Lecture 15
Convolutional Neural Networks(CNN)

GEOL 4397: Data analytics and machine learning for geoscientists

Jiajia Sun, Ph.D.

April. 18th, 2019

UNIVERSITY of HOUSTON

YOU ARE THE PRIDE

EARTH AND ATMOSPHERIC SCIENCES

Course evaluation

Week	Date	Topics	Comments
1	01/15 Tues	Overview of syllabus Lecture: Introduction to Machine learning: applications	
	01/17 Thur	Lecture: Review of linear algebra	
2	01/22 Tues	Lab: Linear algebra in Python	Not graded
	01/24 Thur	Lecture: Introduction to optimization	
3	01/29 Tues	Lab: Gradient descent + Linear regression	Report due on 02/05 at 5:30 pm
	01/31 Thur	Lecture: Introduction to machine learning: concepts	
4	02/05 Tues	Lecture: Logistic regression	
	02/07 Thur	Lab: Logistic regression	Report due on 02/14 at 5:30 pm
5	02/12 Tues	Lecture: Support vector machine	
	02/14 Thur	Lab: Support vector machine	Report due on 02/21 at 5:30 pm
6	02/19 Tues	Lecture: Decision trees	
	02/21 Thur	Lab: Decision trees	Report due on 02/28 at 5:30 pm
7	02/26 Tues	Lecture: Random Forest	
	02/28 Thur	Lab: Random forest	Report due on 03/07 at 5:30 pm
8	03/05 Tues	Lecture: Ensemble learning	
	03/07 Thur	Lab: Ensemble learning	Reprot due on 03/19 at 5:30 pm
9	03/12 Tues	No class due to spring break	
	03/14 Thur	No class due to spring break	
10	03/19 Tues	Review & Recap	
	03/21 Thur	Exam	
11	03/26 Tues	Lecture: Clustering	
	03/28 Thur	Lab: Clustering	Report due on 04/04 at 5:30 pm
12	04/02 Tues	Lecture: Introduction to TensorFlow	
	04/04 Thur	Lab: TensorFlow	Not graded
13	04/09 Tues	Lecture: Introduction to neural networks 1	
	04/11 Thur	Lecture: Introduction to neural networks 2	
14	04/16 Tues	Lab: Deep learning	Report due on 04/23 at 5:30pm
	04/18 Thur	Lecture: Convolutional neural networks 1	
15	04/23 Tues	$\boxed{\text{Guest lecture: Convolutional neural networks 2}}$	
	04/25 Thur	Lab: CNN (optional)	Report due on 05/02 at 5:30 pm
16	04/30 Tues	final presentation??	
	05/02 Thur	final presentation??	
Note	28 class meetings		04/29 last day of class

Learning Objectives

This course focuses on important concepts in machine learning and several widely used machine learning algorithms. This class consists of both lectures and lab exercises. After completion of the class, students can expect to

- Understand basic concepts in machine learning;
- Understand how a machine learning project is typically carried out;
- Be able to explain machine learning concepts to others who are new to machine learning;
- Be able to implement and evaluate several most widely used machine learning algorithms such as logistic regression, support vector machine, decision trees, neural network, etc;
- Be able to program in Python, and use Jupyter Notebook;
- Be able to use modules in Scikit-Learn;
- Be able to implement neural networks using TensorFlow;

Reading materials

- Hyperlinks/weblinks in Jupyter Notebooks
- Suggested reading materials

Completing Online Faculty/Course Evaluation Instructions in-Class

Faculty/Course Evaluation Information & Dates
at www.eval.uh.edu

Instructors:
1. Please conduct the evaluation in class. Allow 10 to 15 minutes of class time for the evaluation to be completed.
2. Please print and provide this instruction sheet to the proctor. The instructions for the proctor are provided below.
3. Only students who are officially enrolled in the course will be able to complete an online evaluation.
4. **Students are not to begin until instructor leaves the room**.

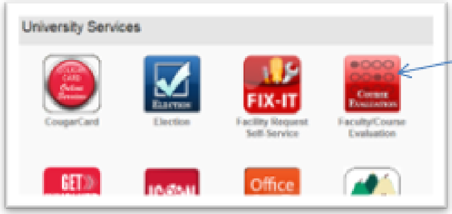
Proctor (Student Handling Teaching Evaluation):

Proctor Name: ______________________________ **Evaluation Date:** ________________

For Course ______________________________

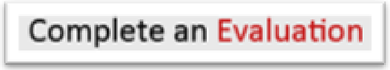
1. Have the Class use their Cell phones, Tablet or Laptop to log into AccessUH:

2. Have students select Course Evaluation:



Select Course Evaluation

3. Have students select:



4. Students will then see their class course evaluation.
Please give students time to answer questions and type comments.

If students can't see the course evaluation icon: **Under Help panel on the left on AccessUH, click to refresh icon list**

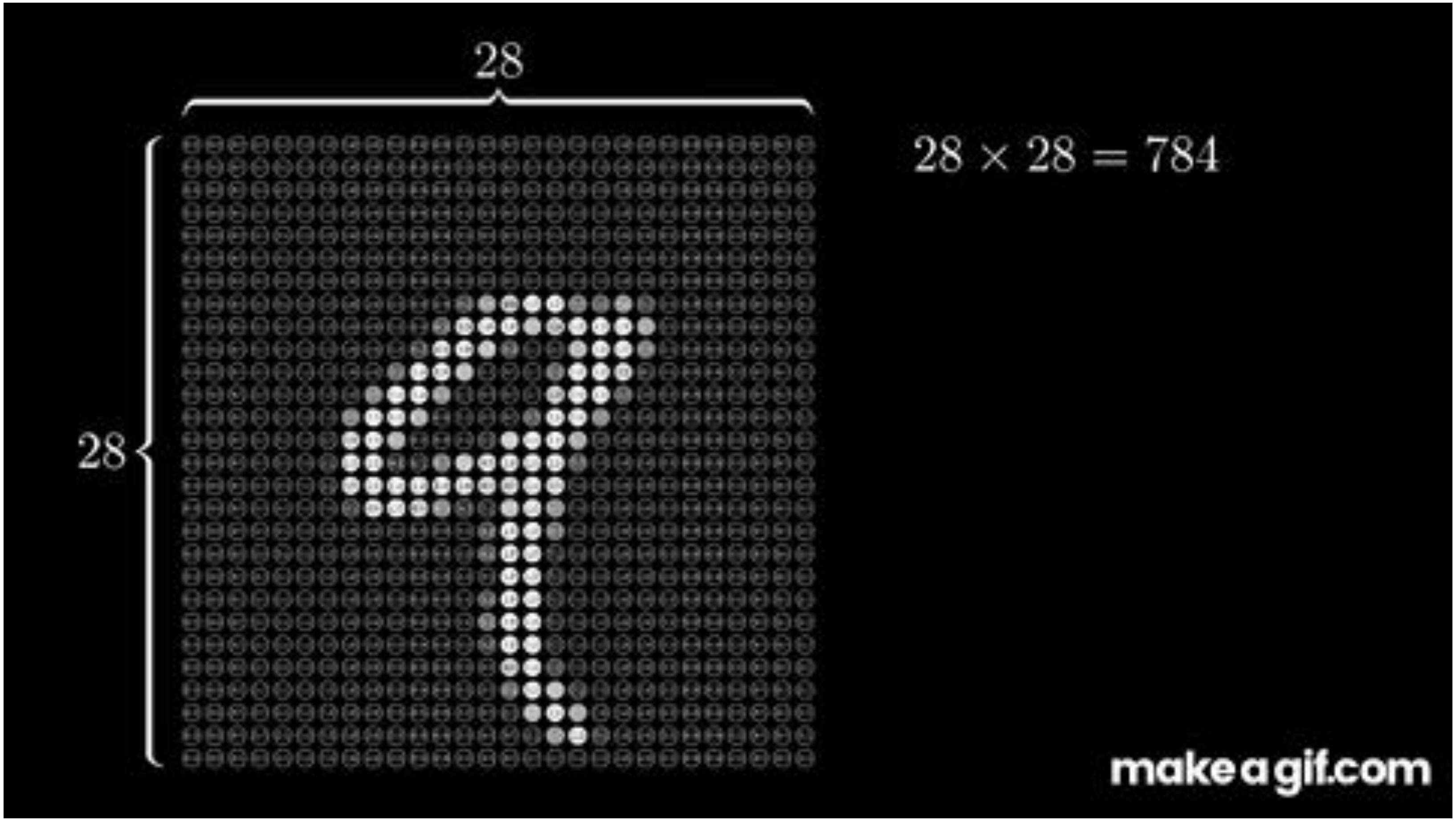
Technical issues with website during evaluation please contact: UH Measurement & Evaluation Center: (713)743-5440 or (713)743-5442 or email MEC@uh.edu.

Thank you for your assistance.

Outline

- Why CNNs?
- What is CNN?
  - Convolution
  - Stride
  - Padding
- Convolution at one layer
- Pooling
- Classis networks
- Implementation in Kera

Why CNNs?



Deep Learning on large images

- Too many parameters to train
- Risk overfitting

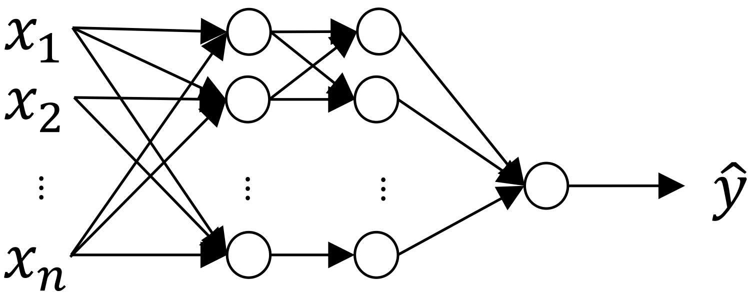
Cat? (0/1)

64x64x3
12288

1000 x 1000 x 3

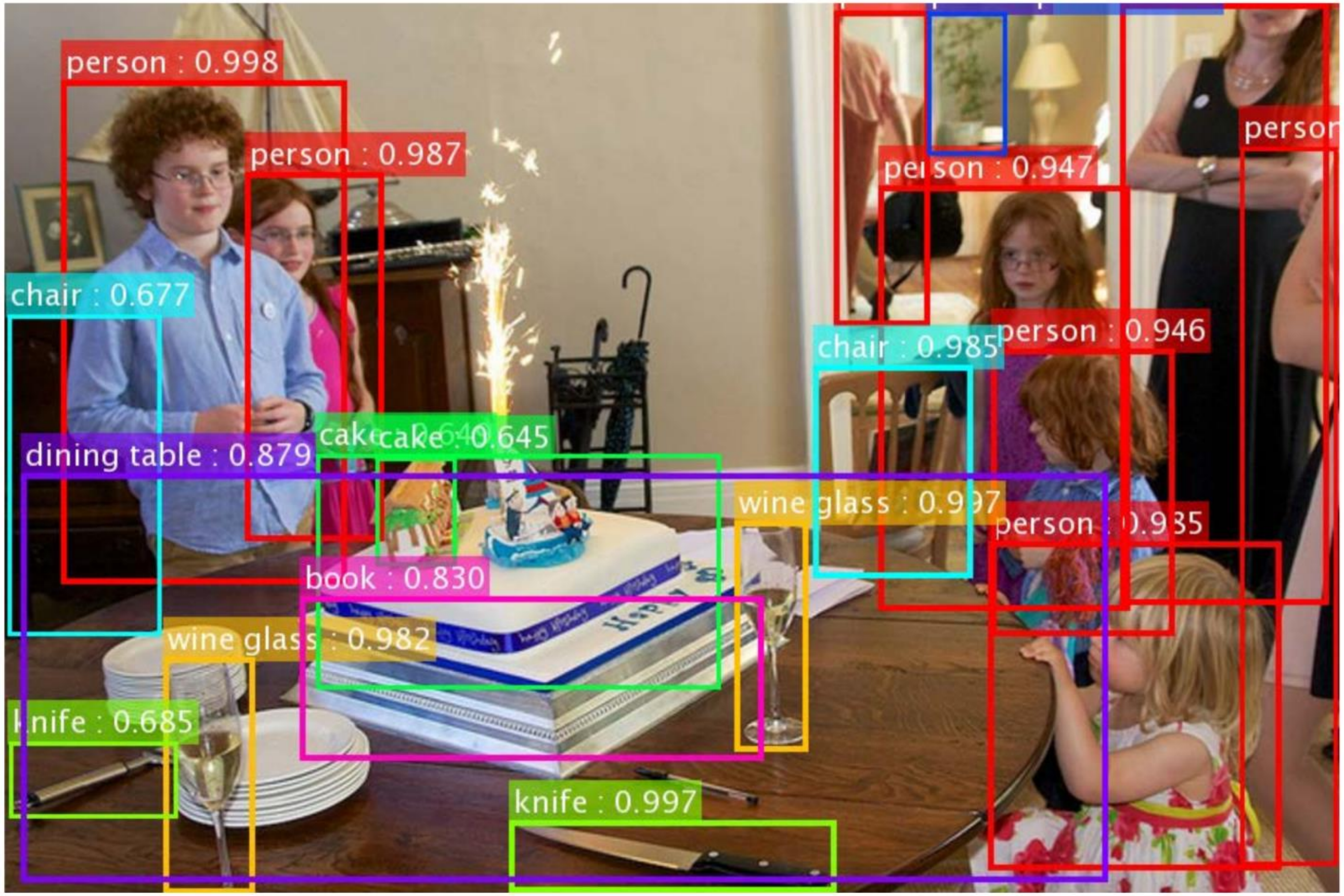
3 million

$W^{[1]}$ is (1000, 3M) = 3 billion parameters to train using FC layers



Credit: Andrew Ng

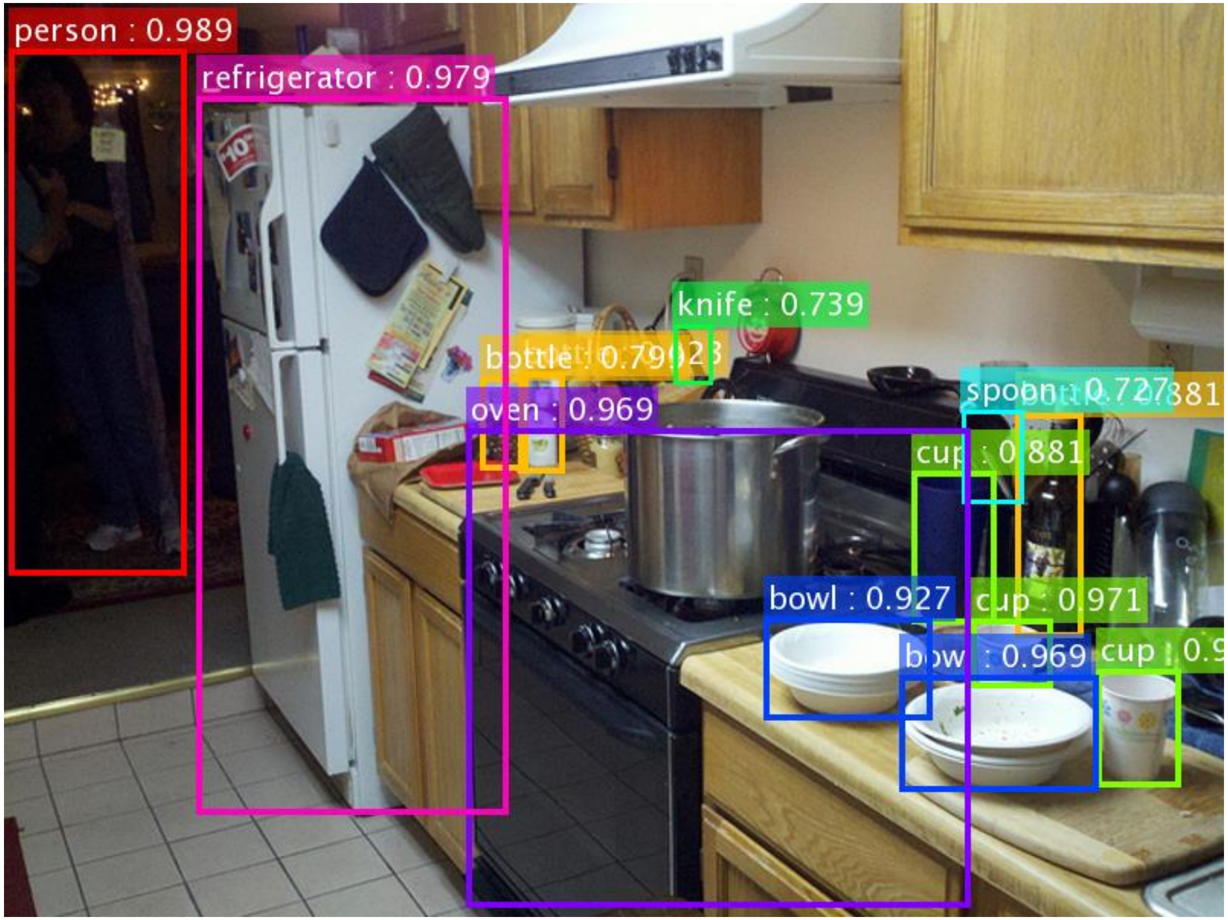
Object detection



ResNet applied to COCO dataset.

Source: He et al., Deep residual learning for image recognition, CVPR, 2016

Object detection

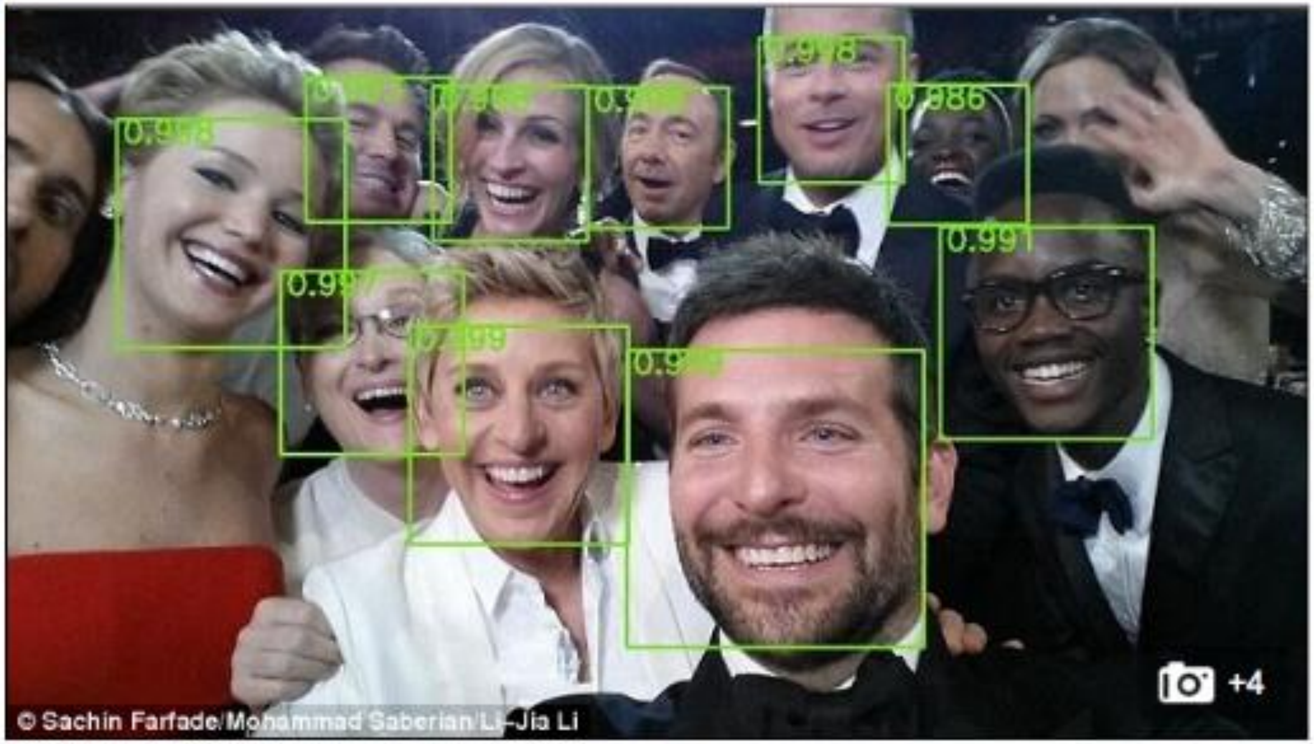


Source: He et al., Deep residual learning for image recognition, CVPR, 2016

Face detection

Convolution Neural Net (CNN)

Yahoo + Stanford example – find a face in a pic, even upside down

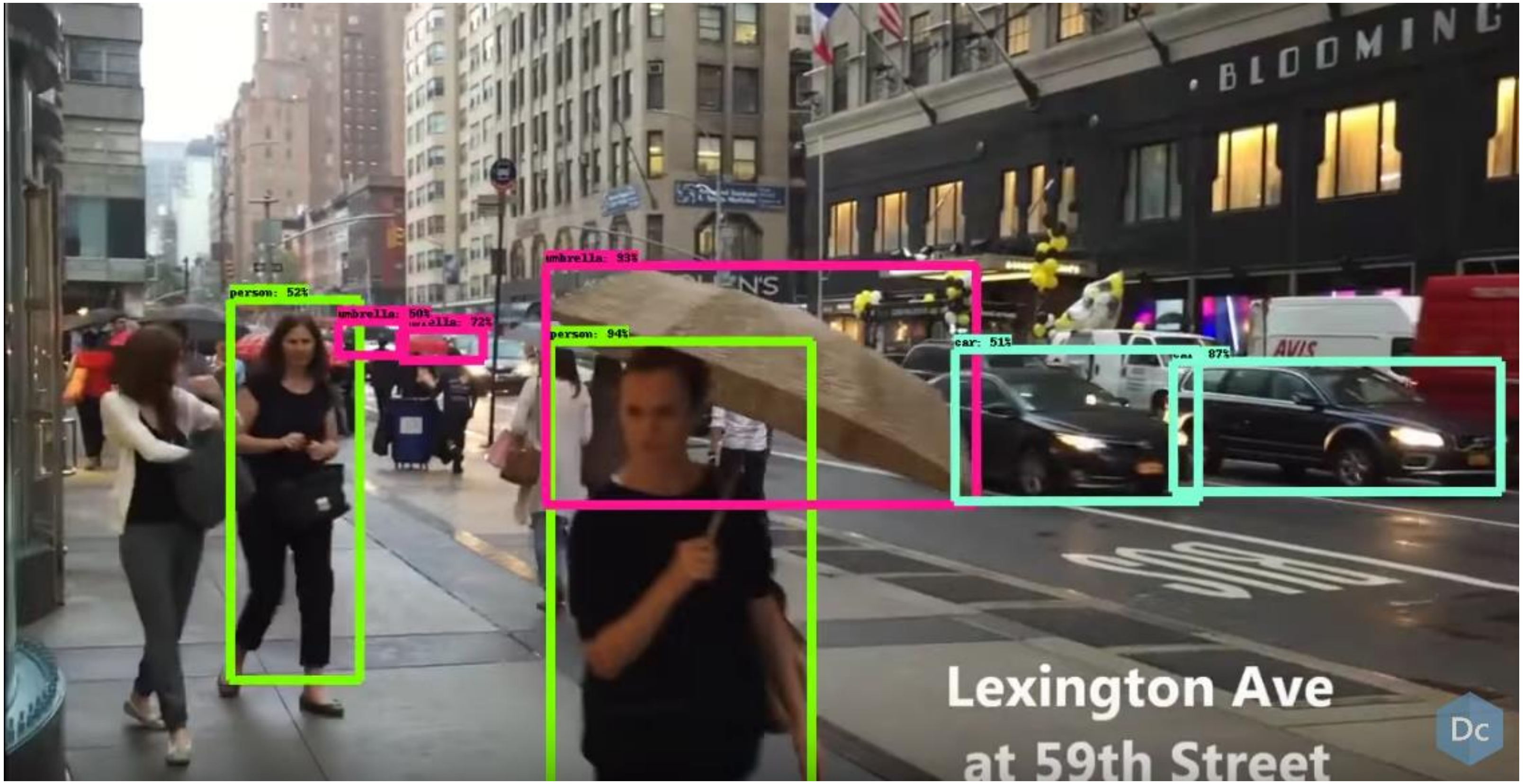


The Deep Dense Face Detector algorithm was built by Yahoo Labs in California and Stanford University. The researchers used a form of machine learning known as a deep convolutional neural network to train a computer to spot facial features (pictured) in a database of images

http://www.da

At the moment, the so-called Deep Dense Face Detector doesn't recognise who the individual faces belong to, just that there is a face.

Real time object detection

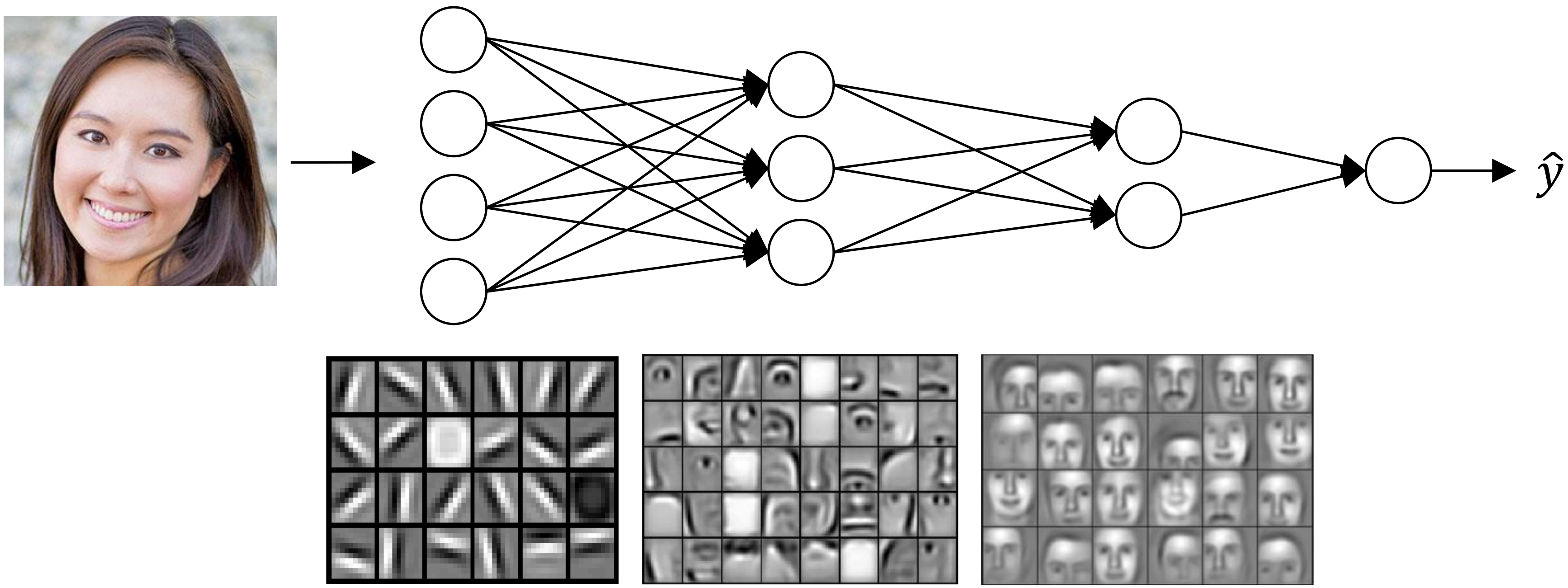


Video online: https://www.youtube.com/watch?v=_zZe27JYi8Y

Why CNNs?

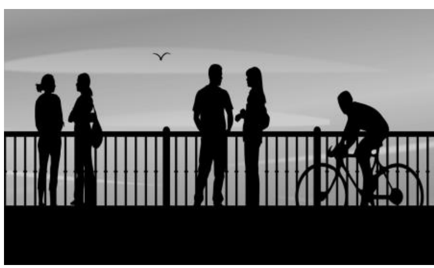
- Preserve and use contextual information in natural images
- Smaller amount of parameters to train
- Have been very successful in solving many real-world problems

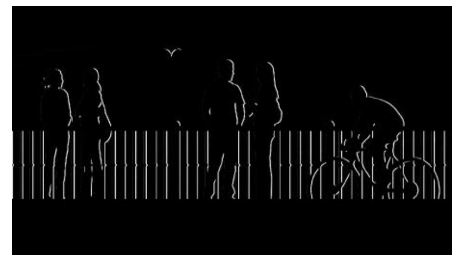
Deep learning



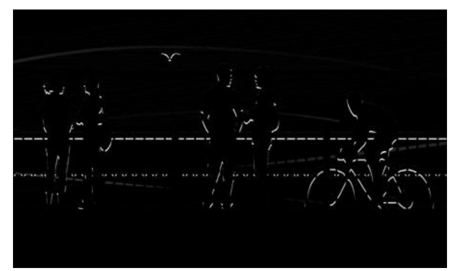
Credit: Andrew Ng

Edge detection





vertical edges



horizontal edges

Credit: Andrew Ng

Vertical edge detection via convolution

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

6 X 6

*

filter

kernel

3 X 3

=

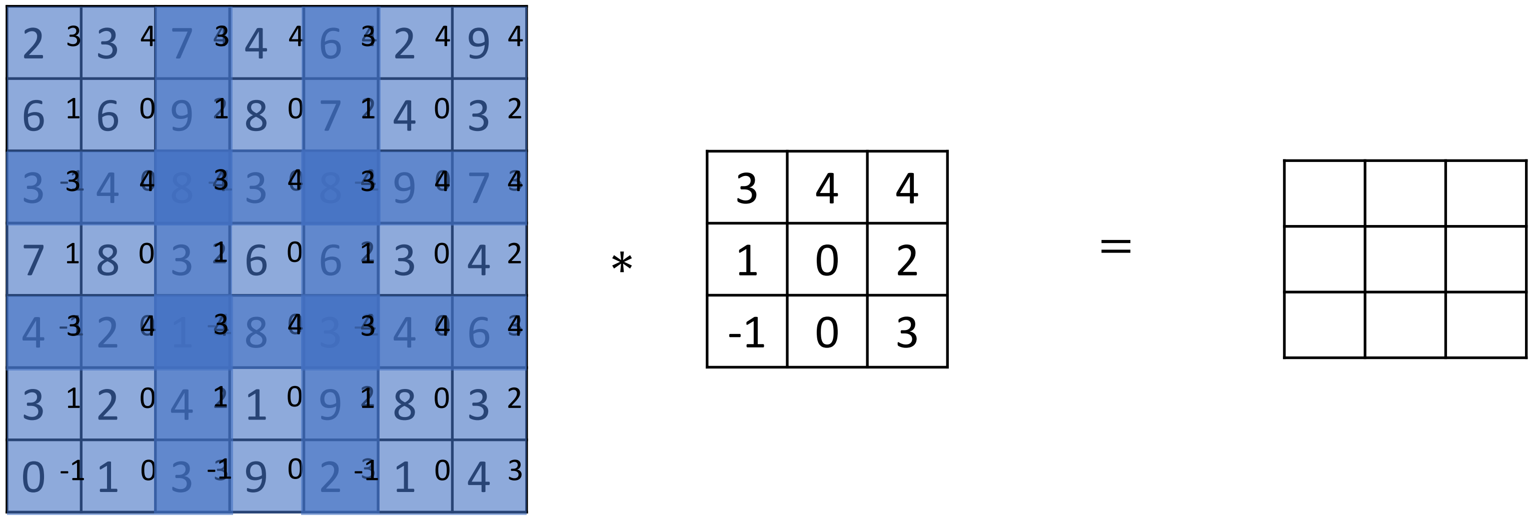
0	-2	-4	-7
-3	-2	-3	-16

4 X 4

Convolution

Credit: Andrew Ng

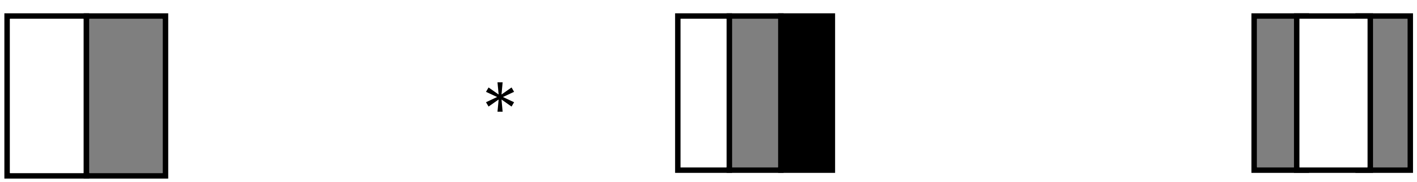
Strided convolution



Stride: the number of pixels before and after you move a filter

Vertical edge detection via convolution

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



Credit: Andrew Ng

What would a horizontal edge detection filter look like?

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

Credit: Andrew Ng

Learning to detect edges

Human defined filters

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

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
-10	0	-10
3	0	3

Schorr filter

Filters automatically learned by NN

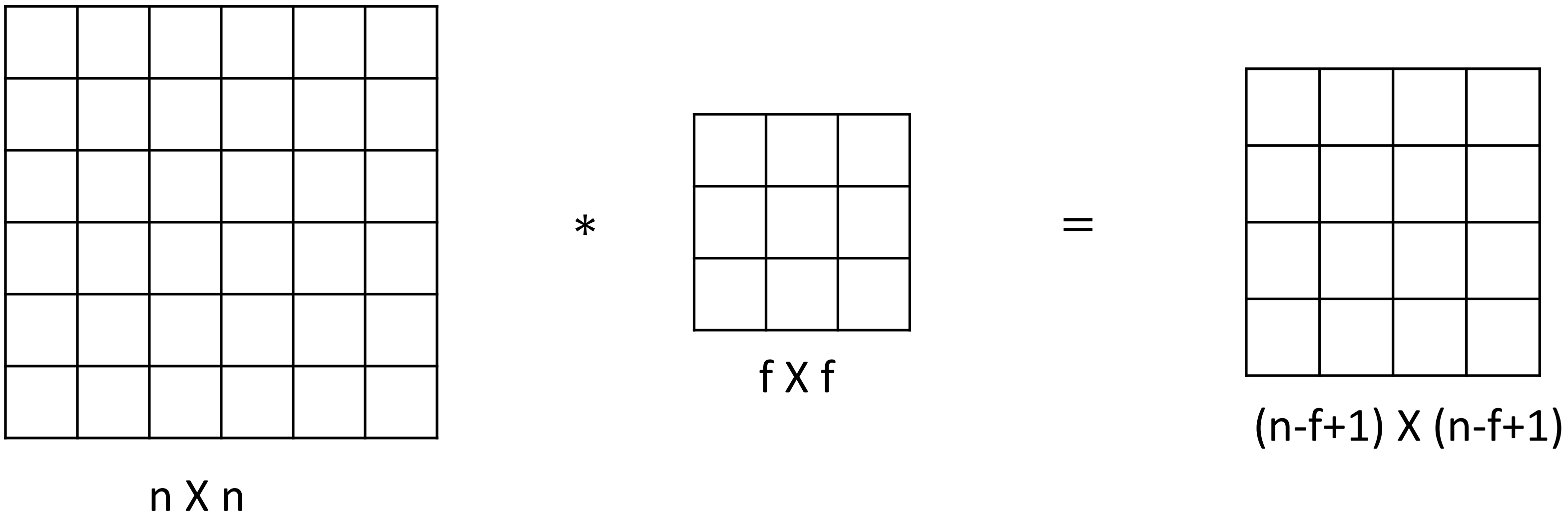
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Credit: Andrew Ng

Padding

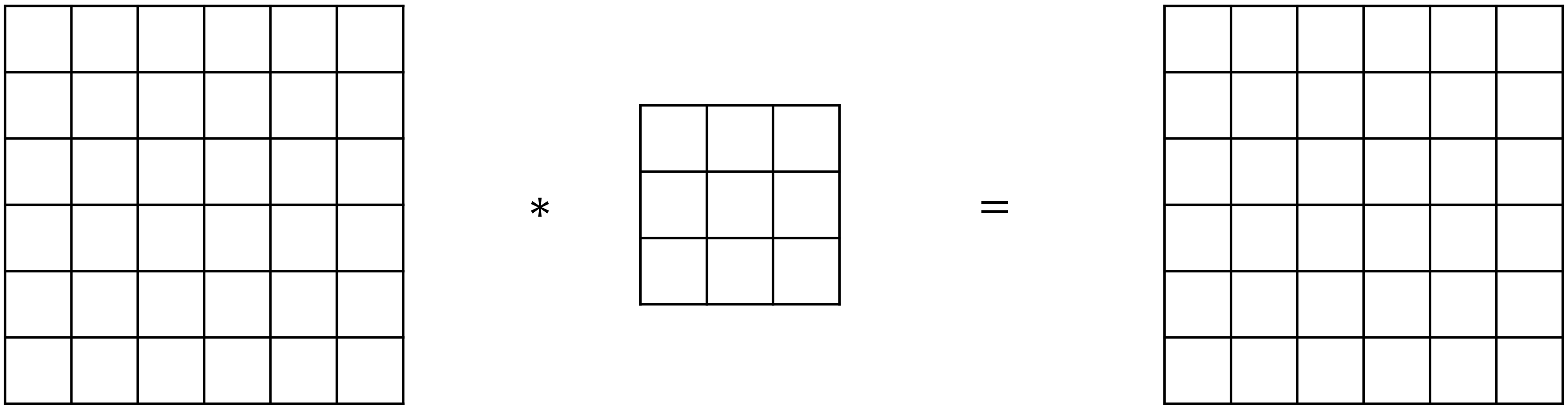
Shrinking effect



- Shrink output. For deep neural networks, quickly shrink to 1 x 1
- Throw away information from edges

Credit: Andrew Ng

Padding



Suppose p cells are padded in each direction, the output image would be (n+2p-f+1) X (n+2p-f+1), if the stride is 1

Credit: Andrew Ng

Padding strategies

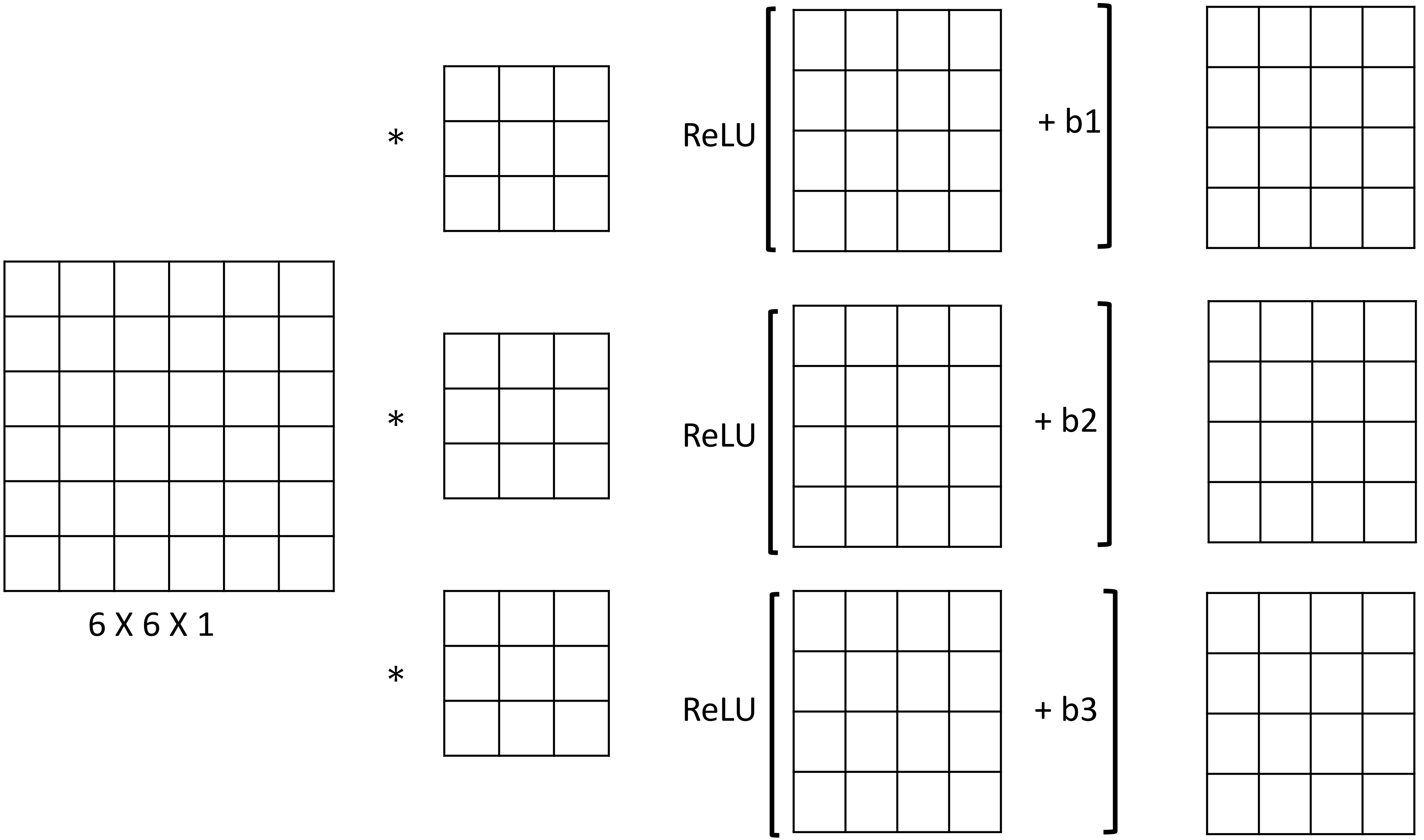
- No padding (Valid)

- Same padding: pad cells so that the output size is the same as the input size.

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

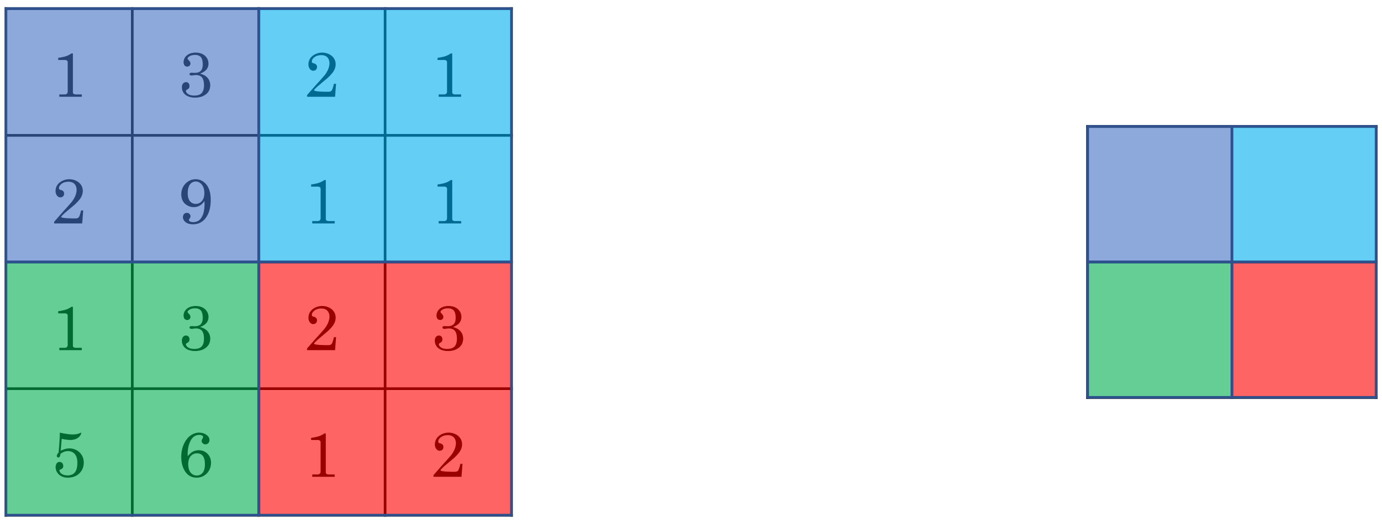
Credit: Andrew Ng

Convolution at one layer



Pooling

Pooling layer: Max pooling



No parameters to learn!

Hyperparameters: f = 2 s = 2

Credit: Andrew Ng

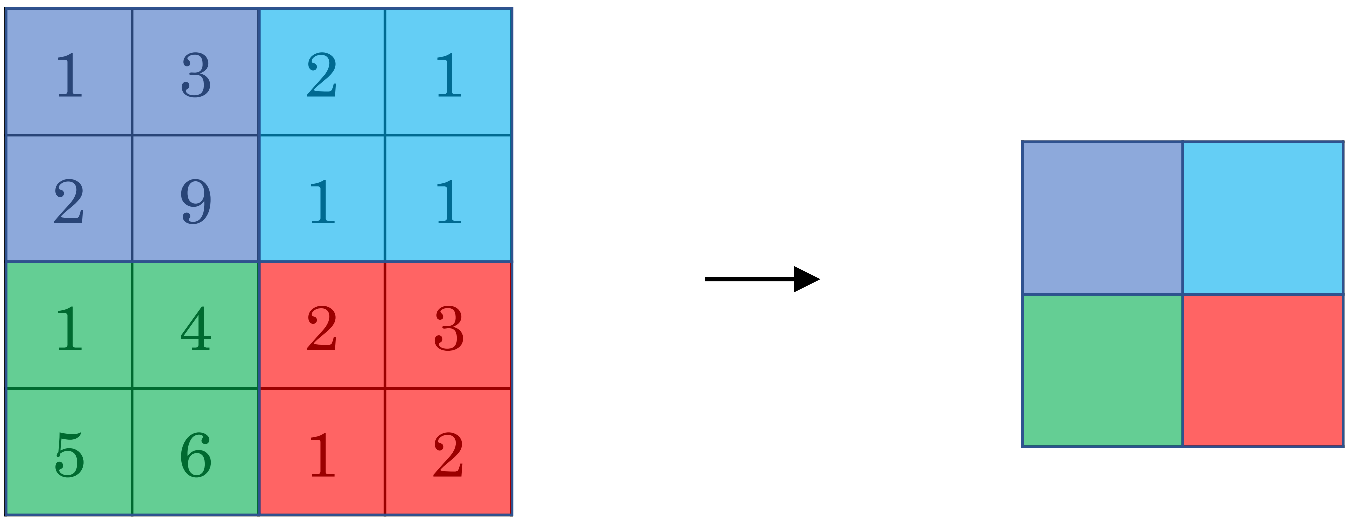
Pooling layer: Max pooling

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

Hyperparameters: $f = 3$ $s = 1$

Credit: Andrew Ng

Pooling layer: Average pooling



Credit: Andrew Ng

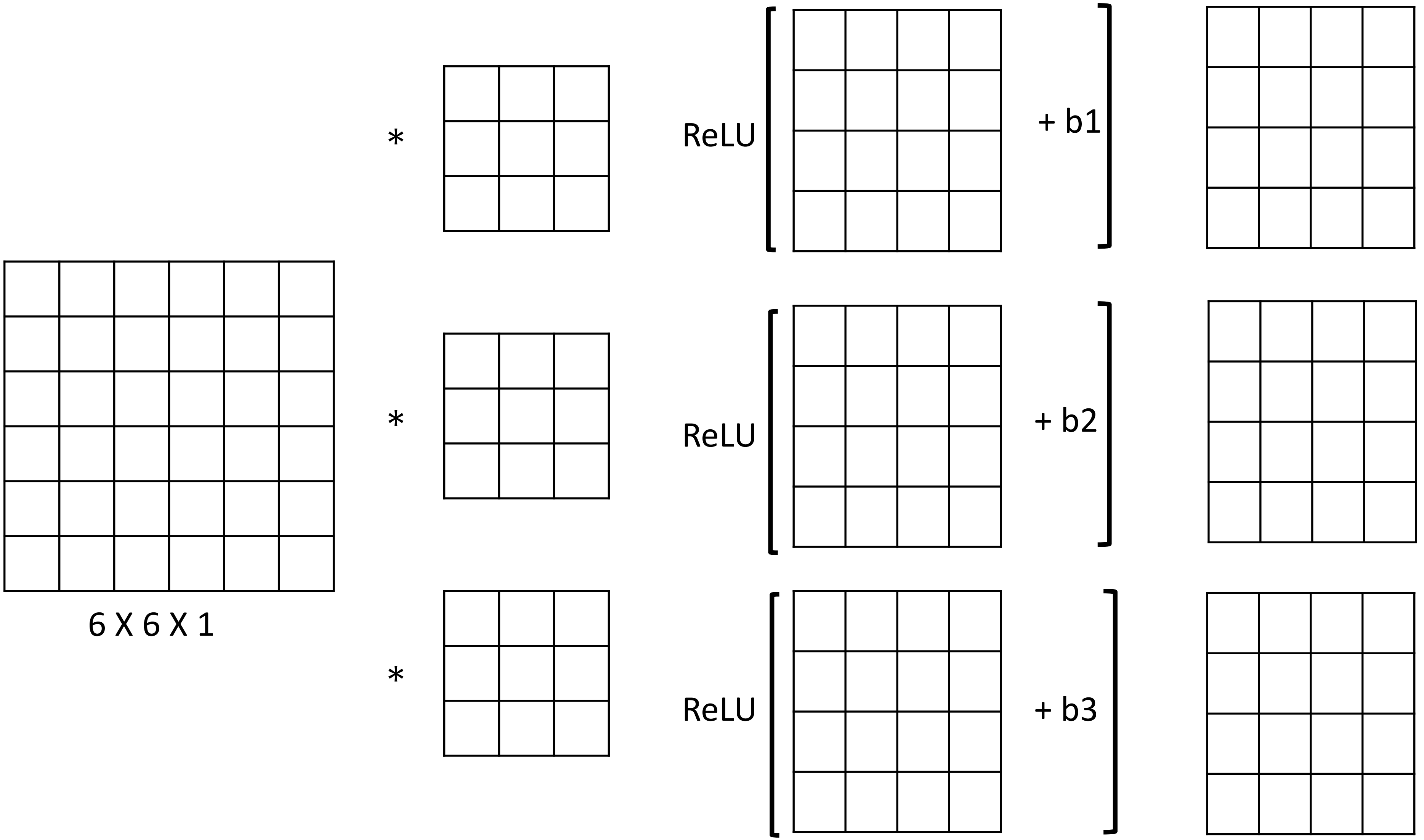
Why pooling?

- Reduce the number of parameters to learn
  - Speed up the computation
  - Avoid overfitting
- Only the strongest features are used in subsequent layers
  - Focus the network's attention to important features while ignoring secondary details
  - Making the features learned robust
- Translation invariance
- Numerous experiments show that it works surprisingly well

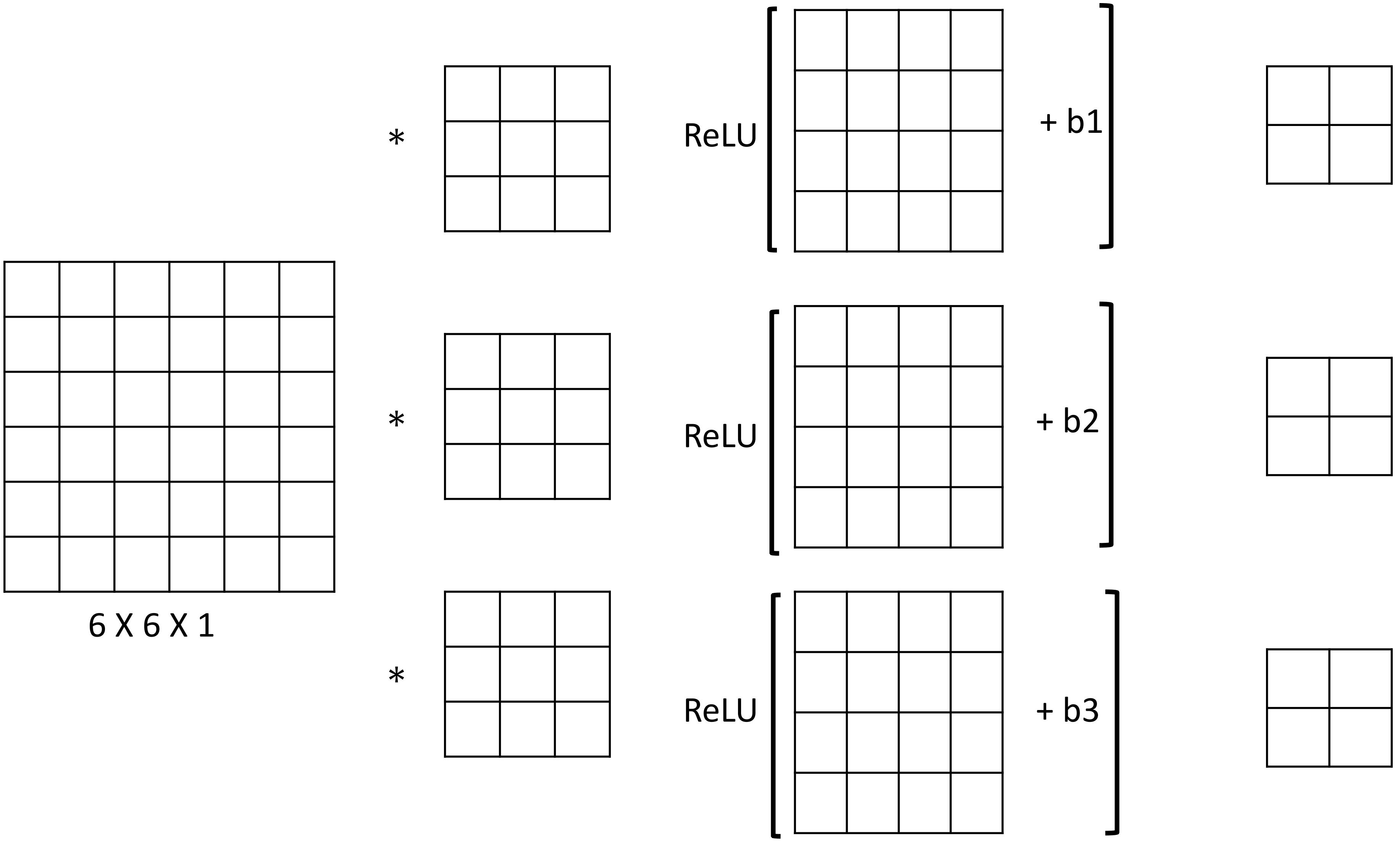
Three types of layers in a CNN

- Convolution (conv)
- Pooling (pool)
- Fully connected (FC)

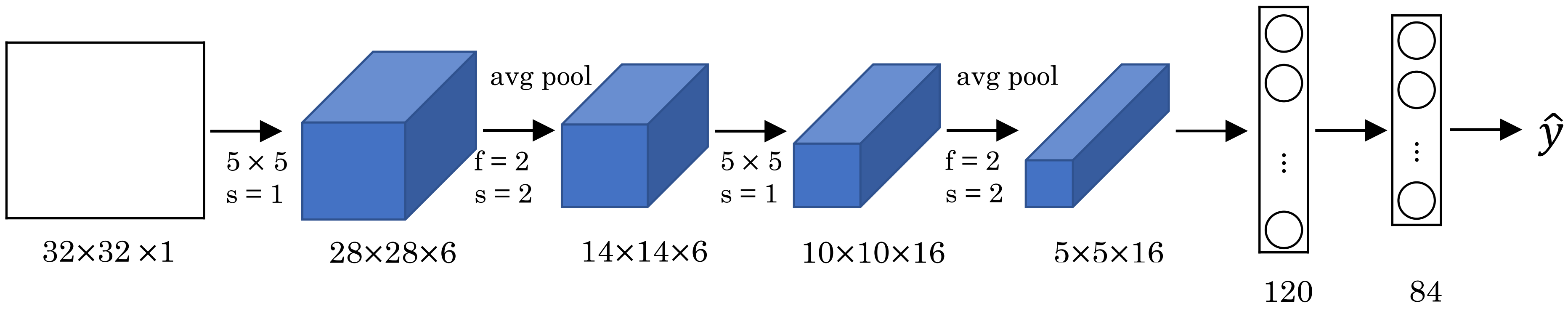
Convolution at one layer without pooling



Convolution at one layer with pooling



LeNet - 5



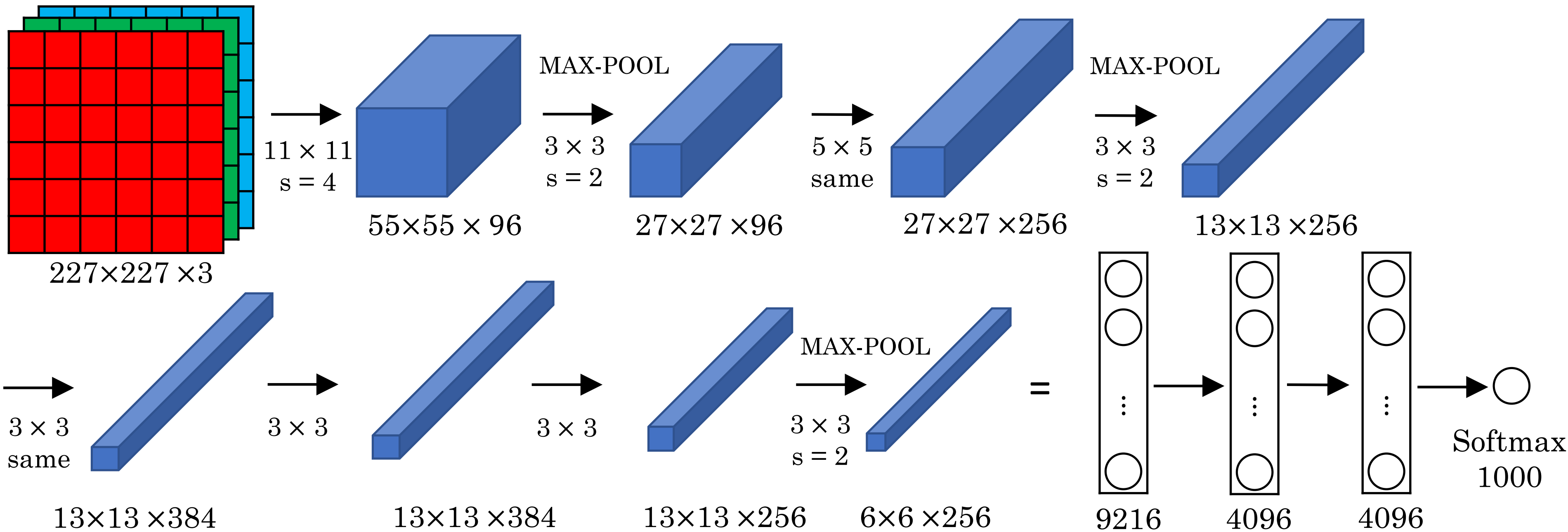
- 60,000 parameters
- Conv – pool – conv – pool – fc – fc – output
- n_h, n_w decrease, and n_c increase
- Activation: sigmoid/tanh (today: ReLU)
- Activation after pooling (today: before)

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

http://yann.lecun.com/exdb/lenet/

Credit: Andrew Ng

AlexNet



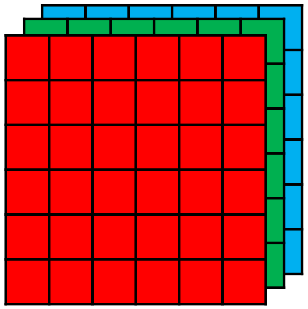
- Similar to LeNet, but much bigger with ~ 60 million parameters
- ReLU was used
- Multiple GPUs

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

Credit: Andrew Ng

VGG - 16

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



224×224 ×3 $\xrightarrow[\text{[CONV 64] ×2}]{}$ 224×224×64 $\xrightarrow[\text{POOL}]{}$ 112×112 ×64 $\xrightarrow[\text{[CONV 128] ×2}]{}$ 112×112 ×128 $\xrightarrow[\text{POOL}]{}$ 56×56 ×128

$\xrightarrow[\text{[CONV 256] ×3}]{}$ 56×56 ×256 $\xrightarrow[\text{POOL}]{}$ 28×28 ×256 $\xrightarrow[\text{[CONV 512] × 3}]{}$ 28×28 ×512 $\xrightarrow[\text{POOL}]{}$ 14×14×512

$\xrightarrow[\text{[CONV 512] ×3}]{}$ 14×14 ×512 $\xrightarrow[\text{POOL}]{}$ 7×7×512 $\rightarrow$ FC 4096 $\rightarrow$ FC 4096 $\rightarrow$ Softmax 1000

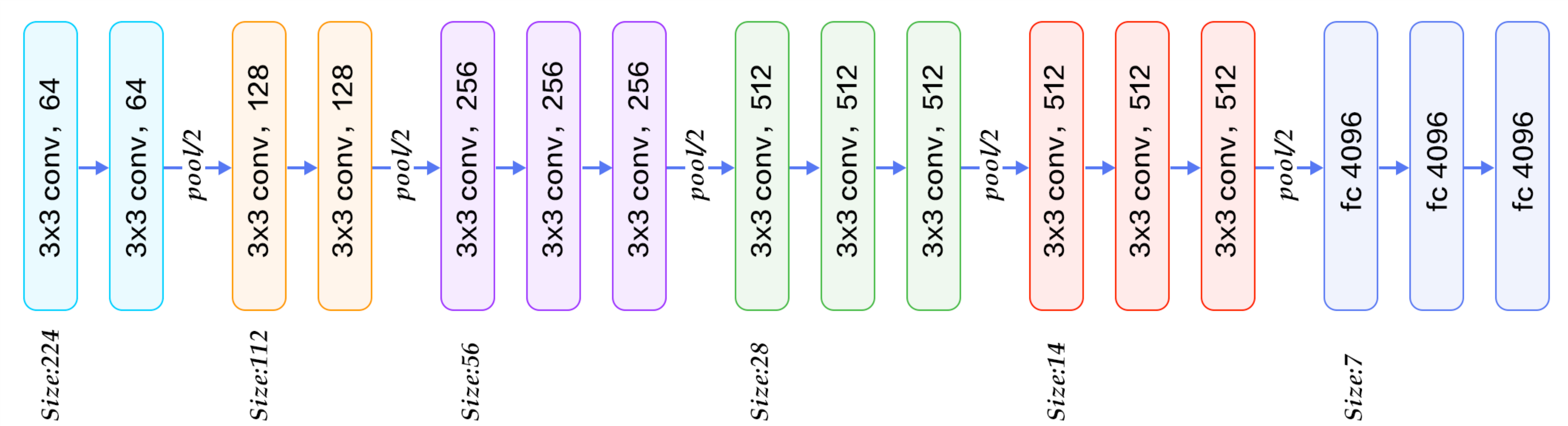
~138 million parameters.

n_h, n_w decrease, and n_c increase

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

Credit: Andrew Ng

VGG-16



https://www.quora.com/What-is-the-VGG-neural-network

CNN With Keras

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()


x_train = x_train.reshape(x_train.shape[0], 28,28,1).astype('float32')/225
x_test = x_test.reshape(x_test.shape[0], 28,28,1).astype('float32')/225
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), strides=1, padding="same", input_shape=(28,28,1), activation="relu"))
model.add(Conv2D(32, (3,3), padding="same", activation="relu"))
model.add(Flatten())
model.add(Dense(10, activation="softmax"))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=128, epochs=10, verbose=1, validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

model.add(MaxPooling2D(pool_size=(2,2), strides=1)

Credit: Felicia Nurindrawati