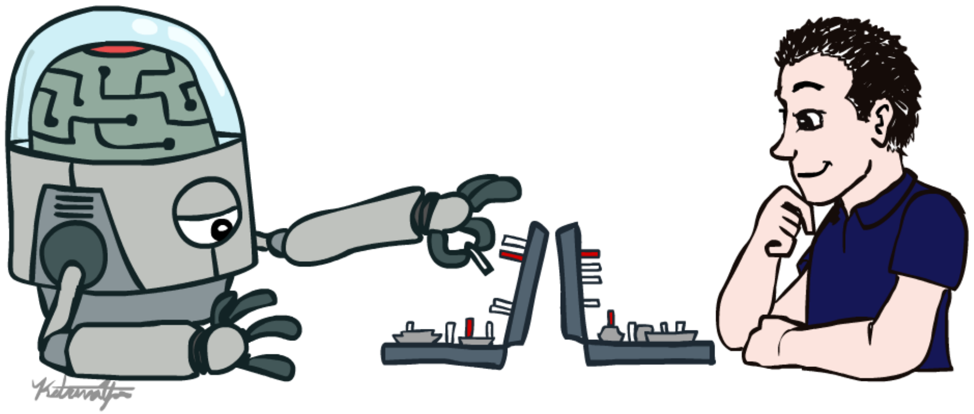
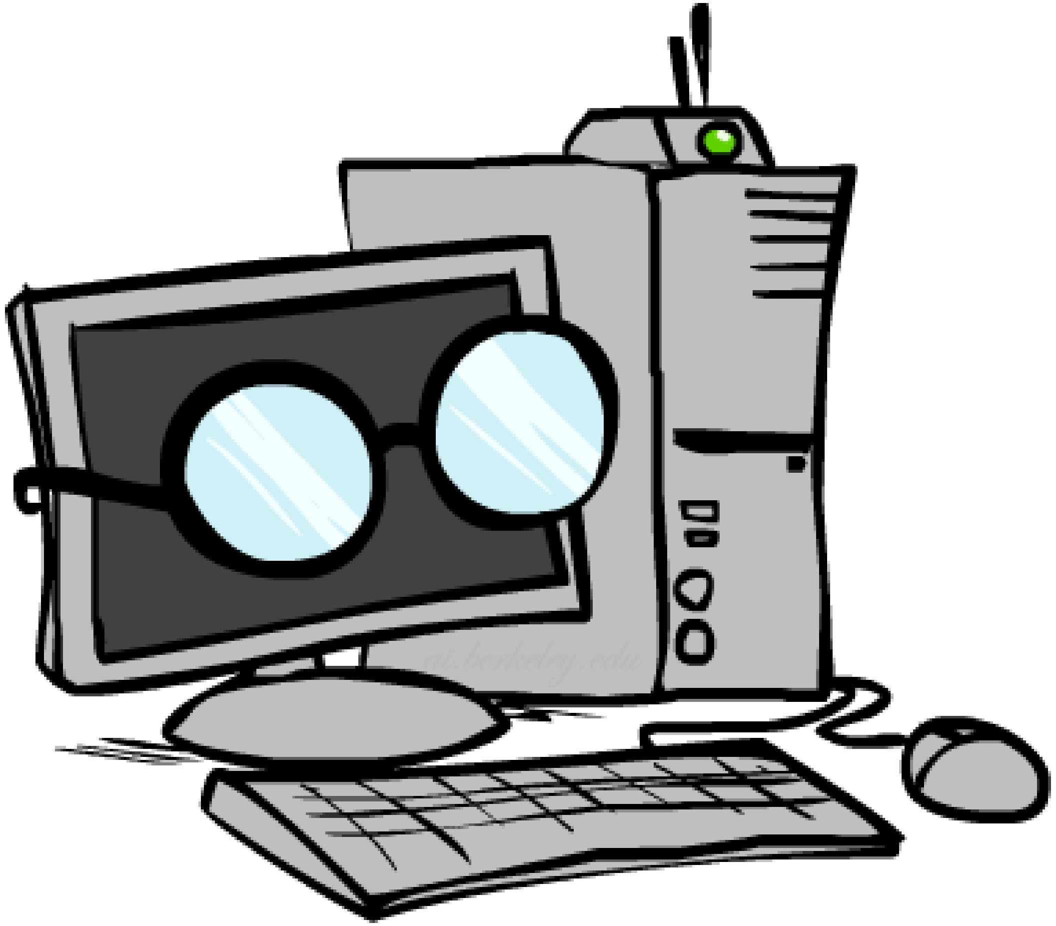
Lecture 12

Ashis Kumar Chanda
chanda@rowan.edu



Rowan University

Computer Vision



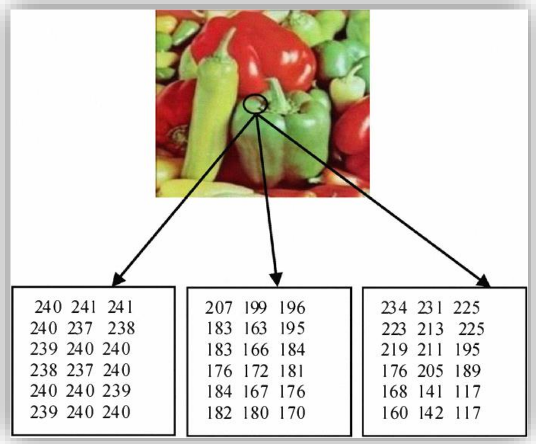
Computer Vision

Grayscale Image

6 x 6

6 x 6

1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2



Picture data is stored in RGB format

Structured data

Computer Vision Problems

Image Classification

64x64

→ Cat? (0/1)

Object detection

Neural Style Transfer


Introduction

Two types of image classification:

- Object (i.e., cat, dog) [background doesn’t matter]
- Background (i.e., grassland, living room)

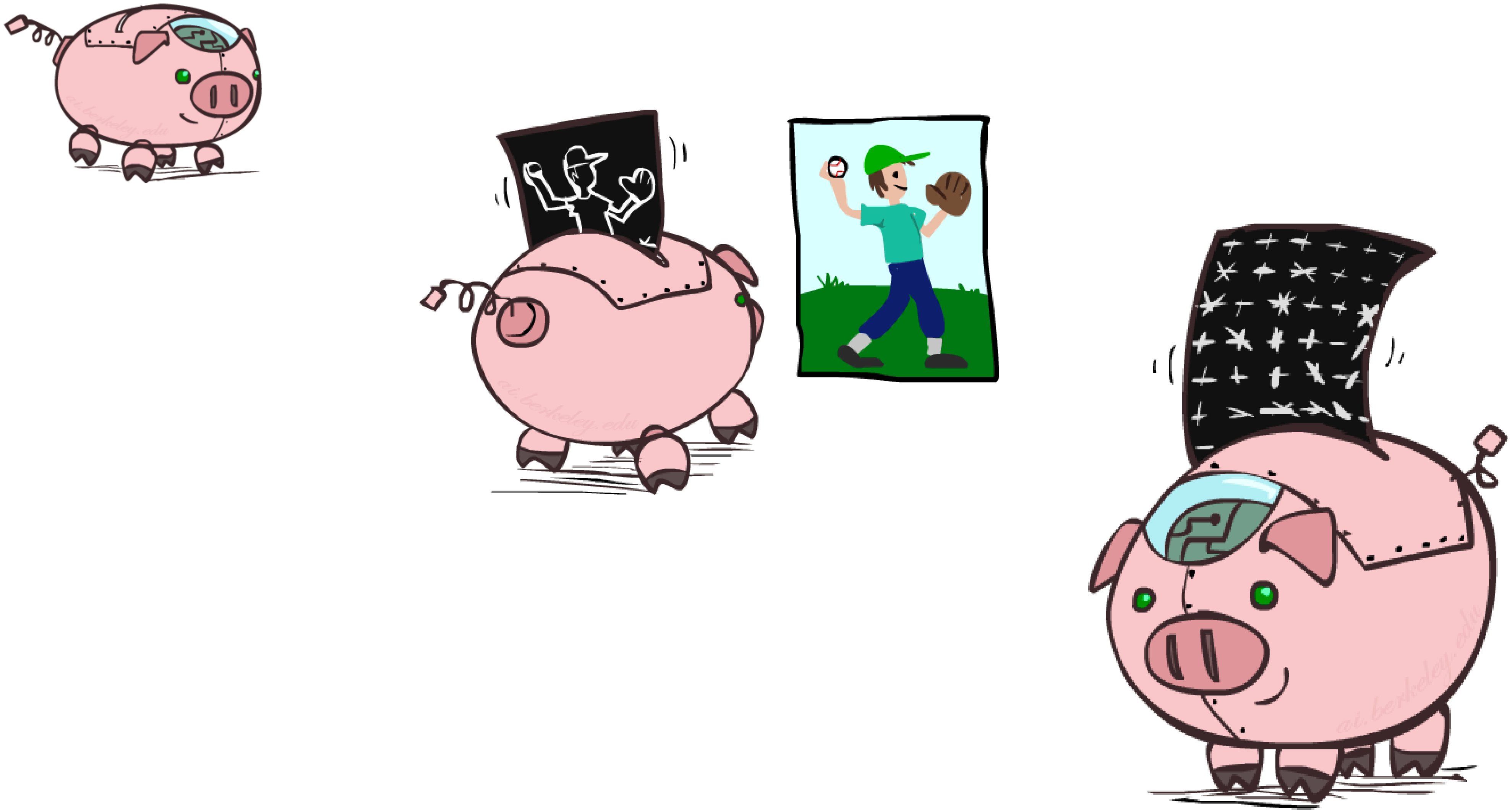
- Effects on images:
  - Lighting: which changes the brightness and color of the image.
  - Aspect: which causes objects to look different when seen from different directions.
  - Occlusion: where some parts of the object are hidden.
  - Deformation: where the object changes its shape.

Introduction

Figure 2. Deformation and scaling problem in product images

Figure 3. Occlusion problem in product images

Manual Feature Design



Introduction

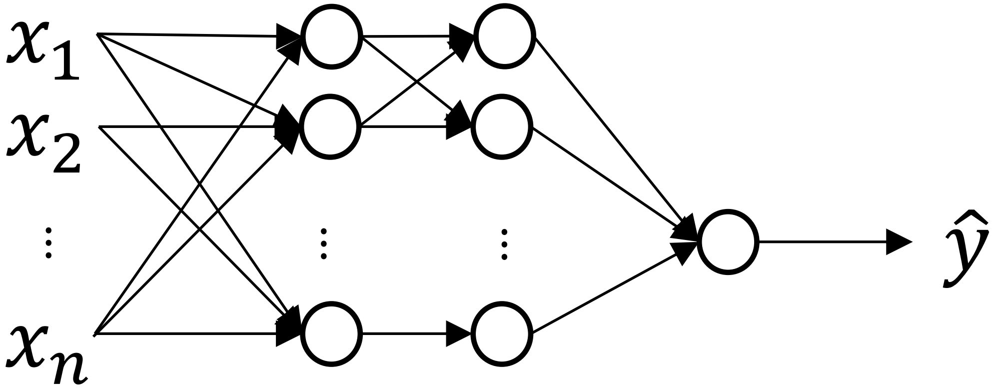
- Individual pixel values are not important.
- Local patterns can be quite informative.
    - The digits 0, 6, 8, 9 have loops.
    - The digits 4, and 8 have crosses.
    - The digits 1, 2, 3, 5, 7 have lines.

- How do we learn about the patterns?

NN on images


64x64

→ Cat? (0/1)



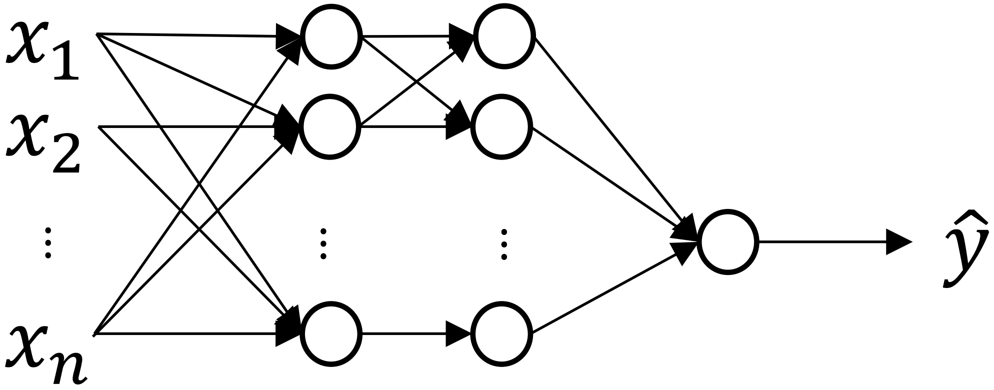
NN on images

64x64×3

→ Cat? (0/1)

12288

1000 x 1000 x 3 = ?



NN on images: problems

64x64x3

Cat? (0/1)

12288

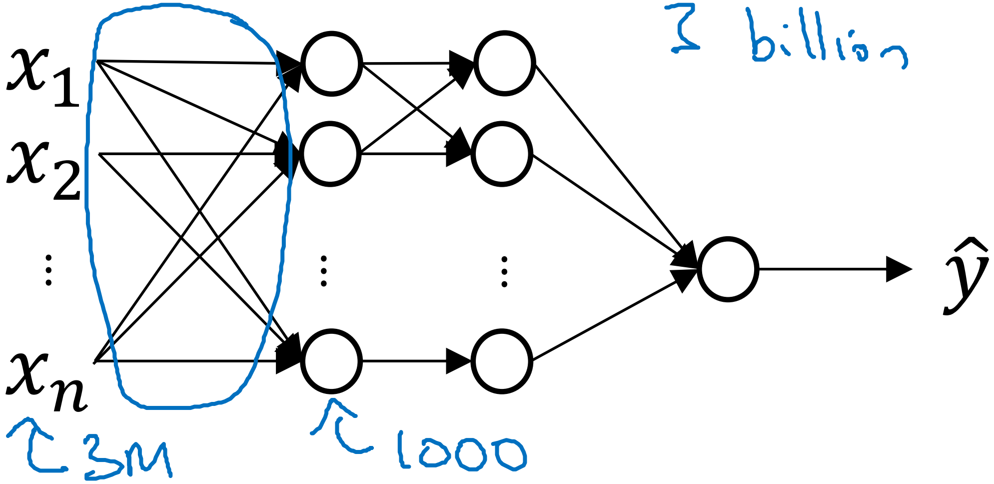
1000x1000x3

= 3 million

$x \in \mathbb{R}^{3M}$

$W^{[1]}$ (1000, 3M)

3 billion



Idea: Reading frame by frame

64x64 × 3

→ Cat? (0/1)

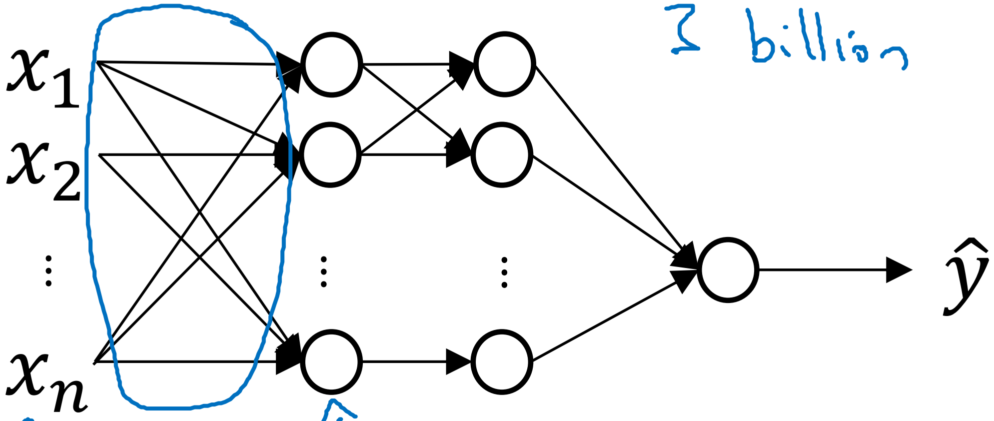
12288

1000 × 1000 × 3
= 3 million

$x \in \mathbb{R}^{3M}$

$W^{[1]}$ (1000, 3M)

3 billion



3M

1000

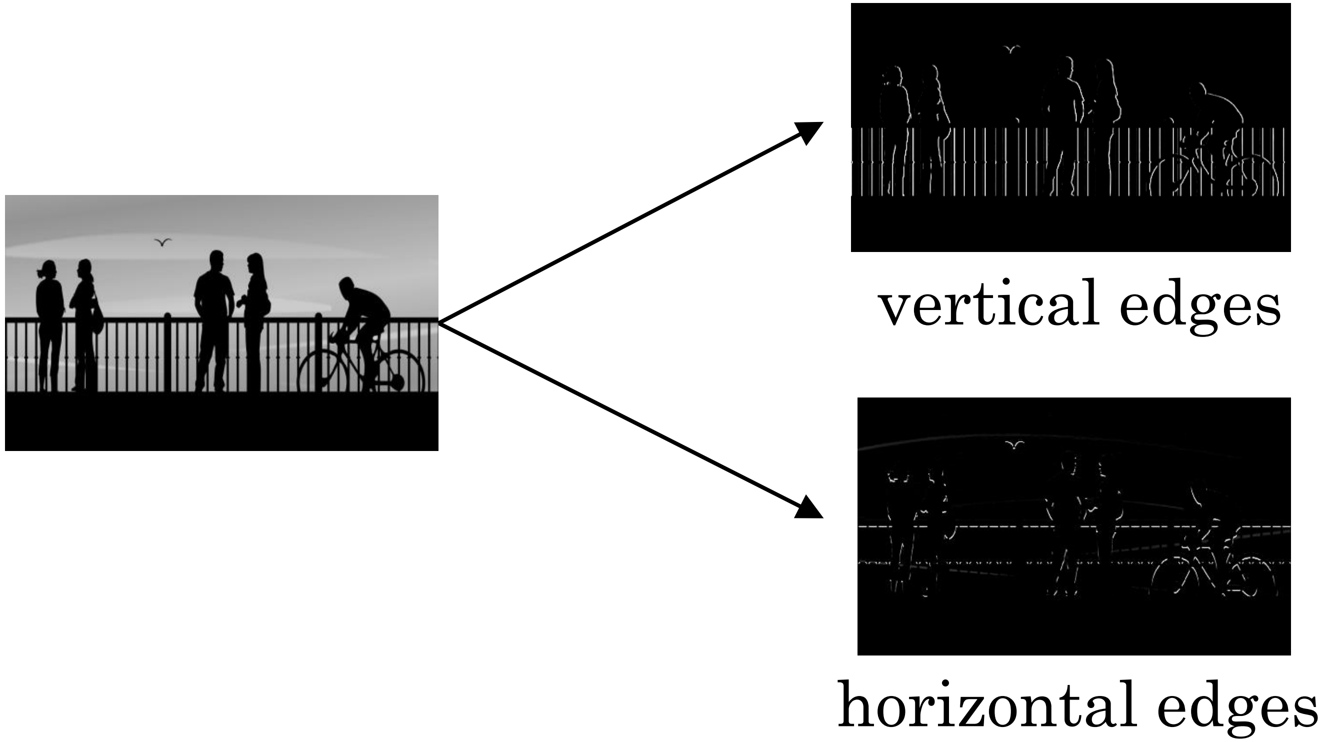
Types of layer in a convolutional network:

- Convolution
- Pooling
- Fully connected

Convolution layer
Edge detection example

Slides adapted from Andrew Ng

Computer Vision Problem



Vertical edge detection

$$
\begin{bmatrix}
3 & 0 & 1 & 2 & 7 & 4 \\
1 & 5 & 8 & 9 & 3 & 1 \\
2 & 7 & 2 & 5 & 1 & 3 \\
0 & 1 & 3 & 1 & 7 & 8 \\
4 & 2 & 1 & 6 & 2 & 8 \\
2 & 4 & 5 & 2 & 3 & 9
\end{bmatrix}
*
\begin{bmatrix}
1 & 0 & -1 \\
1 & 0 & -1 \\
1 & 0 & -1
\end{bmatrix}
=
\begin{bmatrix}
-5 & -4 & 0 & 8 \\
-10 & -2 & 2 & 3 \\
0 & -2 & -4 & -7 \\
-3 & -2 & -3 & -16
\end{bmatrix}
$$

Filter/Kernel
Small region to scan

Stride: the number of gap to slide the filter

Spatial size: [(W – F)/S] +1

Zero padding: adding extra zero column & rows in input to make a desirable output matrix

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

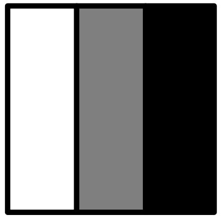
$*$

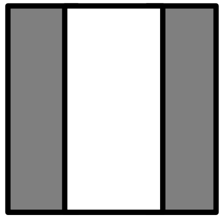
1	0	-1
1	0	-1
1	0	-1

$=$

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

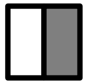
$*$

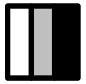


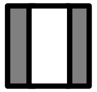


Vertical edge detection examples

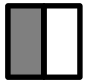
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

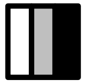


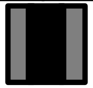




$$\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$







Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

$*$

1	1	1
0	0	0
-1	-1	-1

$=$

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

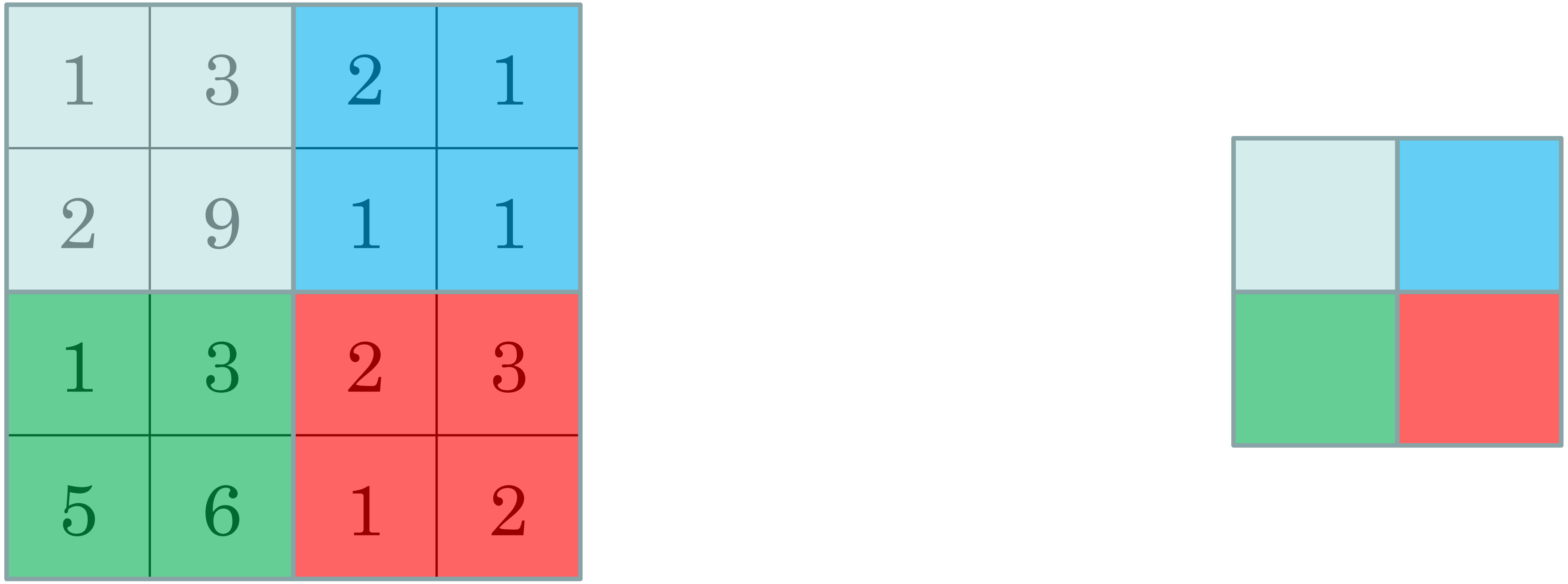
Vertical and Horizontal Edge Detection

(a) Lenna (b) Horizontal edge (c) Vertical edge

Figure : The Lenna image and the effect of different convolution kernels.

Pooling layer

Pooling layer: Max pooling



Reduce the spatial size (Idea: adjacent cells have same similar property)

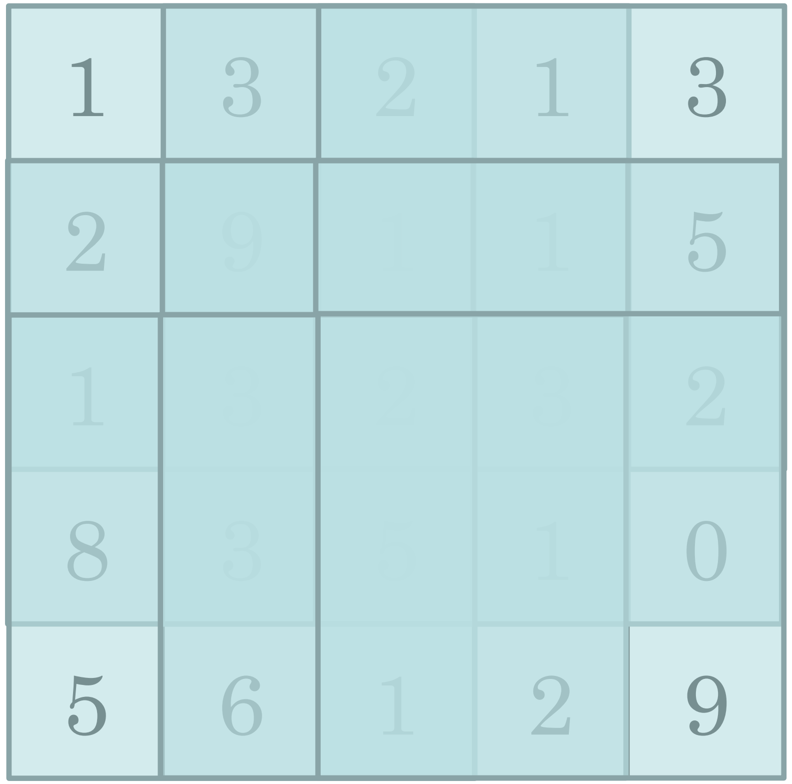
Pooling layer: Max pooling

1	3	2	1	3
2	9	1	1	5
1	3	2	3	2
8	3	5	1	0
5	6	1	2	9

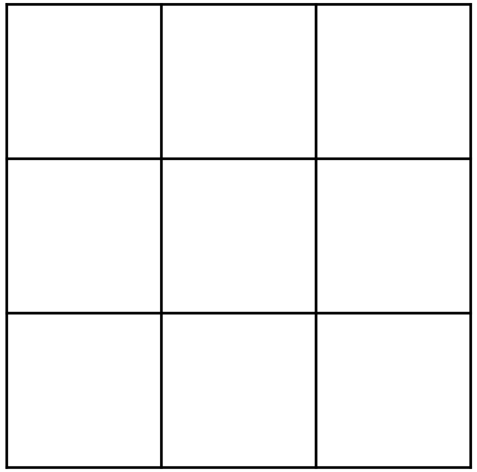
5 * 5

Kernel size is 3 * 3
F = 3
Stride, S = 1

Pooling layer: Max pooling

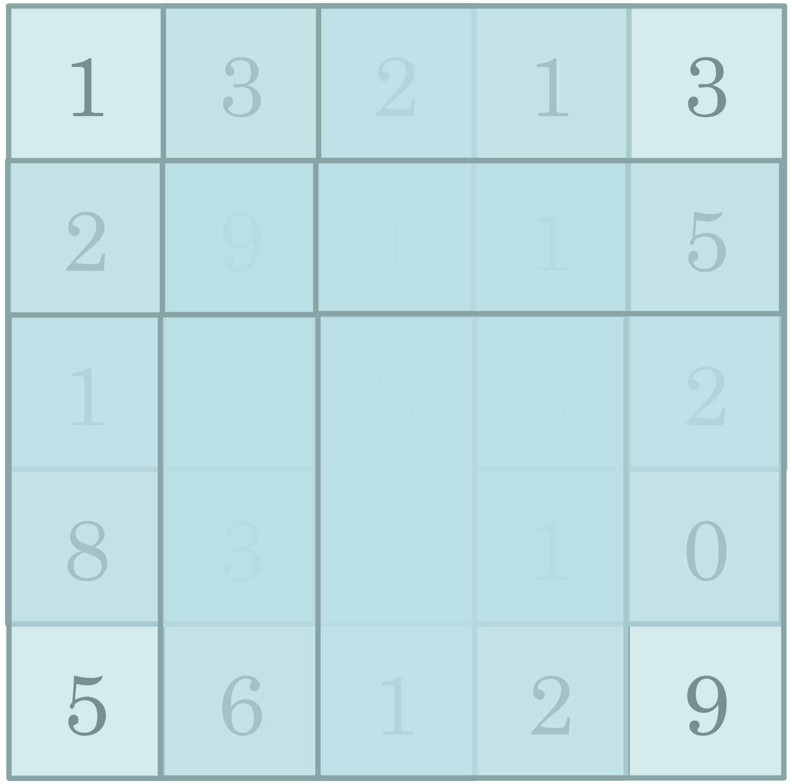


5 * 5



Kernel size is 3 * 3
F = 3
Stride, S = 1

Pooling layer: Max pooling

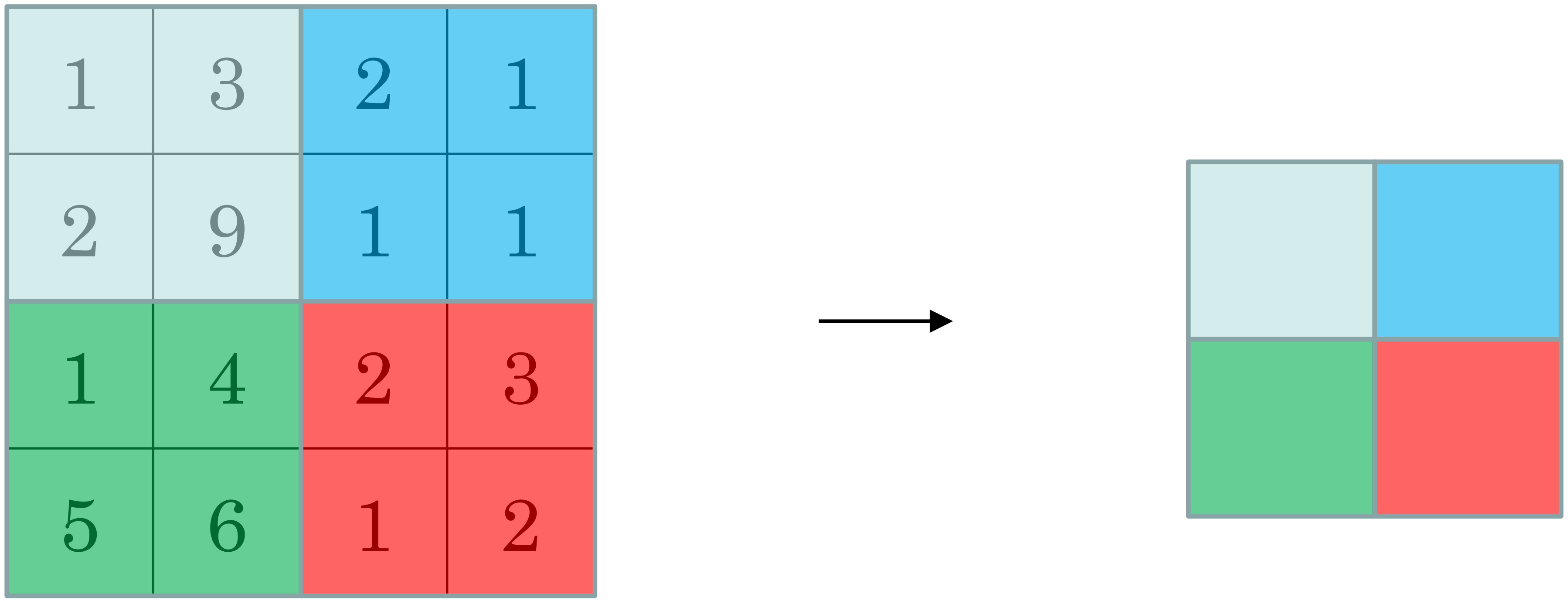


5 * 5

9	9	5
9	9	5
8	6	4

Kernel size is 3 * 3
F = 3
Stride, S = 1

Pooling layer: Average pooling



Kernel size is 2 * 2
F = 2
Stride, S = 2

Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

- Reduce spatial size (memory size)
- Reduce computation cost
- Reduce overfitting

Dropout

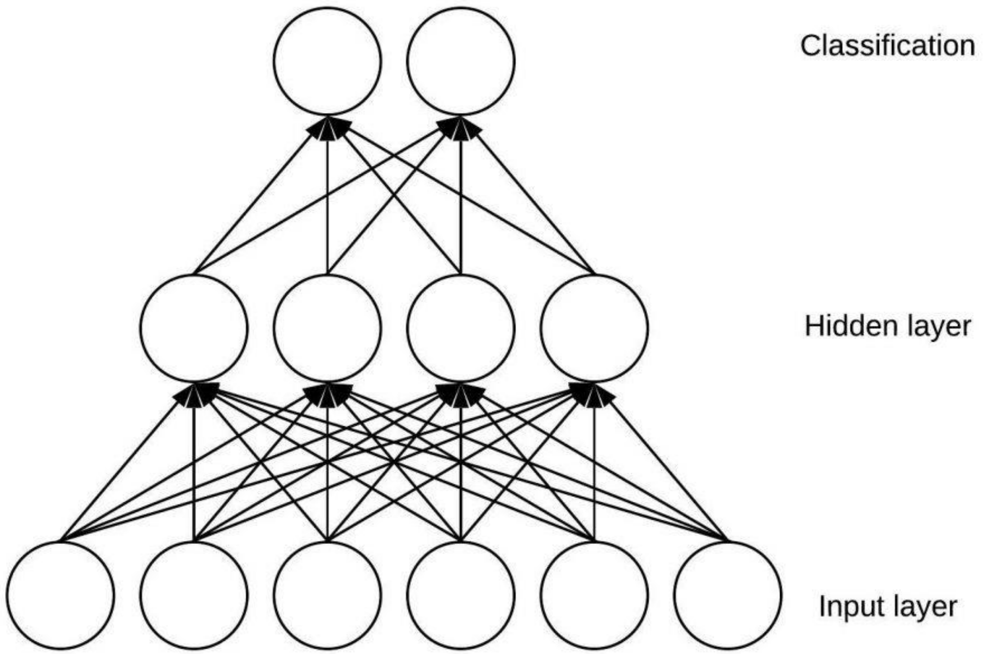
- During training, some number of layer outputs are randomly ignored or "dropped out."

- Dropout is a regularization method.

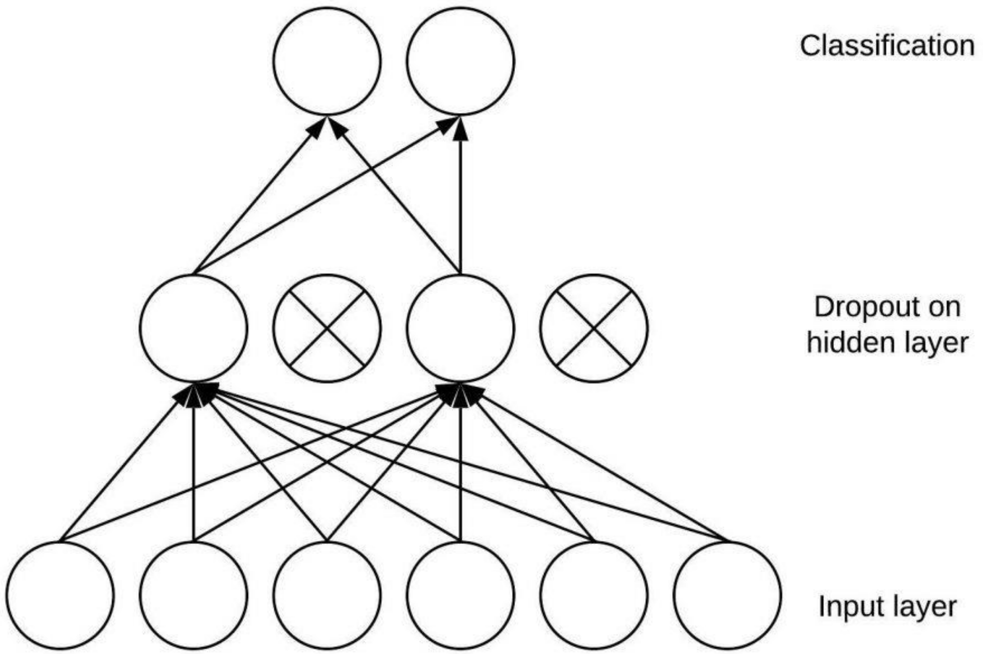
- Dropout is not used after training when making a prediction with the fit network.

- A new hyperparameter is introduced that specifies the probability at which outputs of the layer are dropped out.

Dropout

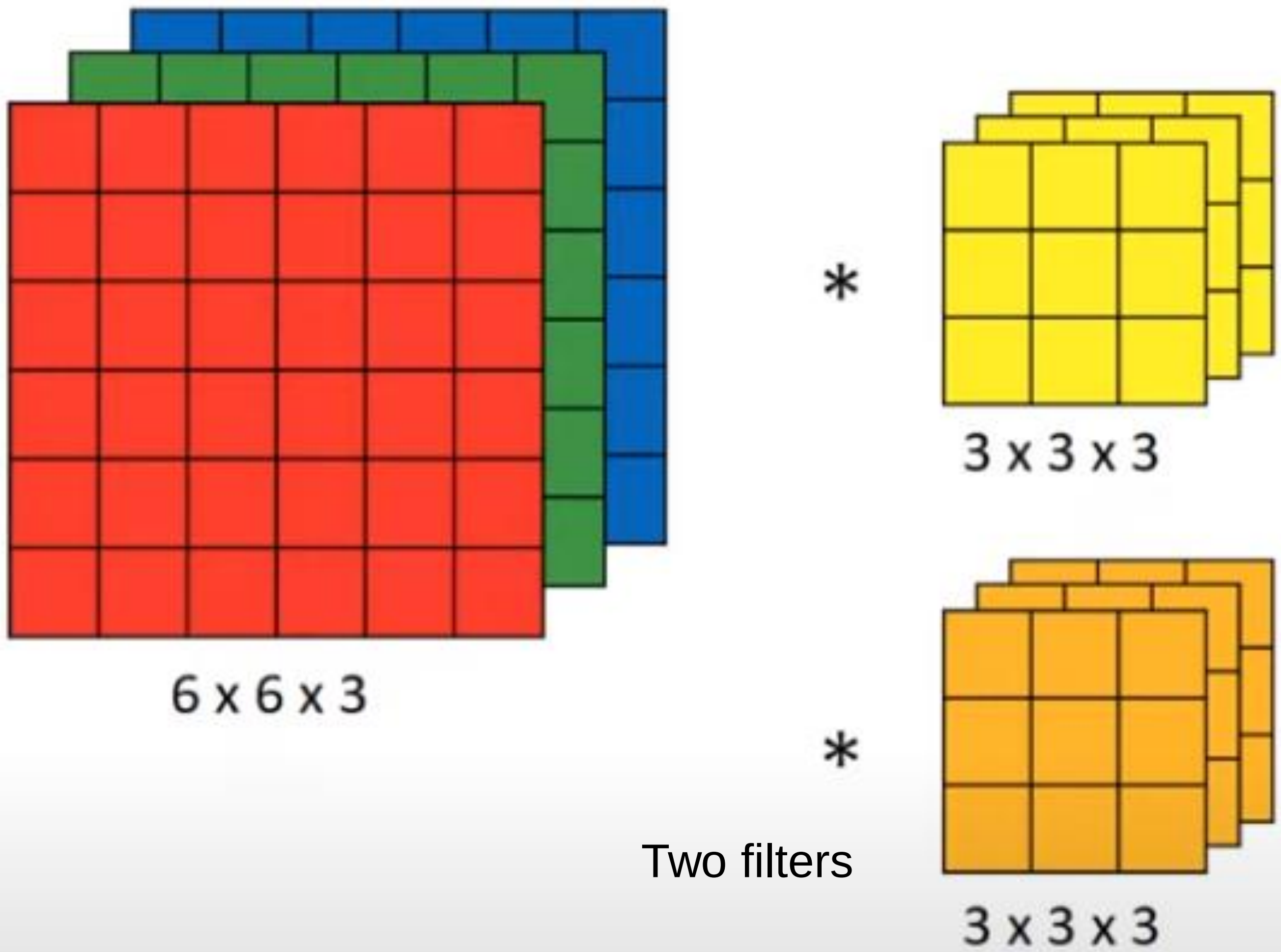


Without Dropout



With Dropout

Example of a layer



Two filters

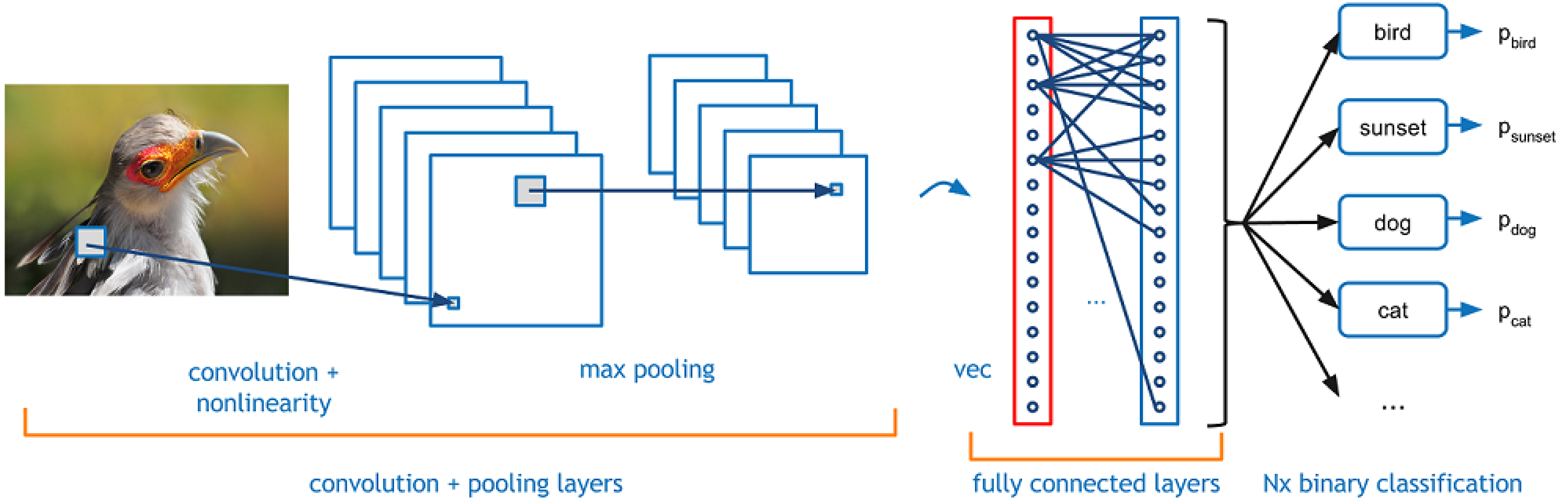
Spatial size: [(W – F)/S] +1
[(6-3)/1] + 1
4

4 by 4

(4 by 4) * 2

4 by 4

CNN: simple architecture



Last layer:
Use sigmoid for binary classification
Use softmax for multi-class problem

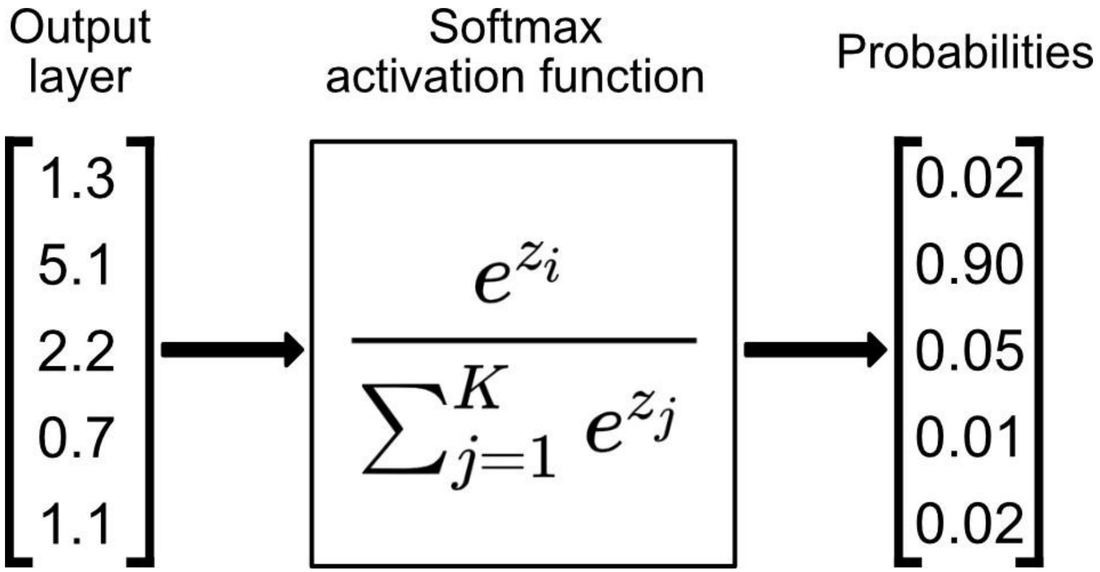
Last/Output layer

$$z = w \cdot x + b$$

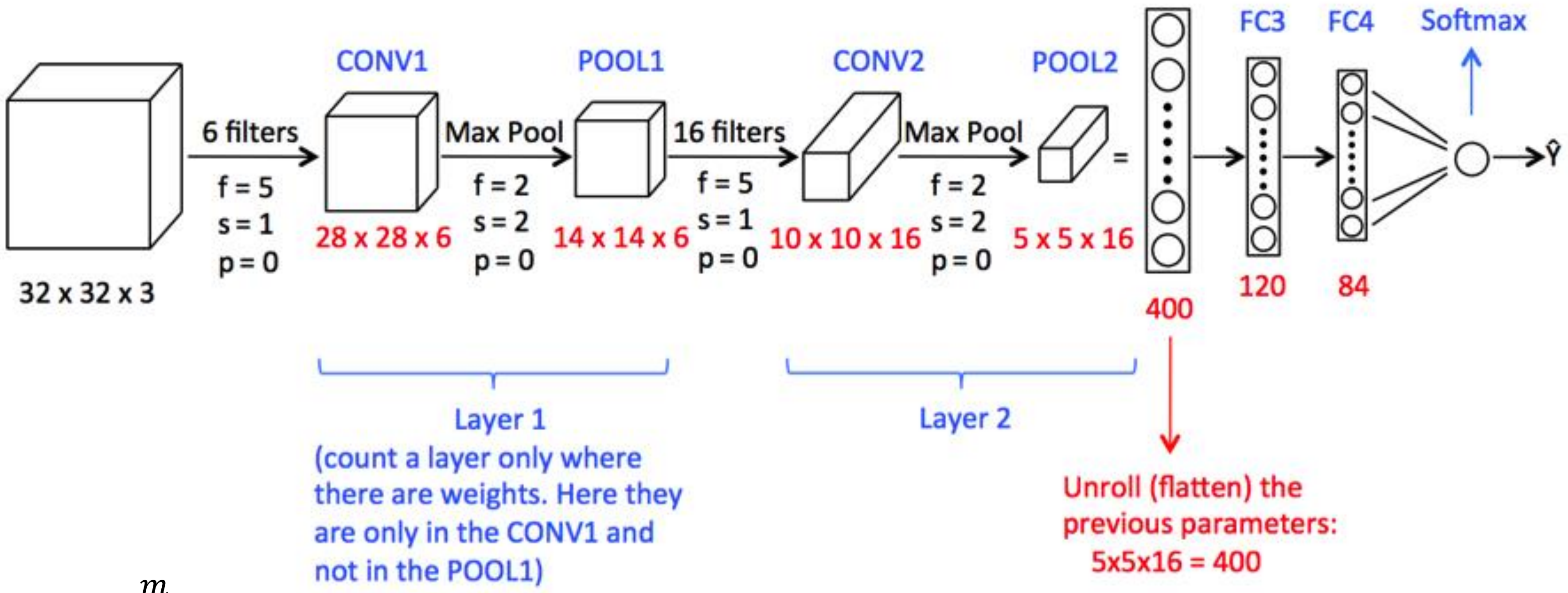
- Solution 1: Sigmoid (use a function of z that goes from 0 to 1)

$$y = s(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)}$$

- Solution 2: Softmax



CNN: adding more layers



$$\text{Cost } J = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

Why convolutions

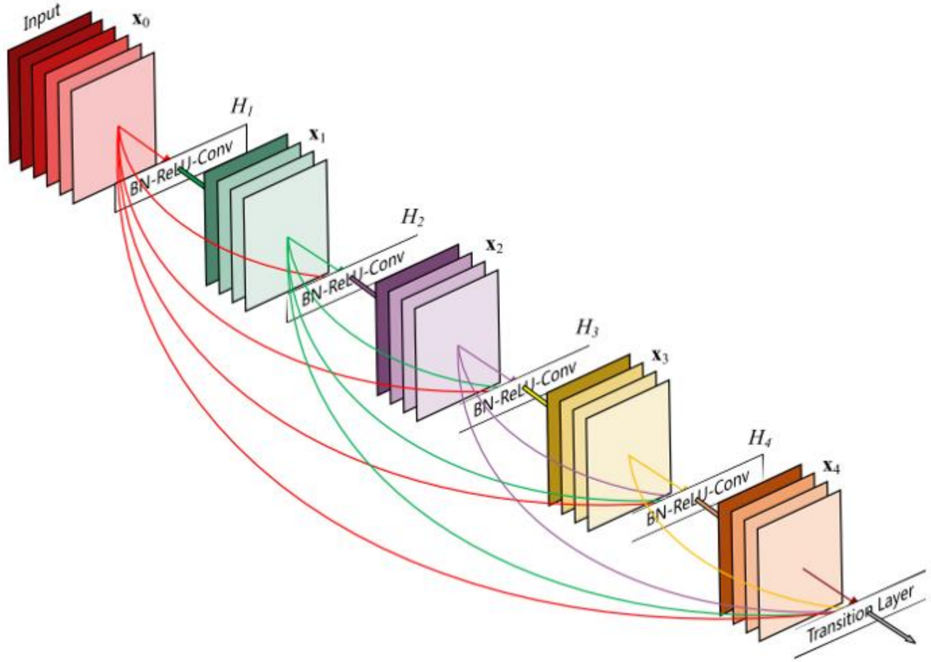
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

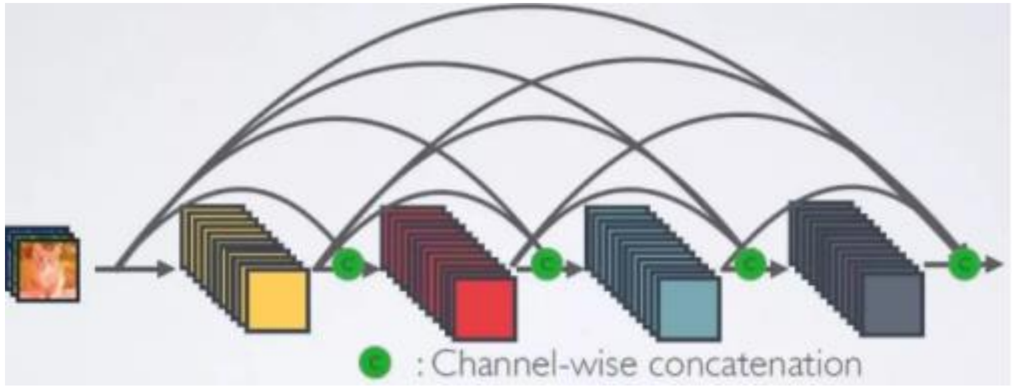
Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

Deep CNN Model

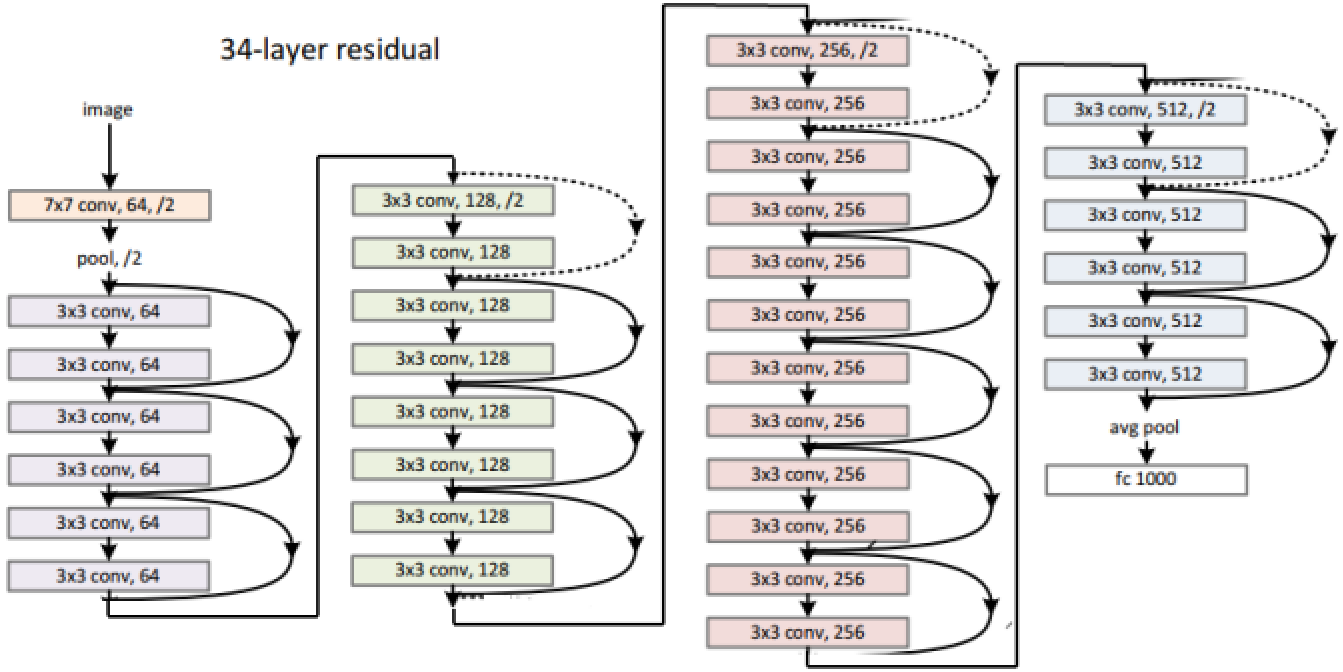
- DenseNet





Deep CNN Model

- Residual Network (ResNet)

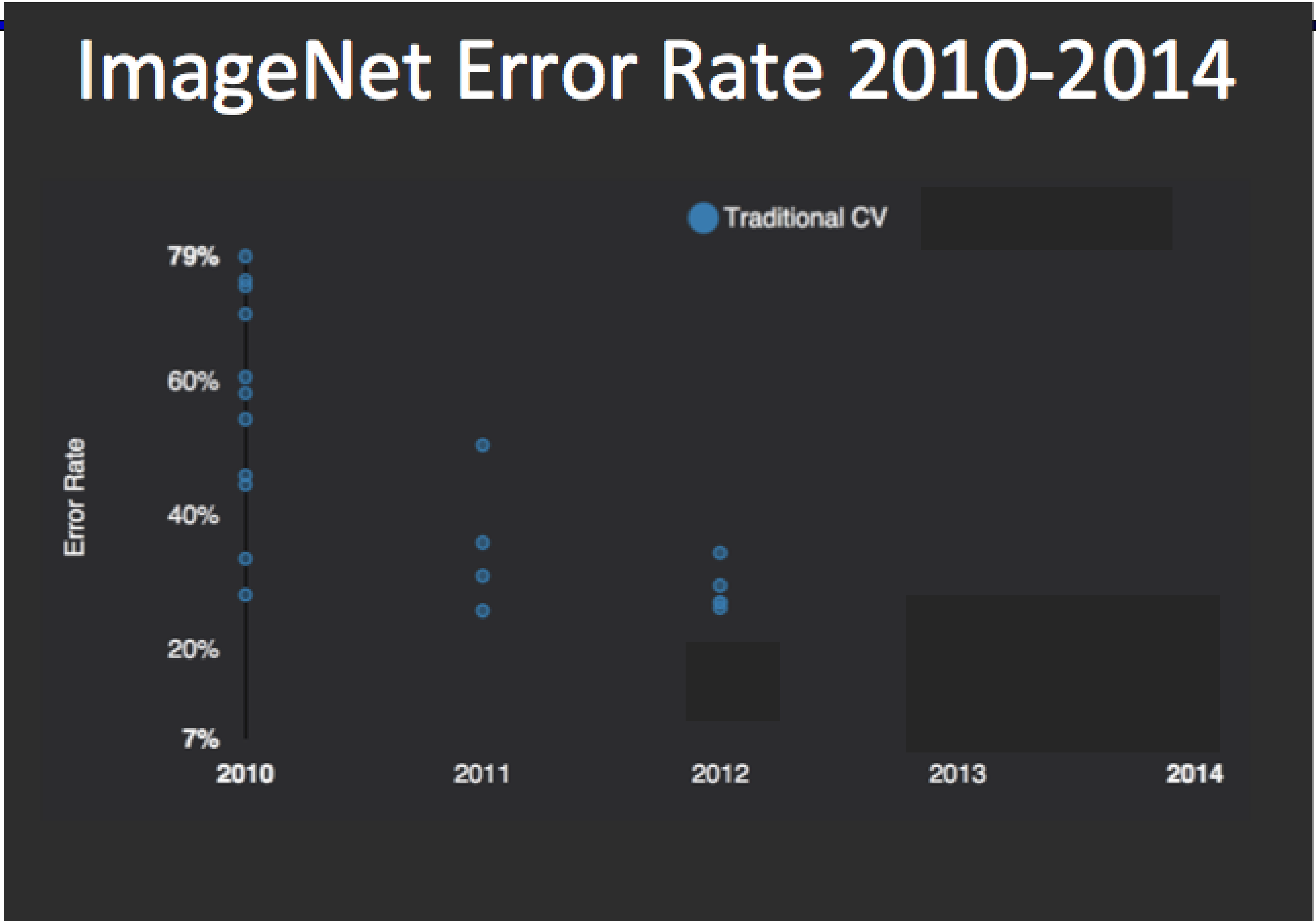


Simplest resNet example given in paper - 34 layers

Deep learning in Image Classification

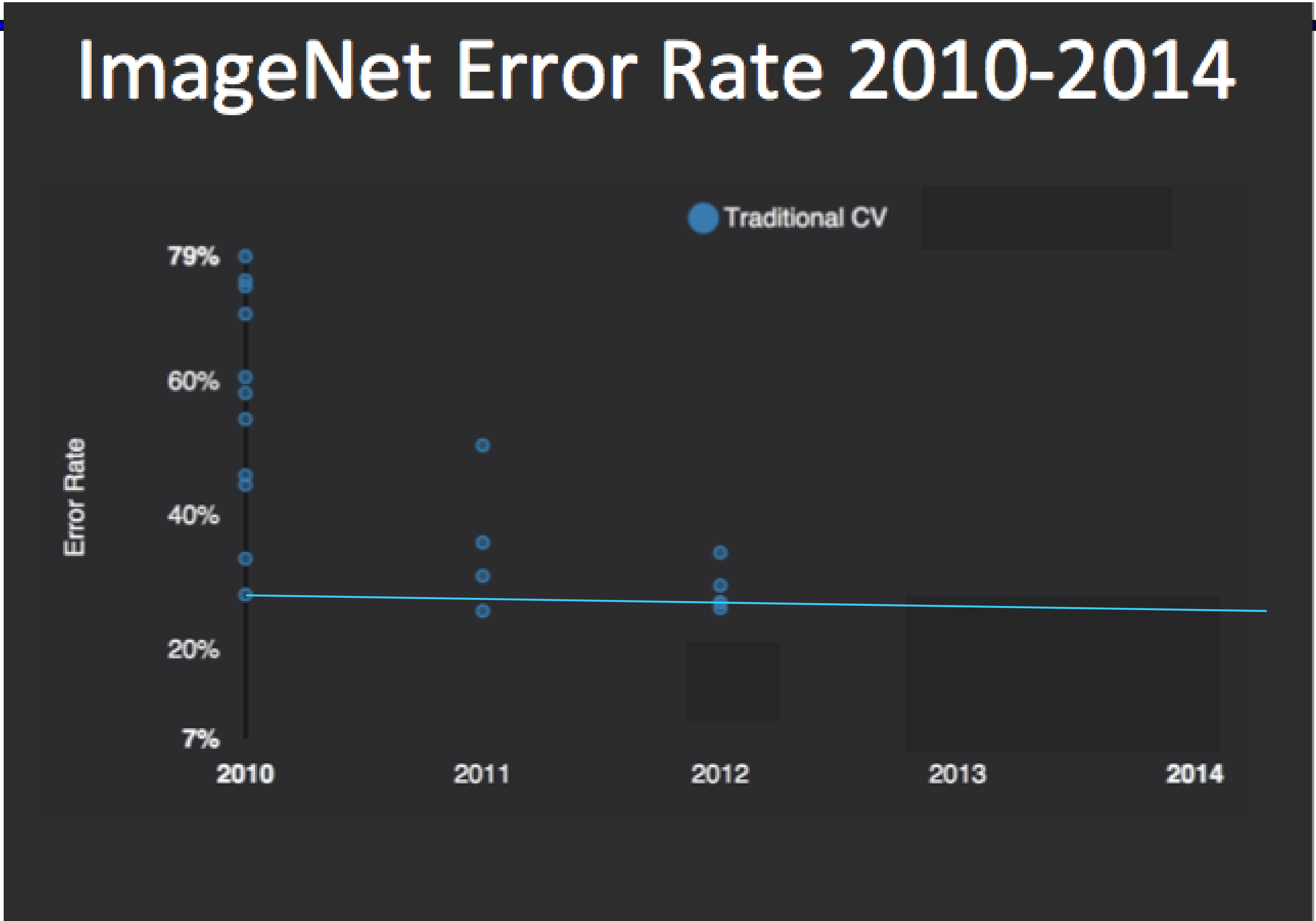
A real example

ImageNet: 14 million images; 30,000 categories



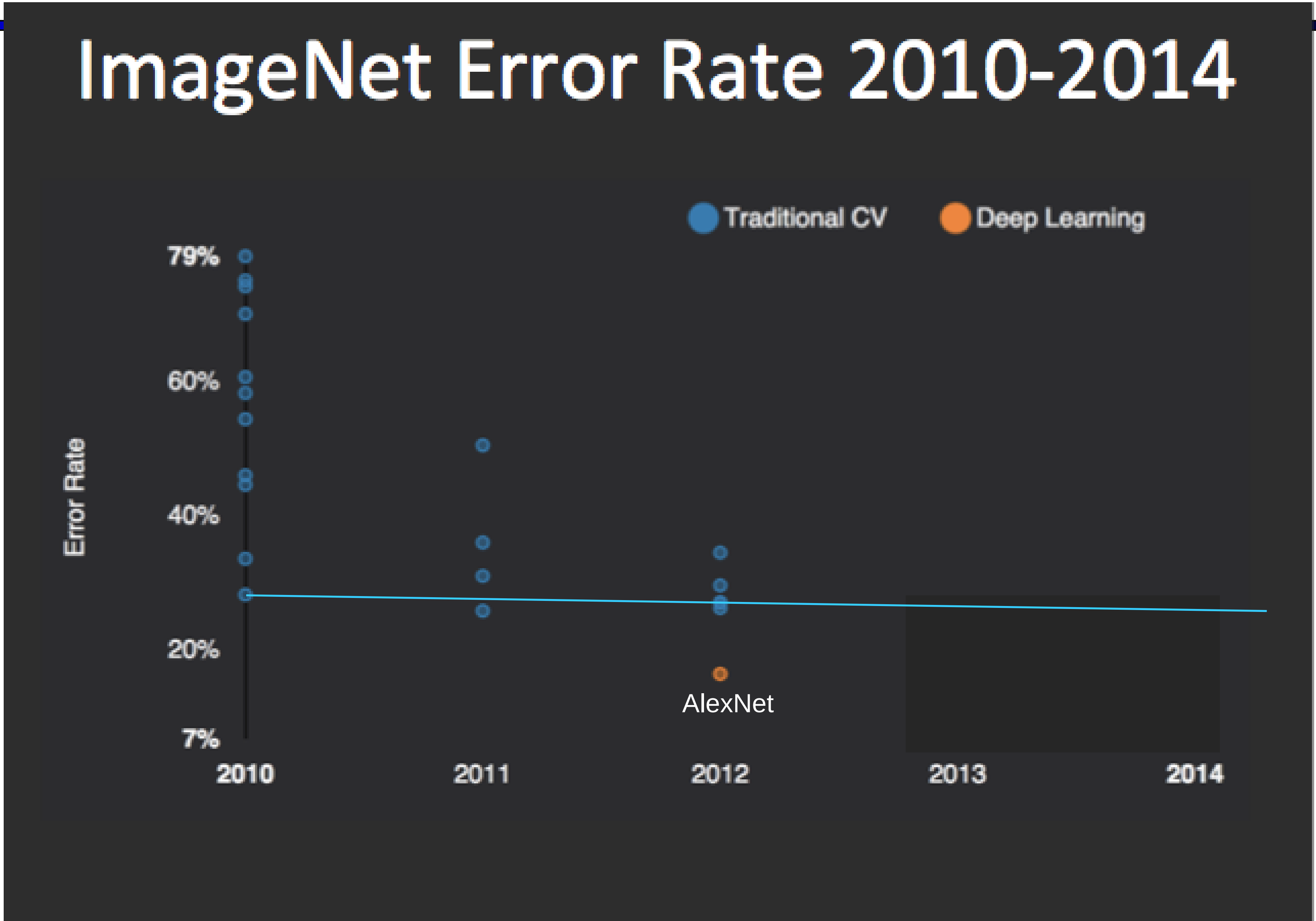
graph credit Matt Zeiler, Clarifai

Performance



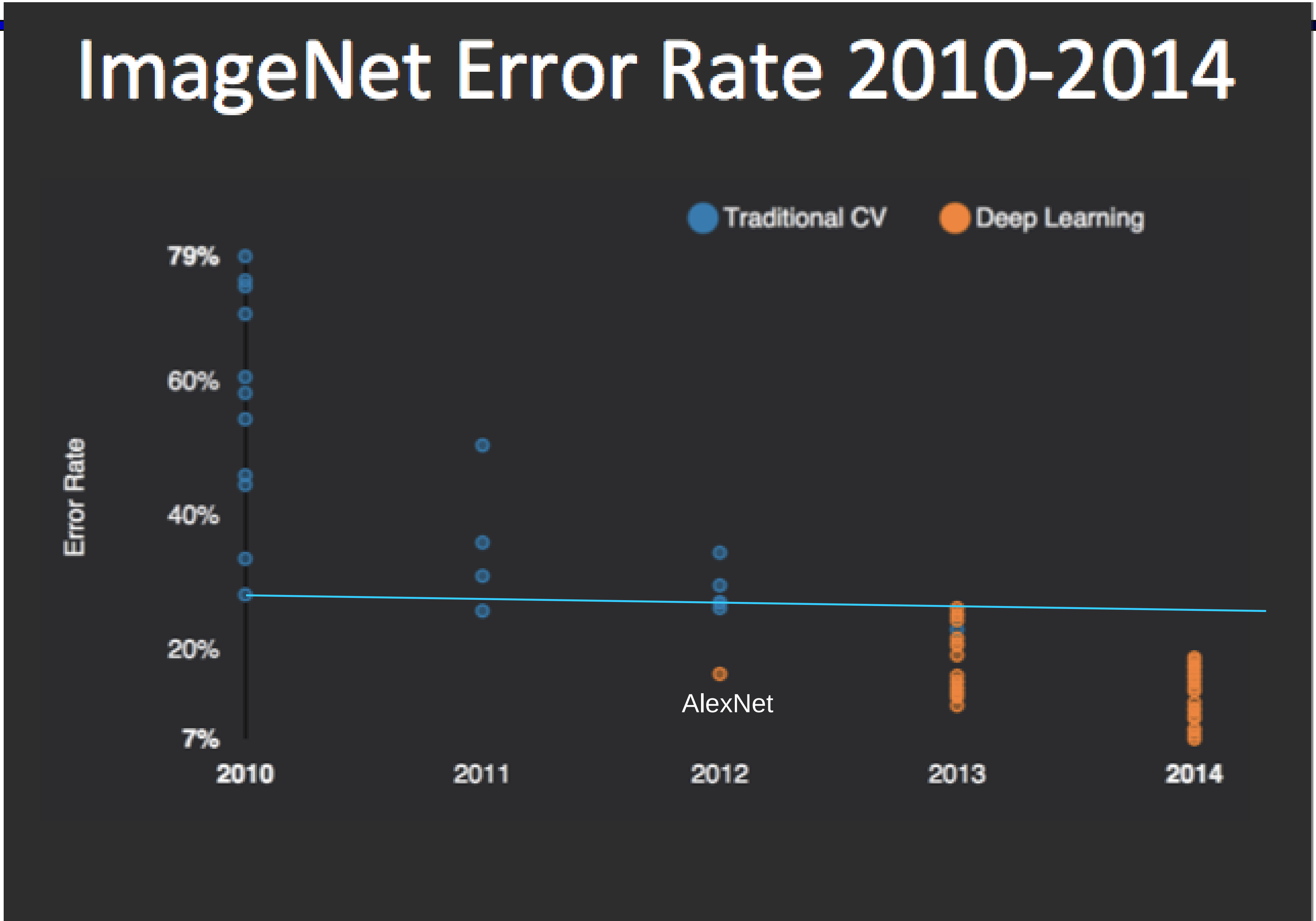
graph credit Matt Zeiler, Clarifai

Performance



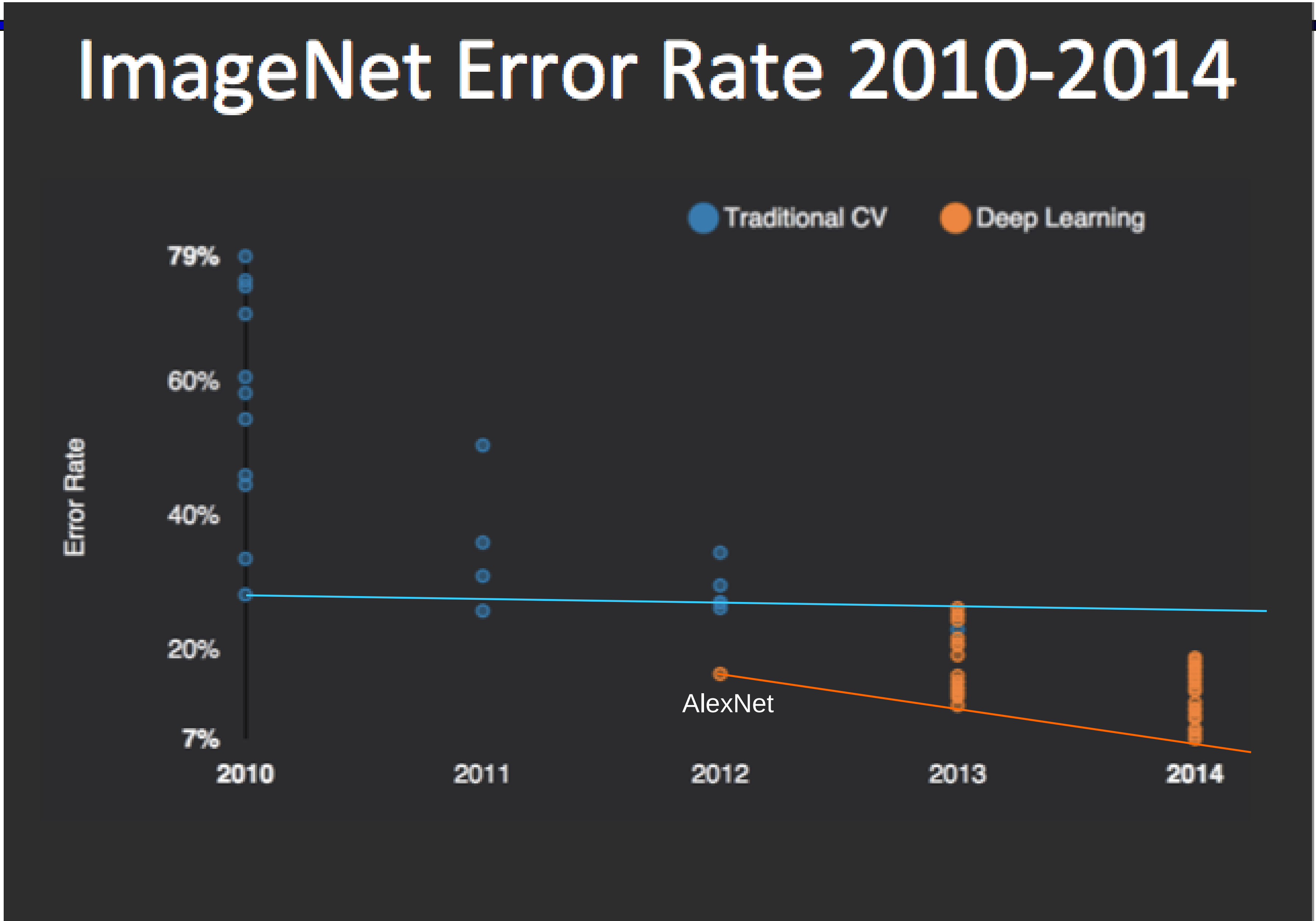
graph credit Matt Zeiler, Clarifai

Performance



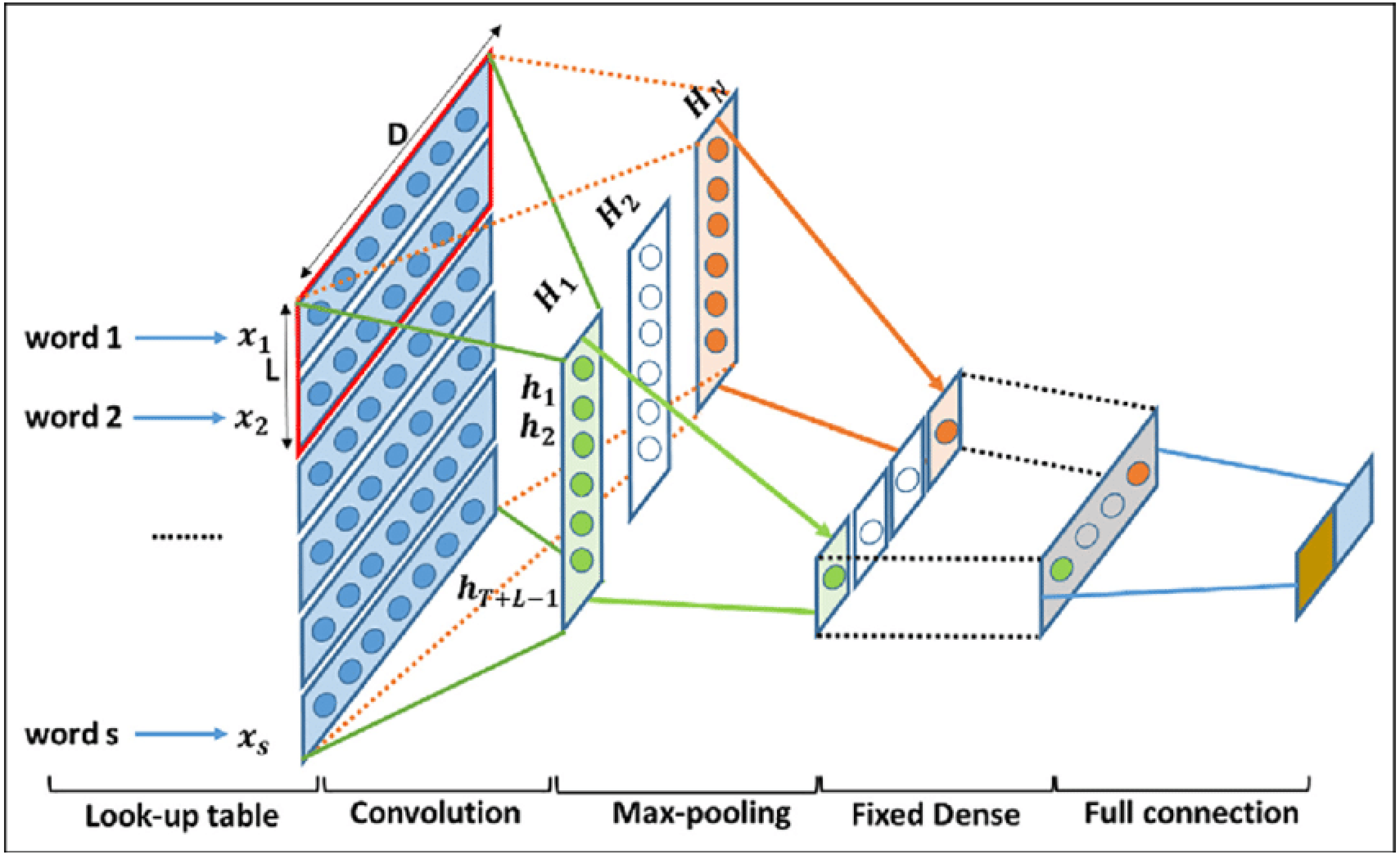
graph credit Matt Zeiler, Clarifai

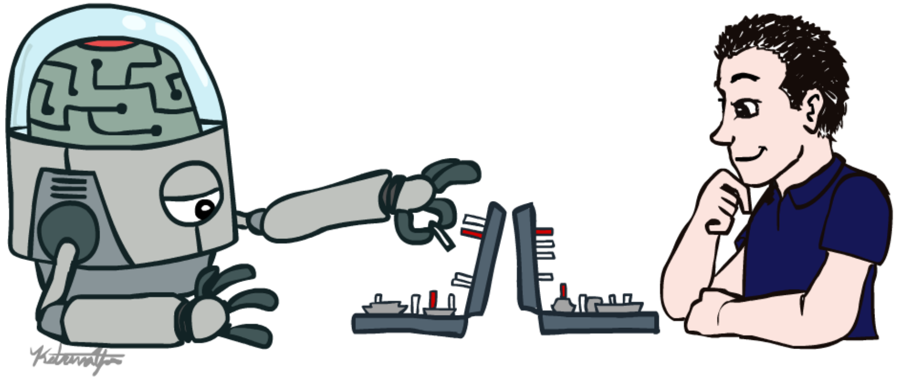
Performance



graph credit Matt Zeiler, Clarifai

CNN in Text Analysis





Thanks!