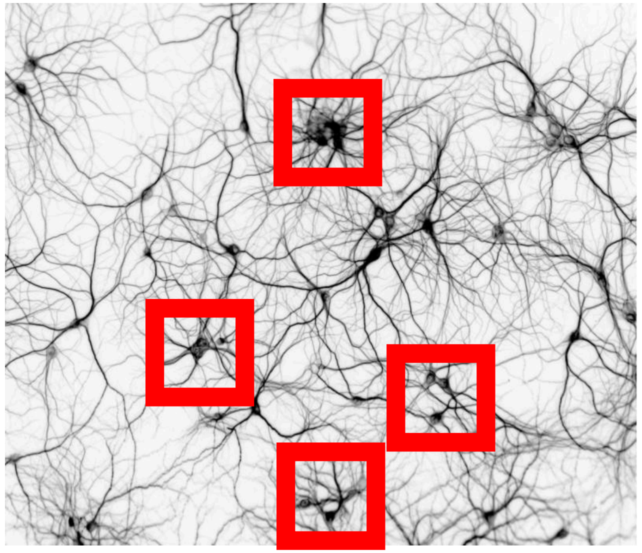
# Recall:



Rise of Neural Networks
1613
Human "Computers"
Early 1800
Linear regression
1945
First programmable machine
1950
Turing Test
K-nearest neighbors
1956
AI
1957
Perceptron
1959
Machine Learning
Naive Bayes
1962
Decision Trees
1974
1980
1rst AI Winter
1987
2nd AI Winter
1993
SVM
Ensemble learners

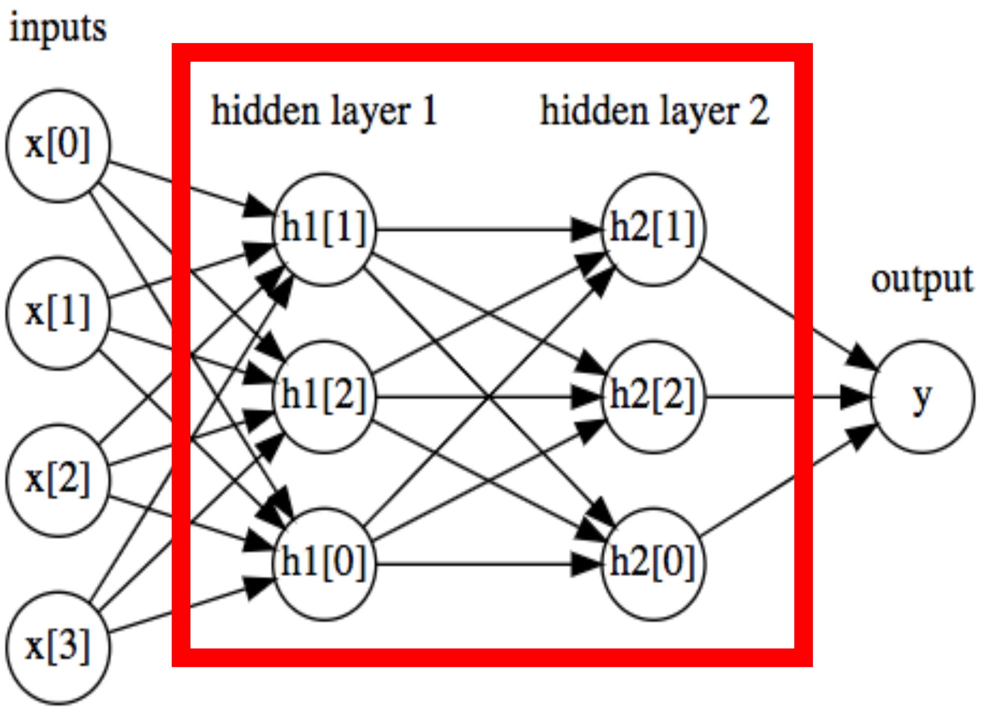
# Recall:

Biological Neural Network:



http://www.rzagabe.com/2014/11/03/an-introduction-to-artificial-neural-networks.html

Artificial Neural Network:




https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb

# Motivation: How Vision System Works

Neuroscientific experiments by Hubel & Weisel to understand how mammalian vision system works

Nobel Prize in Physiology & Medicine to Hubel & Weisel for their accomplishments

Rise of Neural Networks

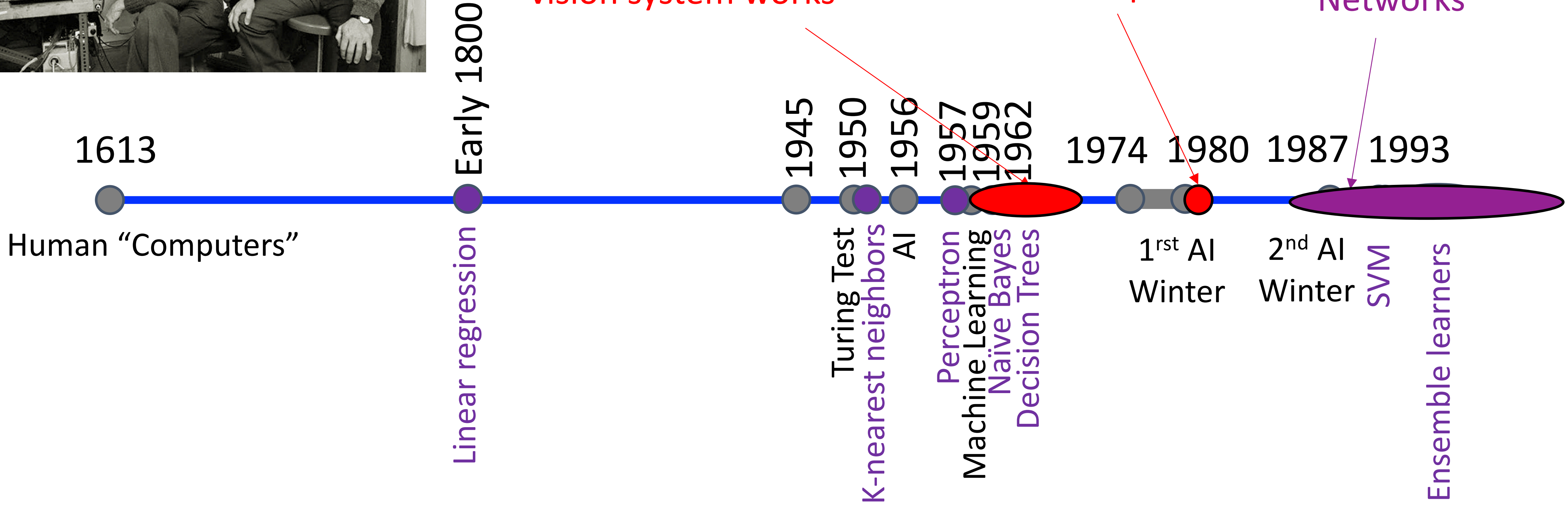
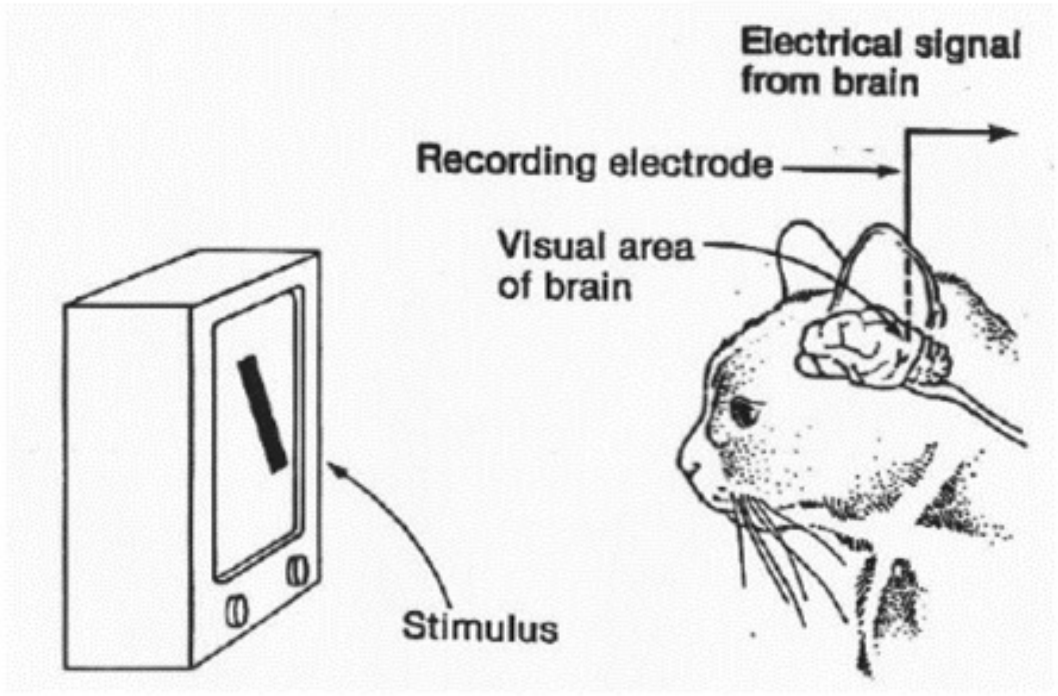



Image Source: https://braintour.harvard.edu/archives/portfolio-items/hubel-and-wiesel

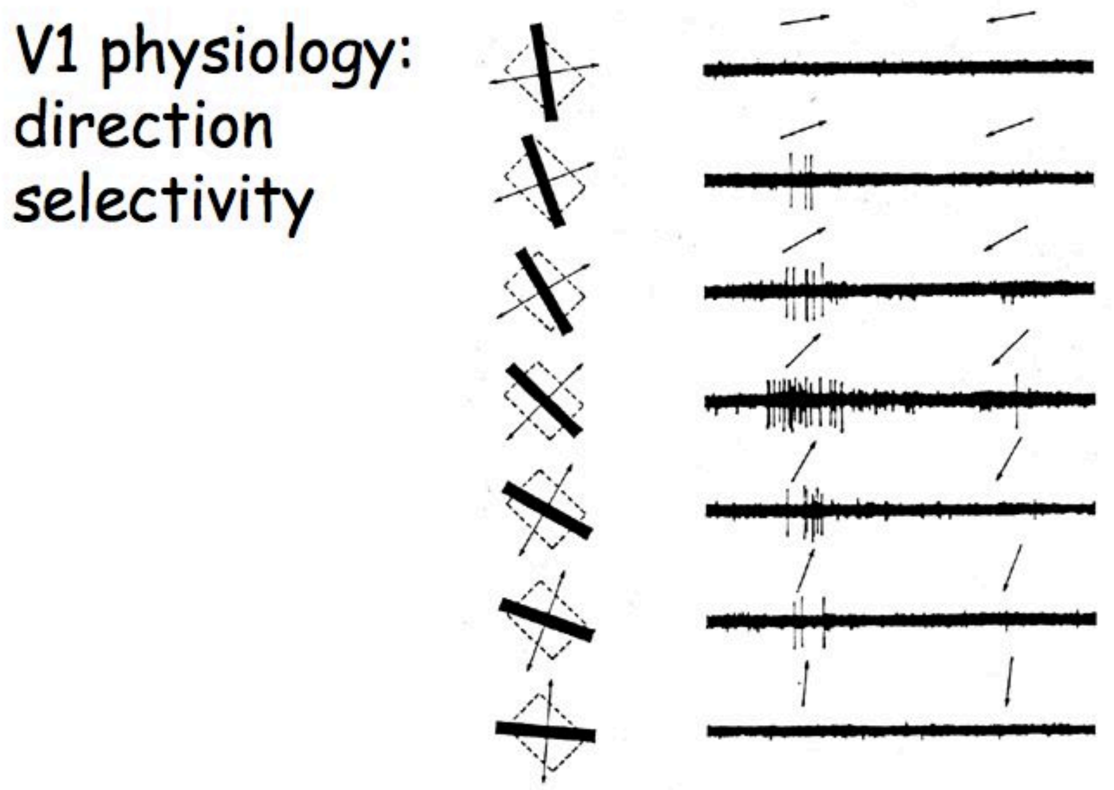
# Motivation: How Vision System Works

Experiment Set-up:




https://www.esantus.com/blog/2019/1/31/convolutional-neural-networks-a-quick-guide-for-newbies

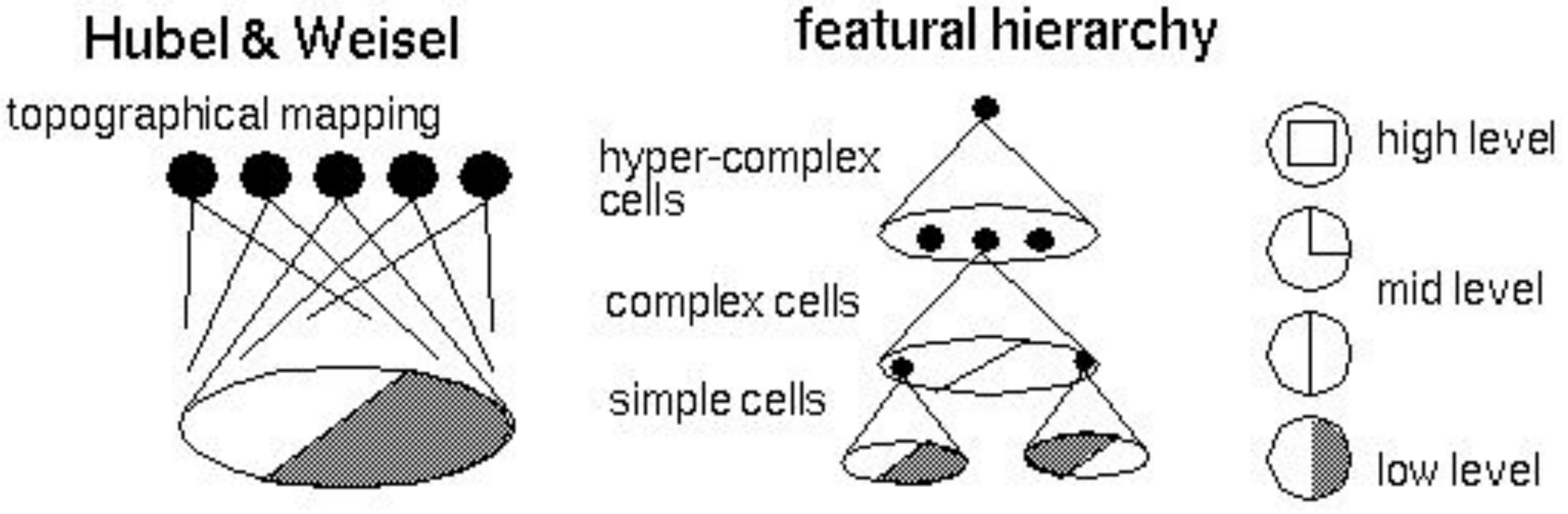
Key Finding: response based on orientation of light stimulus




https://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/lgn-V1.html

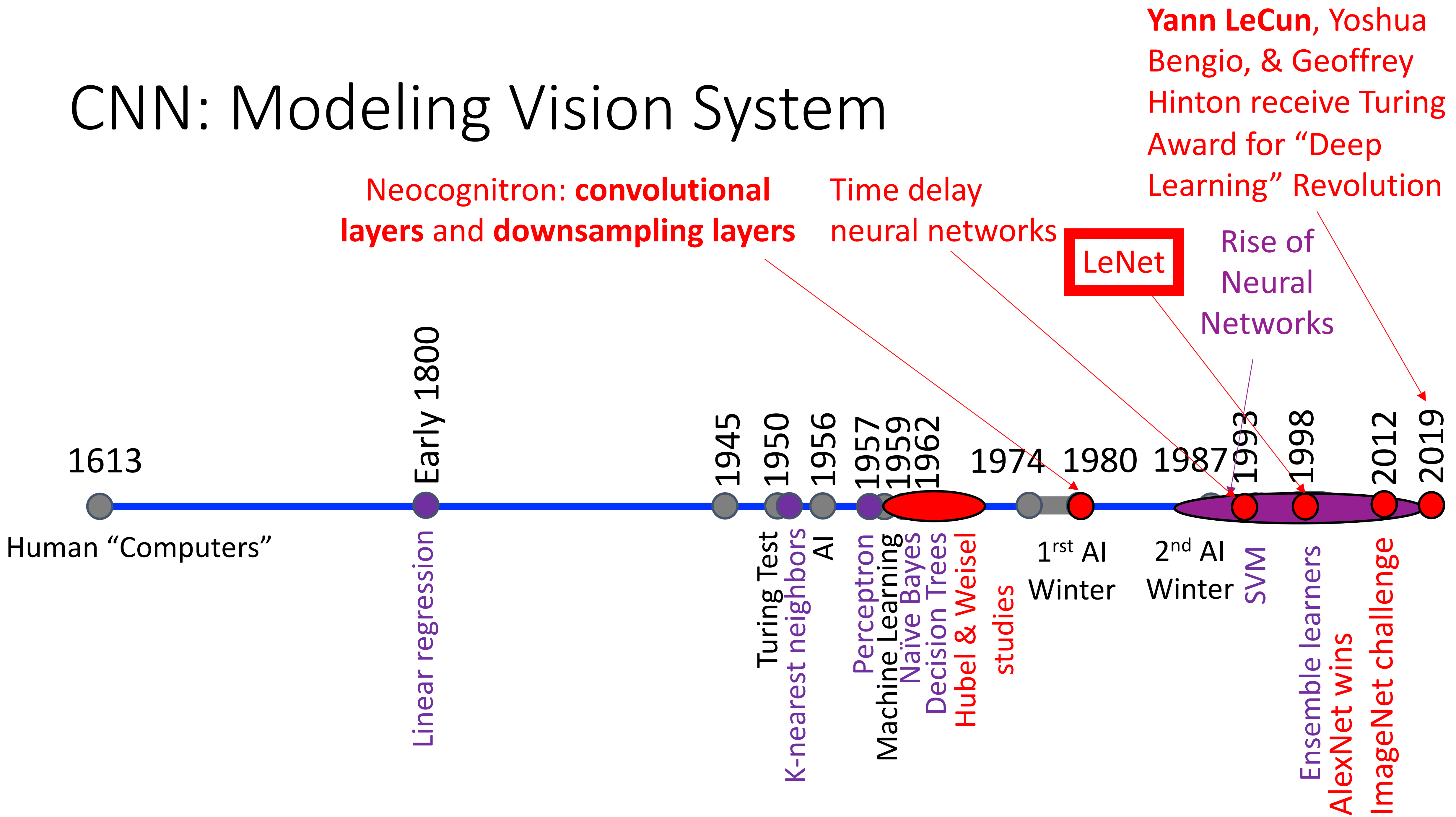
# Motivation: How Vision System Works

Key Finding: cells are organized as a hierarchy of feature detectors, with higher level features responding to patterns of activation in lower level cells

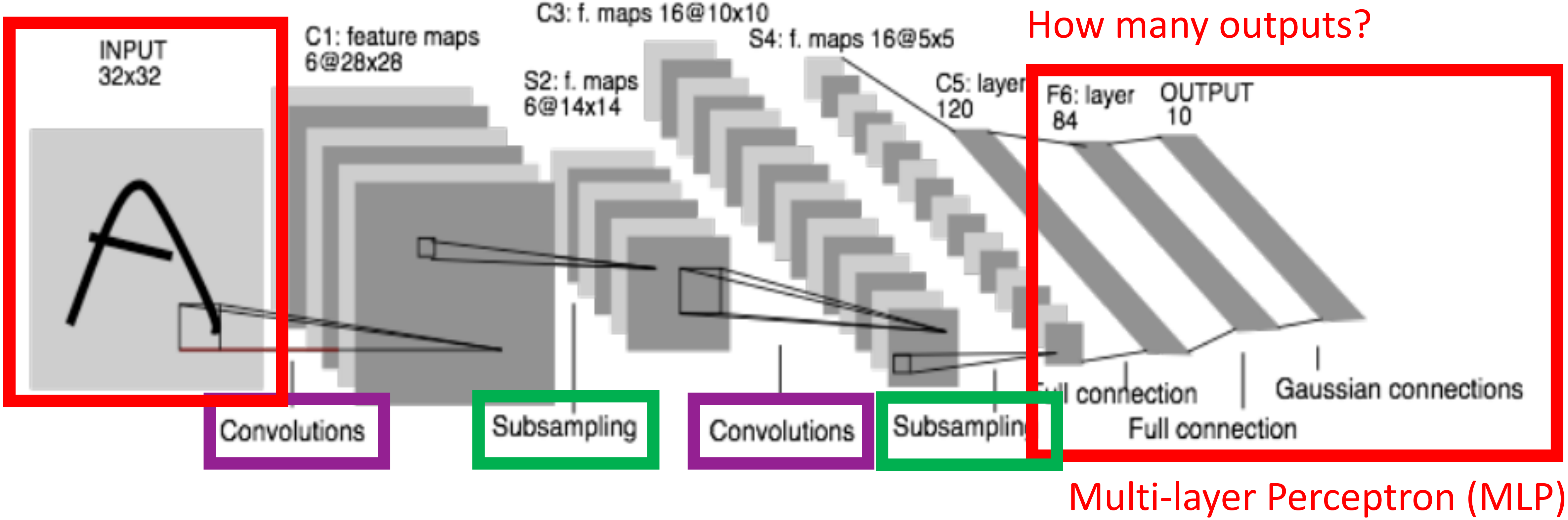



Source: https://bruceoutdoors.files.wordpress.com/2017/08/hubel.jpg

# CNN: Modeling Vision System



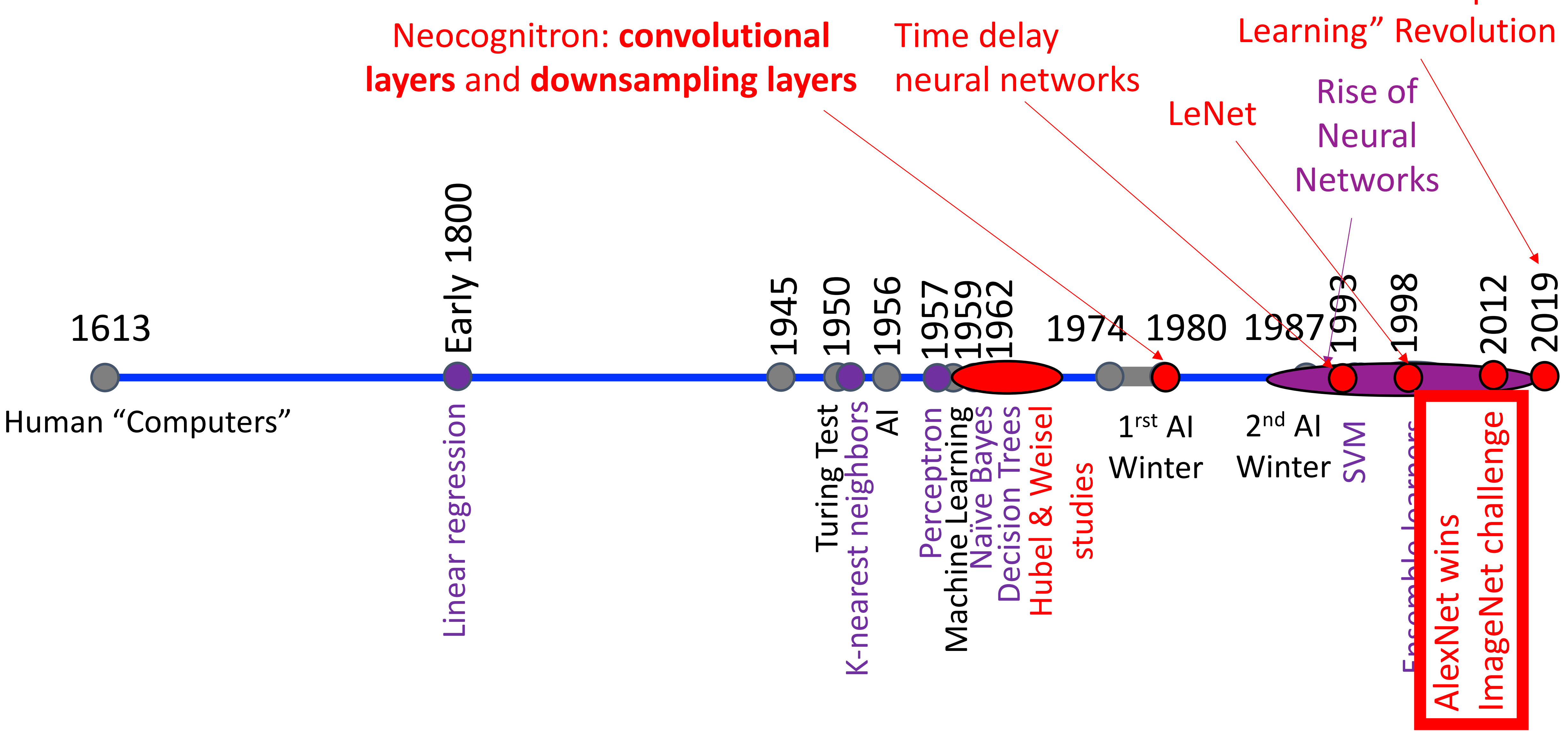
# CNN: Modeling Vision System




Slide Credit: https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

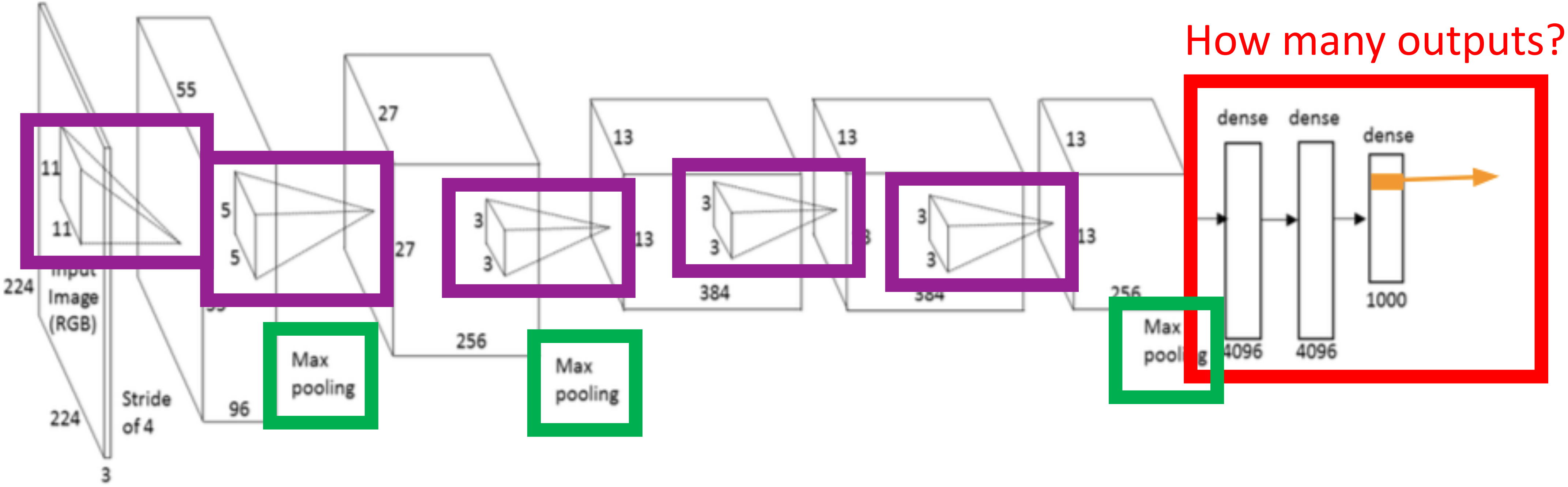
Y. Lecun ; L. Bottou ; Y. Bengio ; P. Haffner; Gradient-based learning applied to document recognition; 1998

# CNN: Modeling Vision System



# CNN: Modeling Vision System

- AlexNet extracts useful features of lower dimension prior to passing it to MLP with:
  - Convolutional layers
  - Pooling Layers



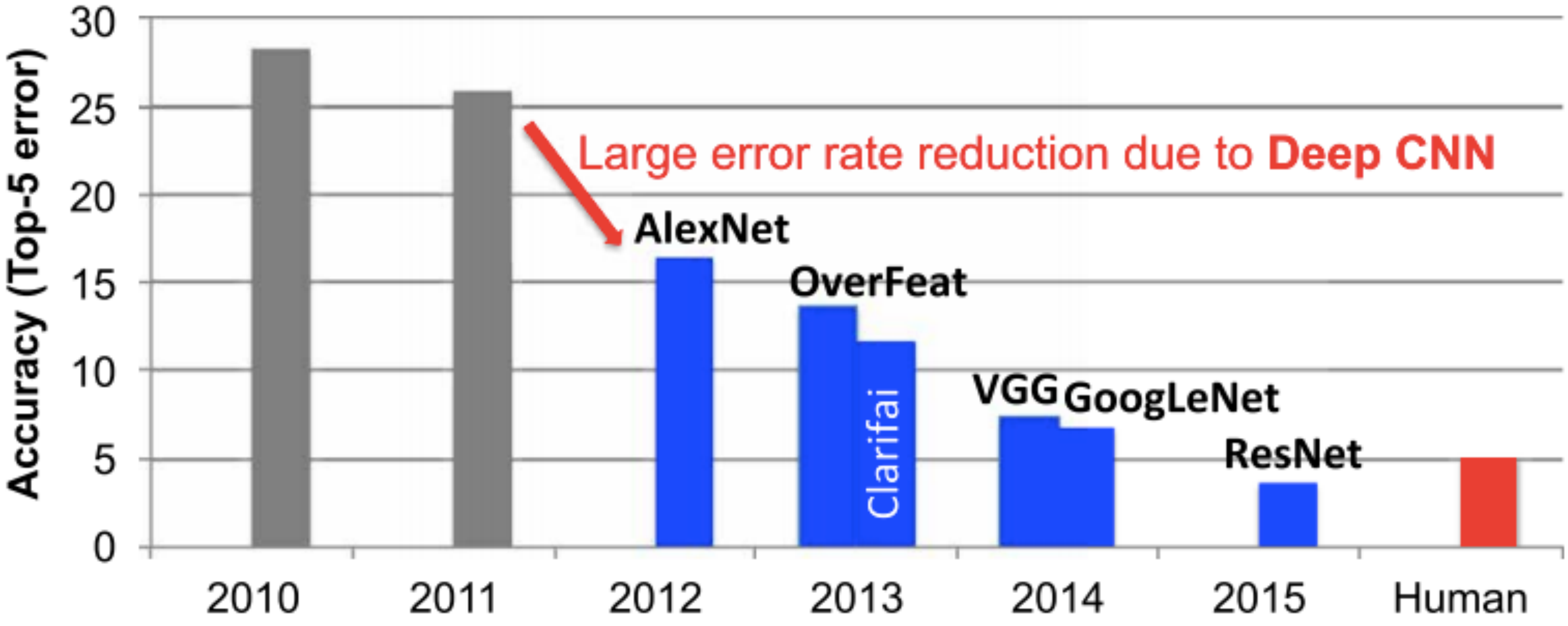

Slide Credit: https://www.slideshare.net/xavigiro/saliency-prediction-using-deep-learning-techniques

A. Krizhevsky, I. Sutskever, G. E. Hinton “ImageNet classification with deep convolutional neural networks”

# ImageNet: Predict Category from 1000 Options

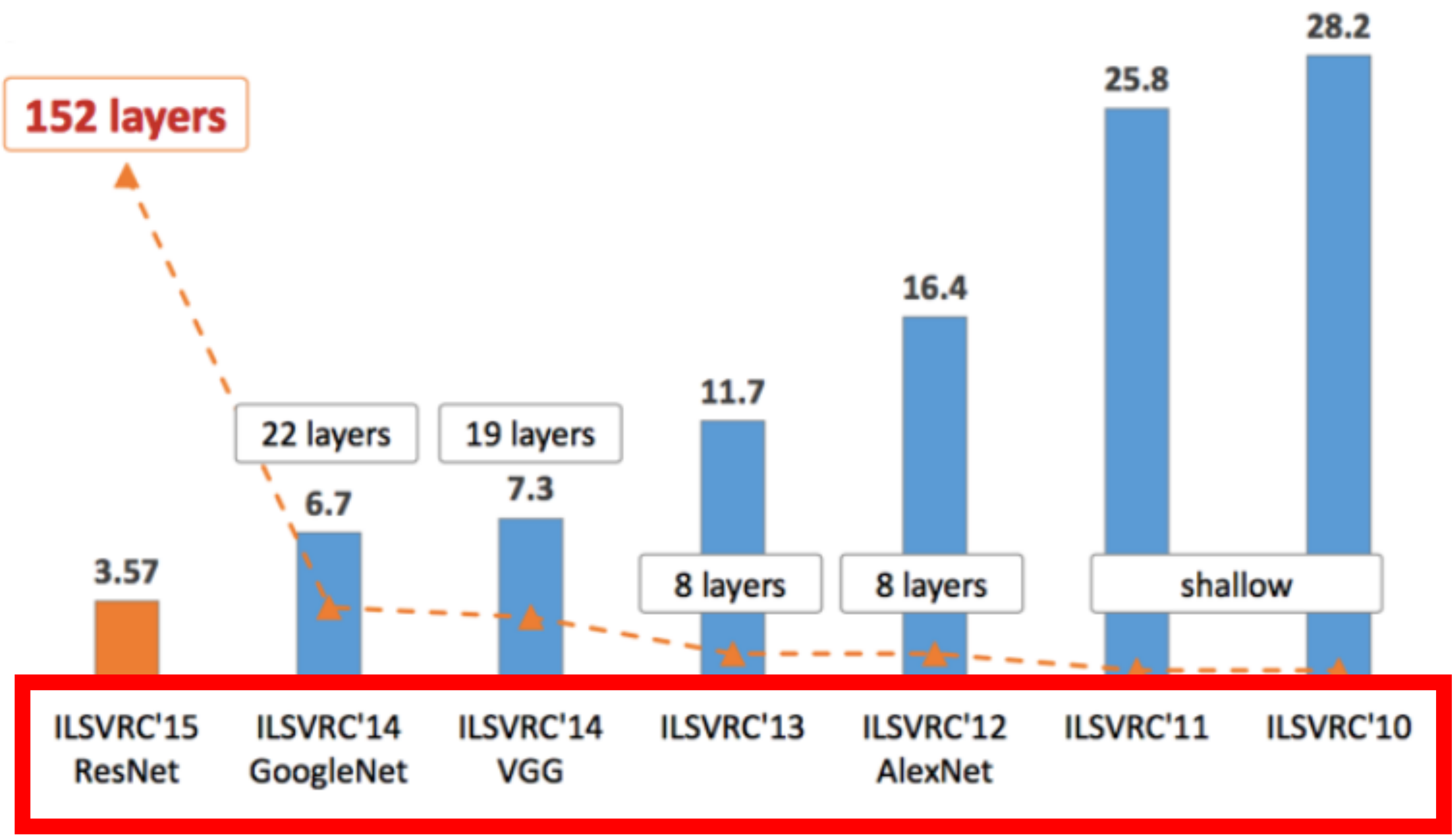

https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11

# ILSVRC: Top CNN Models Over Time



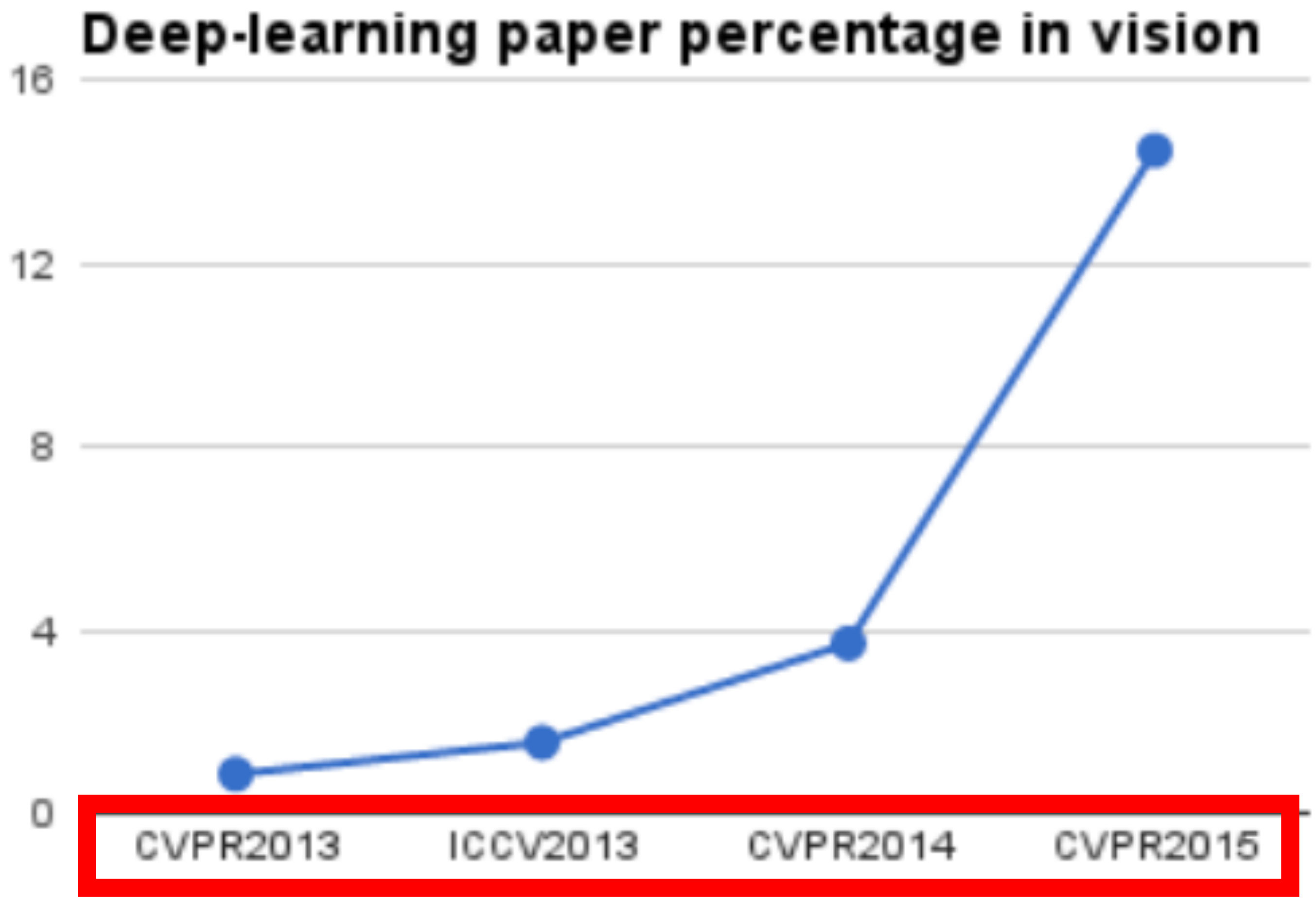

https://people.csail.mit.edu/emer/papers/2017.12.pieee.DNN_hardware_survey.pdf

# ILSVRC: Top CNN Models Over Time



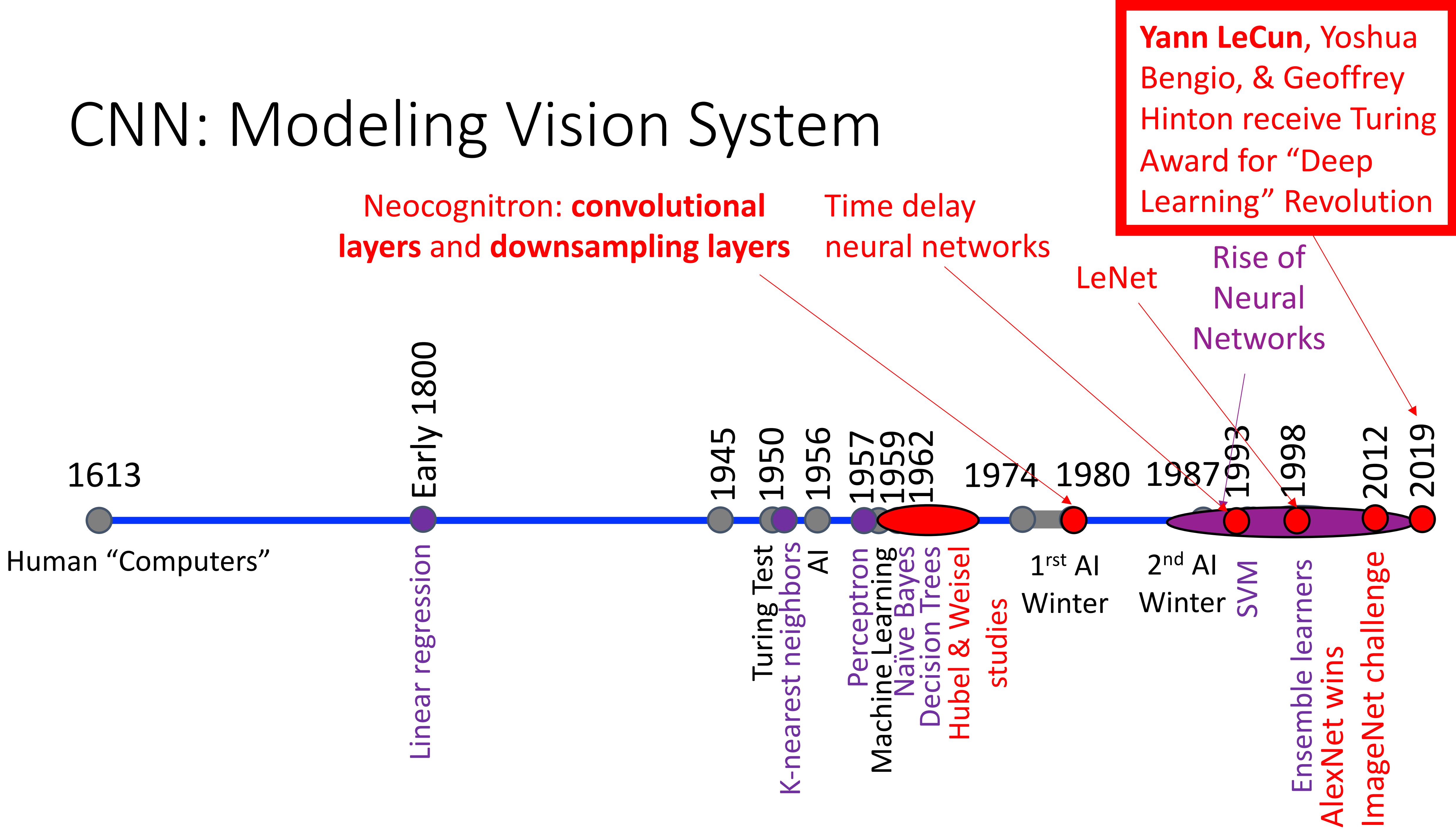

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

# CNN: Modeling Vision System




Slide Credit: https://www.slideshare.net/xavigiro/saliency-prediction-using-deep-learning-techniques

# CNN: Modeling Vision System



# Note: Initial Resistance to this "Revolution"

Yann LeCun's letter to CVPR organizer about 2012 paper submission:
(*Paper ratings: "Definitely Reject," "Borderline", "Weakly Reject"* )

"… I was very sure that this paper was going to get good reviews because: 1) it has two simple and generally applicable ideas for segmentation ("purity tree" and "optimal cover"); **2) it uses no hand-crafted features (it's all learned all the way through. Incredibly, this was seen as a negative point by the reviewers!); 3) it beats all published results on 3 standard datasets for scene parsing; 4) it's an order of magnitude faster than the competing methods.**

If that is not enough to get good reviews, I just don't know what is."

"*Scene Parsing with Multiscale Feature Learning, Purity Trees, and Optimal Covers*" rejected by CVPR but accepted by ICML'12

# Note: Initial Resistance to this “Revolution”

Yann LeCun’s Facebook post on March 28, 2019 after receiving Turing Award (“Nobel Prize” of computing):

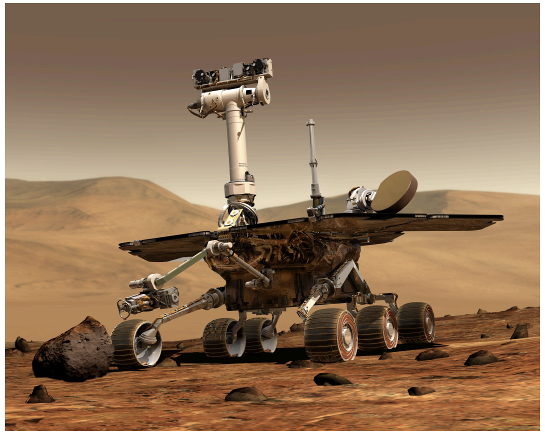
“The injustice of any award is that it has to pick a small number of winners. **But the winners are merely the visible part of an iceberg and wouldn't come to the surface without the much-larger submerged part that supports it…**

I am very thankful to all my mentors, collaborators, postdocs and students over the years. To a large extent, it is their work that the Turing Award rewards… I have been very fortunate to work with incredibly talented people over the years…

Mentors include Maurice Milgram & Françoise Soulié-Fogelman, my PhD advisors, Geoff Hinton with whom I did my postdoc, Larry Jackel and Rich Howard who hired me at Bell Labs, and Lawrence Rabiner my lab director at AT&T Labs…”

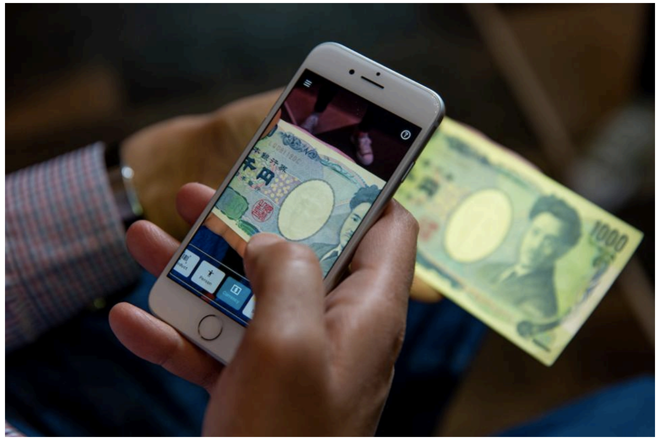
# CNN: Catalyst for Computer Vision Industry Boom

Self-driving cars

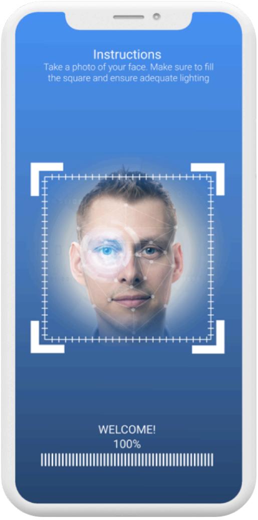


Self-driving vehicle on Mars

Guided surgery



Visual assistance for people who are blind

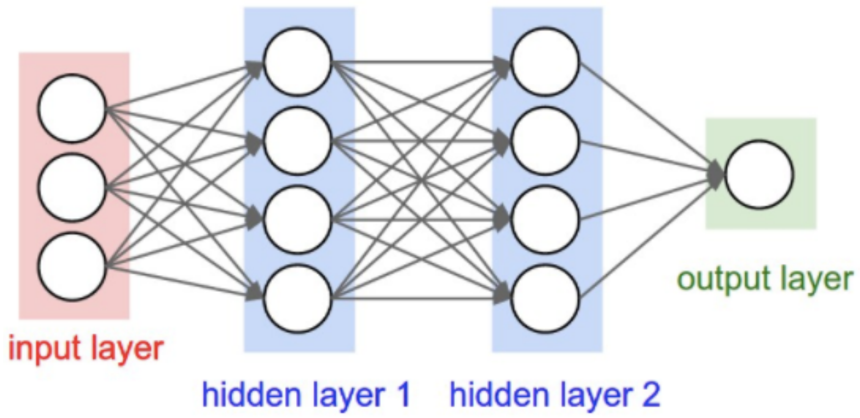


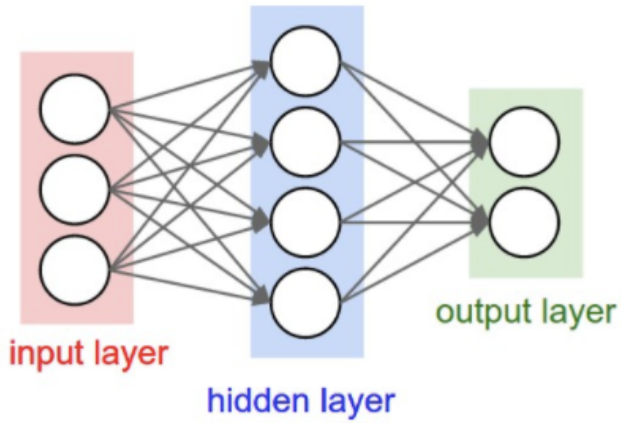

Security

# Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Deep Features
- Guest Speaker: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI

# Recall: Fully-Connected Neural Networks

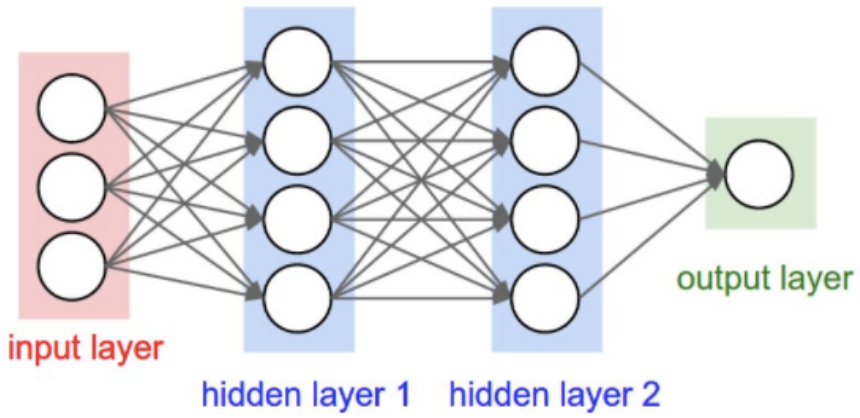


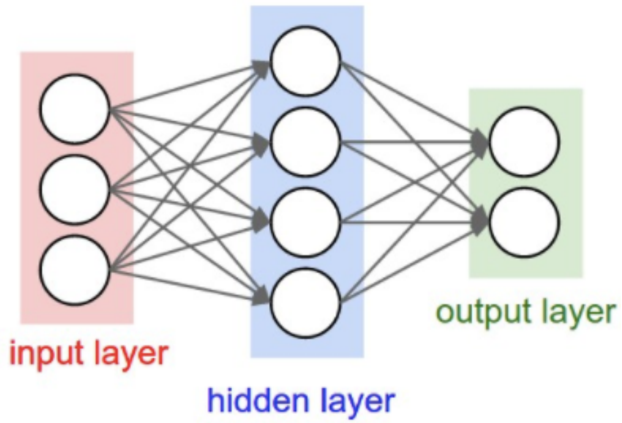




Each node provides input to each node in the next layer

Figure Source: http://cs231n.github.io/neural-networks-1/

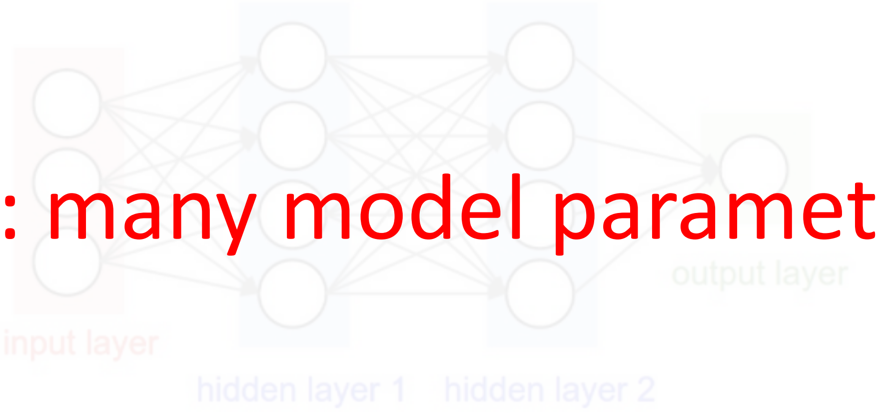
# Recall: Fully-Connected Neural Networks




- Assume 2 layer model with 100 nodes per layer
  - e.g., how many weights are in a 640x480 image?
    - 640x480x3x100 + 100x100 + 100x1 = 92,170,100
  - e.g., how many weights are in a 2048X1536 image (3.1 Megapixel image)?
    - 2048x1536x3x100 + 100x100 + 100x1 = 943,728,500

# Recall: Fully-Connected Neural Networks

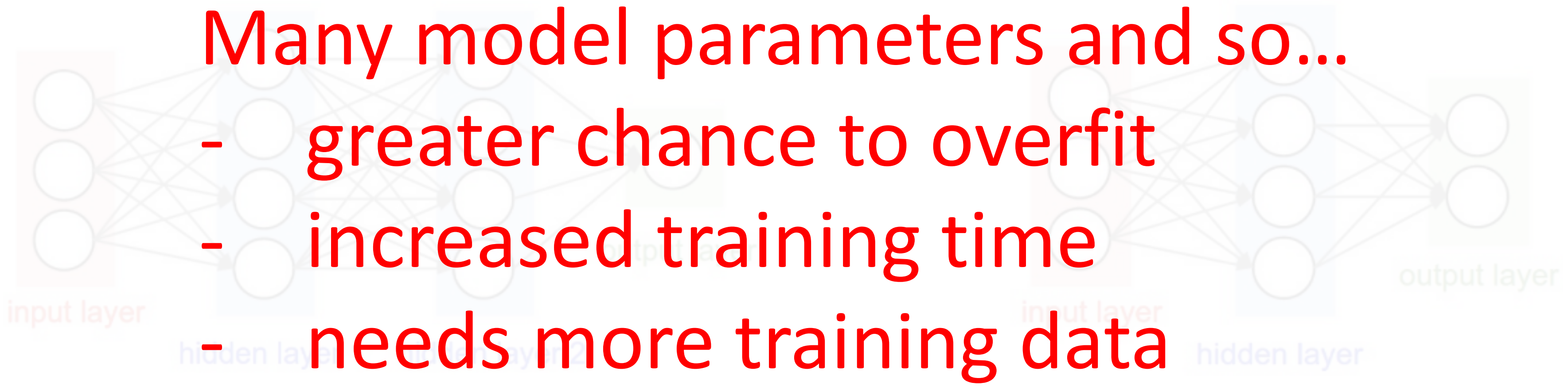




Issue: many model parameters in fully connected networks

- Assume 2 layer model with 100 nodes per layer
  - e.g., how many weights are in a 640x480 image?
    - 640x480x3x100 + 100x100 + 100x1 = 92,170,100
  - e.g., how many weights are in a 2048X1536 image (3.1 Megapixel image)?
    - 2048x1536x3x100 + 100x100 + 100x1 = 943,728,500

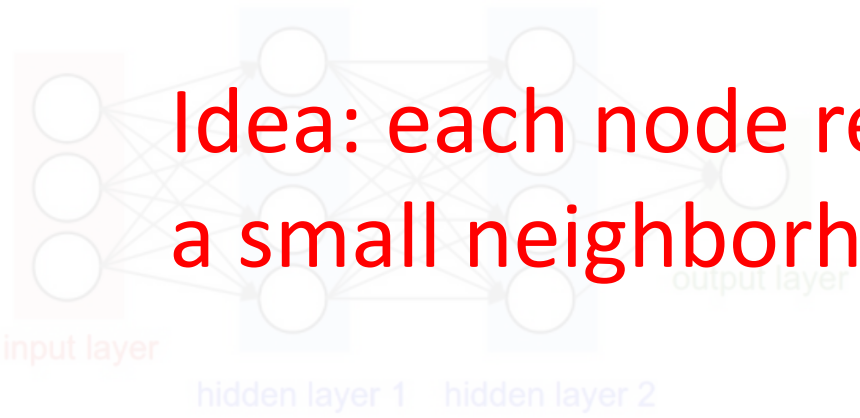
# Recall: Fully-Connected Neural Networks

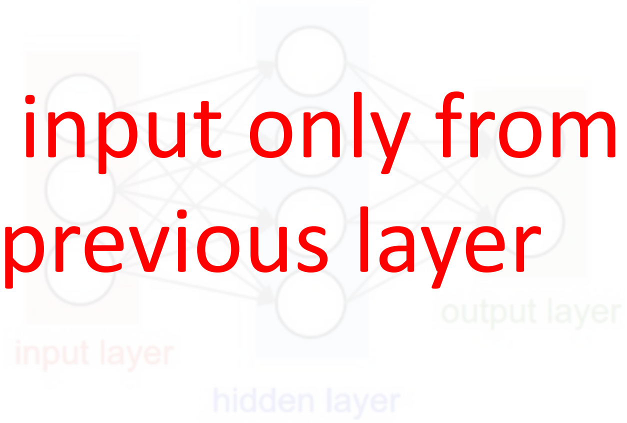



Many model parameters and so…
- greater chance to overfit
- increased training time
- needs more training data

- Assume 2 layer model with 100 nodes per layer
  - e.g., how many weights are in a 640x480 image?
    - 640x480x3x100 + 100x100 + 100x1 = 92,170,100
  - e.g., how many weights are in a 2048X1536 image (3.1 Megapixel image)?
    - 2048x1536x3x100 + 100x100 + 100x1 = 943,728,500

# Convolutional Layer

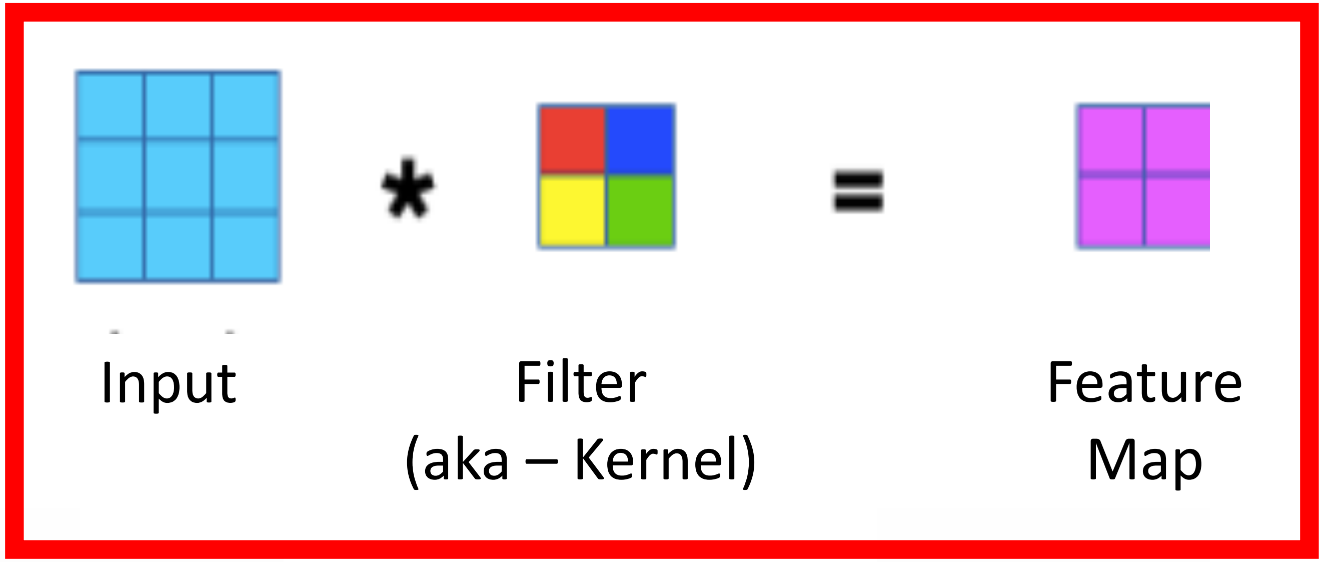


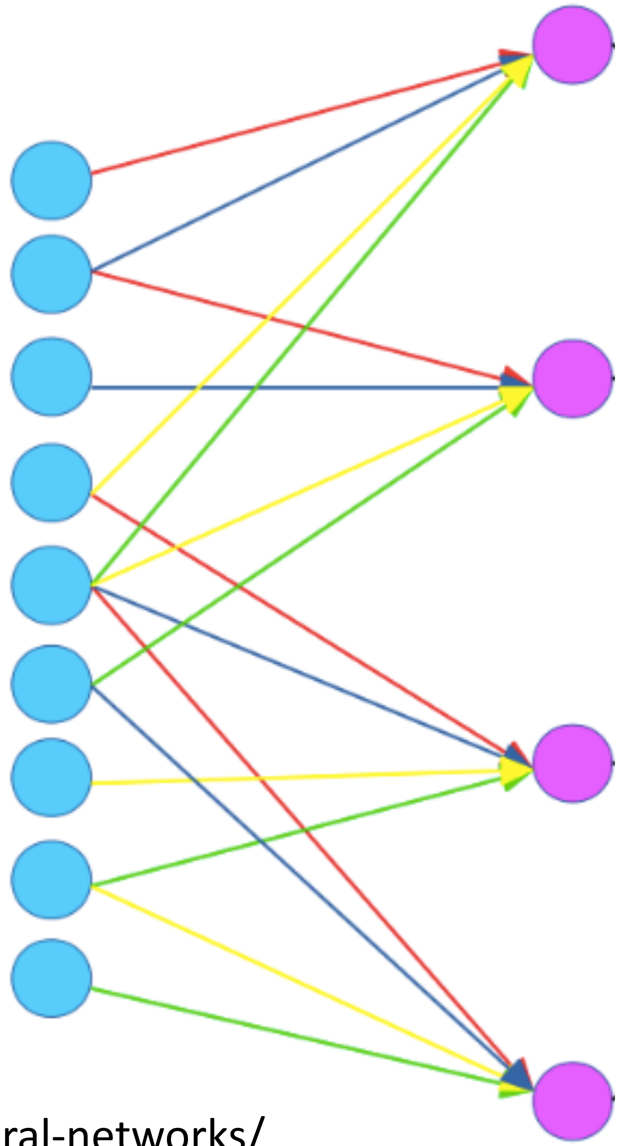

Idea: each node receives input only from a small neighborhood in previous layer

- Assume 2 layer model with 100 nodes per layer
  - e.g., how many weights are in a 640x480 image?
    - 640x480x3x100 + 100x100 + 100x1 = 92,170,100
  - e.g., how many weights are in a 2048X1536 image (3.1 Megapixel image)?
    - 2048x1536x3x100 + 100x100 + 100x1 = 943,728,500

# Convolutional Layer: Applies Linear Filter




Way to Interpret
Neural Network

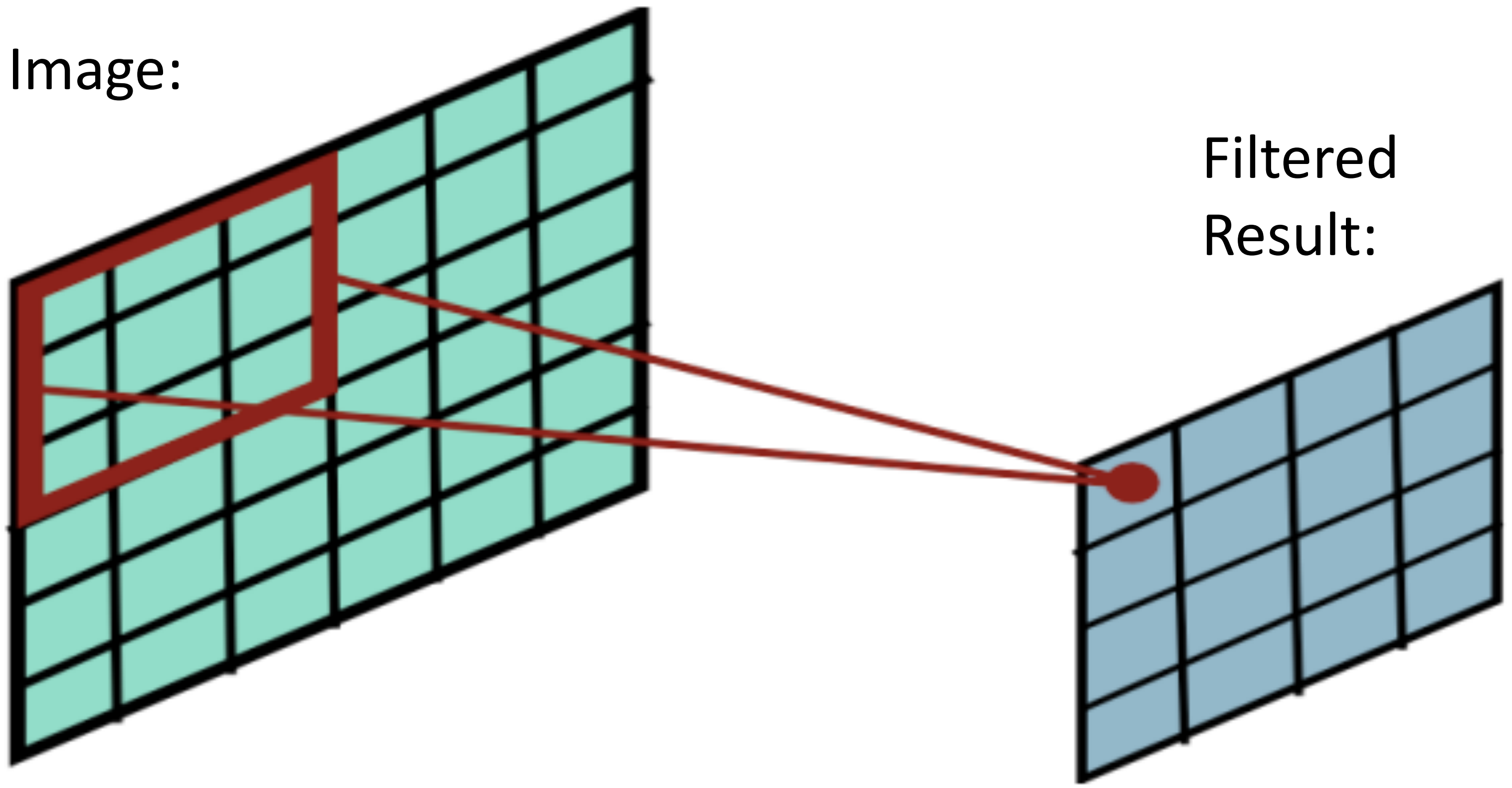


https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

# Image Filtering

- Compute a function of local neighborhood for each pixel in the image
- A filter specifies the function for how to combine neighbors’ values

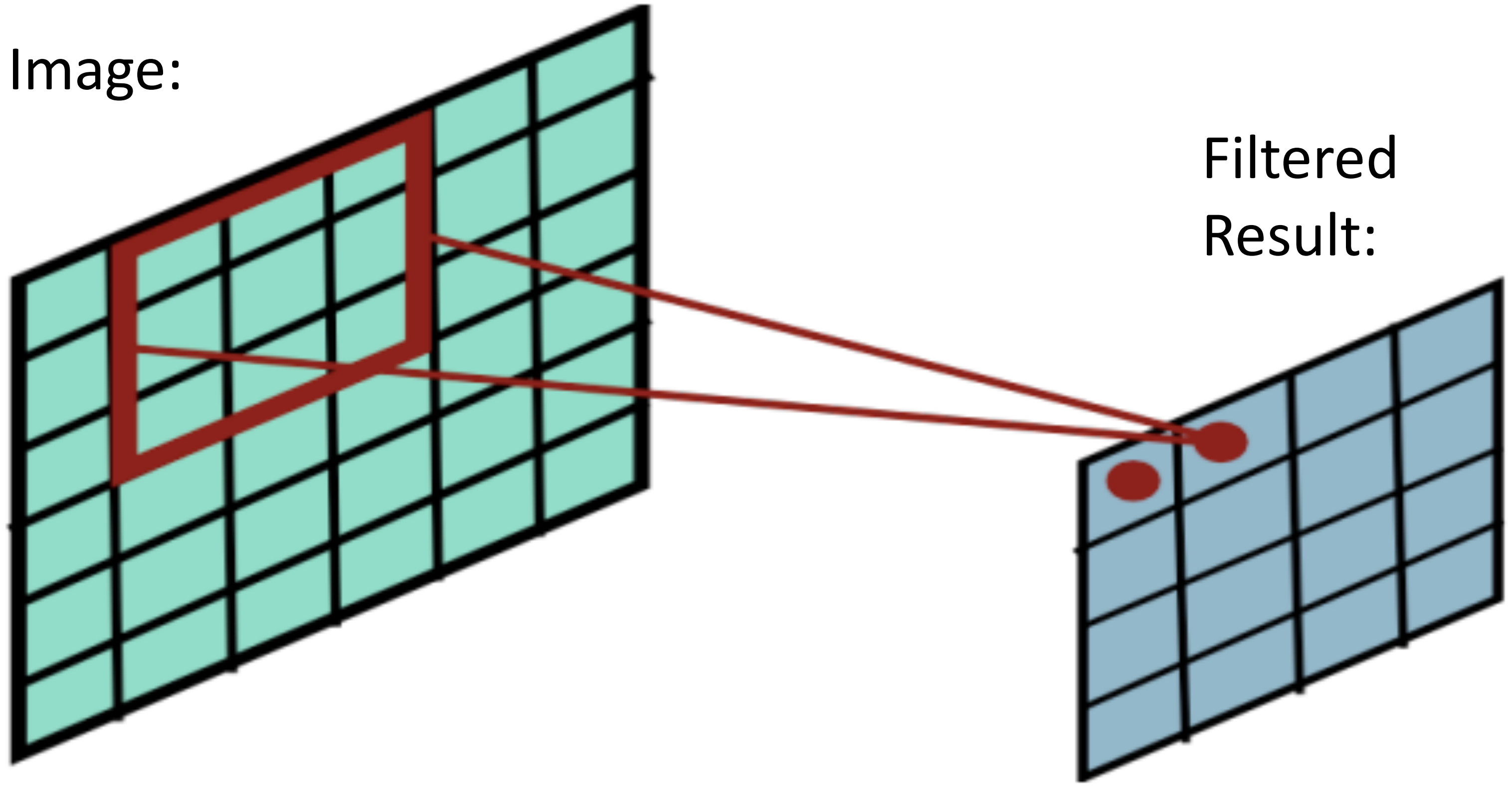
# Image Filtering




Slides filter over the image and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

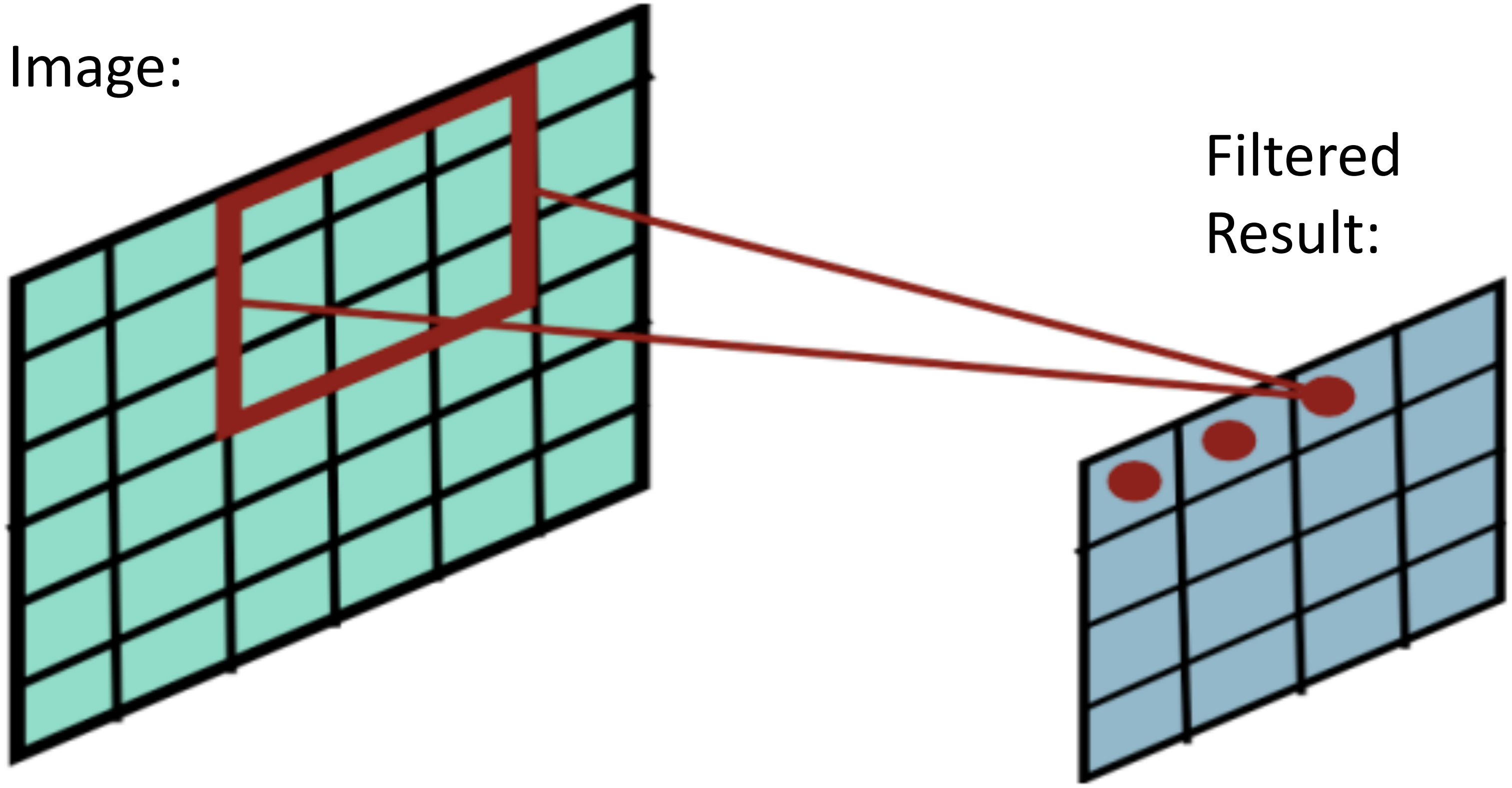
# Image Filtering




Slides filter over the image and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

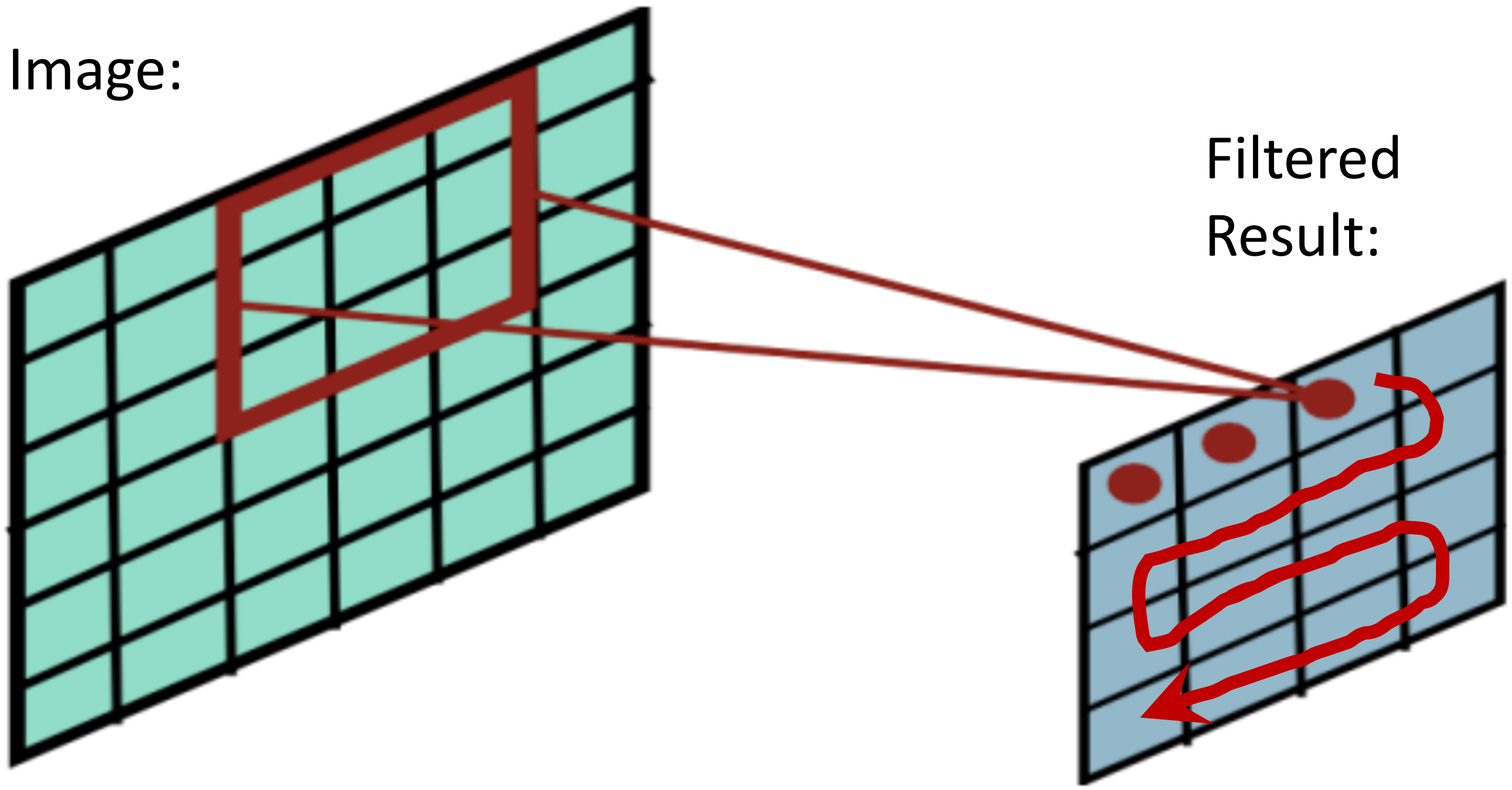
# Image Filtering




Slides filter over the image and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

# Image Filtering




Slides filter over the image and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| ? | ? | ? |
| ? | ? | ? |
| ? | ? | ? |

Dot Product = 1*1 + 1*0 + 1*1 + 0*0 + 1*1 + 1*0 + 0*1 + 0*1 + 0*0 + 0*0 + 1*1

Dot Product = 4

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | ? | ? |
| ? | ? | ? |
| ? | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | ? |
| ? | ? | ? |
| ? | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| ? | ? | ? |
| ? | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | ? | ? |
| ? | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | ? |
| ? | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| ? | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | ? | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | 3 | ? |

# Image Filtering: Toy Example

Image

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Filter

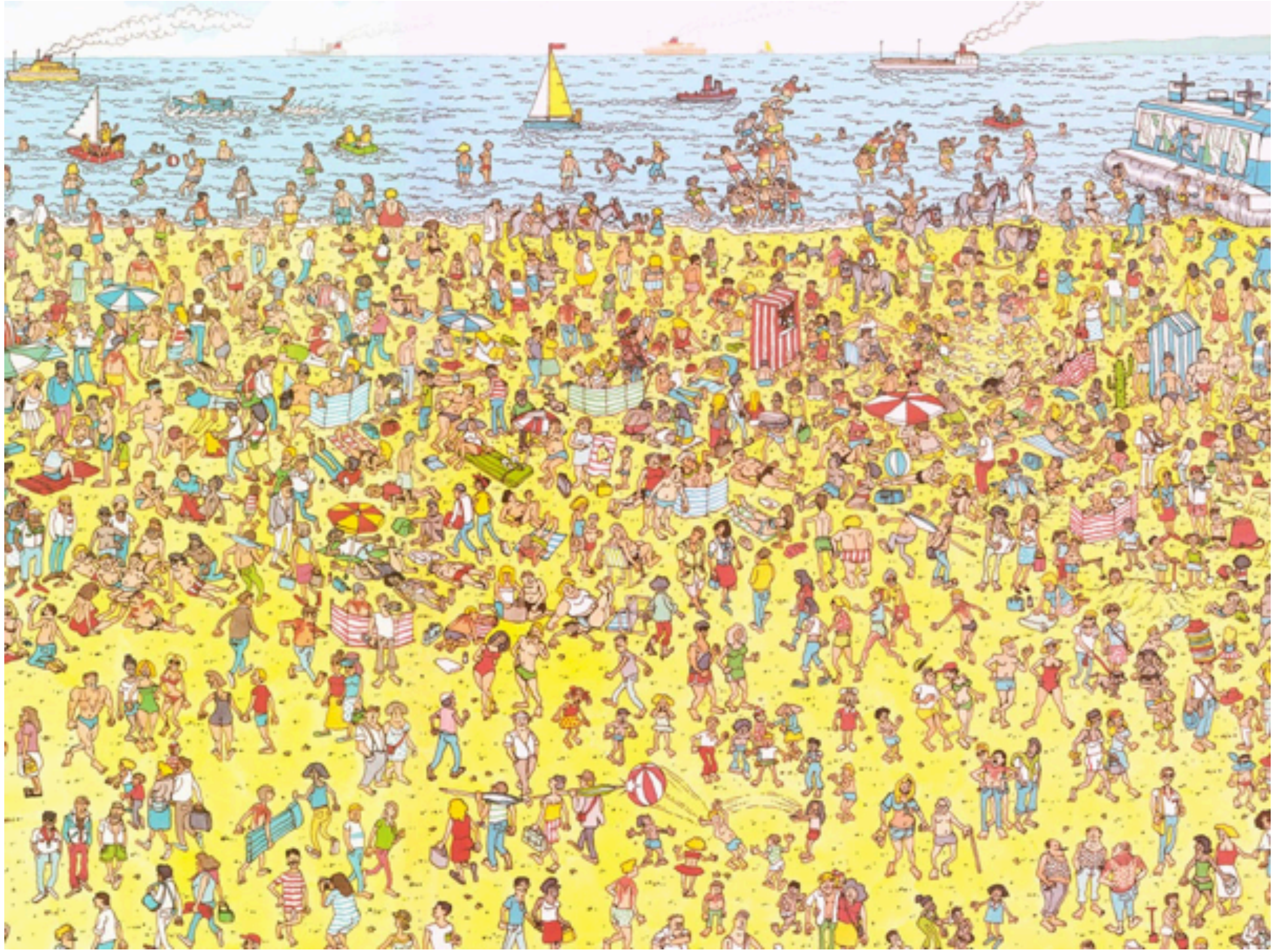
| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Feature Map

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | 3 | 4 |

# Image Filter: What Does It Do? (Where's Waldo?)

Filter



# Image Filter: What Does It Do?

- e.g.,

Filter

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Visualization of Filter

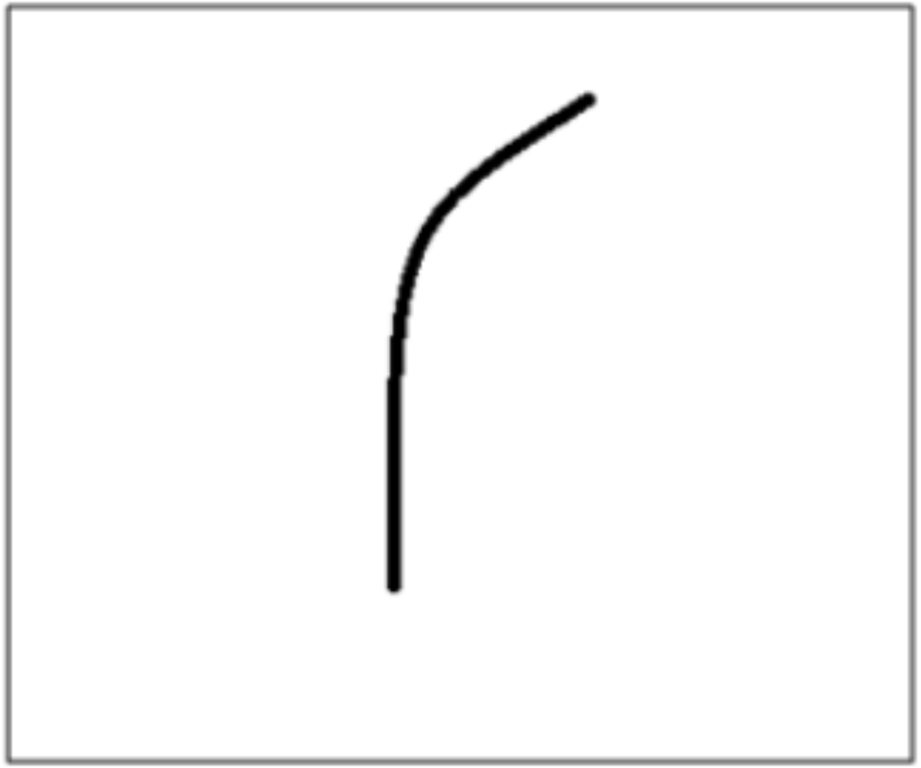
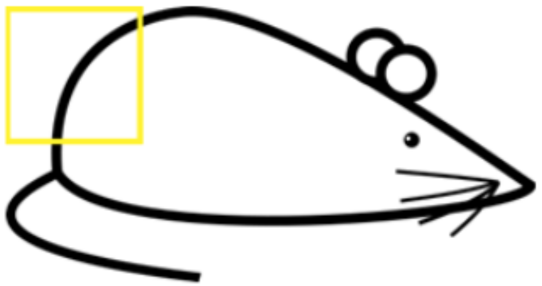


Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

# Image Filter: What Does It Do?

- e.g.,

Filter Overlaid on Image



Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 30 |
| 0 | 0 | 0 | 0 | 50 | 50 | 50 |
| 0 | 0 | 0 | 20 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |

*

Filter

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Weighted Sum = ?

Weighted Sum = (50x30) + (20x30) + (50x30) + (50x3) + (50x30)

Weighted Sum = 6600 **(Large Number!!)**

Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

# Image Filter: What Does It Do?

- e.g.,

Filter Overlaid on Image

Image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 40 | 0 | 0 | 0 | 0 | 0 |
| 40 | 0 | 40 | 0 | 0 | 0 | 0 |
| 40 | 20 | 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 50 | 0 | 0 | 0 | 0 |
| 25 | 25 | 0 | 50 | 0 | 0 | 0 |

*

Filter

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Weighted Sum = ?

Weighted Sum = 0 **(Small Number!!)**

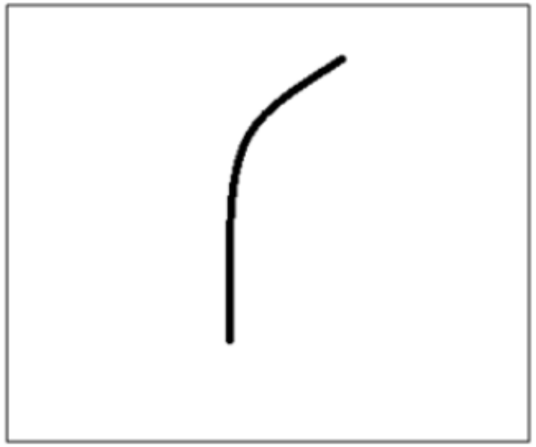
Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

# Image Filter: What Does It Do?

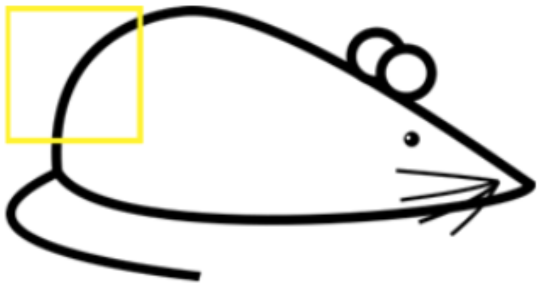
- e.g.,

This Filter is a Curve Detector!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Filter Overlaid on Image (Big Response!)



Filter Overlaid on Image (Small Response!)

Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

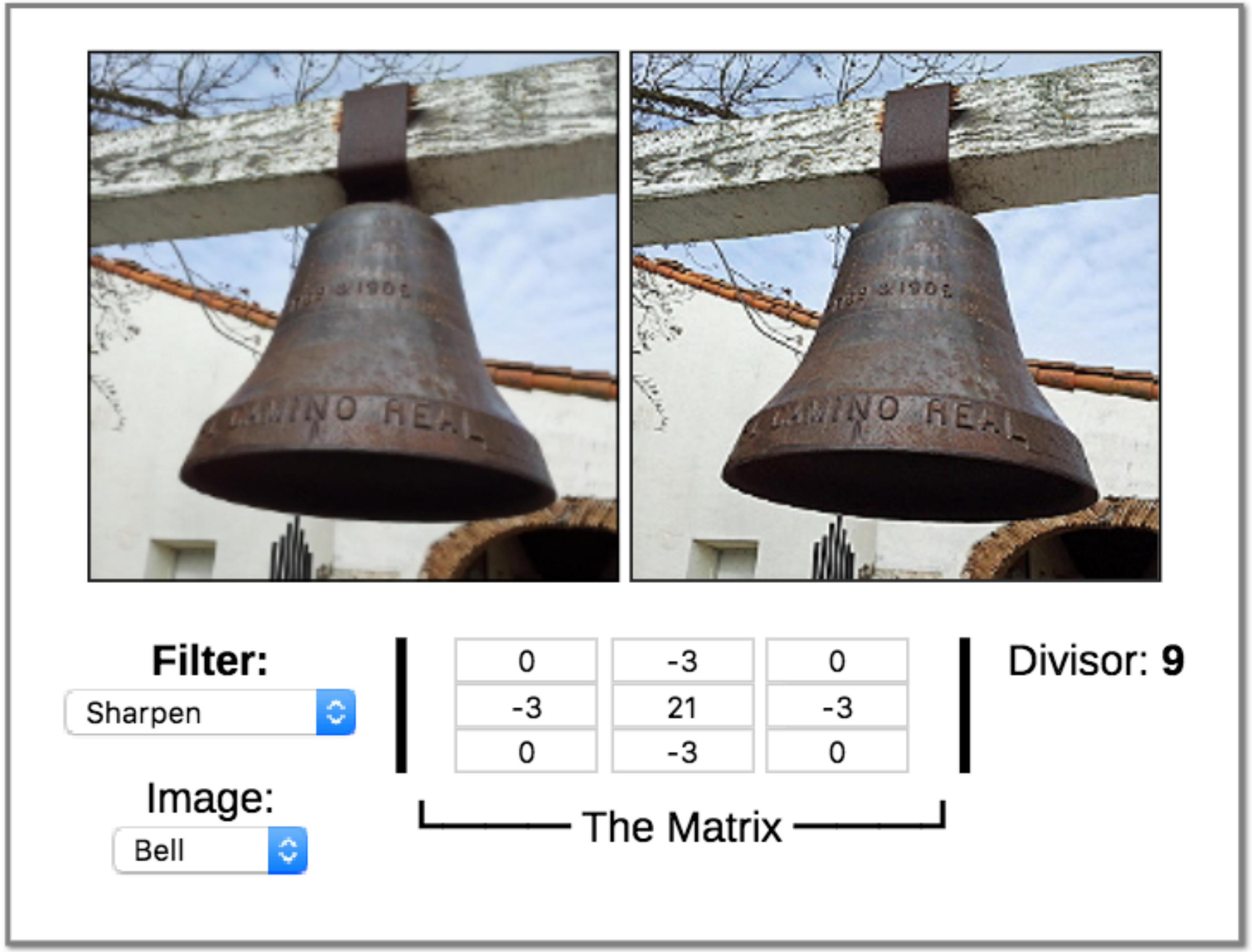
# Different Filters Detect Different Features

| | Filter | Feature Map |
|---|---|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |

| | Filter | Feature Map |
|---|---|---|
| **Sharpen** | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| Box blur (normalized) | $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |
| Gaussian blur (approximation) | $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ |  |

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

# Different Filters Detect Different Features

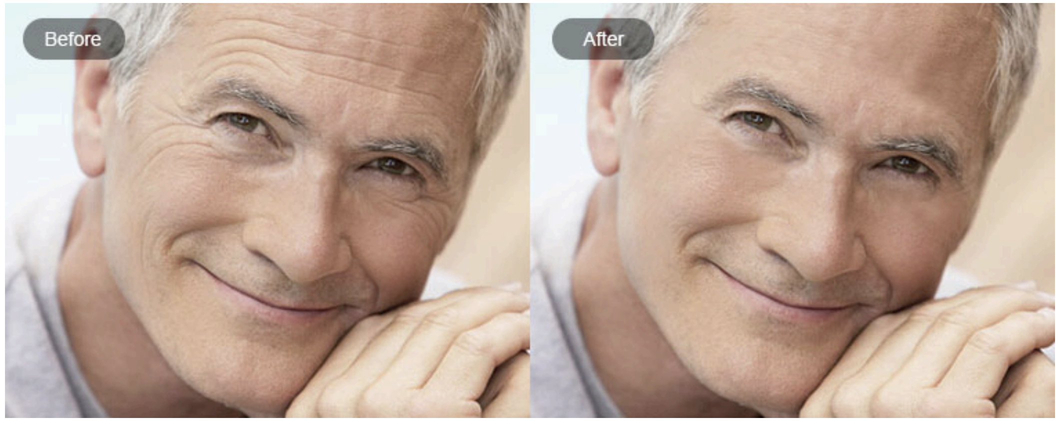


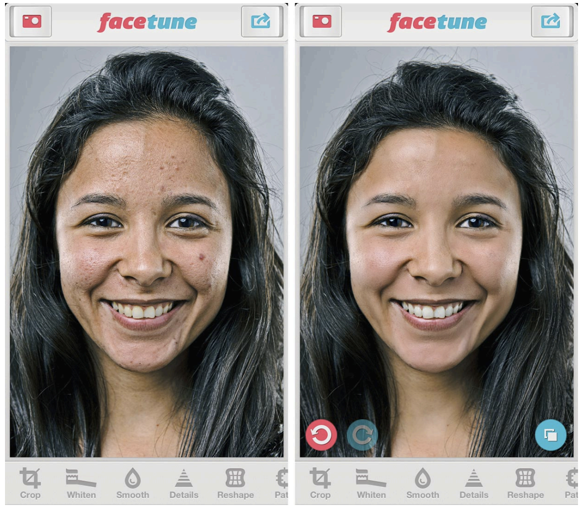

Demo: http://beej.us/blog/data/convolution-image-processing/

# Group Discussion

1. How would you design a linear filter to "brighten" an image

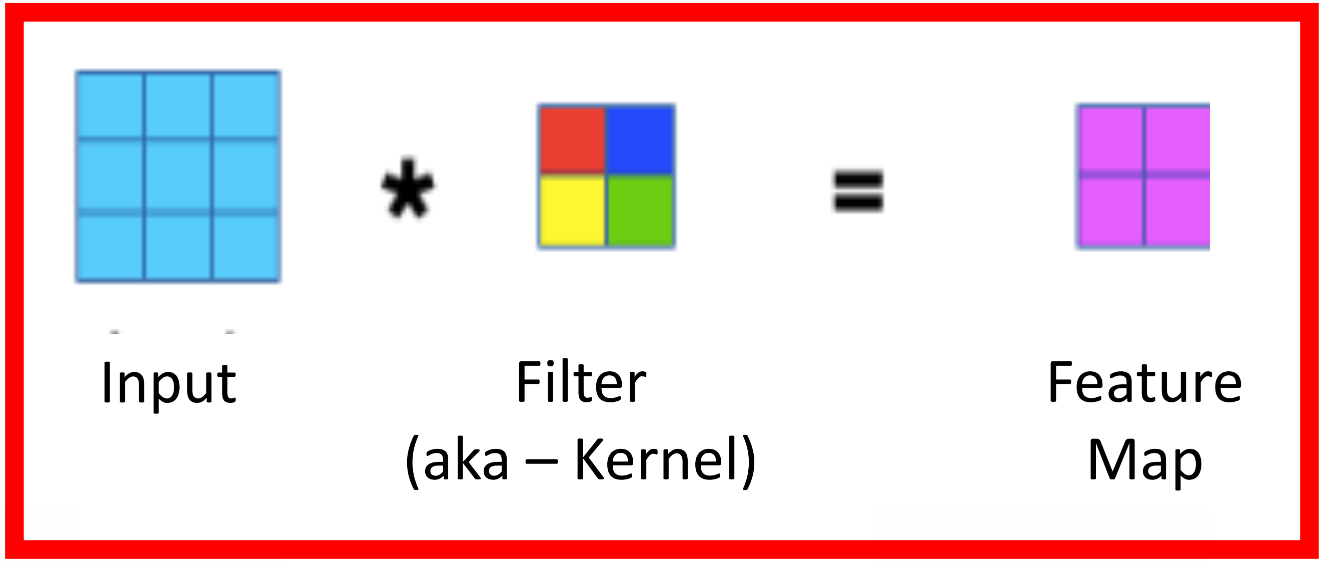

2. How would you design a linear filter to remove wrinkles/blemishes?



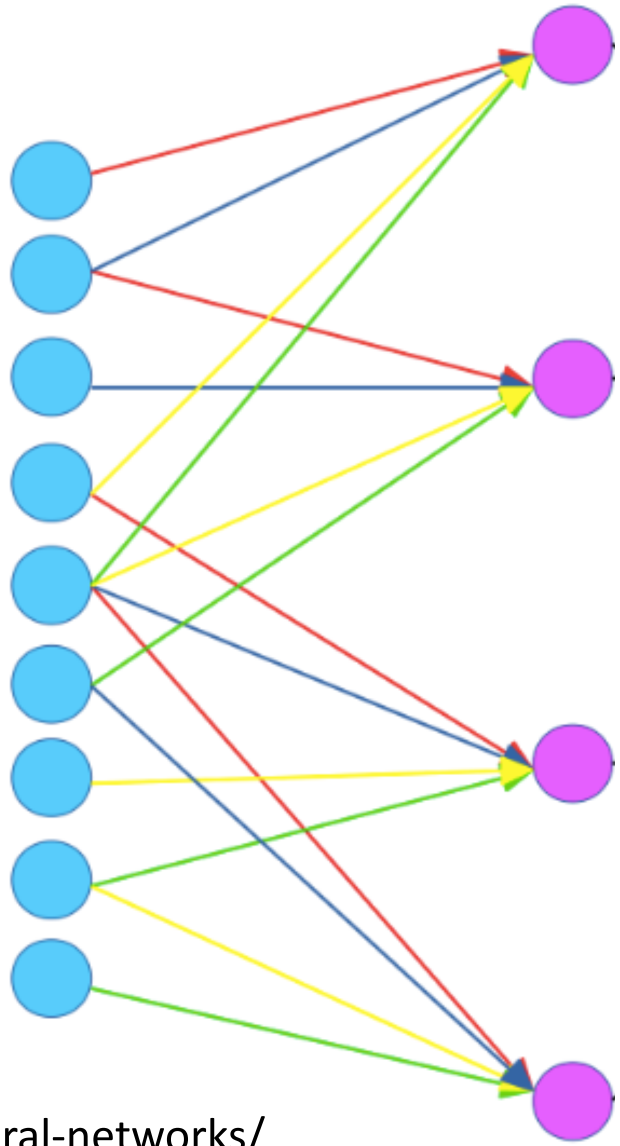



# Convolutional Layer: Applies Linear Filter

- Note, previous examples show the “cross-correlation” function
- Many neural network libraries use “cross correlation” interchangeably with “convolution”; for mathematicians, these are technically different



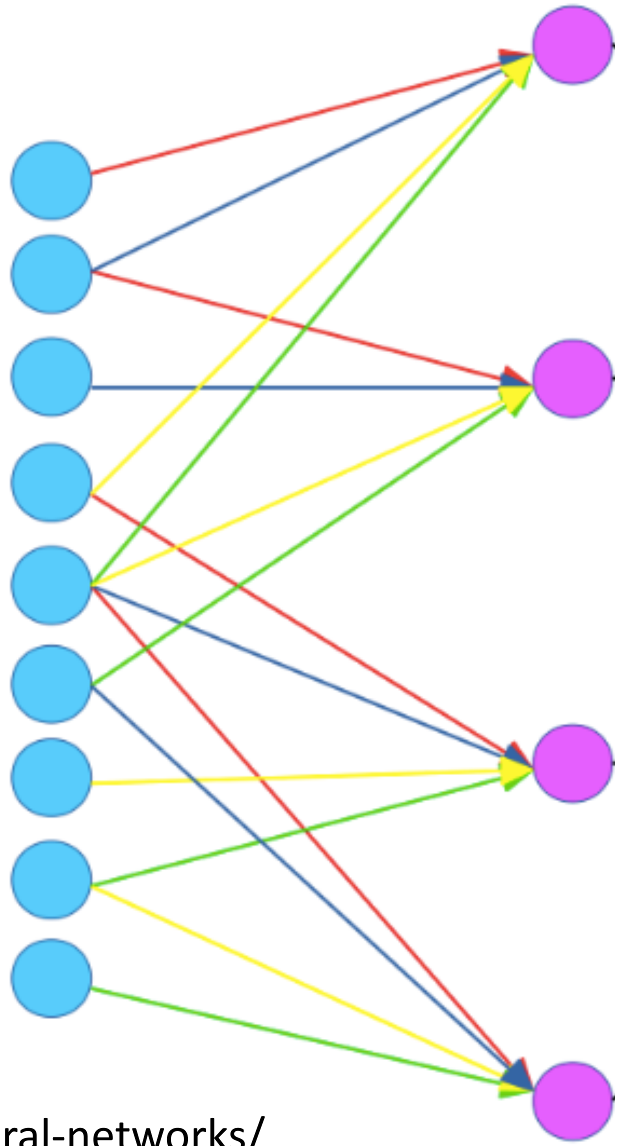

Way to Interpret Neural Network



https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

# Convolutional Layer: Parameters to Learn

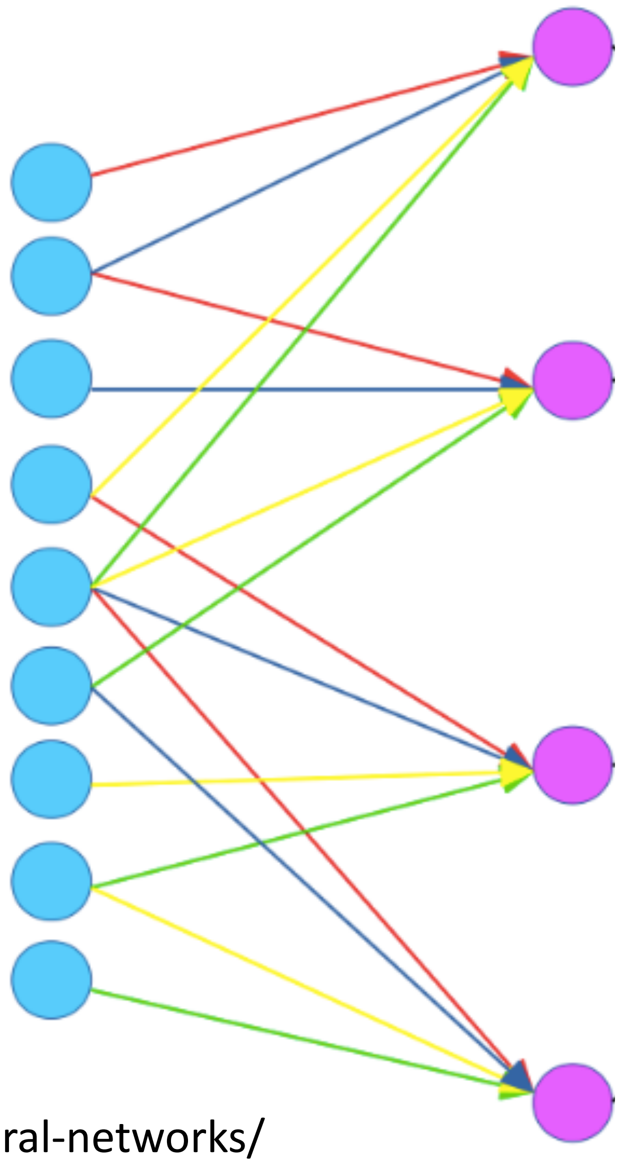
- For shown example, how many weights must be learned?
  - 4 (red, blue, yellow, and green values)
- If we instead used a fully connected layer, how many weights would need to be learned?
  - 36 (9 turquoise nodes x 4 magenta nodes)
- For shown example, how many parameters must be learned
  - 5 (4 weights + 1 bias)
- If we instead used a fully connected layer, how many parameters would need to be learned?
  - 40 (36 weights + 4 bias)



https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

# Convolutional Layer: Parameters to Learn

- Parameter sharing significantly reduces number of weights to learn and so storage requirements
- Sparse (rather than full) connectivity also significantly reduces the number of computational operations required



https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

# Convolutional Layer: Implementation Details

- **Padding**: add values at the image boundaries to preserve image size

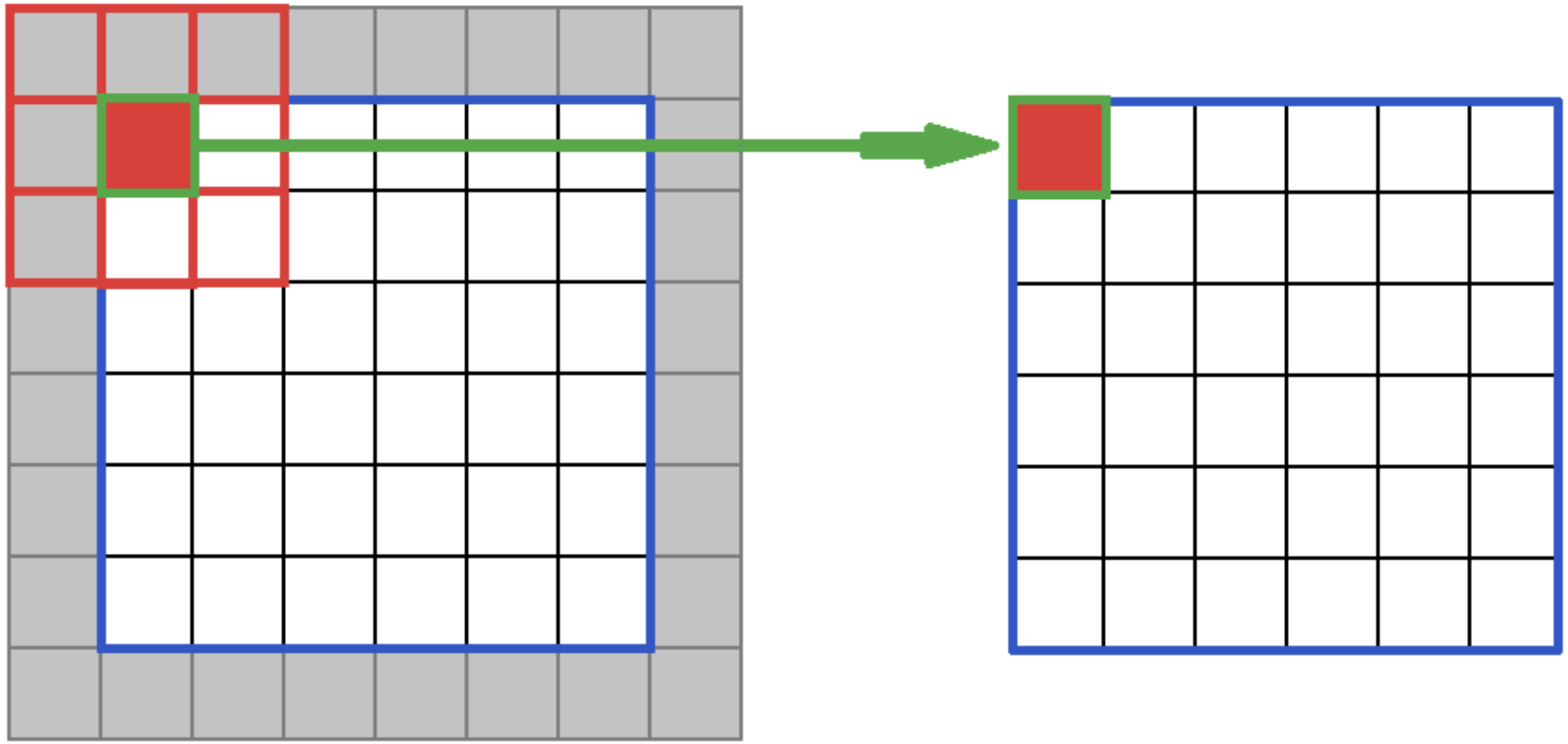
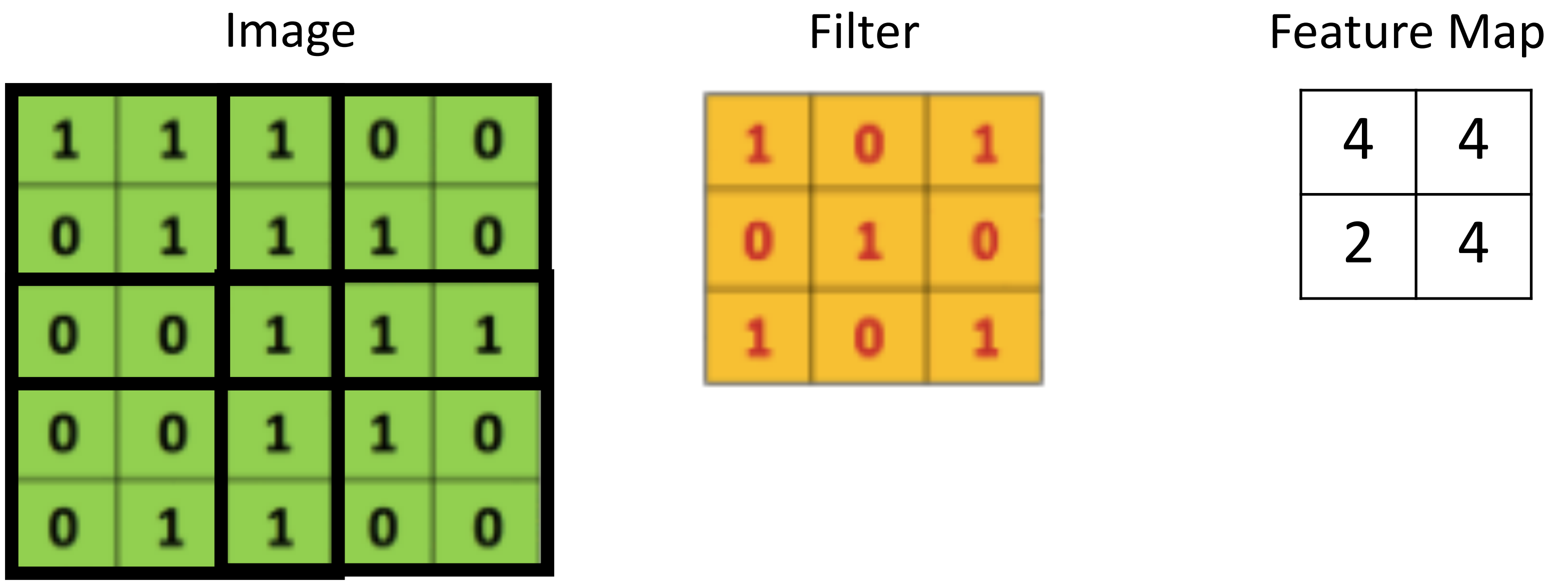


Image Credit: https://software.intel.com/en-us/node/586159

# Convolutional Layer: Implementation Details

- **Stride**: how many steps taken spatially before applying a filter
  - e.g., 2x2



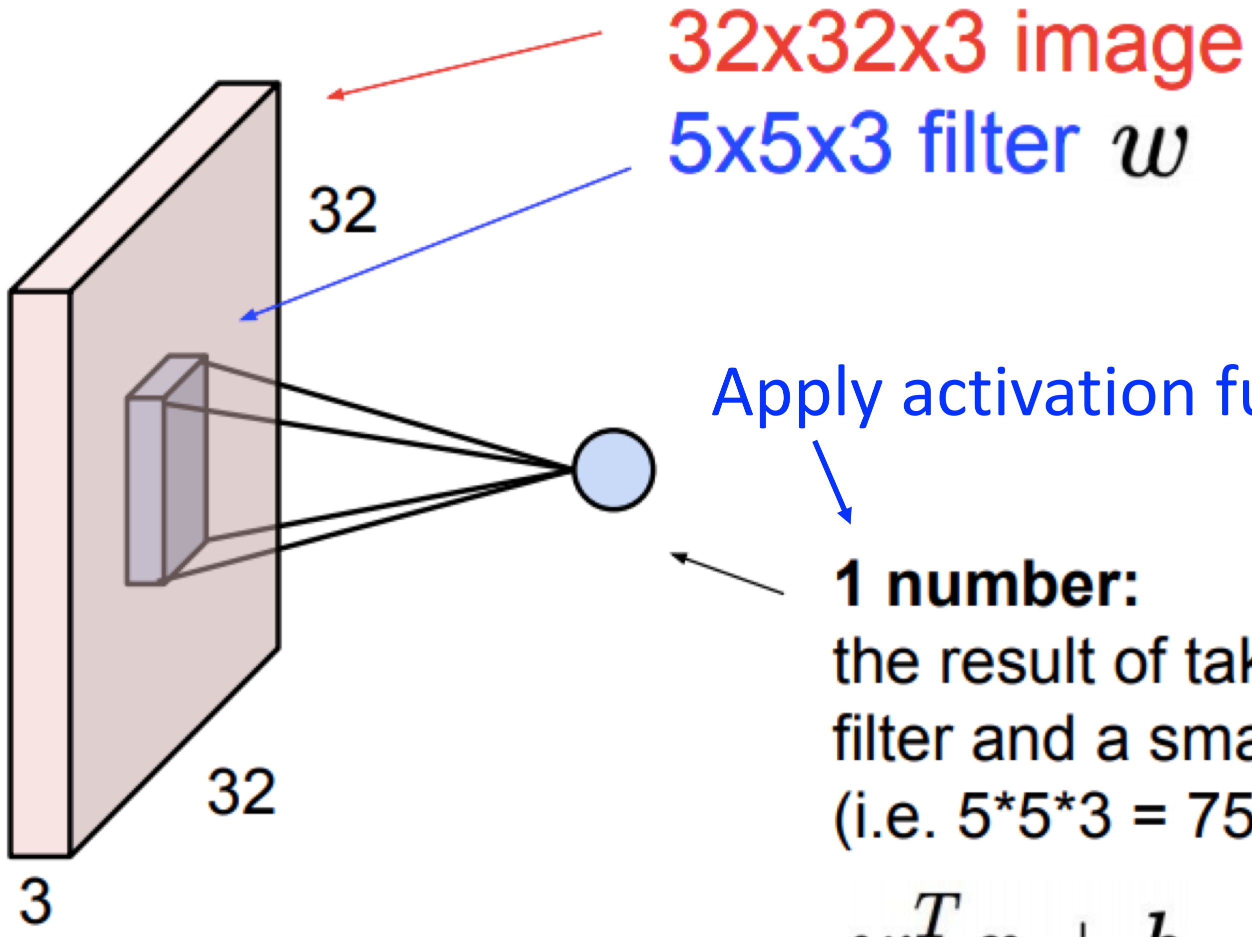

http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

# Convolutional Layer: Implementation Details

- Demo:

http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

# Convolutional Layer: Introduce Non-Linearity



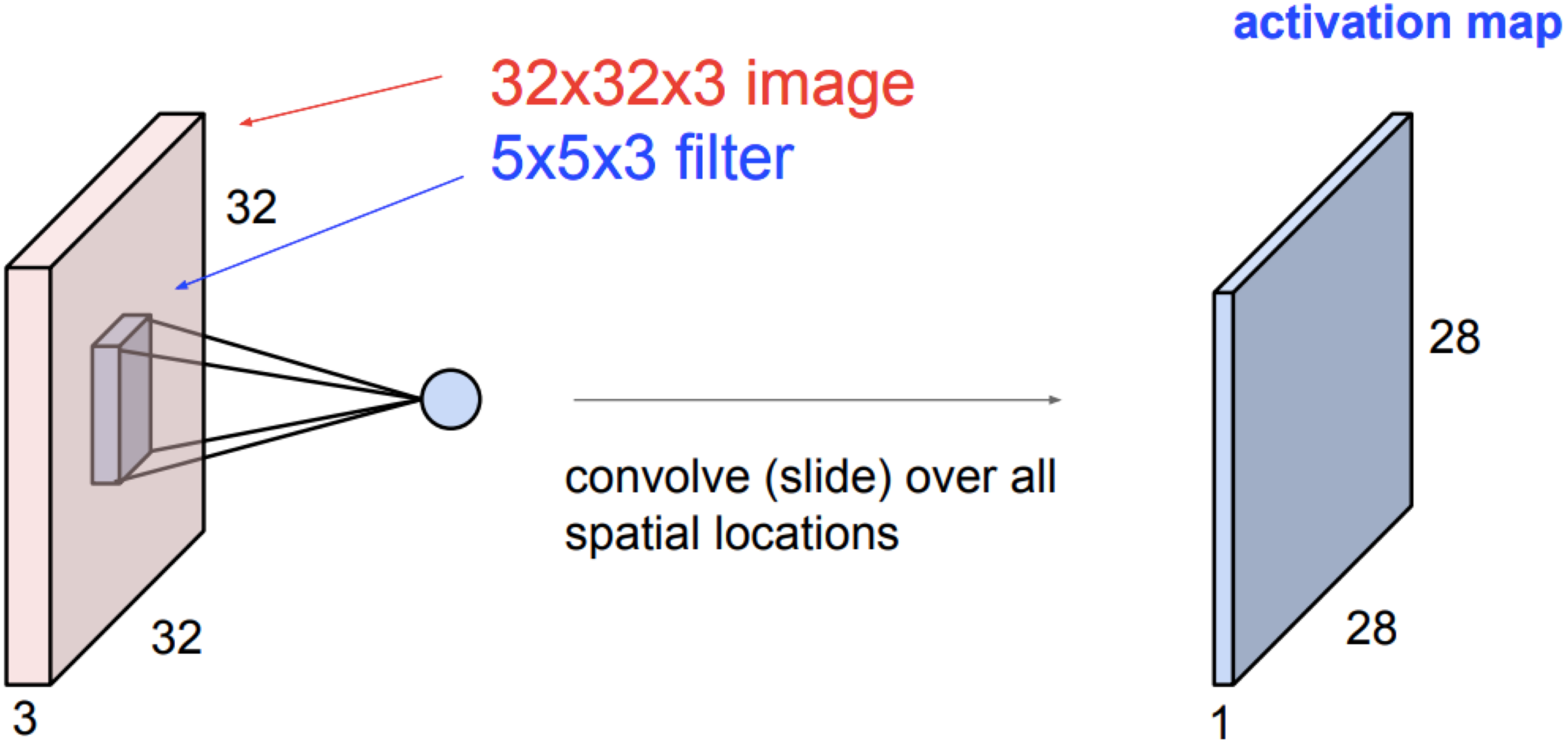

Apply activation function to number; e.g., ReLU

**1 number:**
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

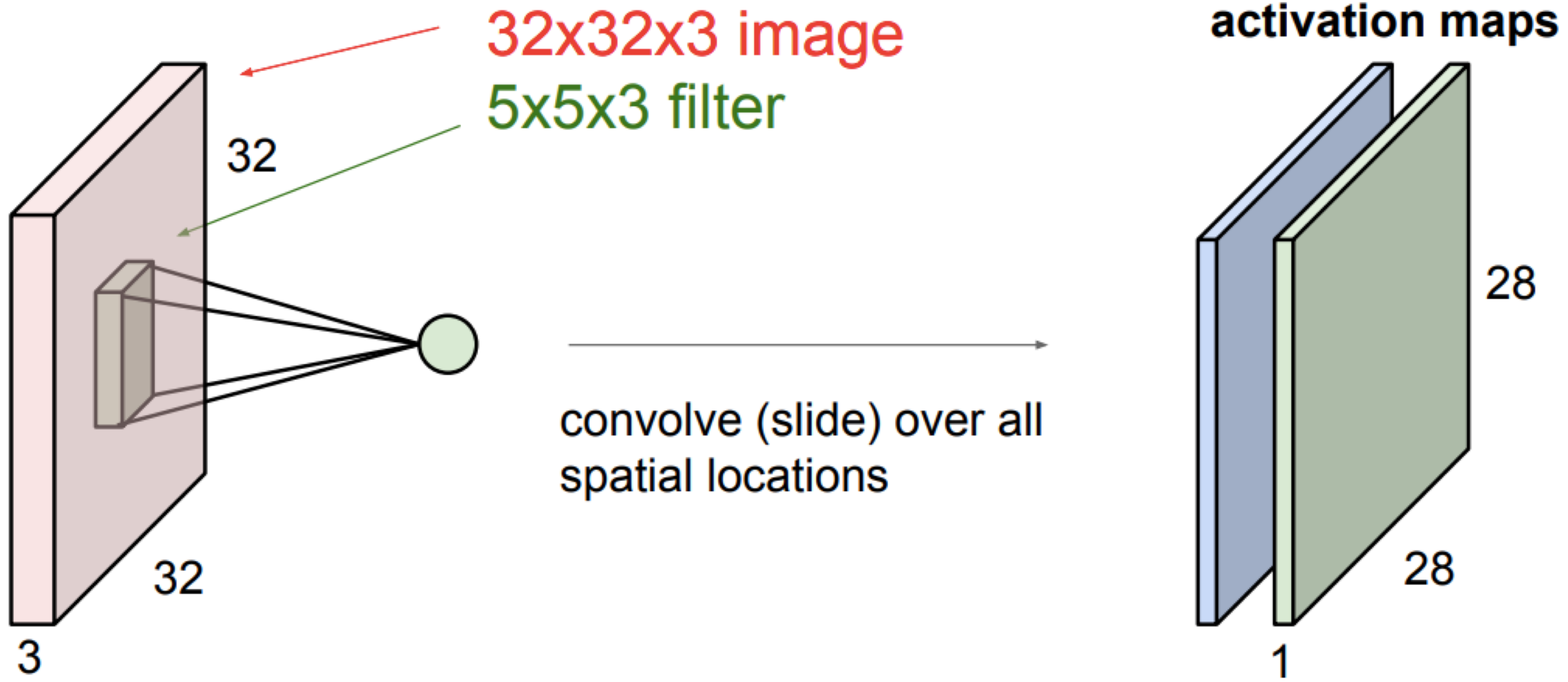
# Convolutional Layer




Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

# Convolutional Layer

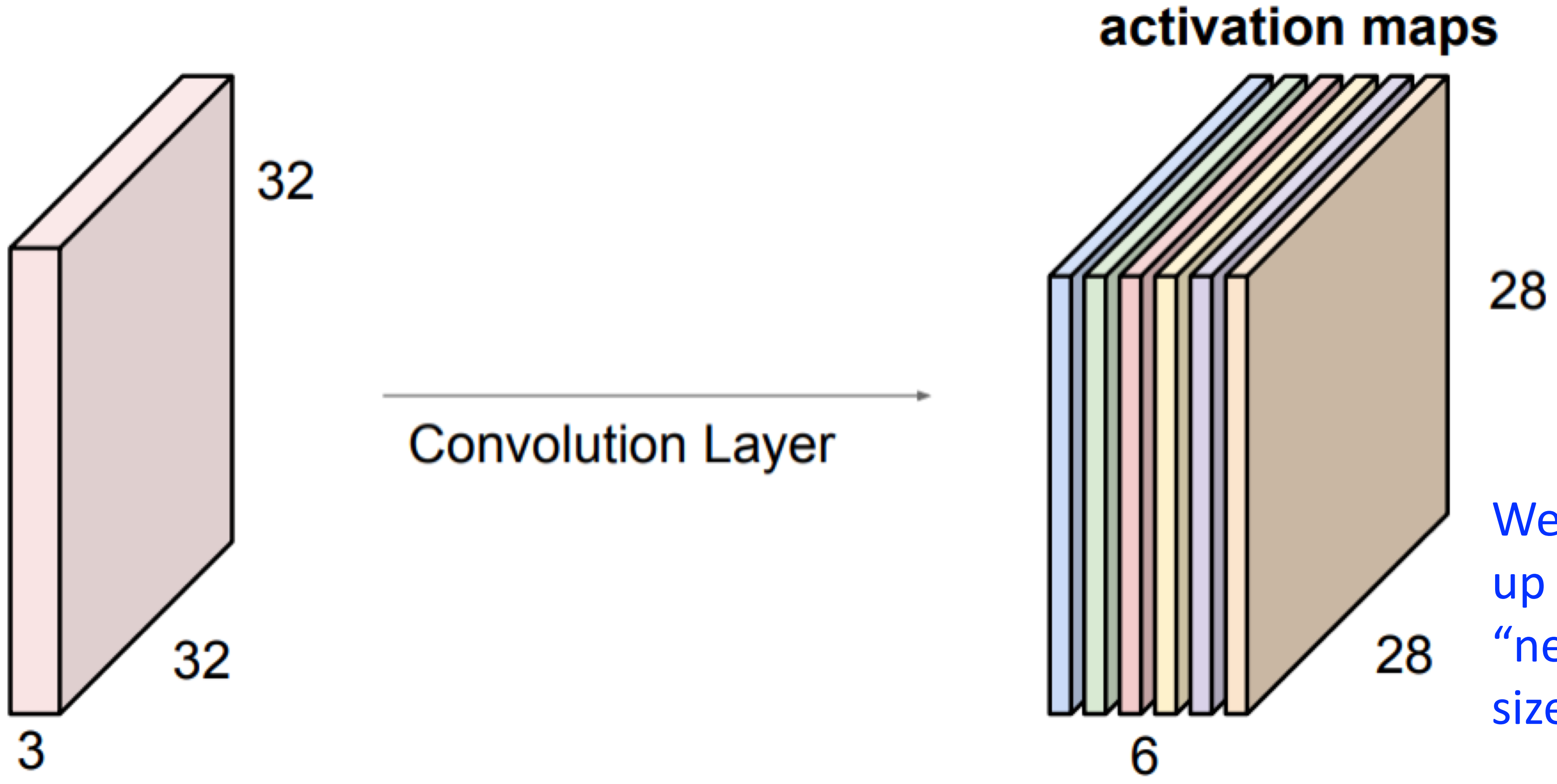
consider a second, green filter




Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

# Convolutional Layer

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

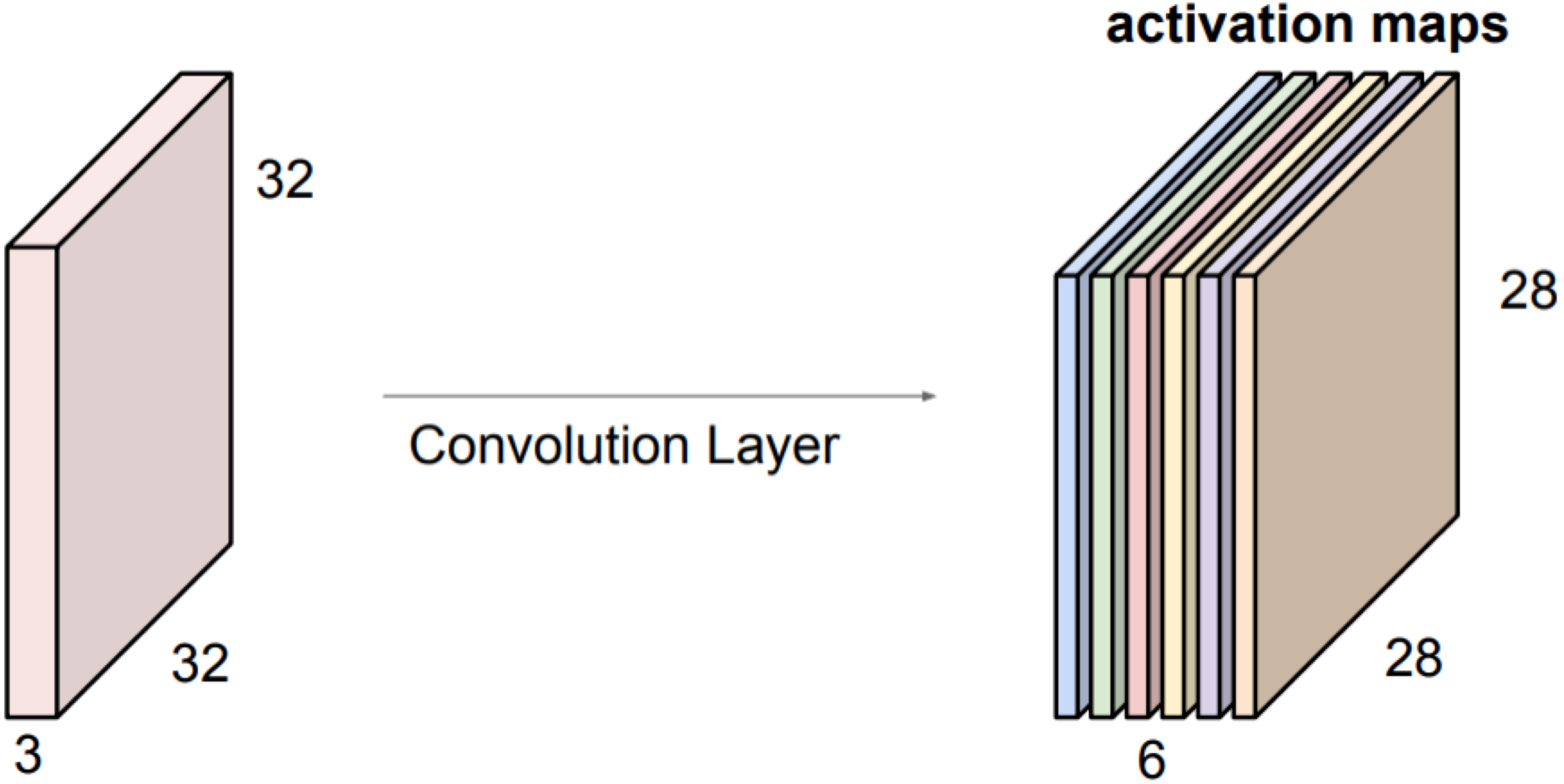



We stack these up to get a "new image" of size 28x28x6!

Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

# Convolutional Layer: Parameters to Learn

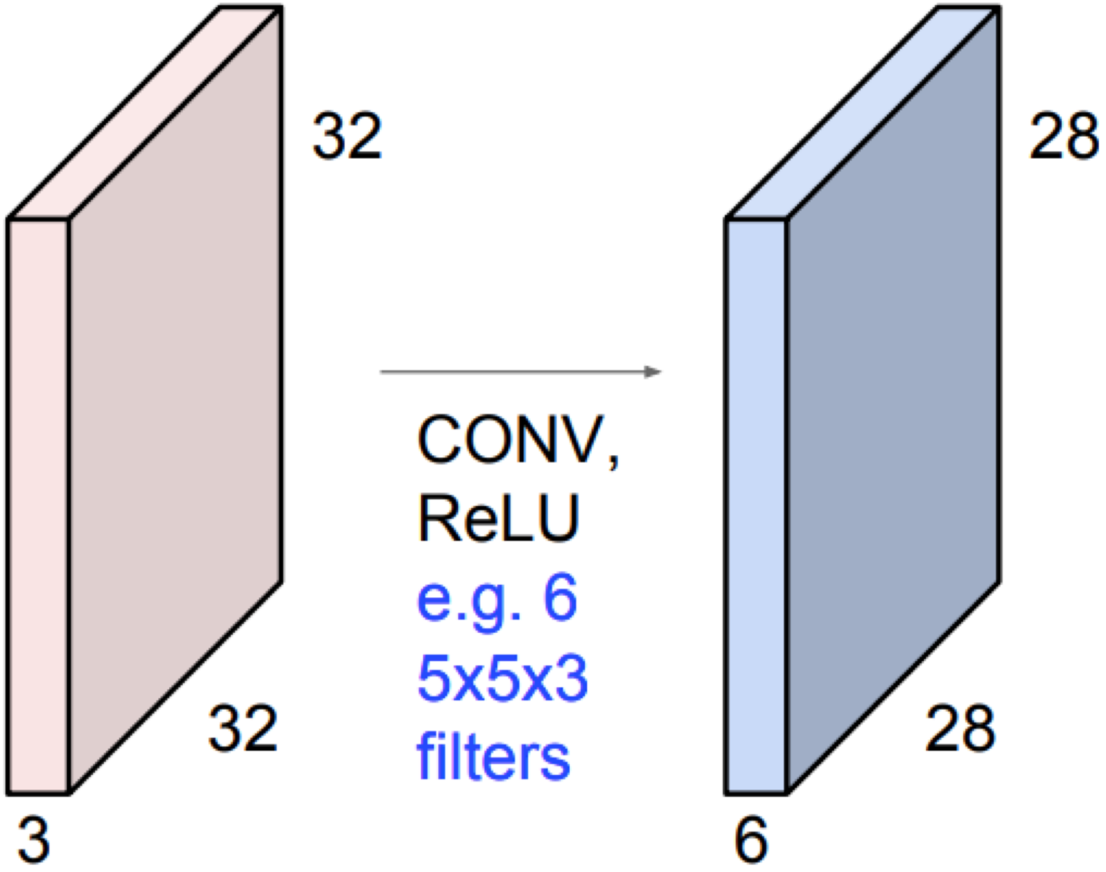
Parameters: bank of filters and biases used to create the activation maps (aka – feature maps)




Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

# Convolutional Neural Networks (CNNs)

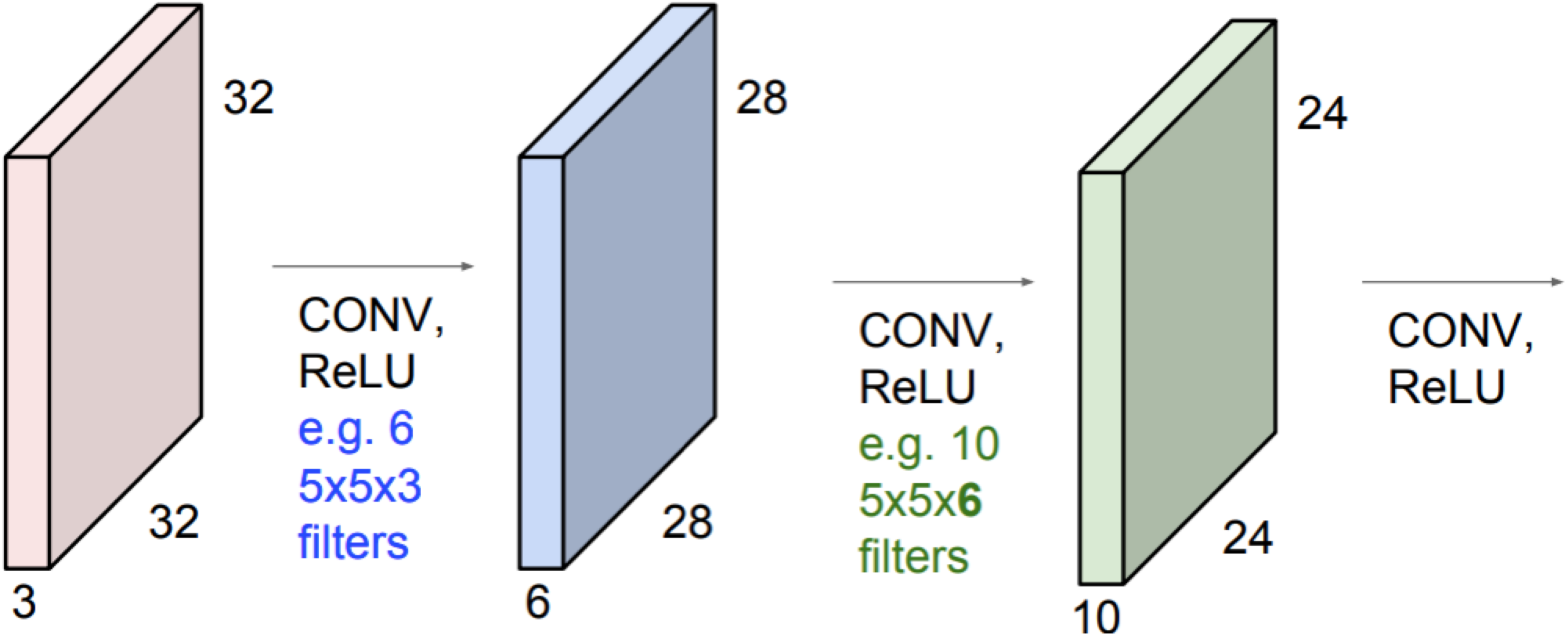
Can then stack a sequence of convolution layers, interspersed with activation functions:




[Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf](http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf)

# Convolutional Neural Networks (CNNs)

Can then stack a sequence of convolution layers, interspersed with activation functions:

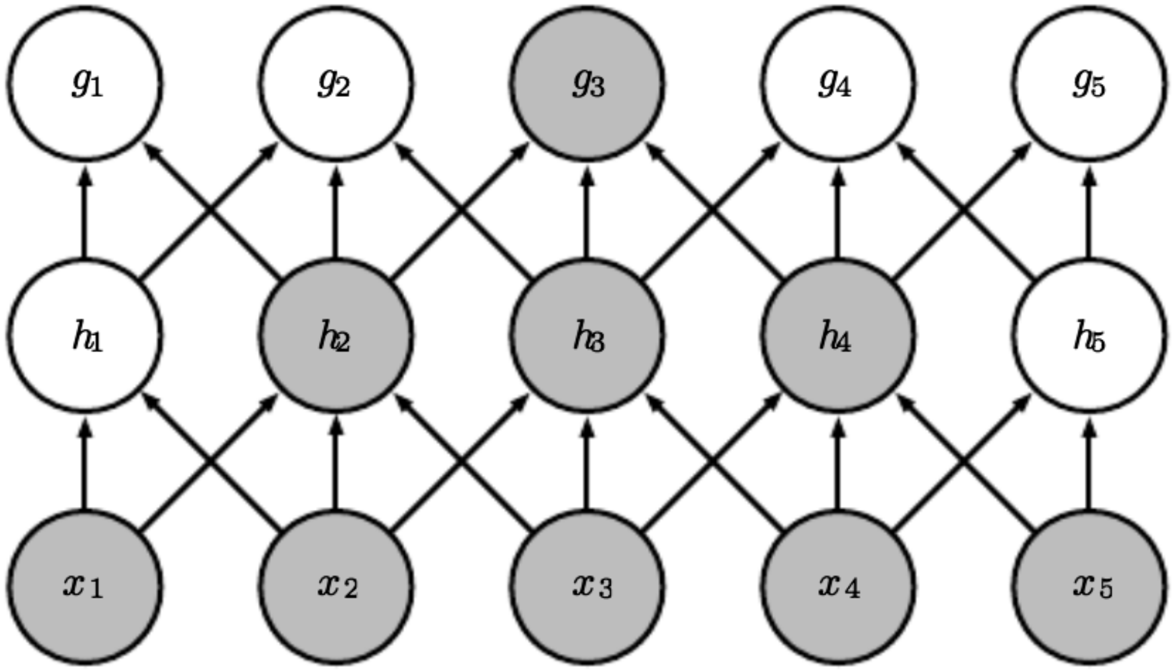



Slide Credit: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

# Convolutional Neural Networks (CNNs)

Can then stack a sequence of convolution layers, interspersed with activation functions:

Stacking many convolutional layers leads to learning patterns in increasingly **larger regions of the input (e.g., pixel) space.**




https://www.deeplearningbook.org/contents/convnets.html

# Convolutional Layer: Training

1. Forward Pass:
    - For convolutional layers:
        1. Apply convolution operation with each filter
        2. Add biases (one per each output image)
        3. Apply an activation function to all the pixels of the output images

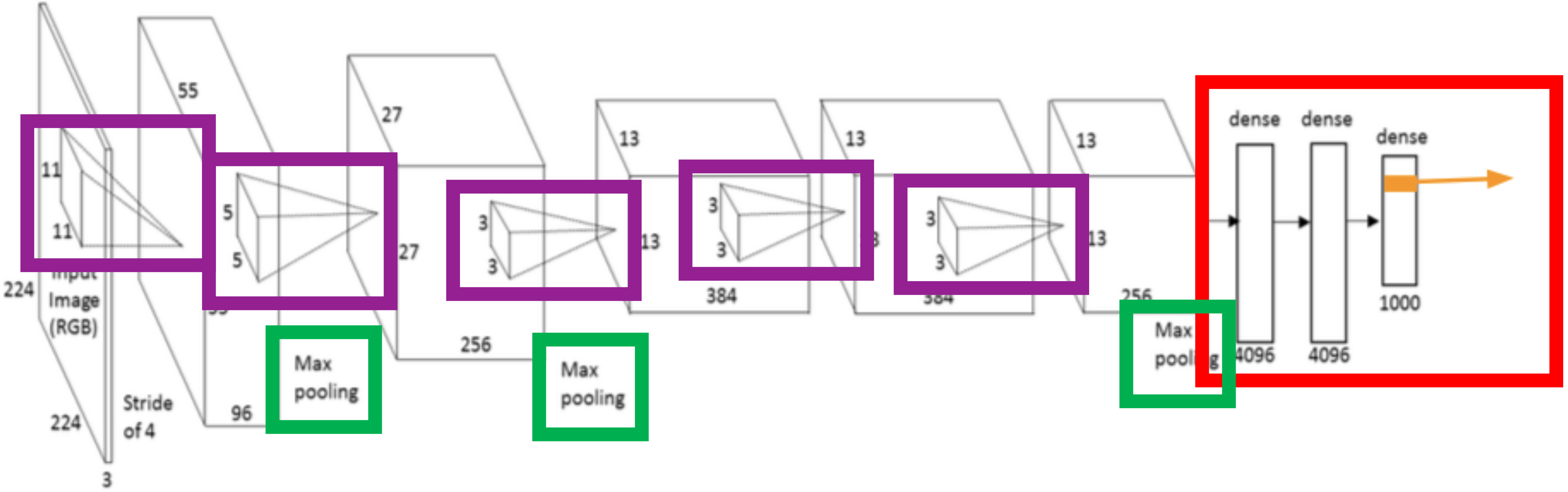
2. Compute prediction error (with respect to a loss function)

3. Backpropagate error to all model parameters (determine how changing a single pixel in the weight kernel affects the loss function)

4. Update all model parameters (kernel weights, biases)

# CNN: Summary of Convolution Layers

- e.g., AlexNet extracts useful features of lower dimension prior to passing it to MLP with:
  - **Convolutional layers**
  - Pooling Layers




Slide Credit: https://www.slideshare.net/xavigiro/saliency-prediction-using-deep-learning-techniques

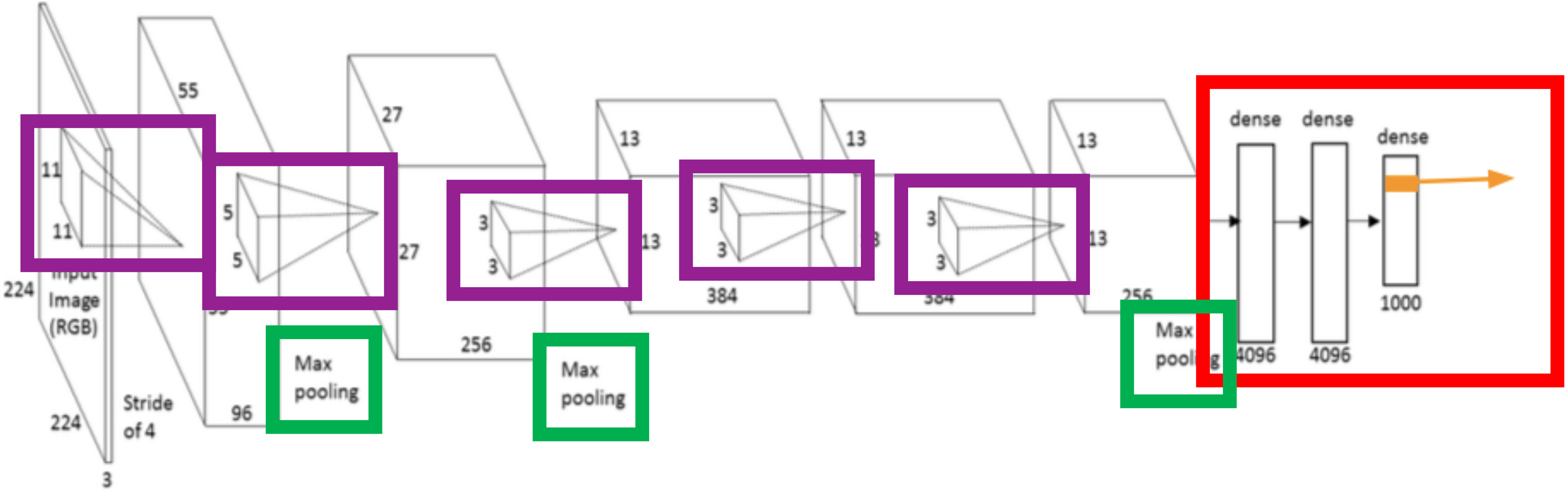
A. Krizhevsky, I. Sutskever, G. E. Hinton “ImageNet classification with deep convolutional neural networks”

# Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Deep Features
- Guest Speaker: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI

# CNN: Pooling Layers

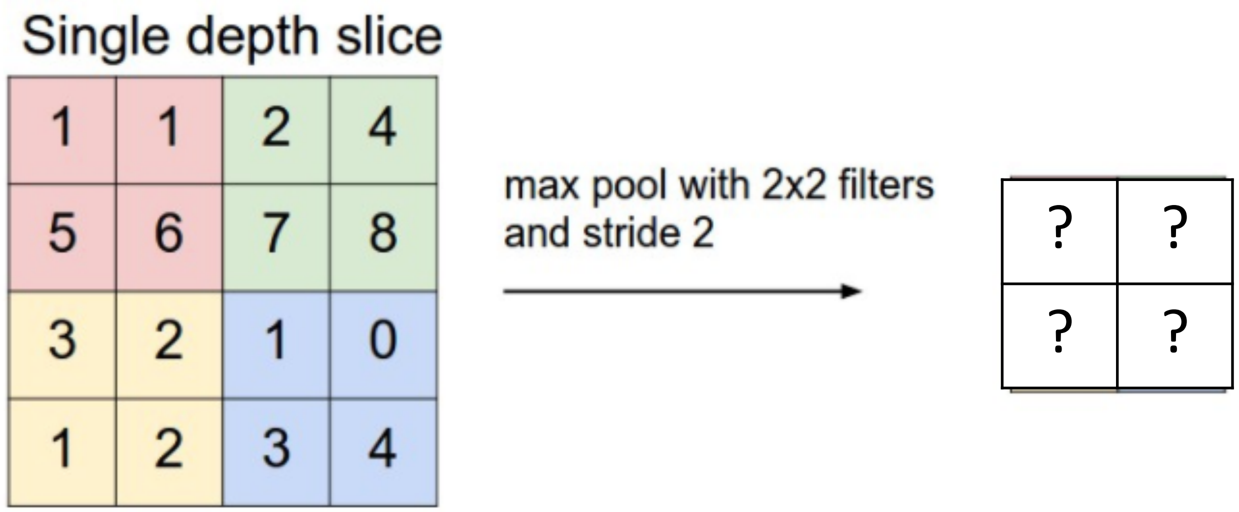
- AlexNet extracts useful features of lower dimension prior to passing it to MLP with:
  - Convolutional layers
  - Pooling Layers




Slide Credit: https://www.slideshare.net/xavigiro/saliency-prediction-using-deep-learning-techniques
A. Krizhevsky, I. Sutskever, G. E. Hinton “ImageNet classification with deep convolutional neural networks”

# Pooling Layer: Summarizes Neighborhood

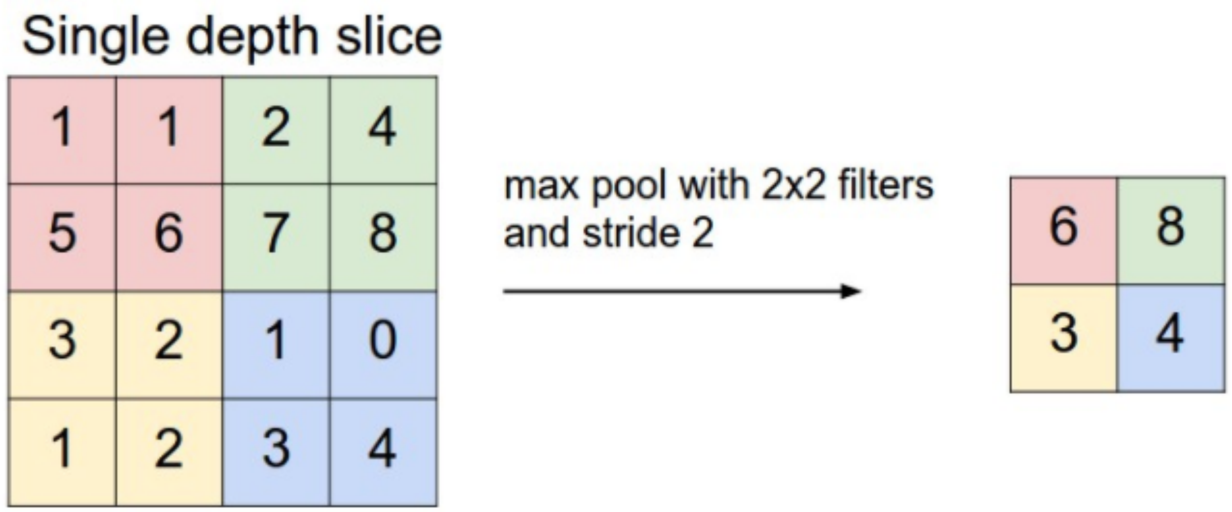
- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk




http://cs231n.github.io/convolutional-networks/#pool

# Pooling Layer: Summarizes Neighborhood

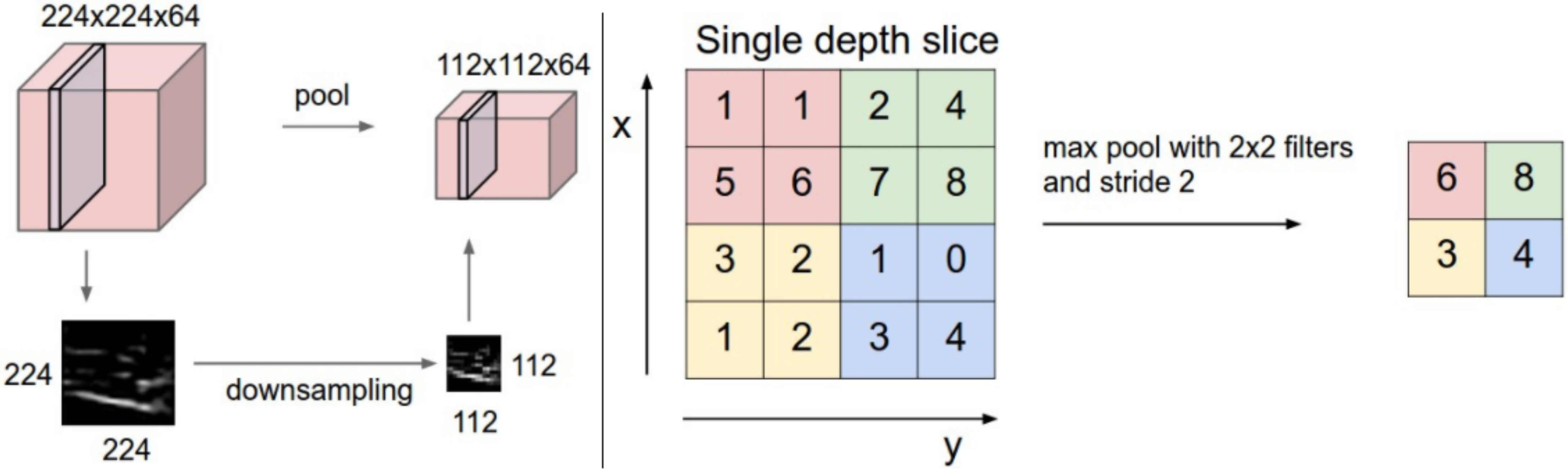
- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk




http://cs231n.github.io/convolutional-networks/#pool

# Pooling Layer: Summarizes Neighborhood

- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk

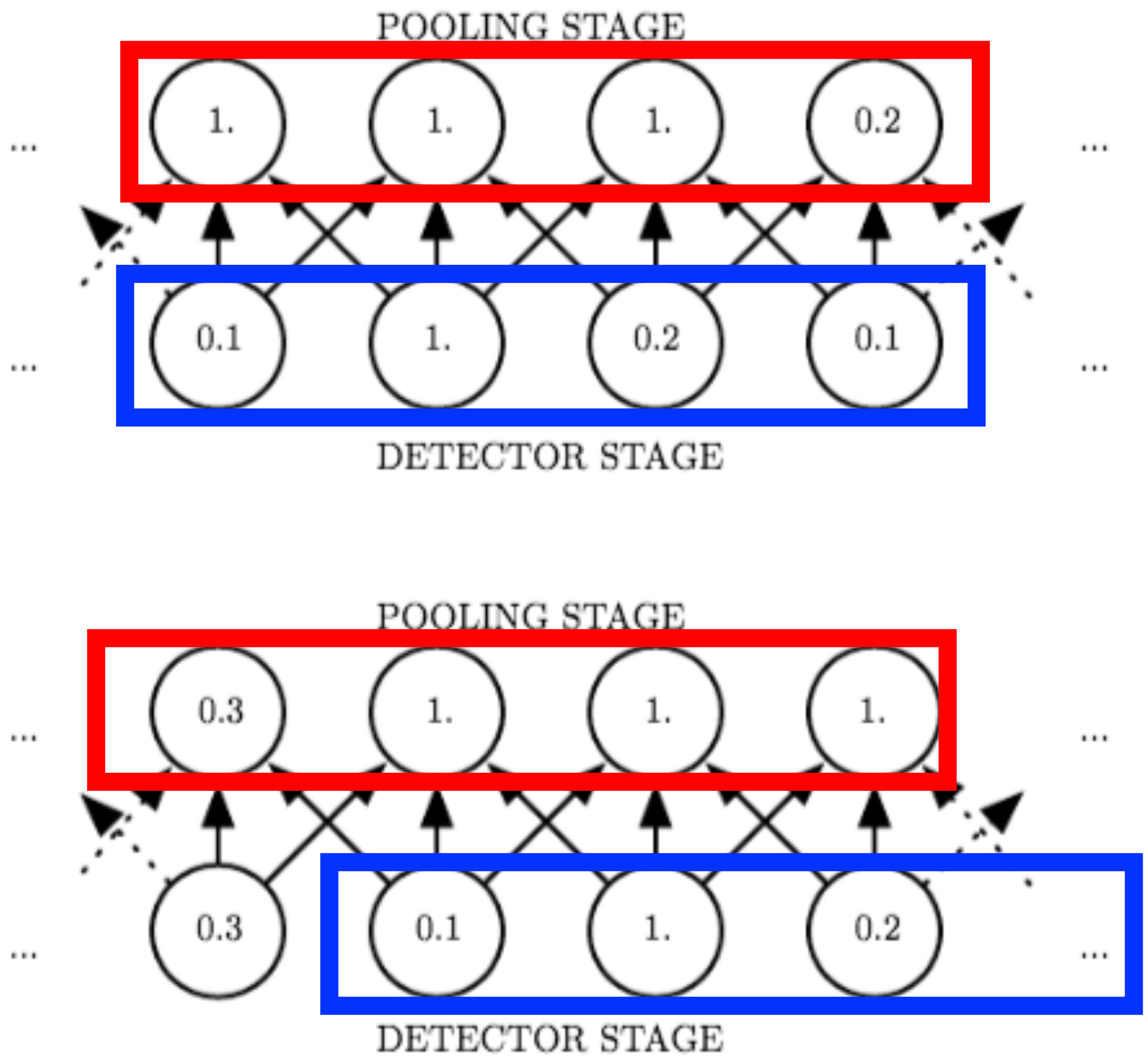



http://cs231n.github.io/convolutional-networks/#pool

# Pooling Layer

- Resilient to small translations

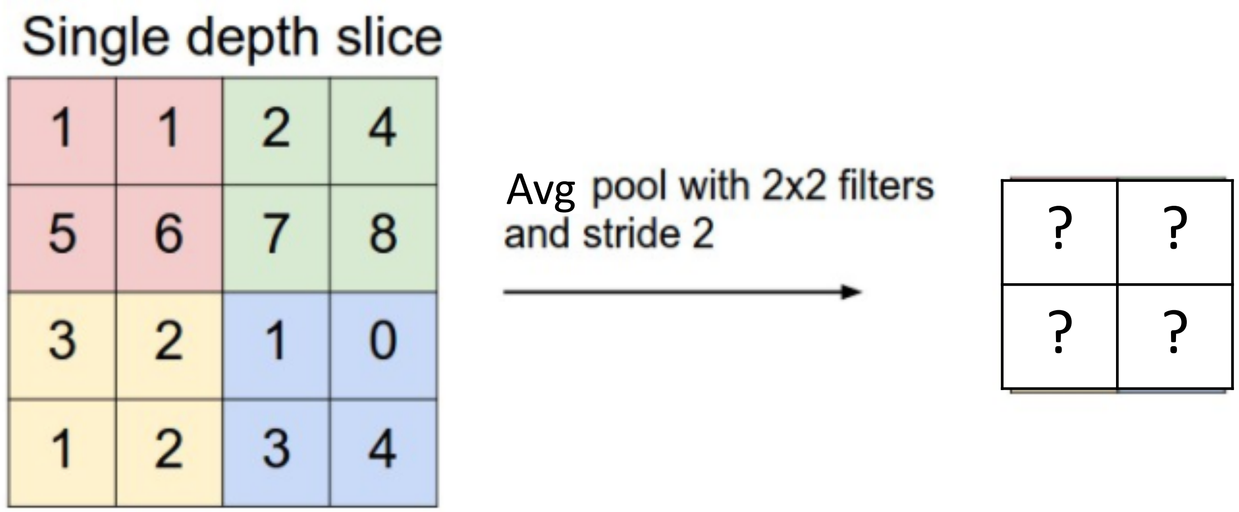
- e.g.,
  - Input: all values change (shift right)
  - Output: only half the values change




https://www.deeplearningbook.org/contents/convnets.html

# Pooling Layer: Summarizes Neighborhood

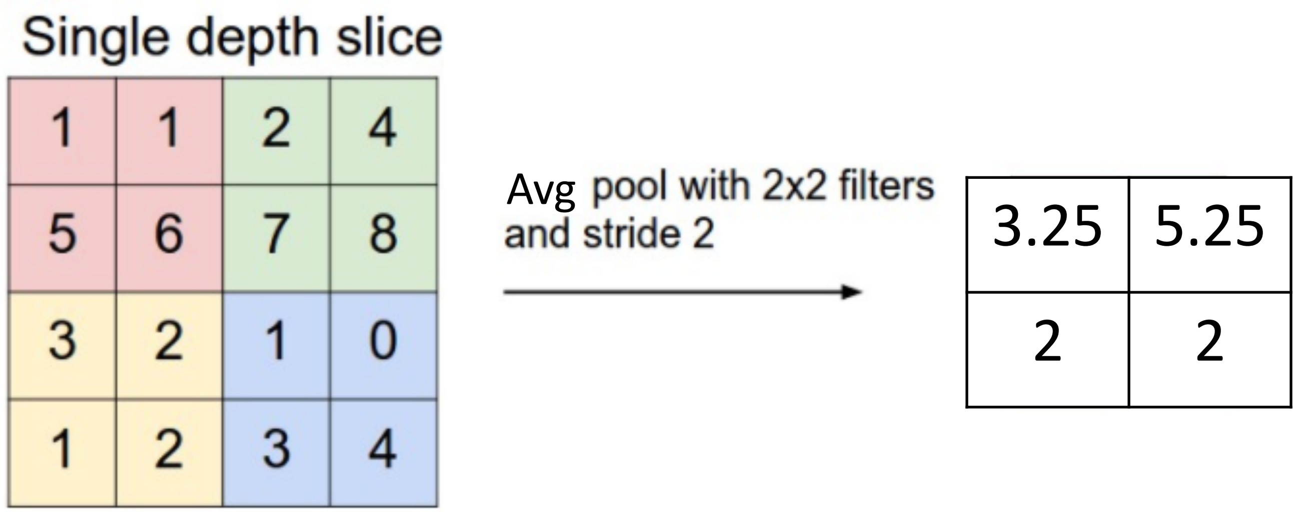
- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk
- **Average-pooling**: partitions input into a set of non-overlapping rectangles and outputs the average value for each chunk




http://cs231n.github.io/convolutional-networks/#pool

# Pooling Layer: Summarizes Neighborhood

- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk
- **Average-pooling**: partitions input into a set of non-overlapping rectangles and outputs the average value for each chunk




http://cs231n.github.io/convolutional-networks/#pool

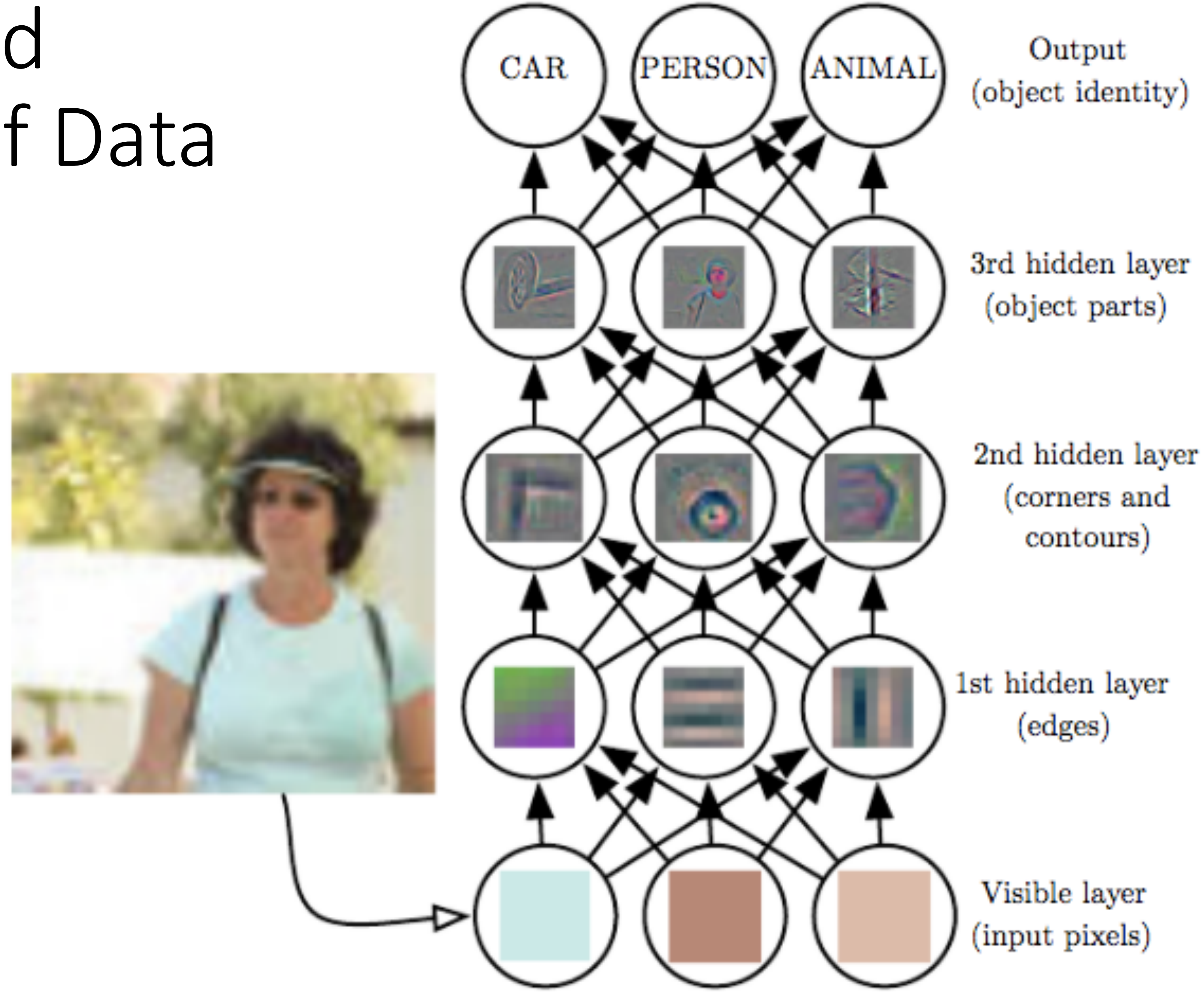
# Pooling Layer: Benefits

- How many parameters must be learned?
  - None
- Benefits?
  - Builds in invariance to translations of the input
  - Reduces memory requirements
  - Reduces computational requirements

# Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Deep Features
- Guest Speaker: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI

# CNN: Learns Good Representation of Data

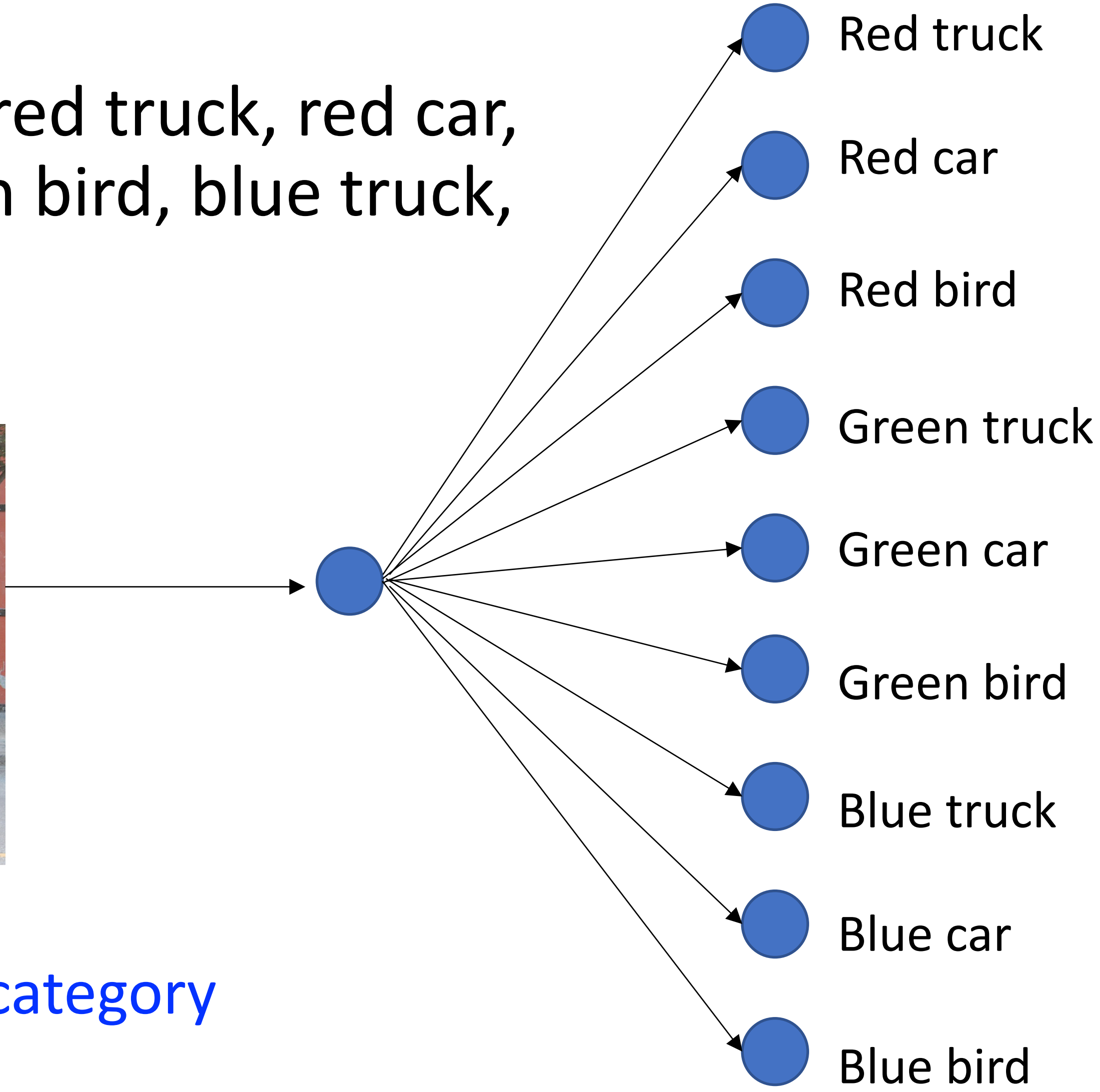



Deep Learning, Ian Goodfellow, Yoshua Bengio, and Aaron Courville

# How to Efficiently Describe/Represent Images?

- e.g., predict for given image if it is a: red truck, red car, red bird, green truck, green car, green bird, blue truck, blue car, & blue bird

e.g., novel image

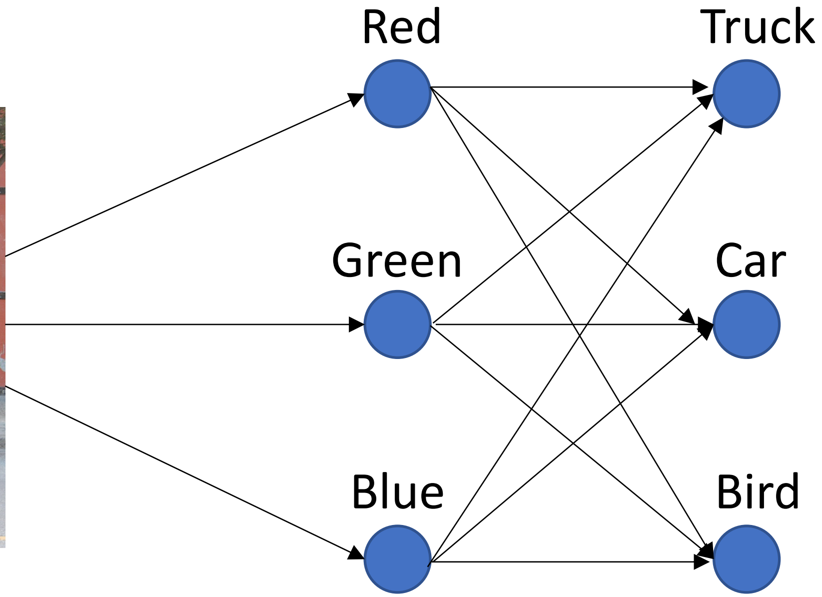




Can train a model to predict each category

# How to Efficiently Describe/Represent Images?

- e.g., predict for given image if it is a: red truck, red car, red bird, green truck, green car, green bird, blue truck, blue car, & blue bird

e.g., novel image


Can design a more efficient model to first capture color and then objects (greater parameter efficiency using hierarchical layers of features)!

# CNN: Learns Good Representation of Data

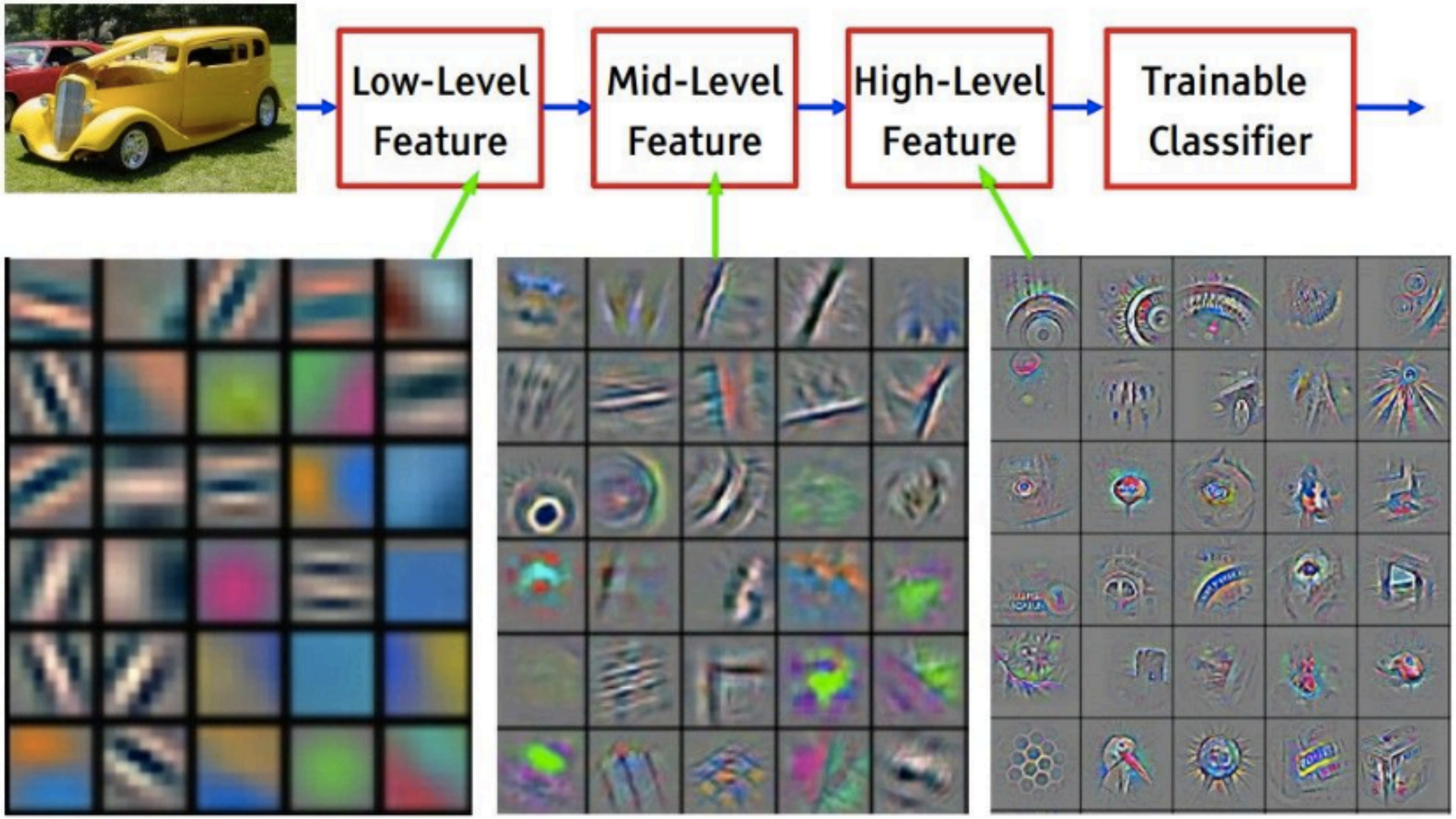
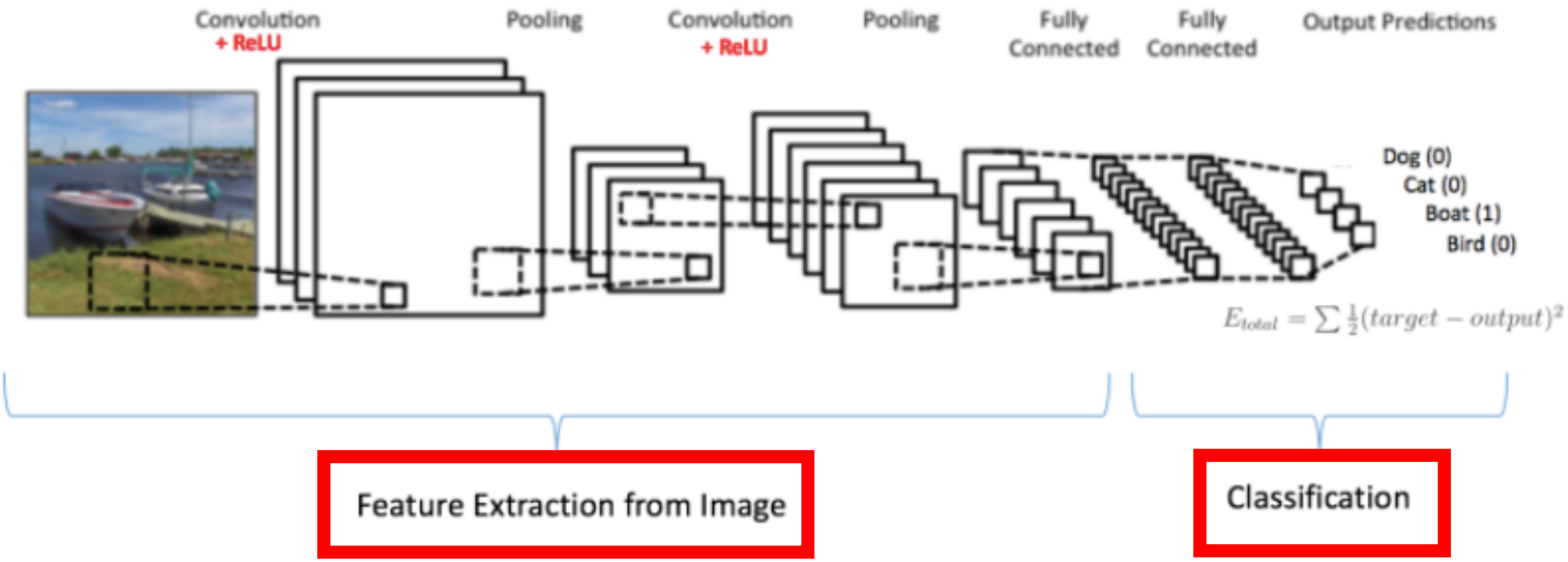



Figure Credit: Yann LeCun

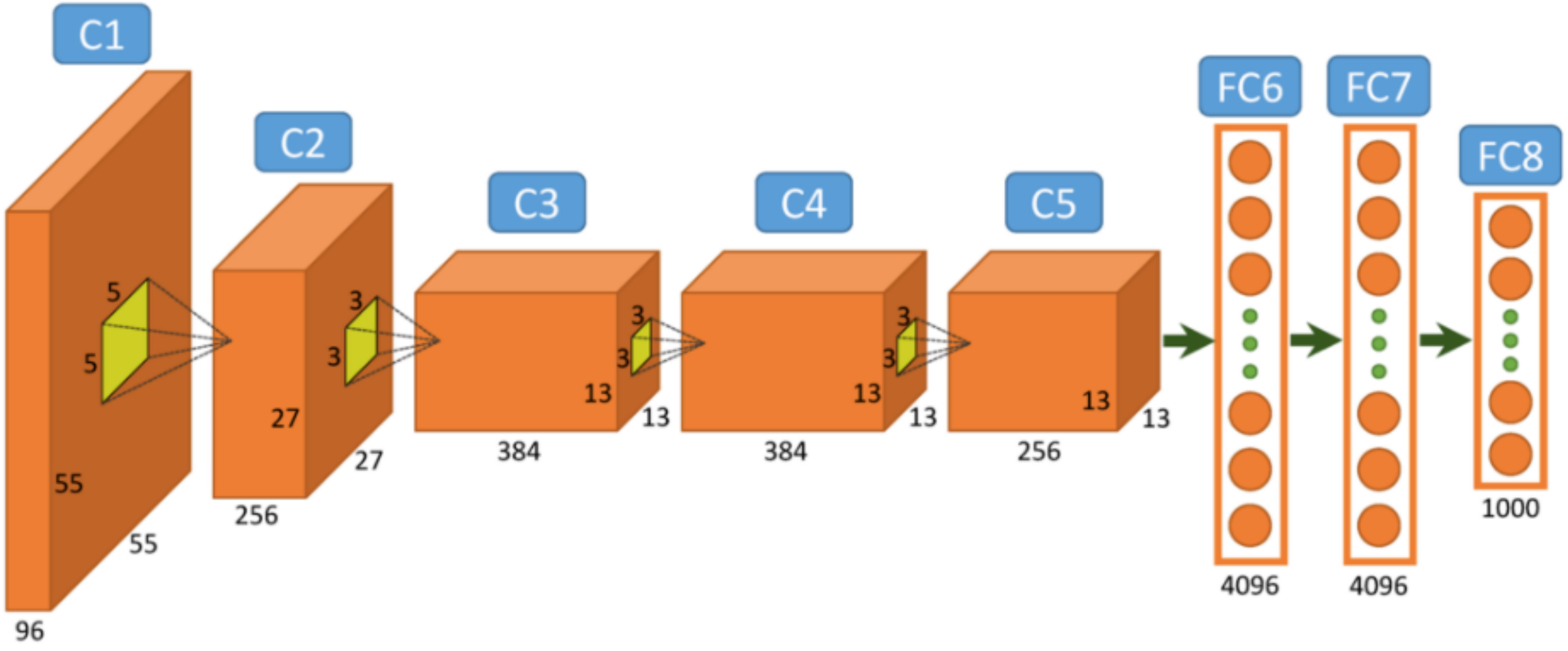
# CNN: Intuition of Different Layers




https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

# AlexNet Deep Features

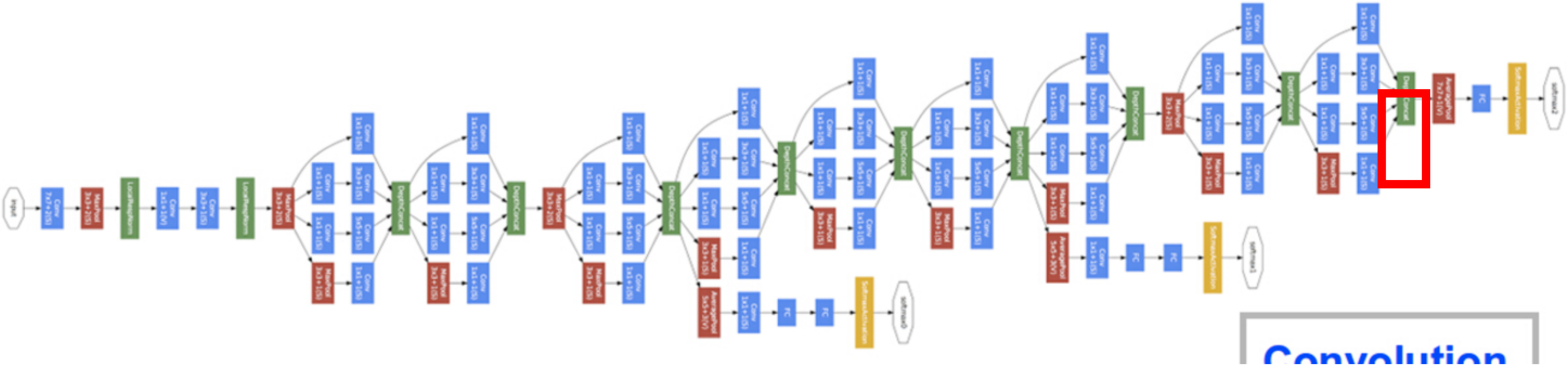
- What is the dimensionality of the fc6 feature?
- What is the dimensionality of the fc7 feature?




https://www.researchgate.net/figure/Architecture-of-Alexnet-From-left-to-right-input-to-output-five-convolutional-layers_fig2_312303454

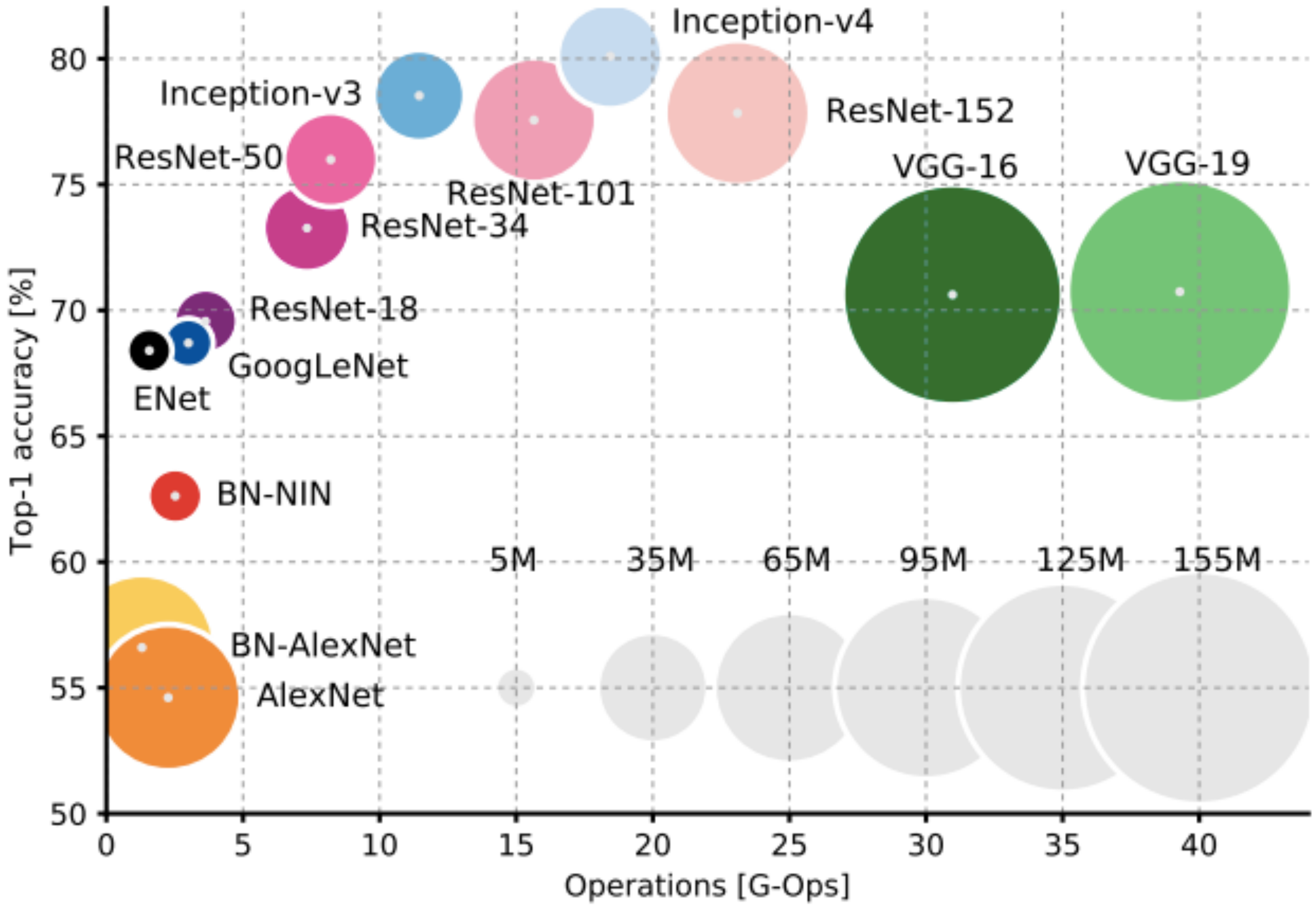
# GoogleNet (Inception) Deep Features

- What is the dimensionality of the inception features?



http://joelouismarino.github.io/blog_posts/blog_googlenet_keras.html

# And Many More Features From...




- VGG16
- VGG19
- ResNet
- Enet
- ...

# CNN Architectures: Input Beyond Images…

- Acoustic/Speech: input treated as an image, with one axis corresponding to time and the other to frequency of spectral components

- Video: one axis corresponds to time, one to the height of the video frame, and one to the width of the video frame

# Google Form: Guest Speaker

- Google form
  - Guest: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI (http://suyogjain.com/): list one question for him for tomorrow's visit

# Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Deep Features
- Guest Speaker: Dr. Suyog Jain, Senior Machine Learning Scientist at PathAI