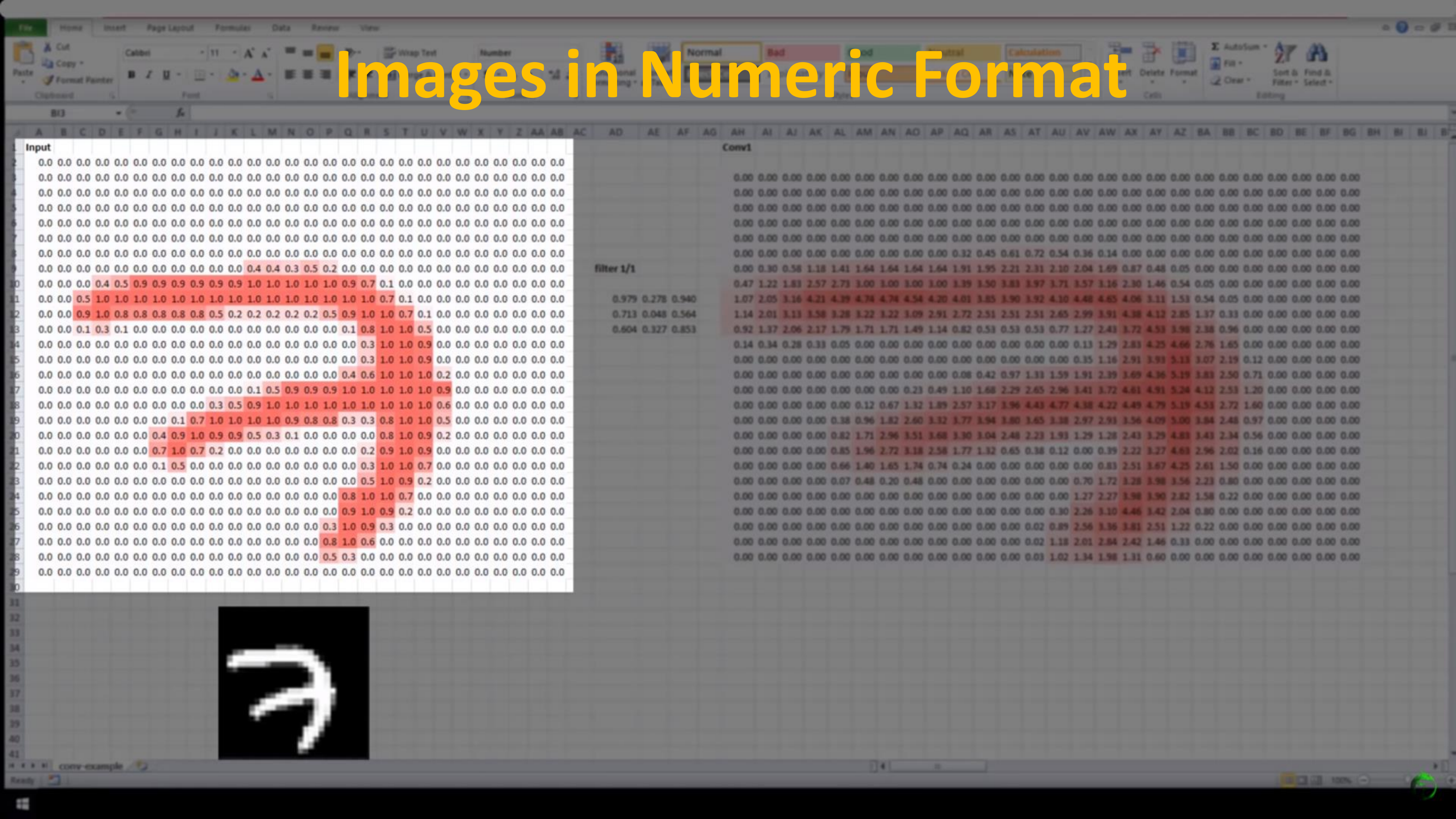
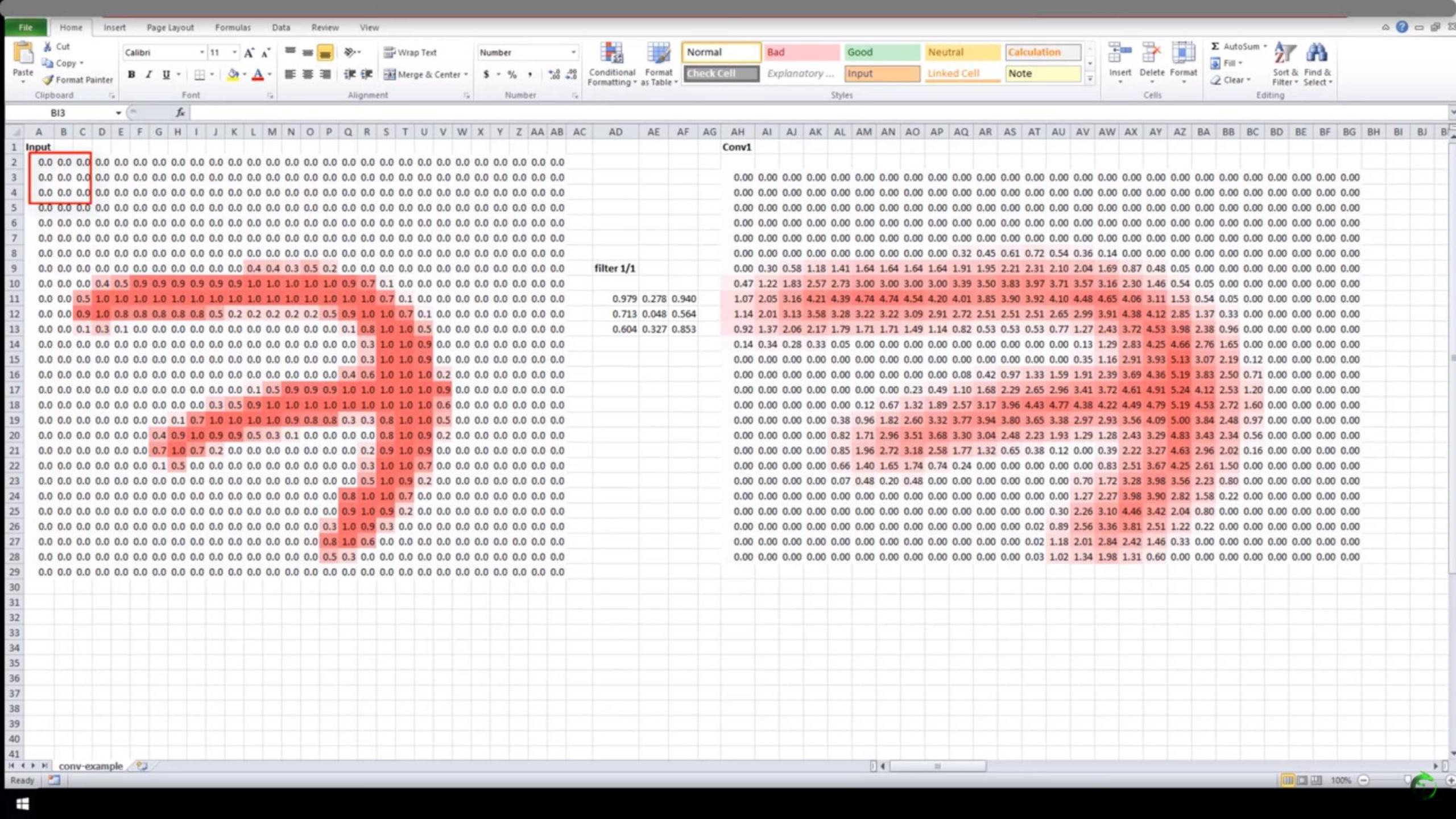
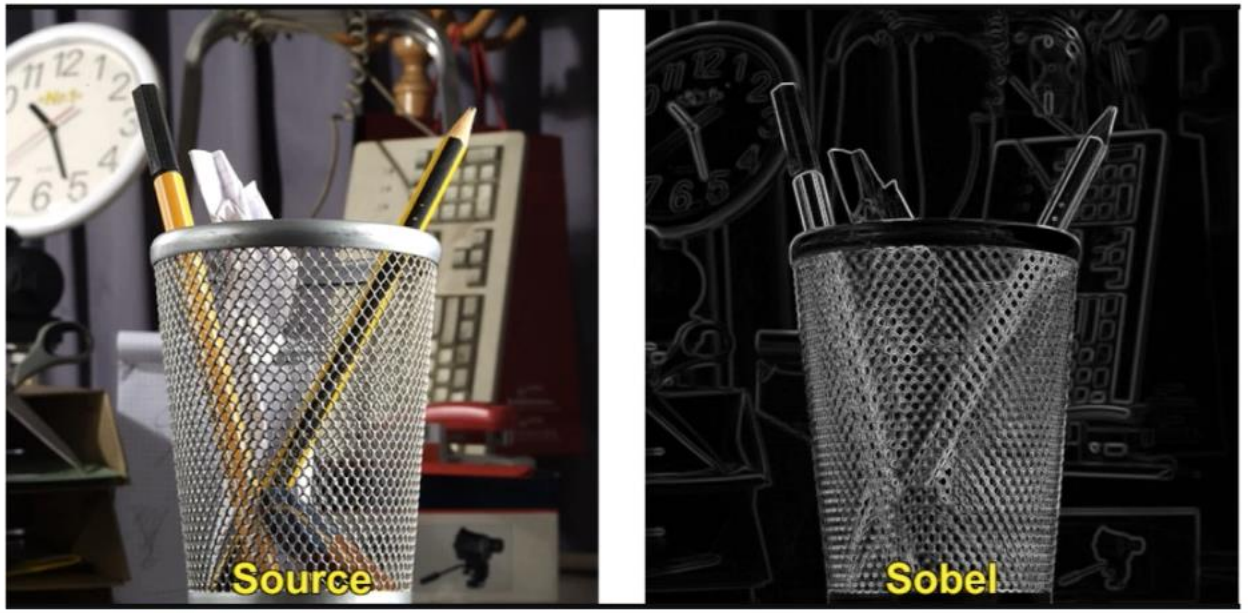
Images in Numeric Format




File
Home
Insert
Page Layout
Formulas
Data
Review
View
Cut
Copy
Format Painter
Paste
Calibri
11
Wrap Text
Merge & Center
Number
Normal
Bad
Good
Neutral
Calculation
Check Cell
Explanatory ...
Input
Linked Cell
Note
Conditional Formatting
Format as Table
Insert
Delete
Format
AutoSum
Fill
Clear
Sort & Filter
Find & Select
Clipboard
Font
Alignment
Number
Styles
Cells
Editing
BI3
Input
Conv1
filter 1/1
0.979 0.278 0.940
0.713 0.048 0.564
0.604 0.327 0.853
conv-example
Ready
100%

What is a CNN?

- Convolutional neural network - a neural network containing convolutional layers
- Normally, convolution in Deep Learning is used for images
- An example of convolution is edge detection



Deep Learning on large images

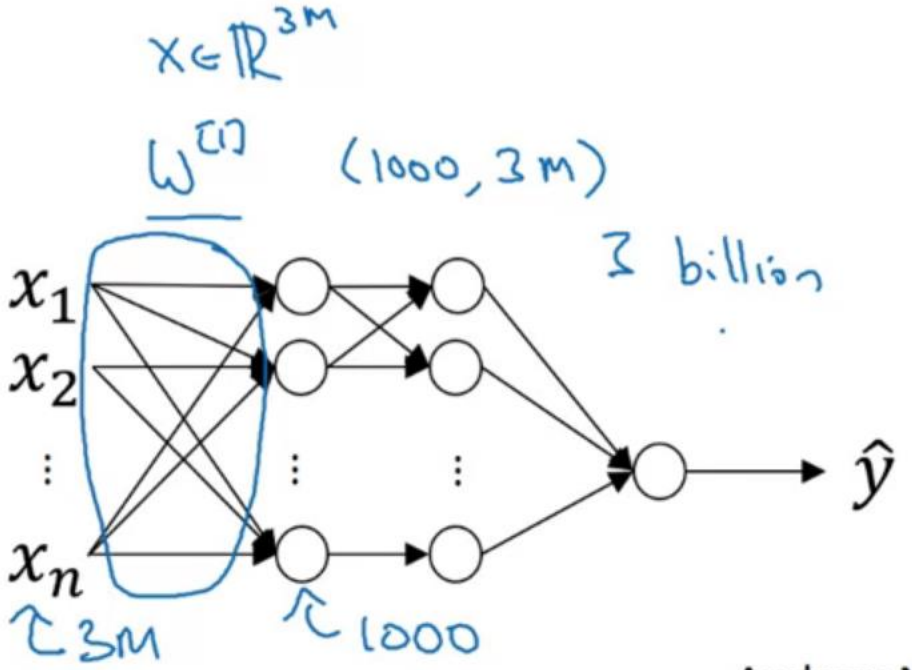
Why CNN not regular NN?

64x64 × 3

→ Cat? (0/1)

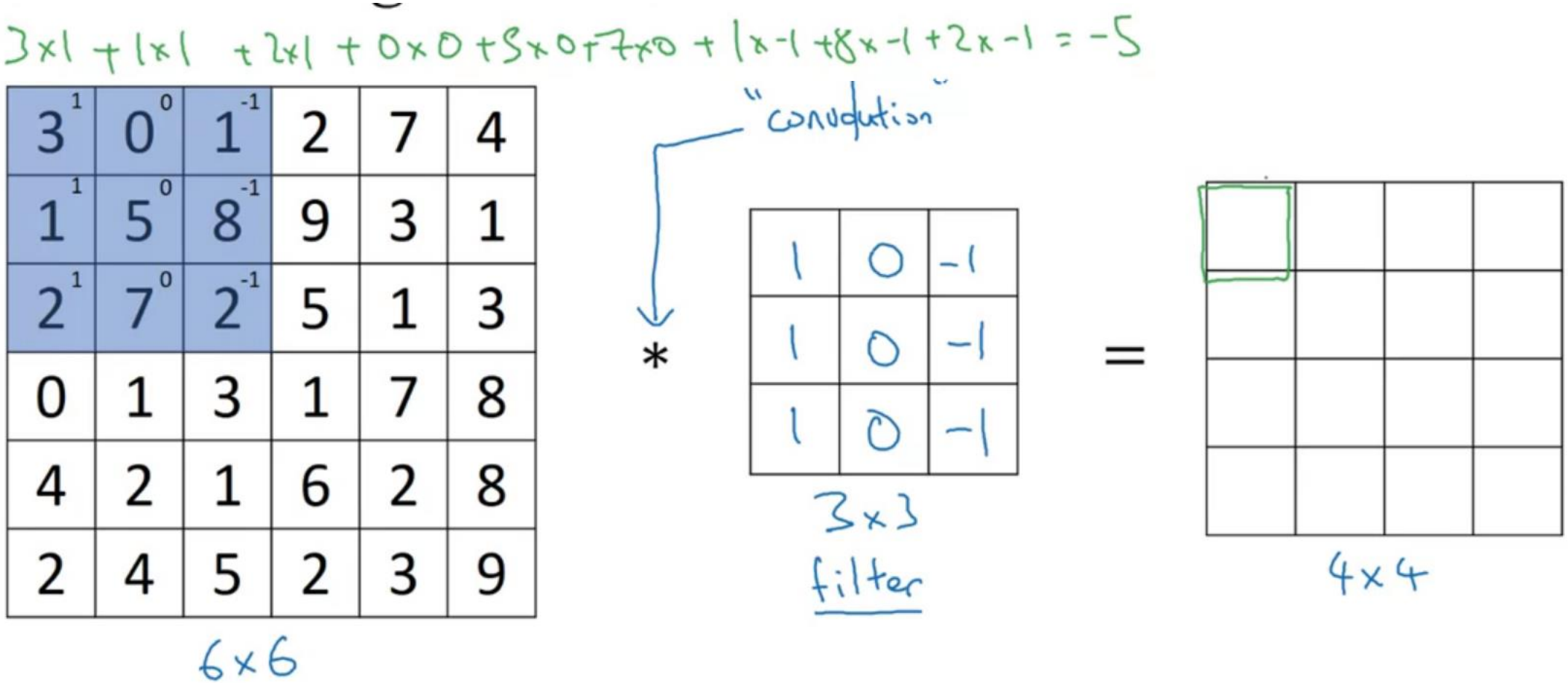
12288

1000 × 1000 × 3
= 3 million

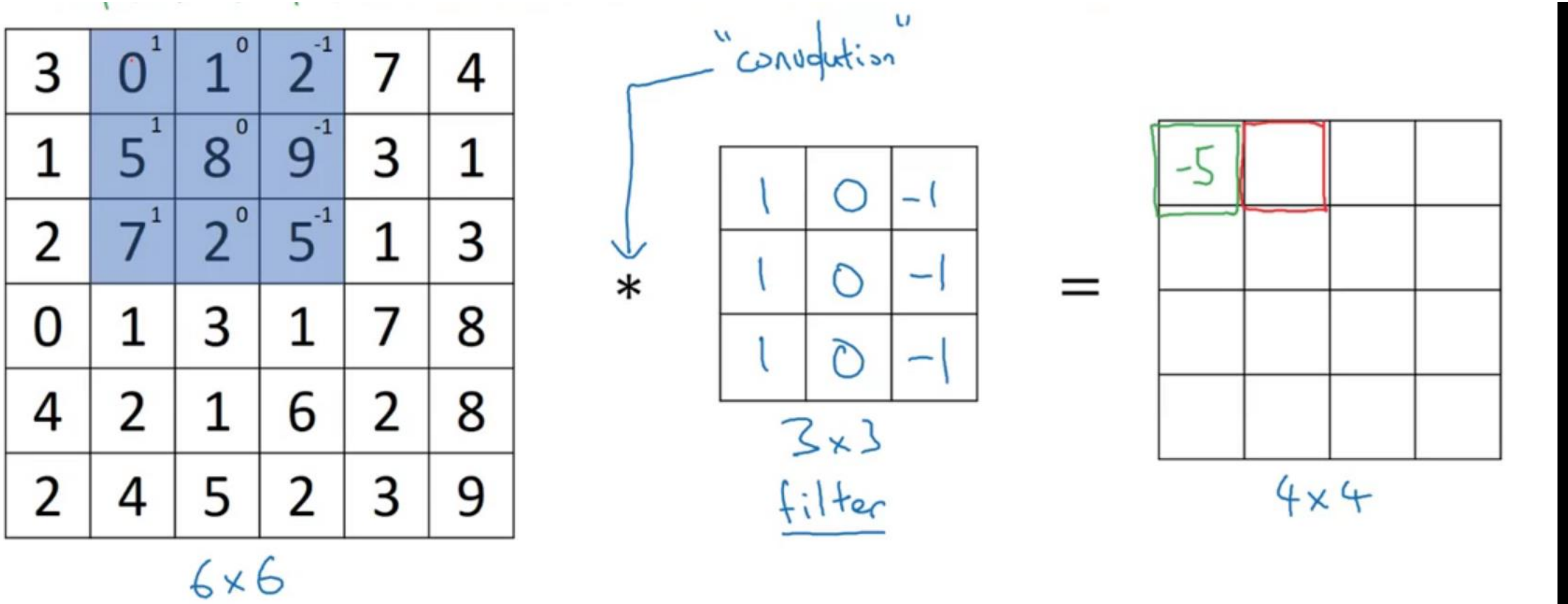


Andrew Ng

Convolution Architect: Components 1

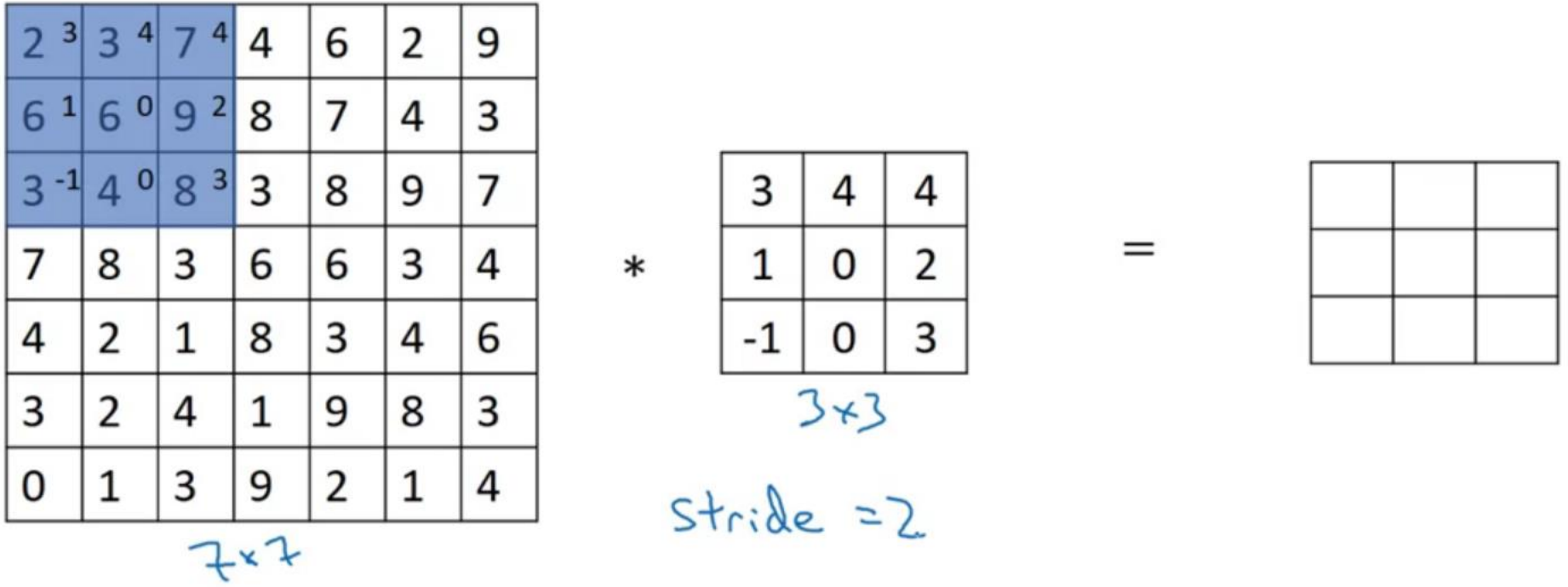


Convolution: Elementwise Multiplication



Andrew Ng

Strided convolution



Strided convolution

2	3	7 ^{3}	4 ^{4}	6 ^{4}	2	9
6	6	9 ^{1}	8 ^{0}	7 ^{2}	4	3
3	4	8 $^{-1}$	3 ^{0}	8 ^{3}	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

7×7

*

3	4	4
1	0	2
-1	0	3

3×3

Stride = 2

=

91	100	

Summary of convolutions

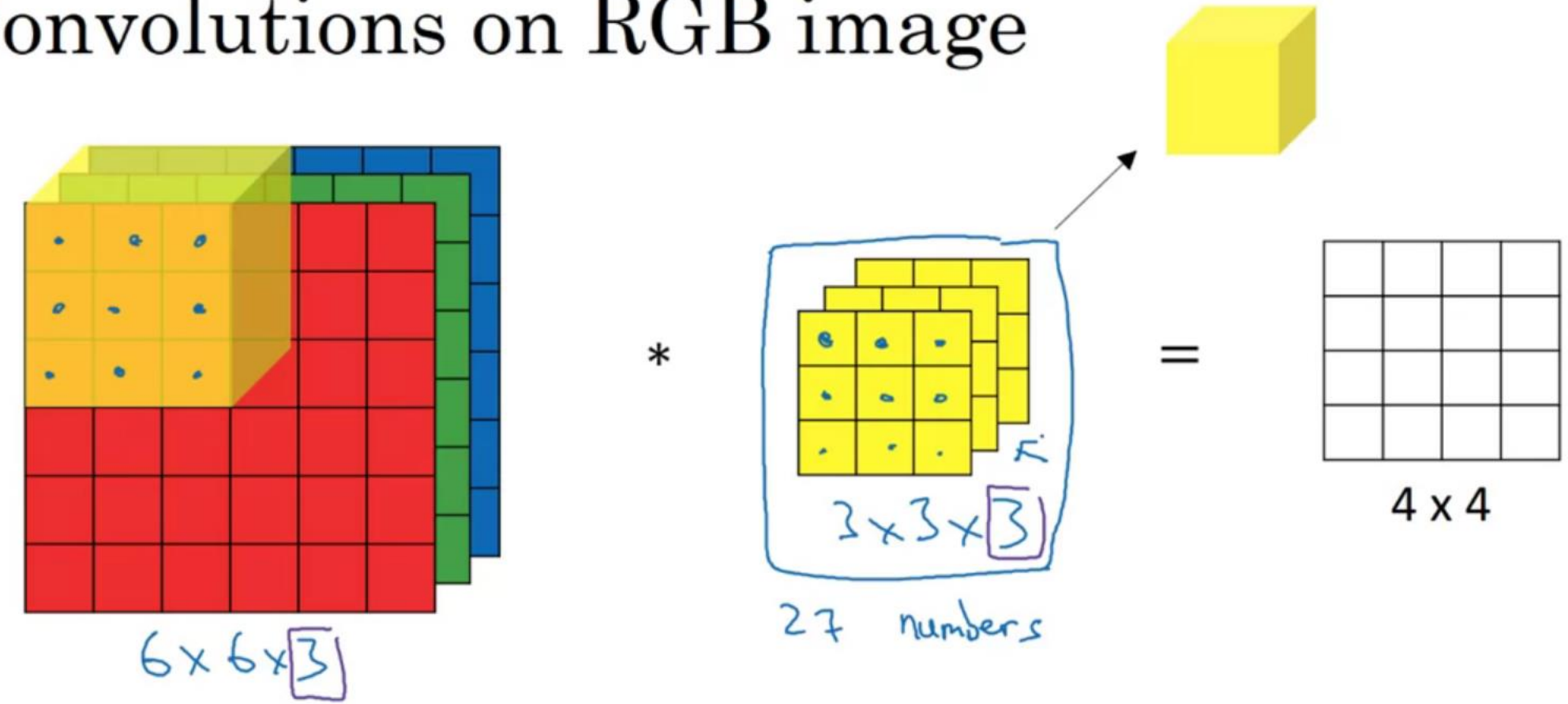
$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Andrew Ng

Convolutions on RGB image



Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1,1,1,n_c^{[l]})$ ← #filters in layer l.

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$ ←

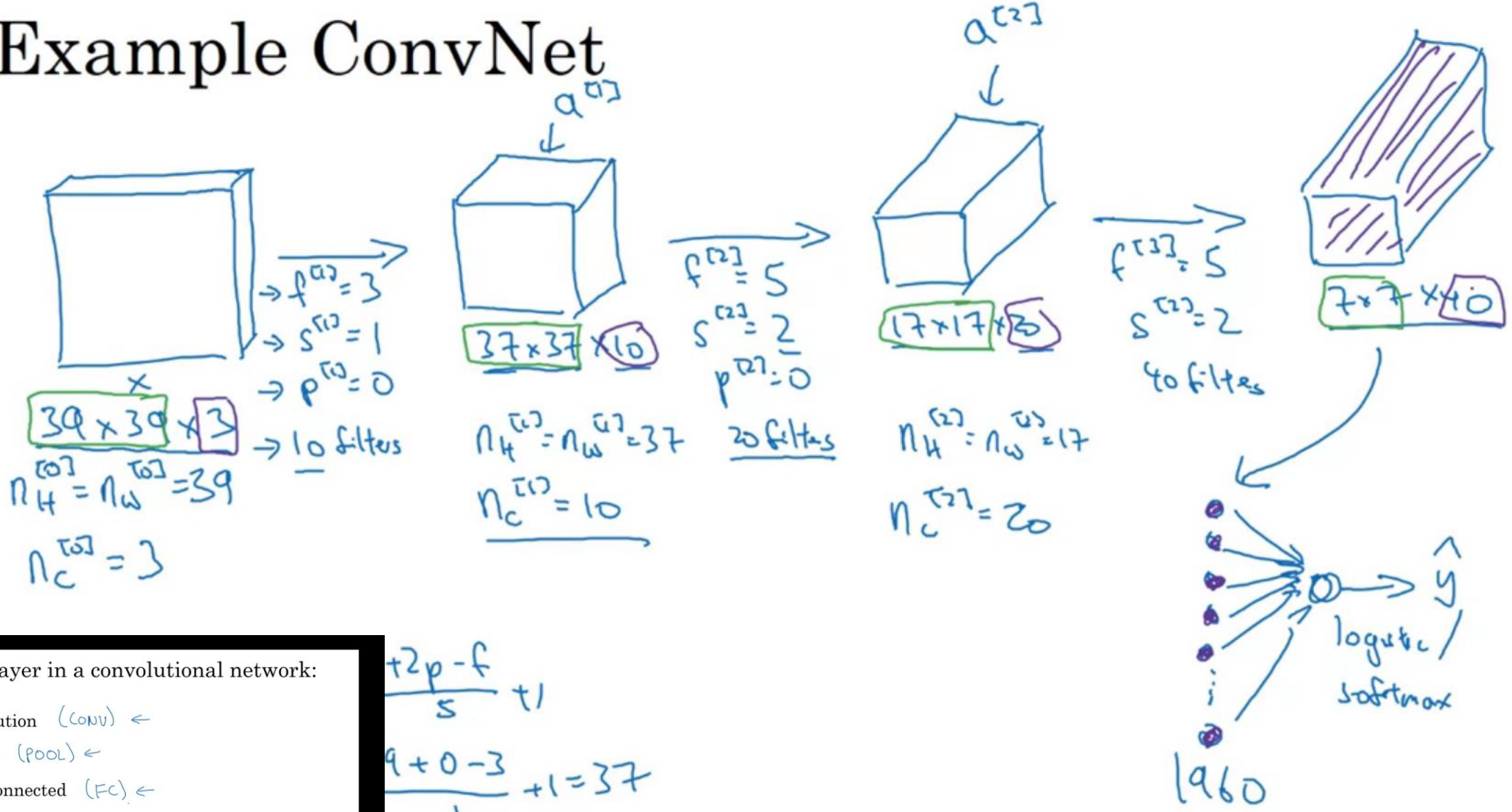
Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ ←

$$n_{H/W}^{[l]} = \left\lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Andrew Ng

Example ConvNet



$\frac{n+2p-f}{s}+1$

$\frac{9+0-3}{1}+1=37$

Types of layer in a convolutional network:

- Convolution (CONV) ←
- Pooling (POOL) ←
- Fully connected (FC) ←

Andrew Ng

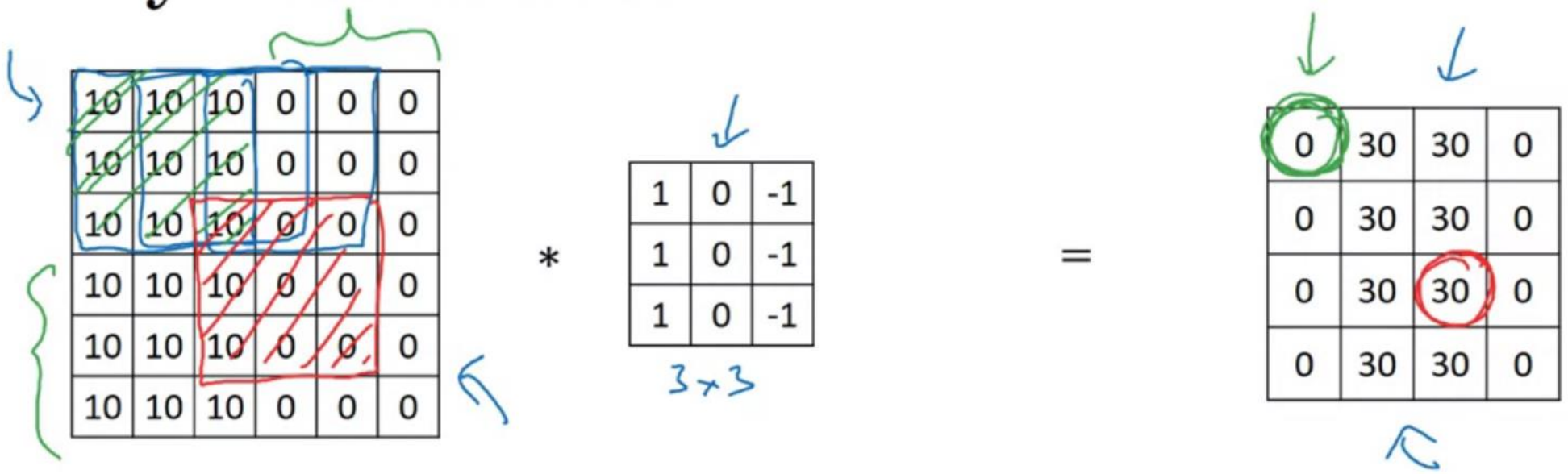
Andrew Ng

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

Andrew Ng

Why convolutions

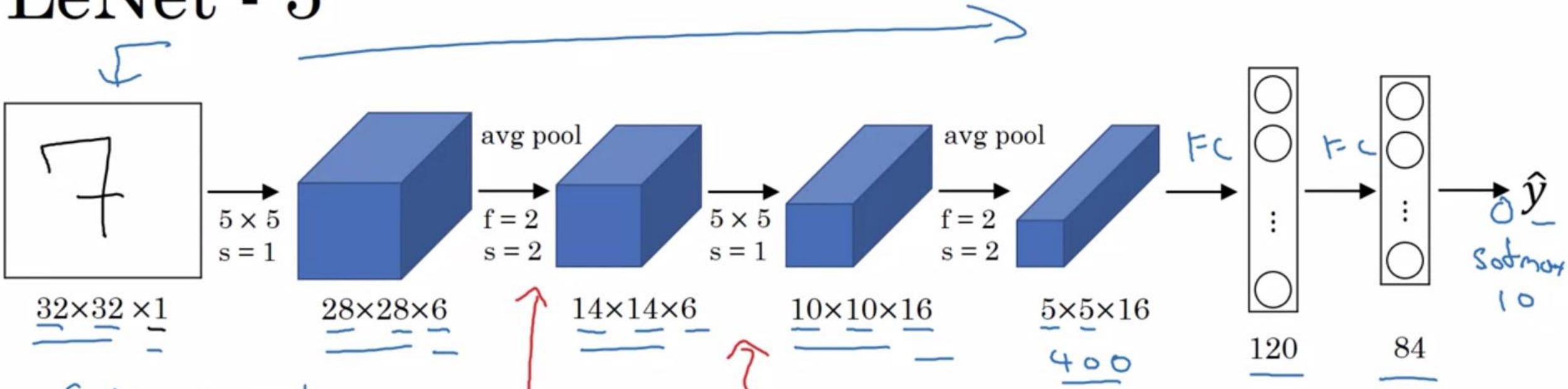


Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

Andrew Ng

LeNet - 5



60k parameters.

$n_H, n_w \downarrow$ $n_c \uparrow$

Non-linearity after pooling

$n_H \times n_w \times n_c$ $f \times f \times n_c$

conv pool conv pool fc fc output

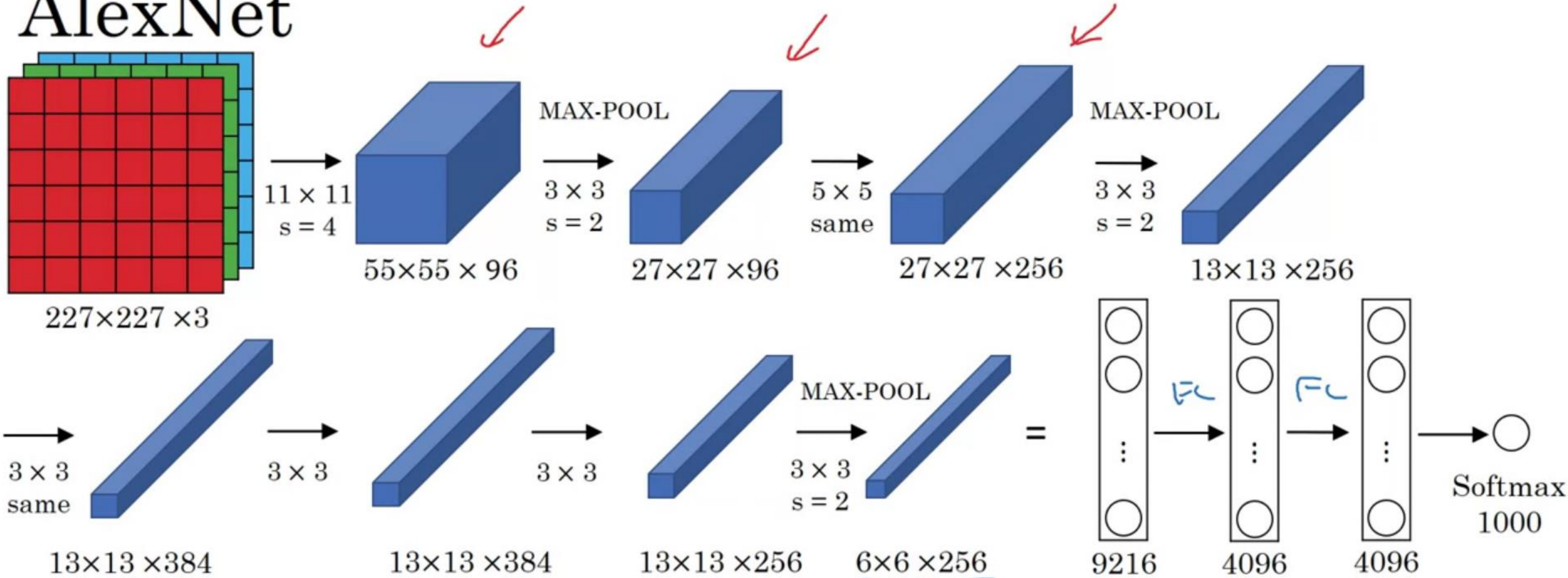
Activation: Sigmoid/tanh ReLU

II III

[LeCun et al., 1998. Gradient-based learning applied to document recognition]

Andrew Ng

AlexNet



- Similar to LeNet, but much bigger.
- ReLU
- Multiple GPUs.
- Local Response Normalization.

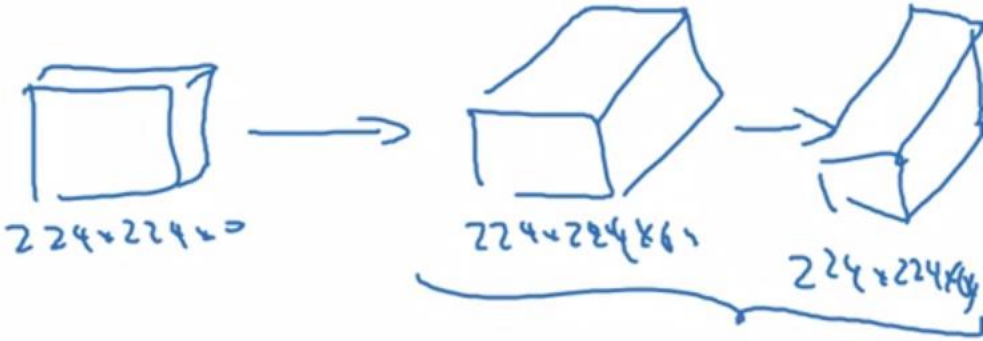
60M parameters

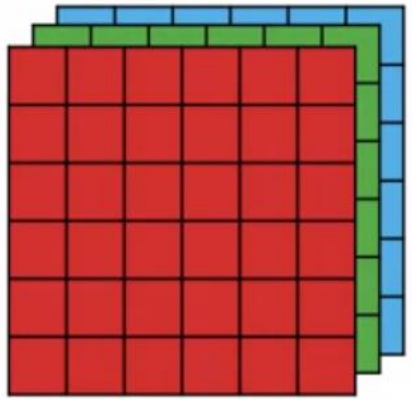
[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

VGG - 16

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2





224×224 ×3

→ 224×224×64 (×2 [CONV 64]) → POOL → 112×112 ×64 → 112×112 ×128 (×2 [CONV 128]) → POOL → 56×56 ×128

→ 56×56 ×256 ([CONV 256] ×3) → POOL → 28×28 ×256 → 28×28 ×512 ([CONV 512] × 3) → POOL → 14×14×512

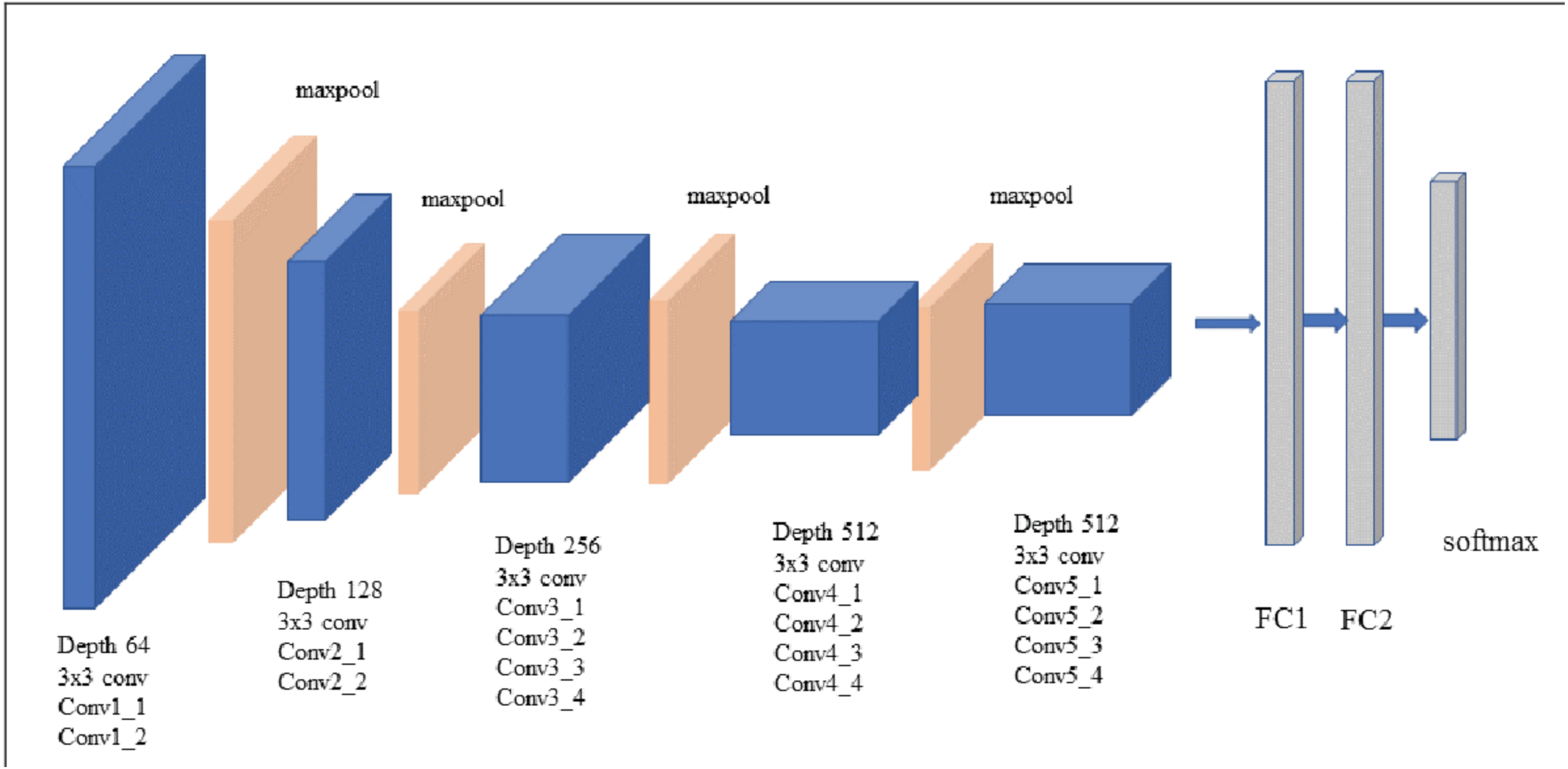
→ 14×14 ×512 ([CONV 512] ×3) → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

~138M

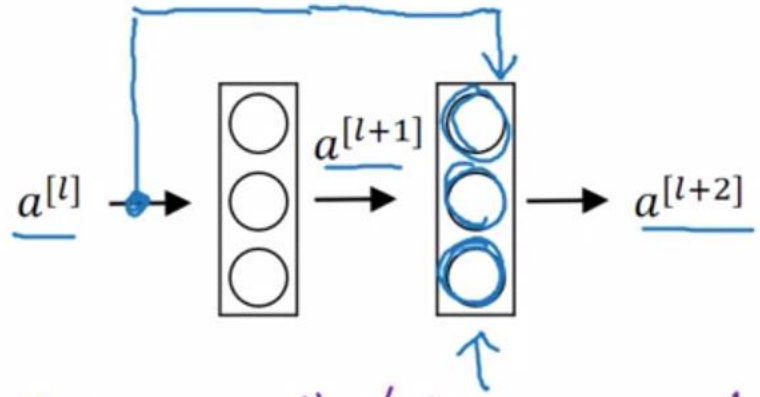
[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Andrew Ng

VGG19


maxpool
maxpool
maxpool
maxpool
Depth 64
3x3 conv
Conv1_1
Conv1_2
Depth 128
3x3 conv
Conv2_1
Conv2_2
Depth 256
3x3 conv
Conv3_1
Conv3_2
Conv3_3
Conv3_4
Depth 512
3x3 conv
Conv4_1
Conv4_2
Conv4_3
Conv4_4
Depth 512
3x3 conv
Conv5_1
Conv5_2
Conv5_3
Conv5_4
FC1
FC2
softmax

Residual block



$a^{[l]}$ → Linear → ReLU → $a^{[l+1]}$ → Linear → (+) → ReLU → $a^{[l+2]}$

"main path"

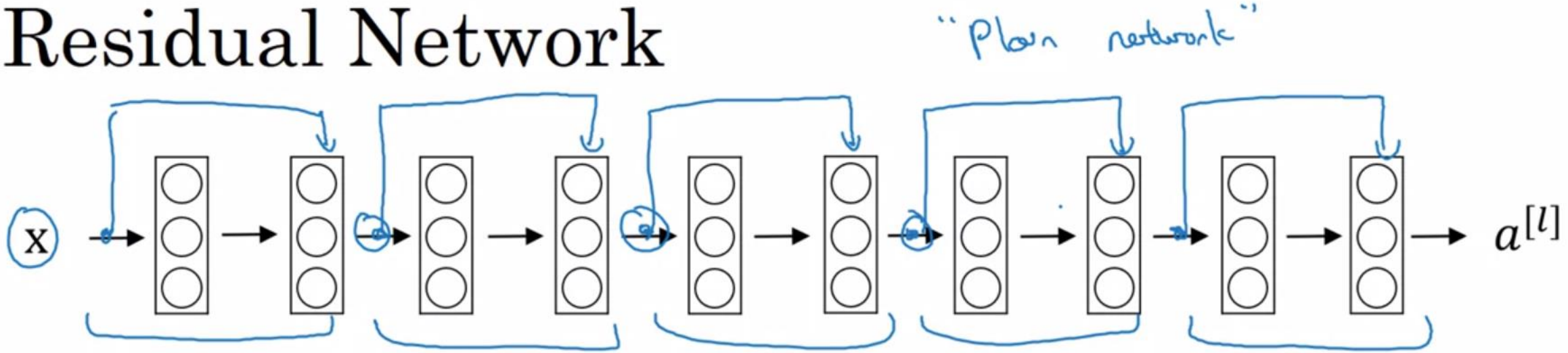
$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad \cancel{a^{[l+2]} = g(z^{[l+2]})}$$

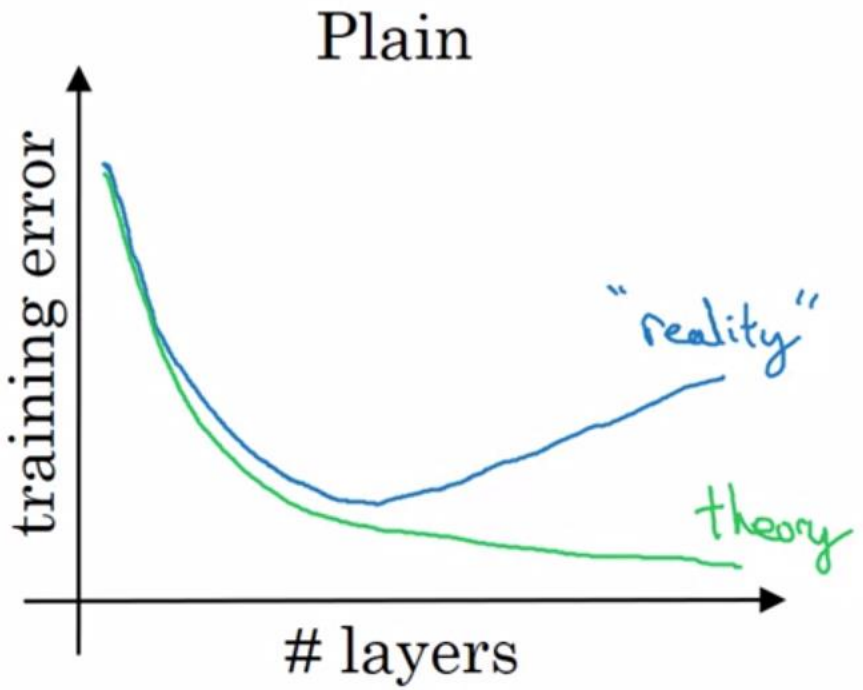
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

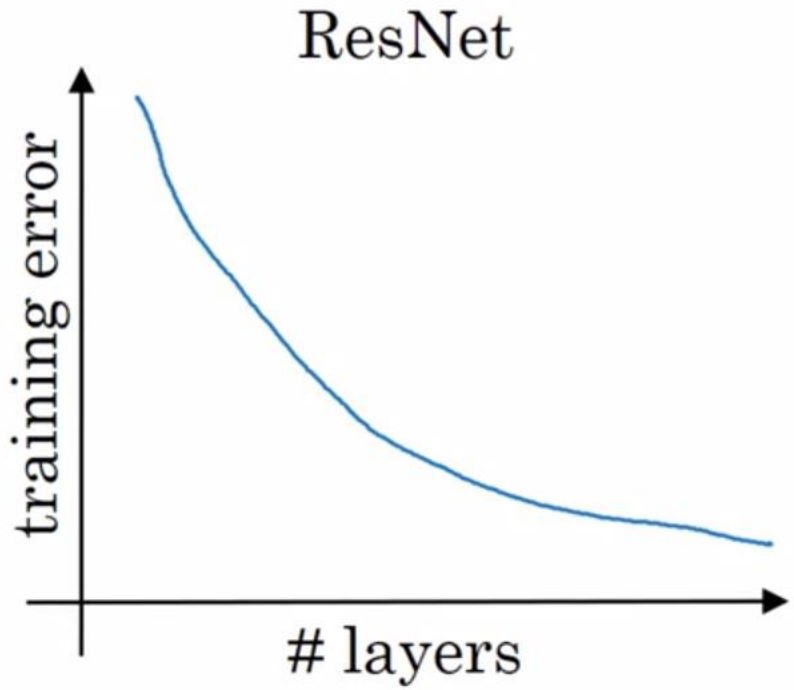
[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

Residual Network







[He et al., 2015. Deep residual networks for image recognition]

Appendix

```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

Shear

Zoom