Trustworthy AI Autonomy

M1-1: Basics of Intelligent Autonomy

Ding Zhao

Assistant Professor

Carnegie Mellon University

Carnegie Mellon University

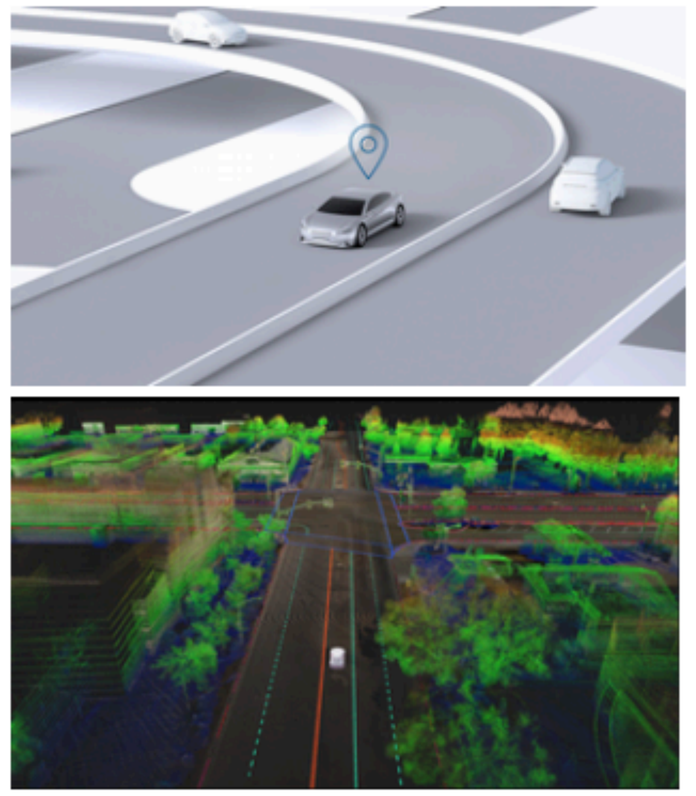
Safe AI Lab @CMU

Contents

- Basics of autonomy
    - Example: self-driving cars
- Review of deep learning basics
    - Case study: traffic sign recognition
    - Training: backpropagation, stochastic gradient descent (SDG)
    - Structure design: Convolution, pooling, and dropout

Key component of an autonomy

Localization



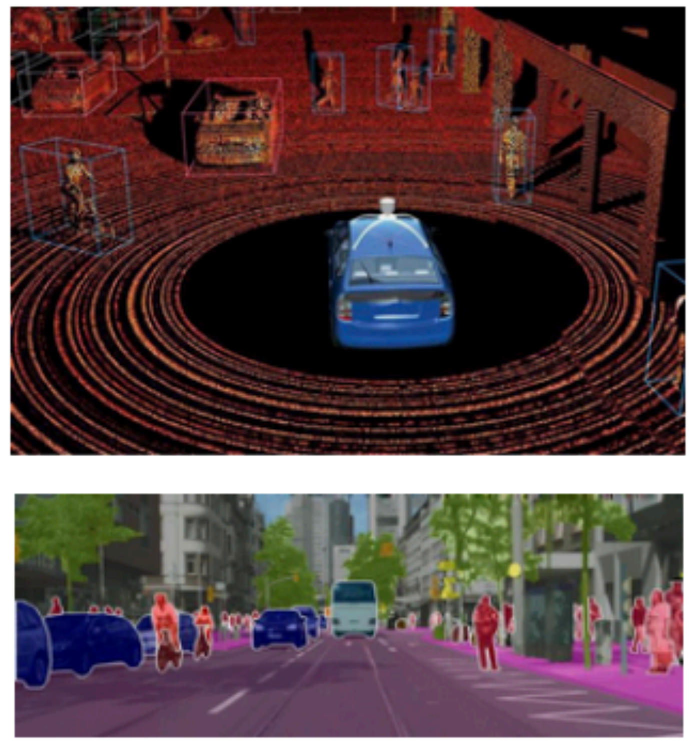
Where am I?

Where am I heading?

HD maps (static objects)

Trustworthy Localization

Dynamic Scene Understanding



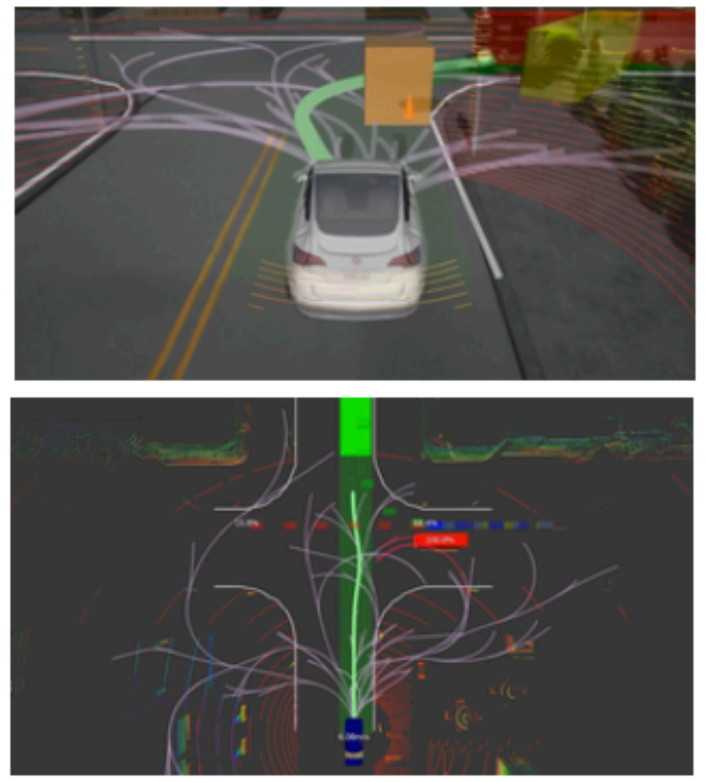
Dynamic agents

Action/trajectory prediction

Free space

Trustworthy Scene Understanding

Path planning



Trajectories

Environment

Future agents' trajectories (predictions)

Trustworthy Path Planning

Control

Steering

Speed

Lateral & Longitudinal Control

Trustworthy Control

User Interaction

User commands

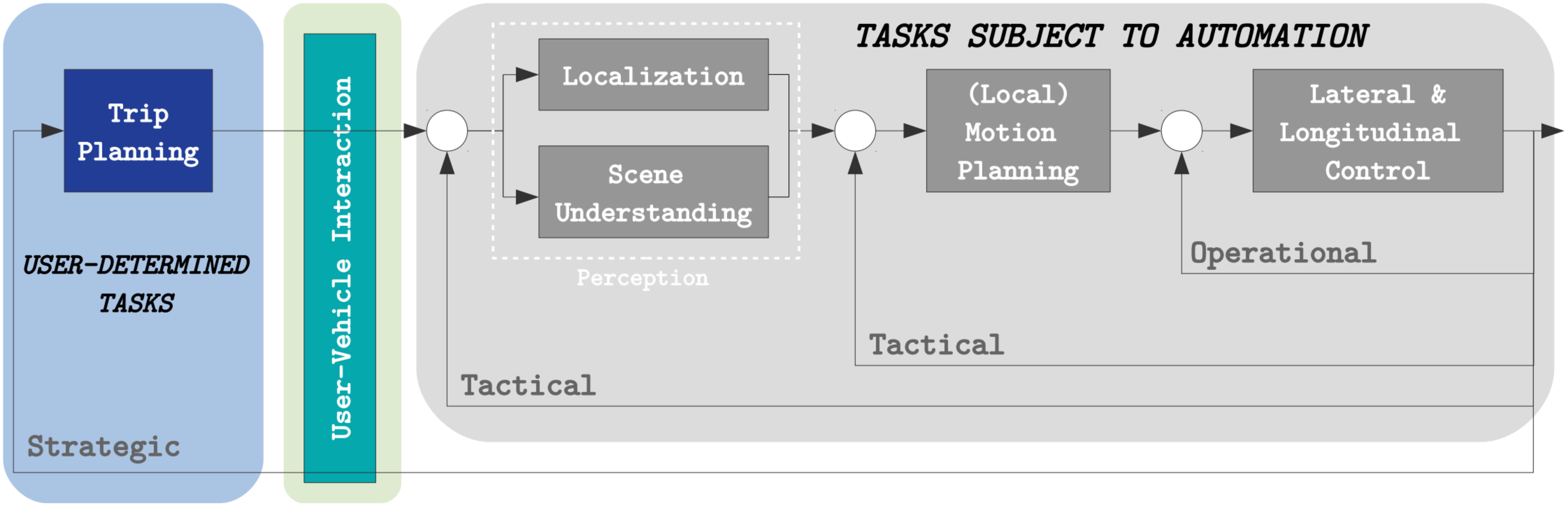
Request to intervene

Other agents' interaction

Trustworthy Interaction

Fernández Llorca D, Gómez E. Trustworthy Autonomous Vehicles. Joint Research Centre (Seville site); 2021 Dec.

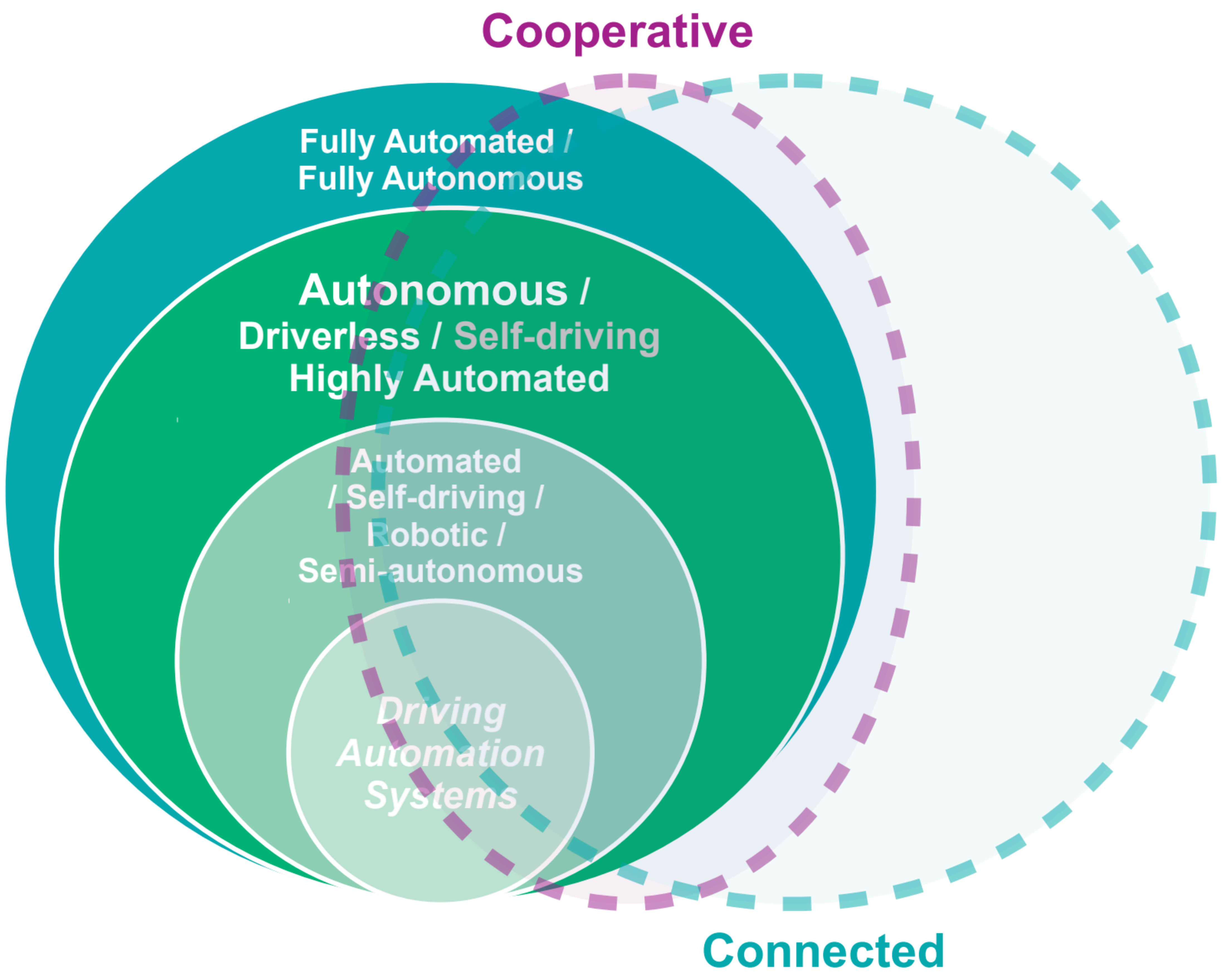
Algorithmic structure



Fernández Llorca D, Gómez E. Trustworthy Autonomous Vehicles. Joint Research Centre (Seville site); 2021 Dec.

Technology Readiness Levels (TRLs) for each Level of Automation of AVs

SAE Level	TRLs	Examples
0	[9]	Conventional modern cars.
1	[9]	Adaptive Cruise Control, Stop & Go, Park Steering Assist, Lane Keeping Assist (proved in operational environment).
2	[9]	Traffic Jam Assist, Automatic Parking Assist, Tesla's Autopilot (proved in operational environment).
3	[5 - 9]	Traffic Jam Chauffeur / Pilot, Automated Lane Keeping Systems, Highway Chauffeur / Pilot, Robotaxis (proved in operational environment).
4	[4 - 7]	Highway Autopilot, Automatic Valet Parking, Autonomous Urban Shuttles, Autonomous Delivery Vehicles, Driverless Robotaxis (demonstrated in operational environment).
5	[1 - 4]	Formulated, experimental proofs of concept, validated in the lab.



SAE Level	Driving Mode	Driver's role	Driver's responsibility
Level 0	Manual Driving	Driver	Full
Levels 1-2	Assisted Driving	Assisted Driver	Full
Level 3	Automated Driving	Assistant/Backup Driver	Shared
Levels 4-5	Autonomous Driving	Passenger	None

Fernández Llorca D, Gómez E. Trustworthy Autonomous Vehicles. Joint Research Centre (Seville site); 2021 Dec.

An example of a self-driving car

Velodyne 16 channel Lidar
360° Horizontal FOV, 30° Vertical FOV, up to 100 meters, Dual Returns

MovAtel SPAN A1
GPS+GLONASS, IMU, RTK

2 Nexcom Lumina
Intel i7, 120GB
SATA HDD

PointGrey Camera

Webcam

IBEO LiDAR
1 front 200 meters
110°Horizontal
3.2°Vertical

Delphi ESR 2.5:
FOV: 60 m / 90°, 175m / 20°

10 Dataspeed:
By wire control: throttle, brake, gear, steering, turning signal

Neobotix Ultrasonic
Sensor: 12 at front and rear, 4 at side

Drive
PX 2

- Computers
- Sensors
- Actuator

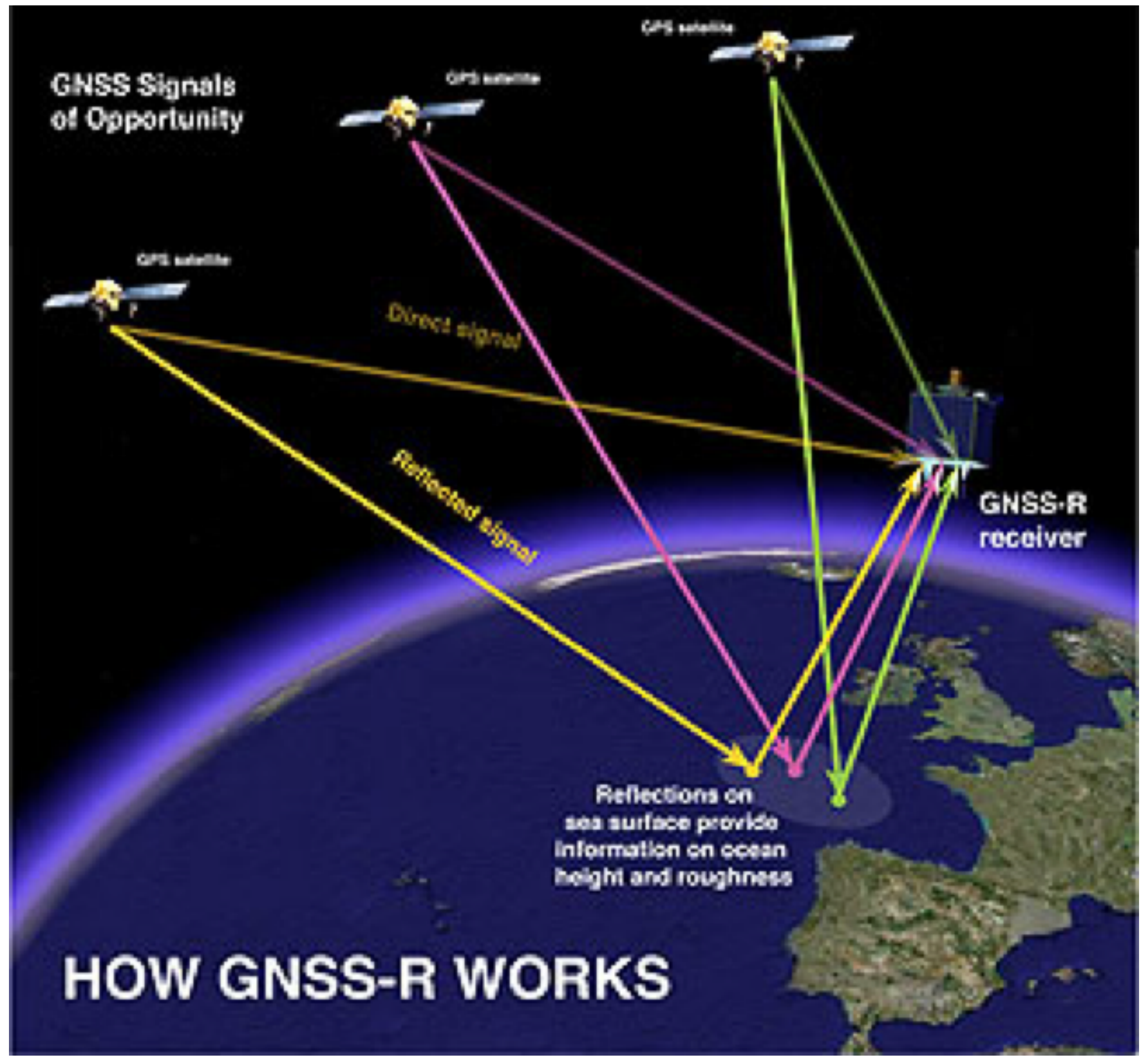
Global Navigation Satellite Systems

US (GPS) : 24 / 1995

Russa (GLINASS): 24 / 2011

India (IRNSS): 7 / 2017

Over 110 satellites



Thousands of base stations and repeaters

EU (GALILEO): 30 / 2013

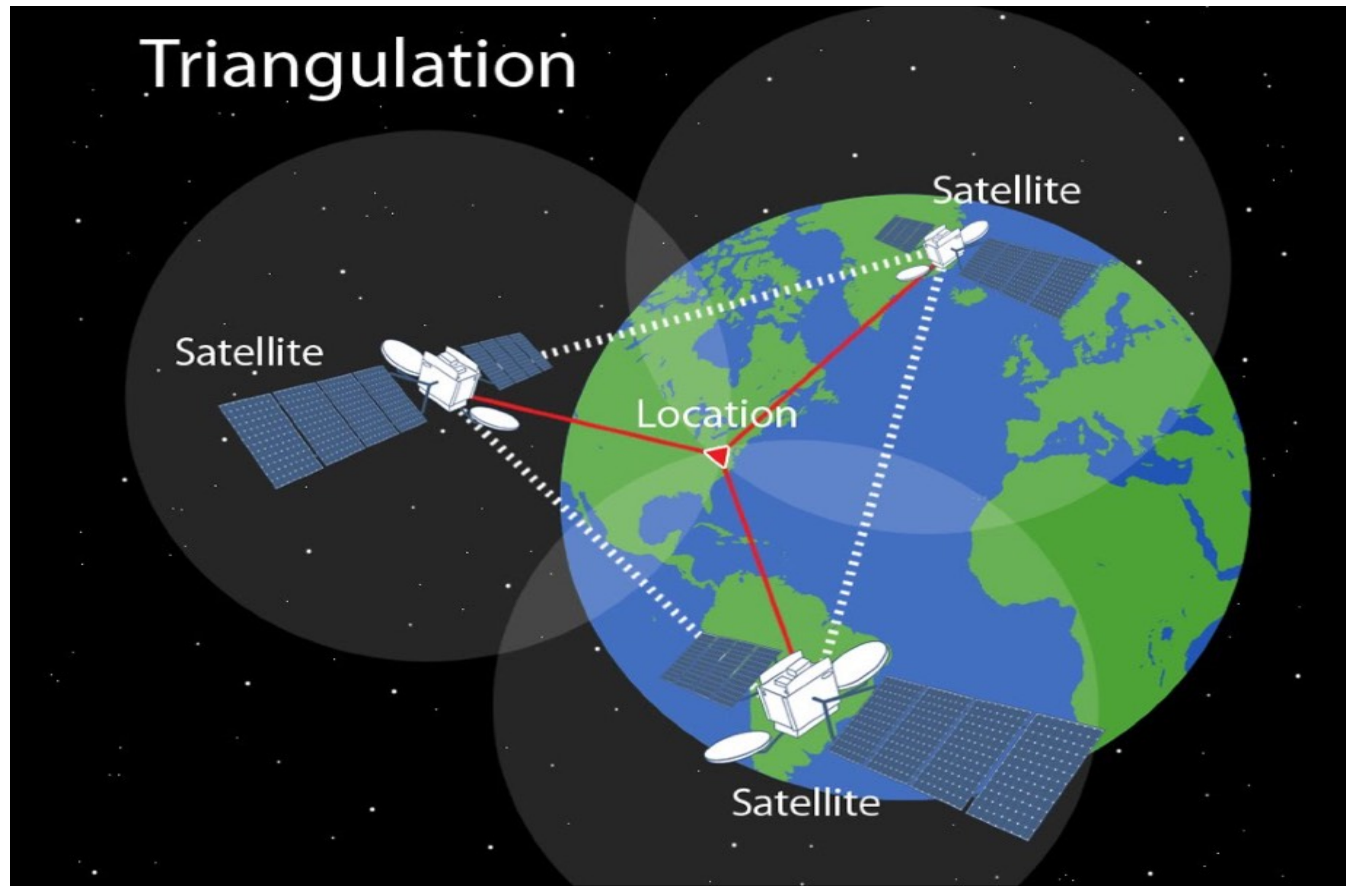
China (BeiDou): 35 / 2020

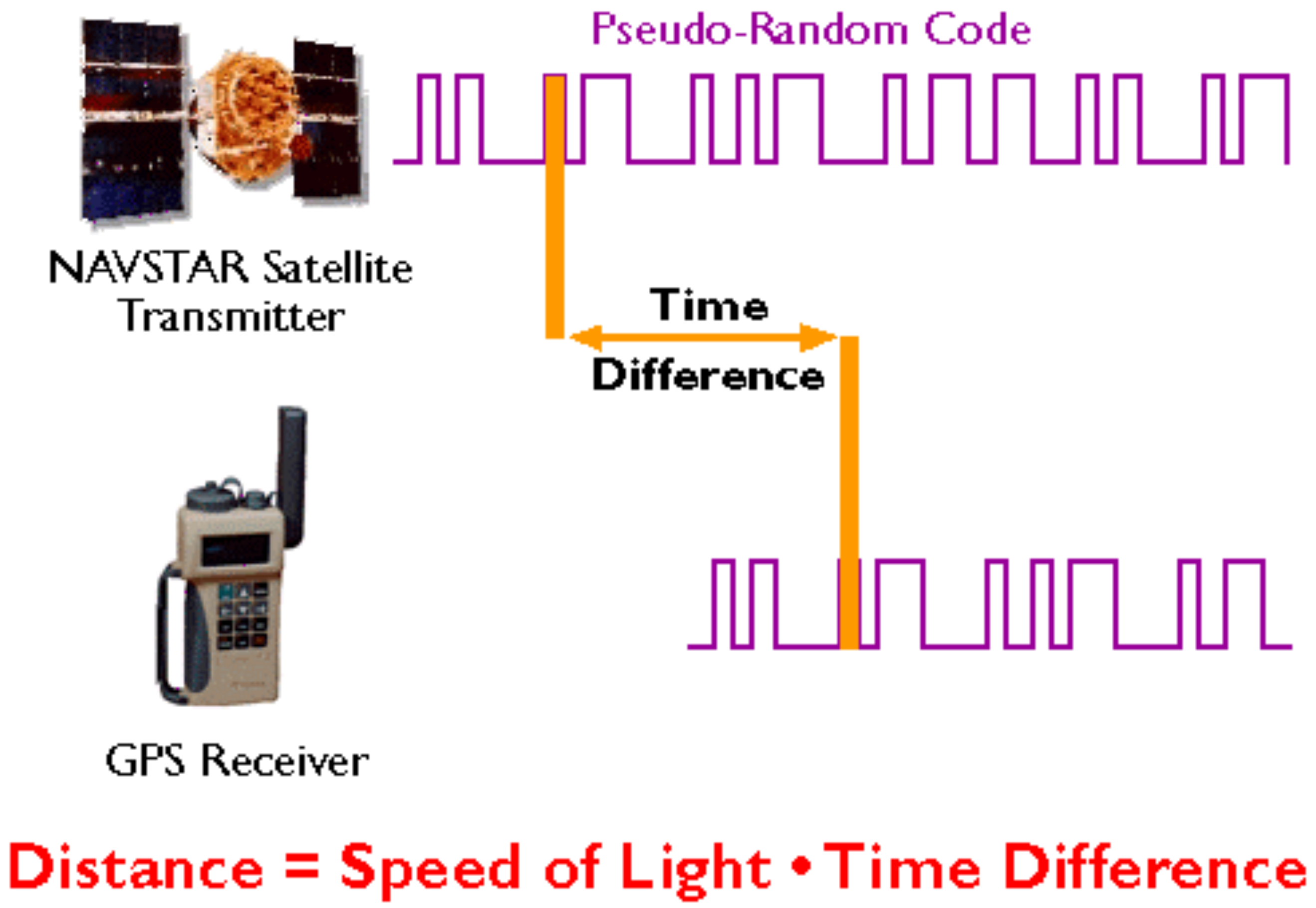
Japan (QZSS): 4 / 2017

Num of satellites may be out of dated

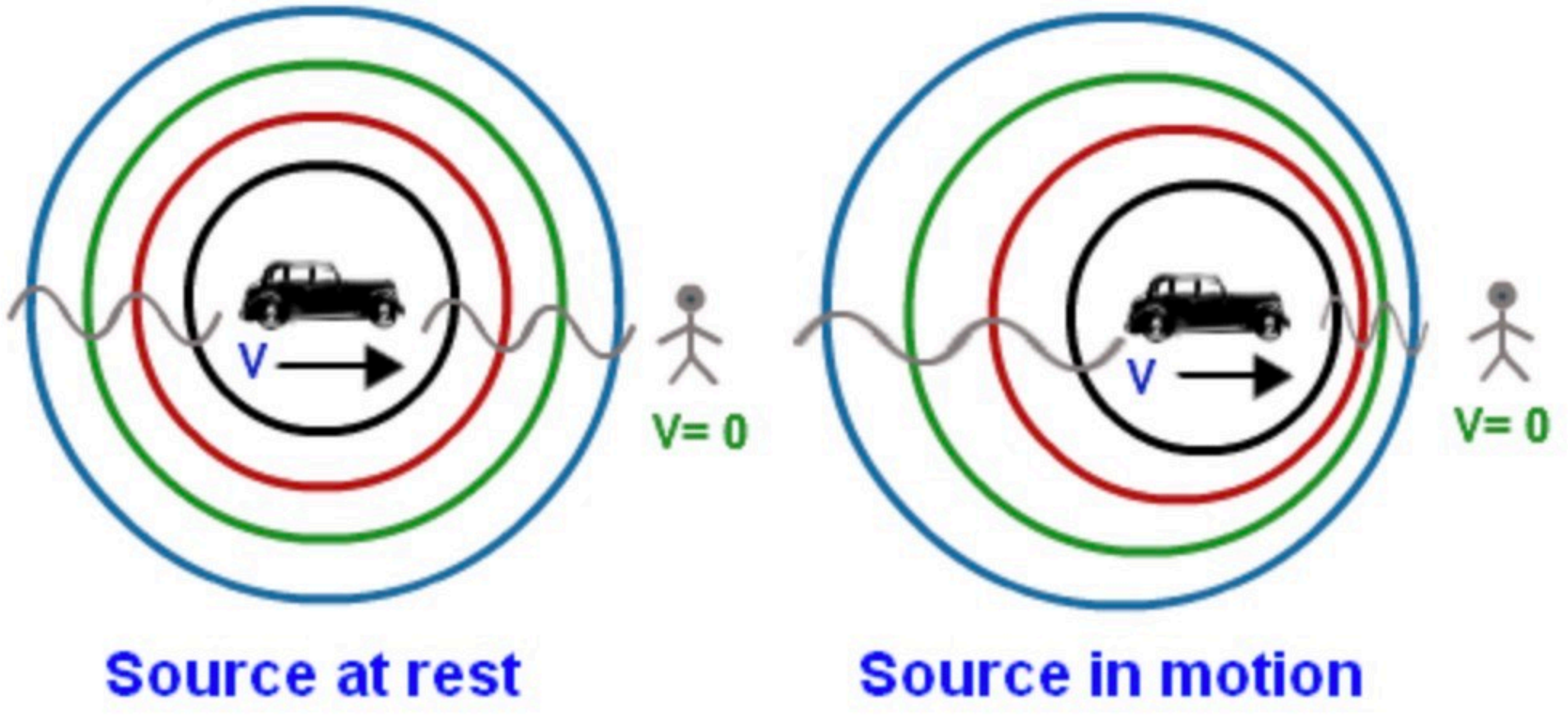
How GNSS Works

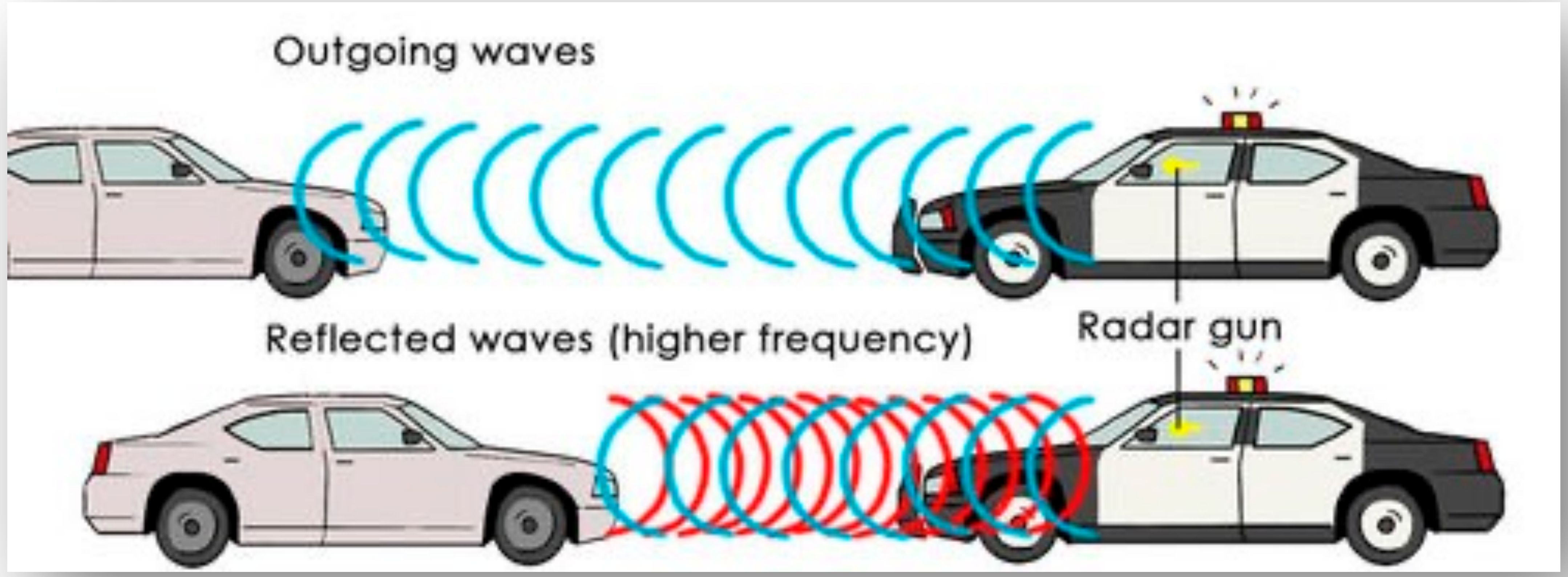
- The Global Positioning System (GPS)
  - 27 satellites orbiting the Earth at an altitude of 20,000 km (24 in operation and three extras in case one fails).
  - At least four satellites "visible" in the sky from anywhere on the earth.
  - Satellites broadcast radio signals can be received by GPS units
- How GPS works: Trilateration
  - Needs at least three satellites
- How to measure the distance to a satellite
  - Use a pseudo-random code
  - On a satellite: atomic clock; on a receiver: quartz clock
  - Measure time lag to calculate distance





Fundamental of Radar





Lidar and Automated Vehicles

Google car

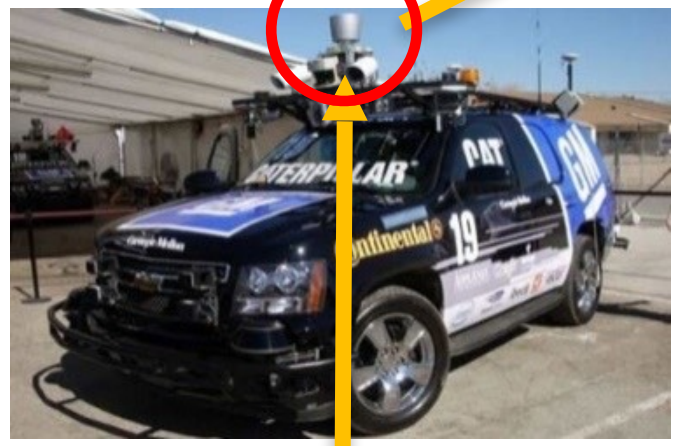
Stanford's Stanley

Sebastian Thrun

DARPA Urban Challenge (2007)

CMU's Sandstorm (7.32/150 mi)

Boss



CMU-GM Cadillac SRX

DARPA Grand Challenge (2004) → **(2005)**

No one finished the test…

MIT's Talos

UM-Ford Fusion

30 inches in diameter, 100 lbs

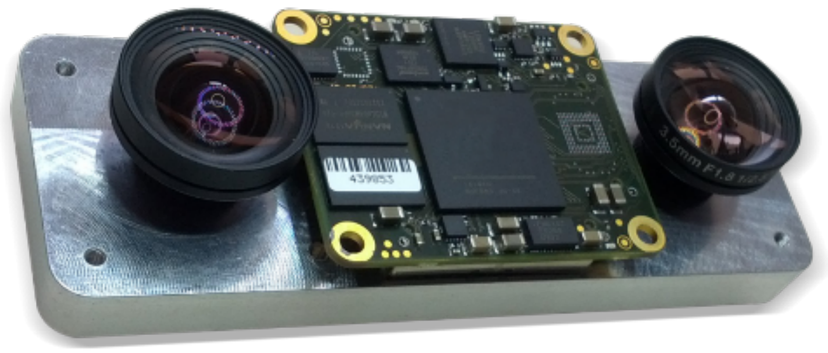
Team DAD (Digital Audio Drive)

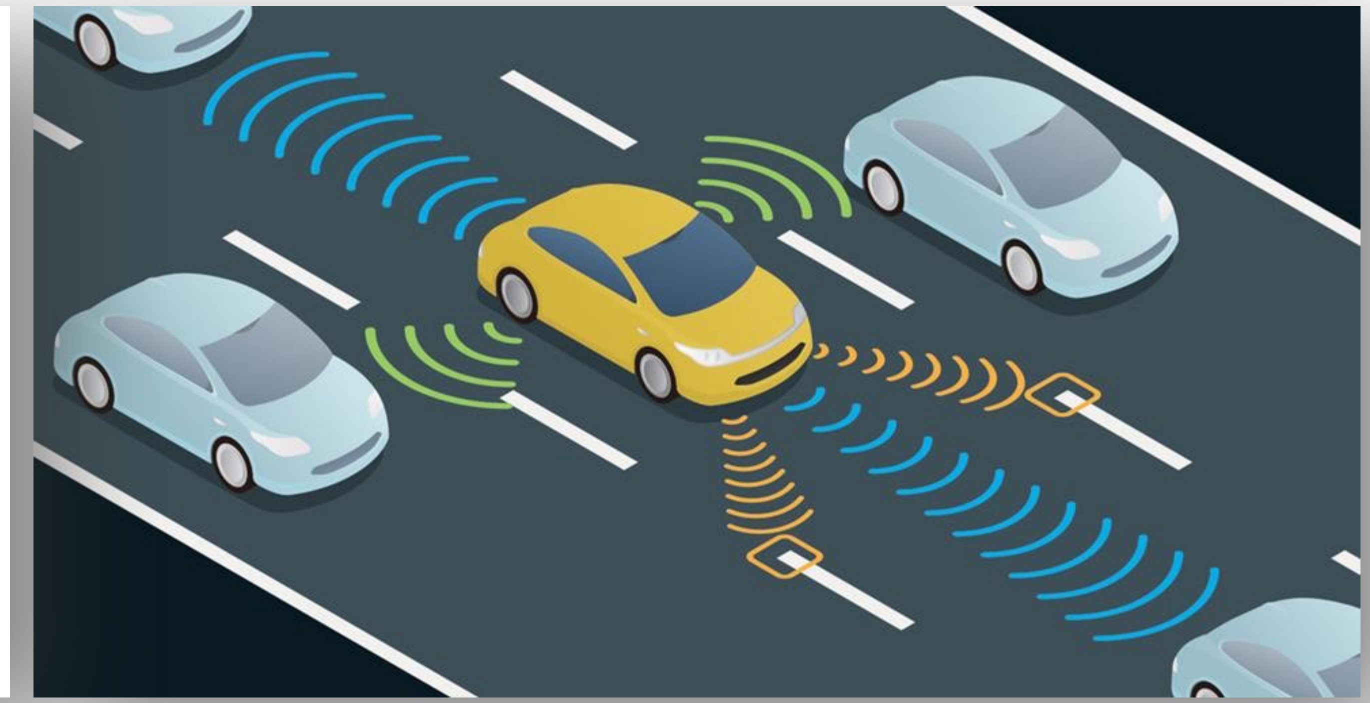
5 out of 6 teams that finished the course use Velodyne Lidar

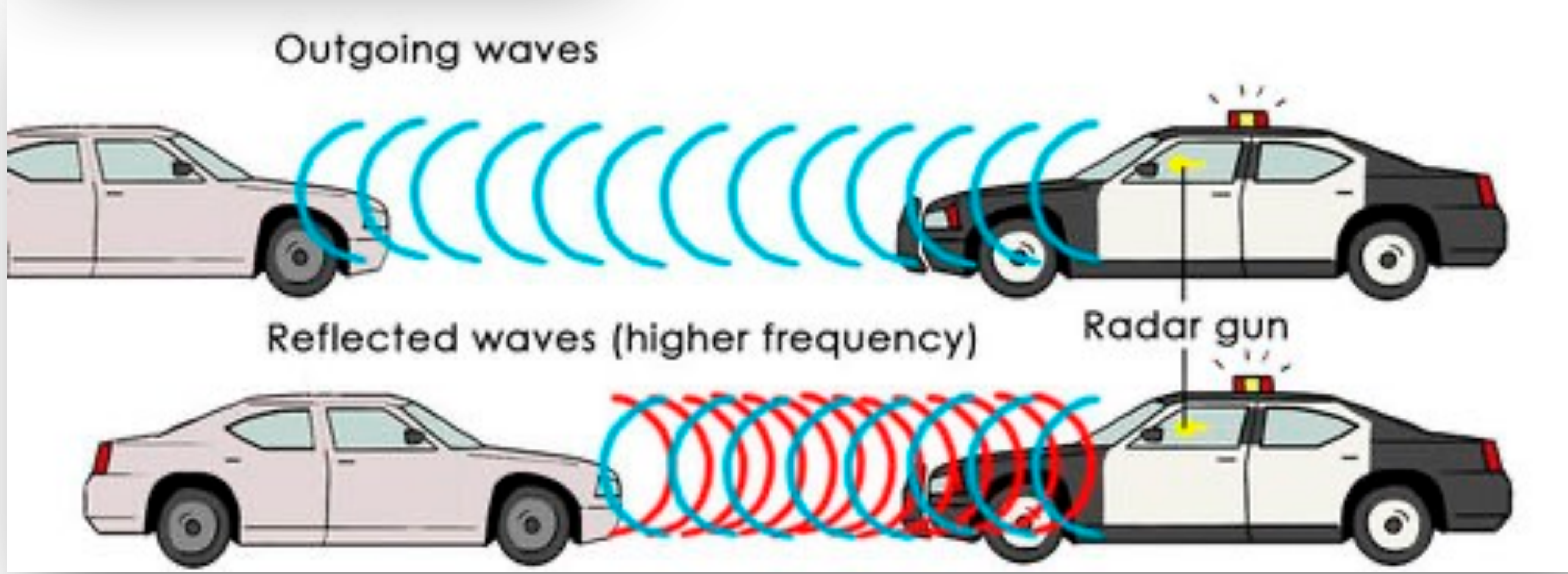
Lidar types by laser channels

Solid state

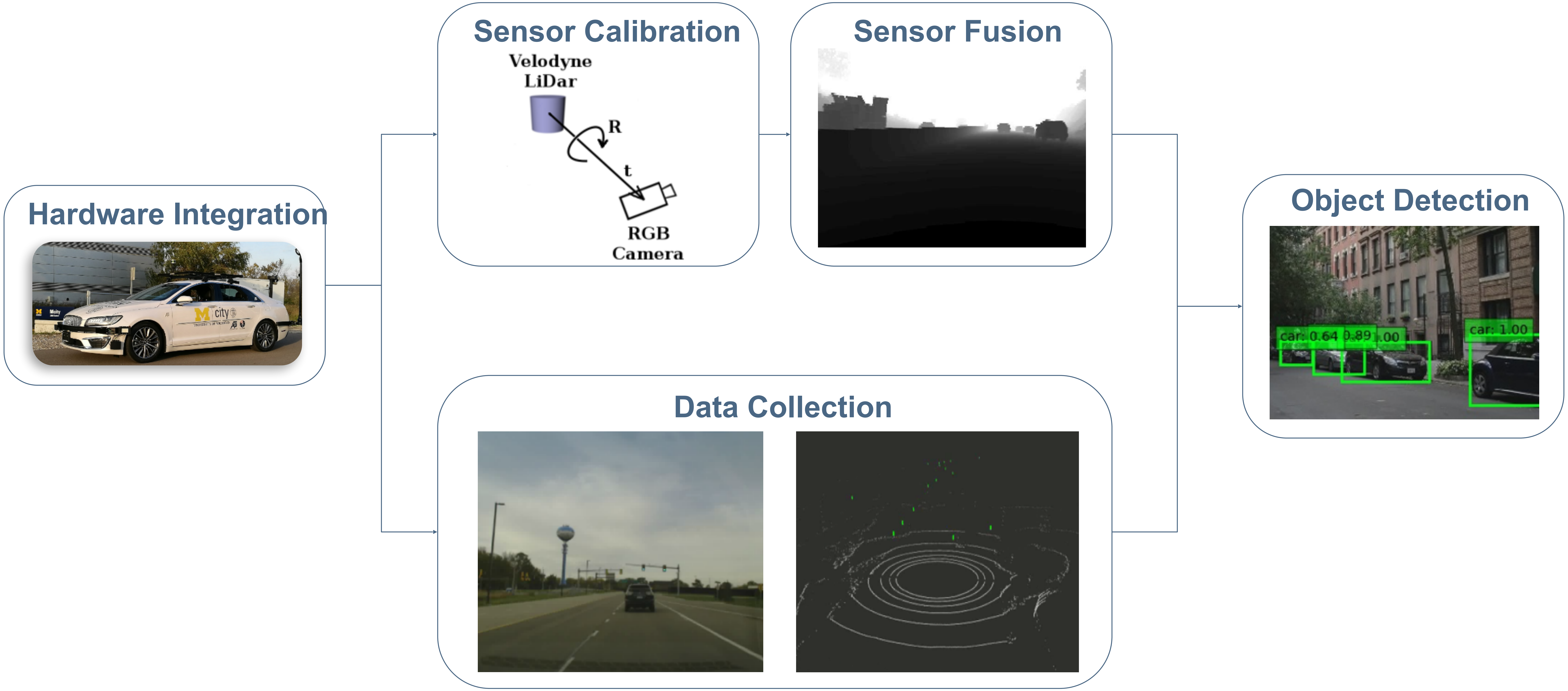
Radar+ LiDAR+Camera





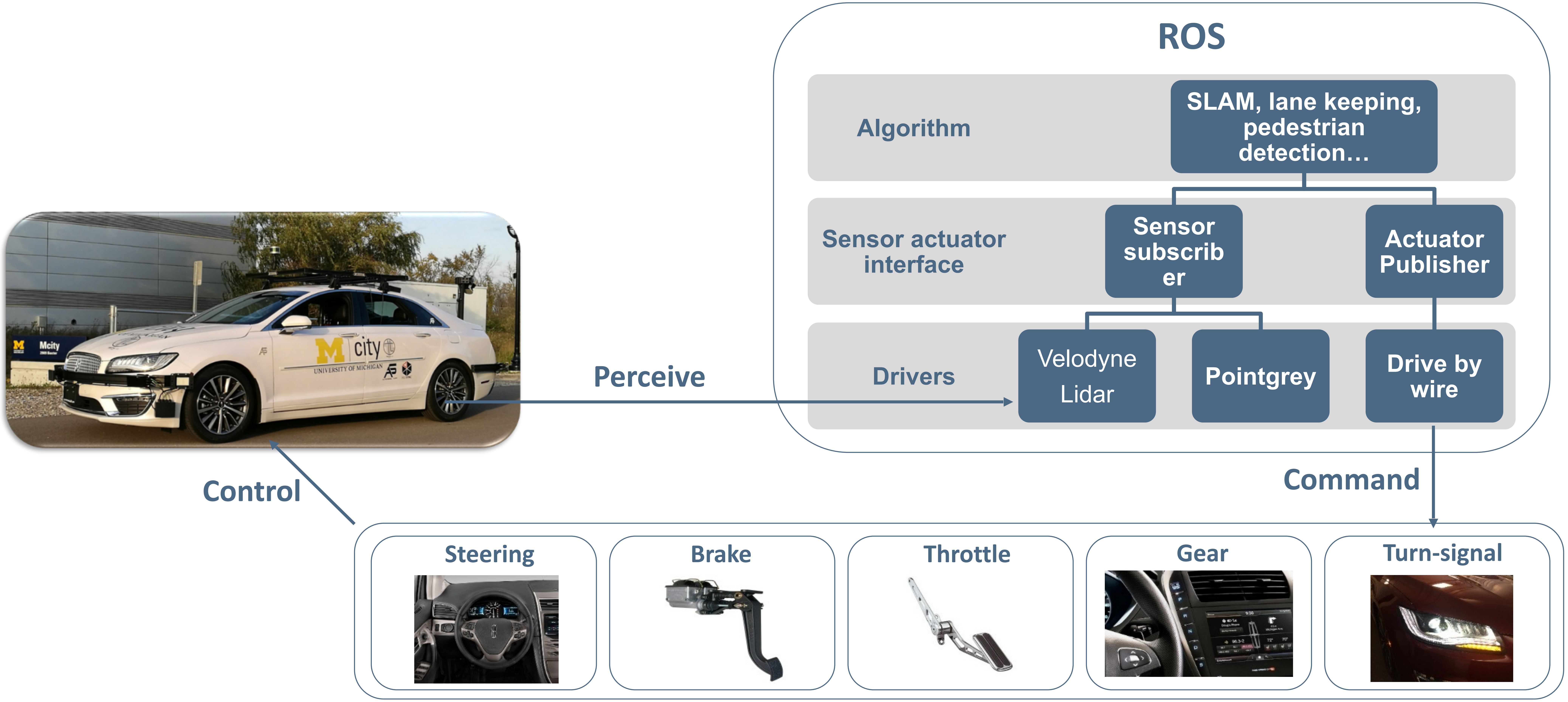


How to Use the Sensors

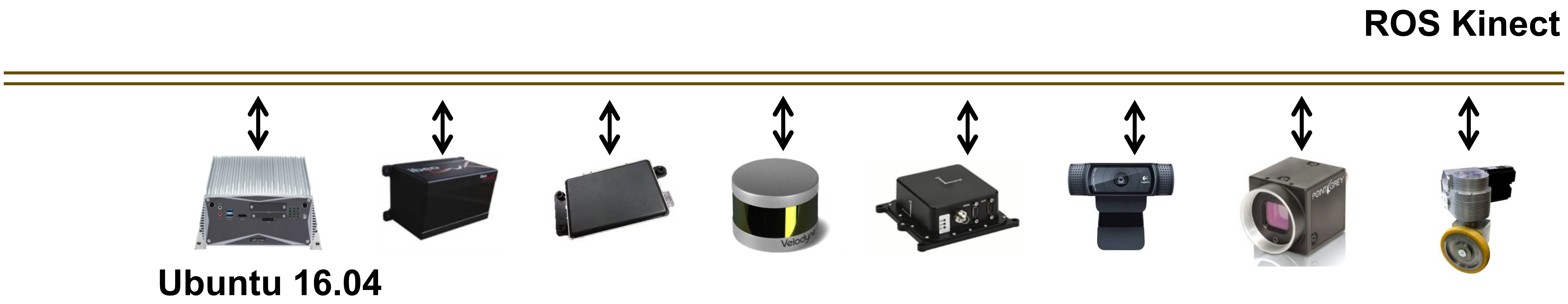


Camera Lidar Fusion Result

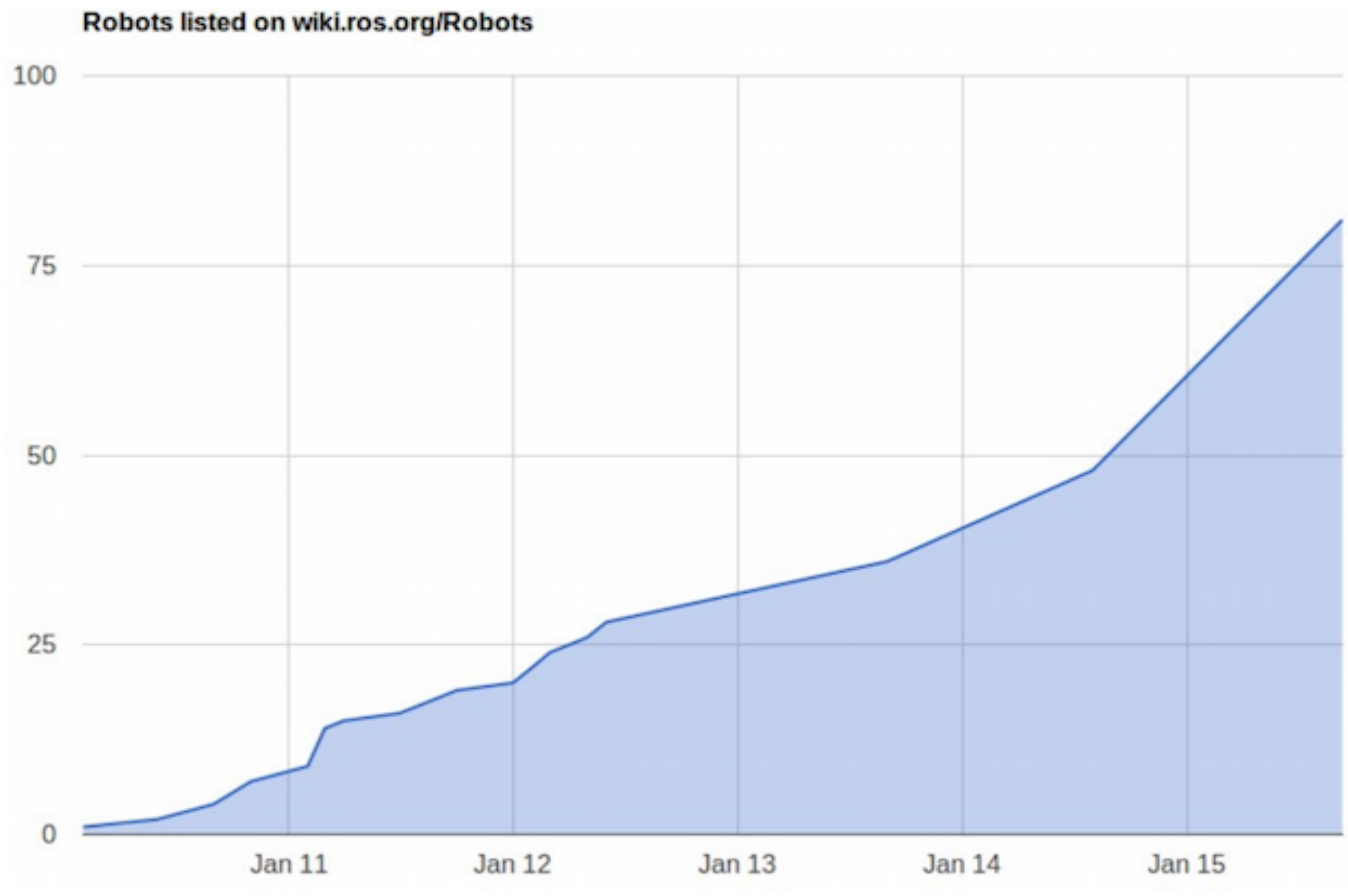
Structure of the Automated Vehicle



ROS Network



What is ROS (Robotic Operational System)



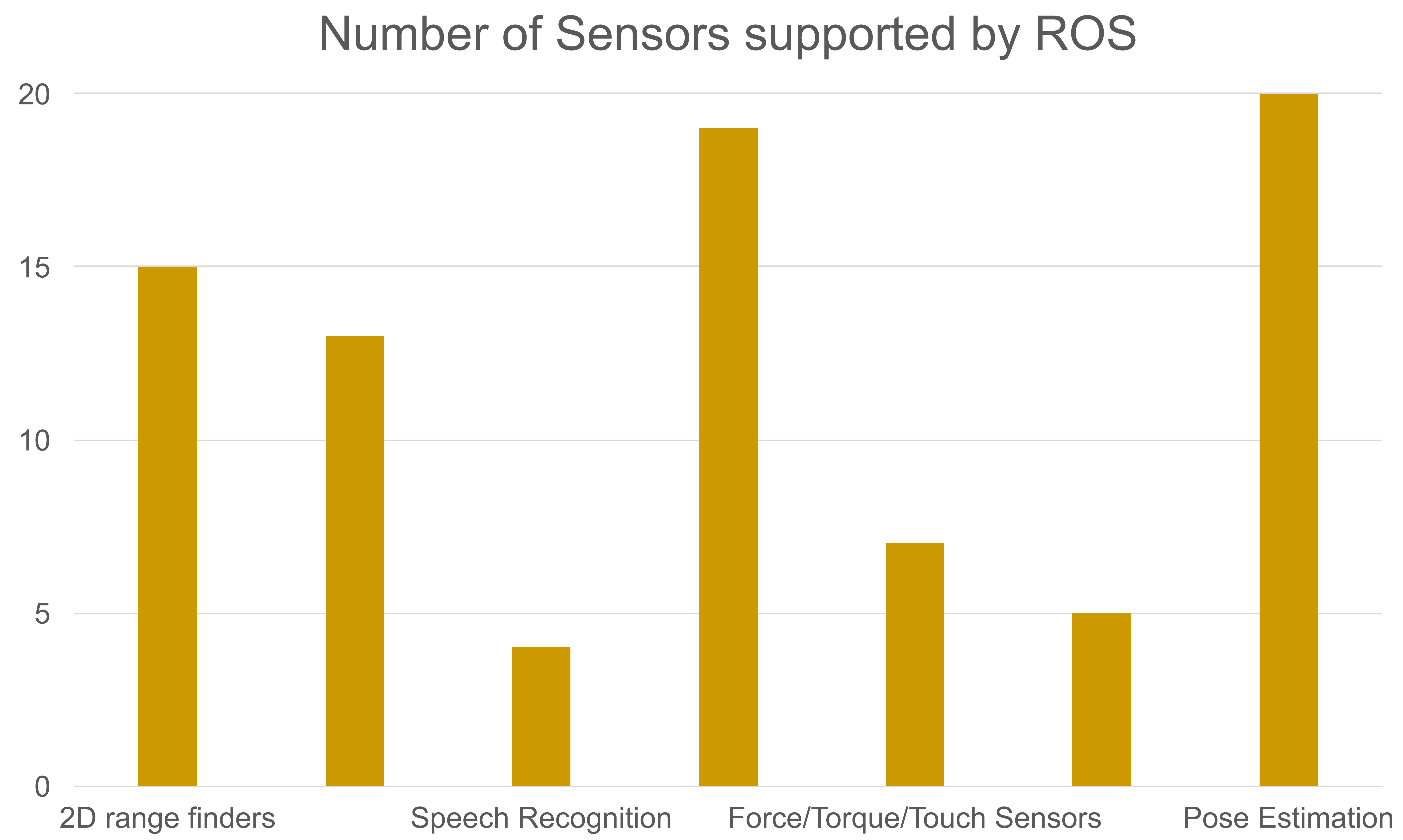
ROS provides 2000+ software libraries

The total line count is over 14 million lines of code

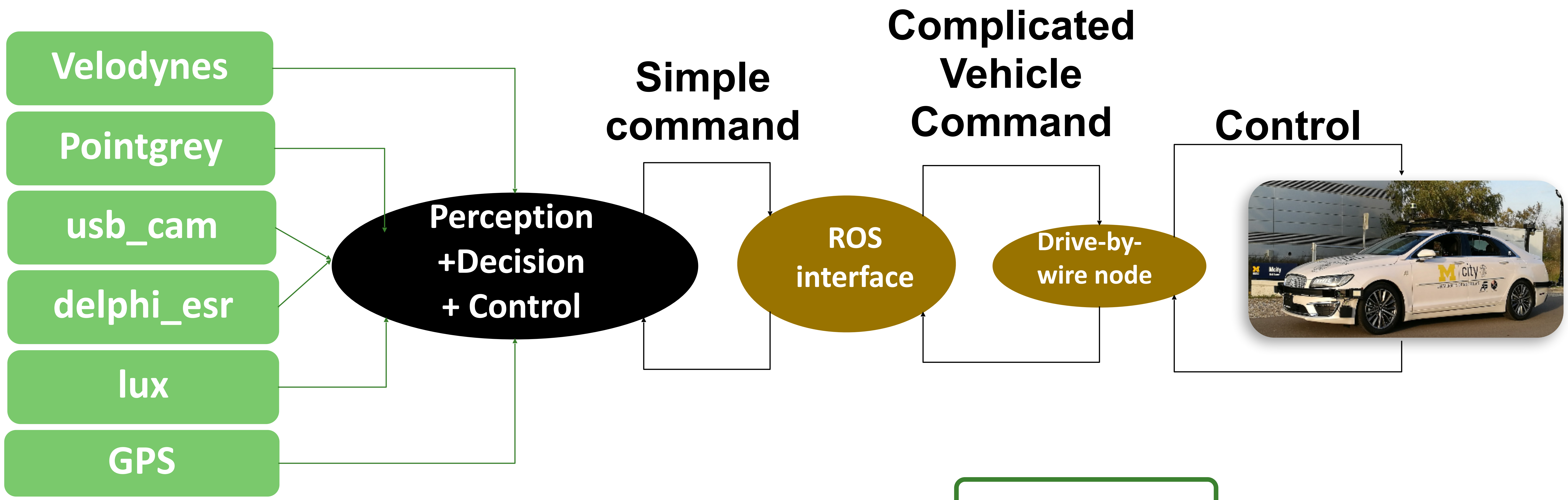
There have been 2,477 authors

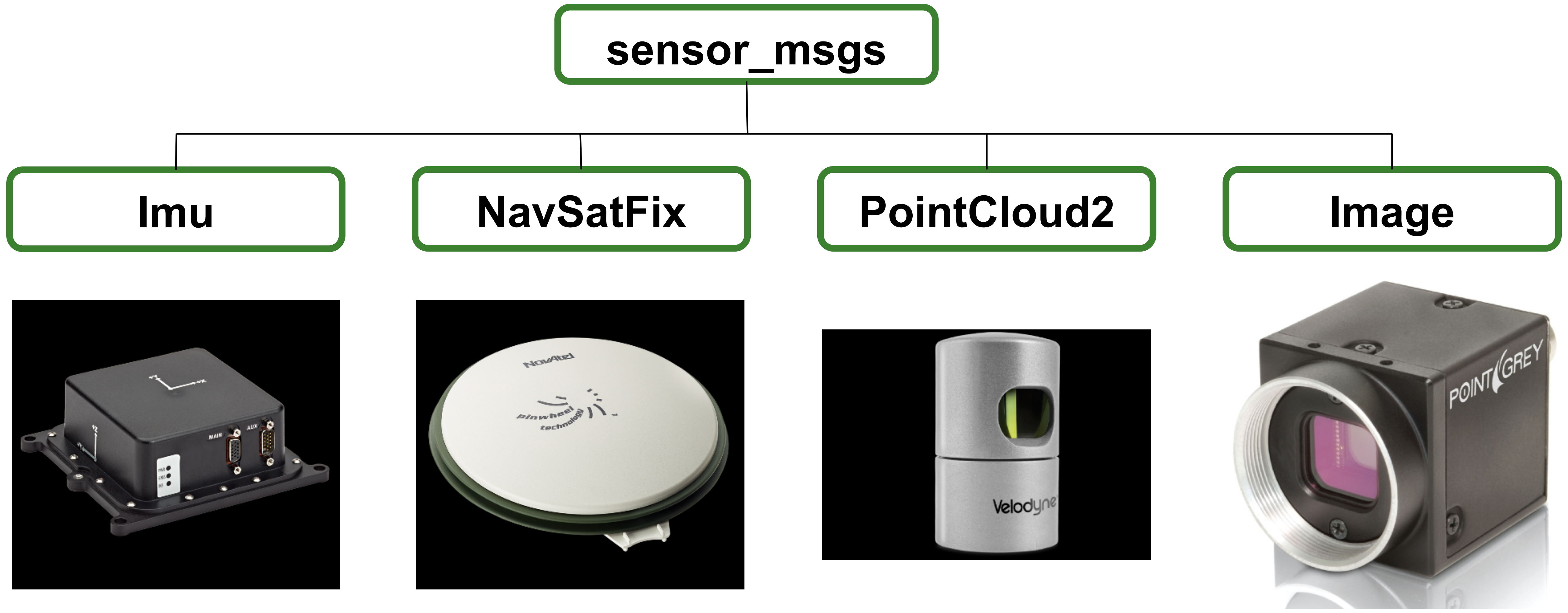
In total 181,509 commits

ROS Hardware Support



Real Time ROS Structure





[BONUS] Real Interview Case Study

- How to Build an AV from Scratch?
  - Decide driving environment (urban, highway)
  - Decide driving functional requirement (longitudinal, lateral, intersection)
  - Decide hardware (sensors, actuators, computational units)
  - Decide communication approach (middleware)
  - Sensor calibration, synchronization
  - Design high level algorithms (localization, detection, tracking, decision, control)
  - Tests (simulation, on-track, naturalistic driving)

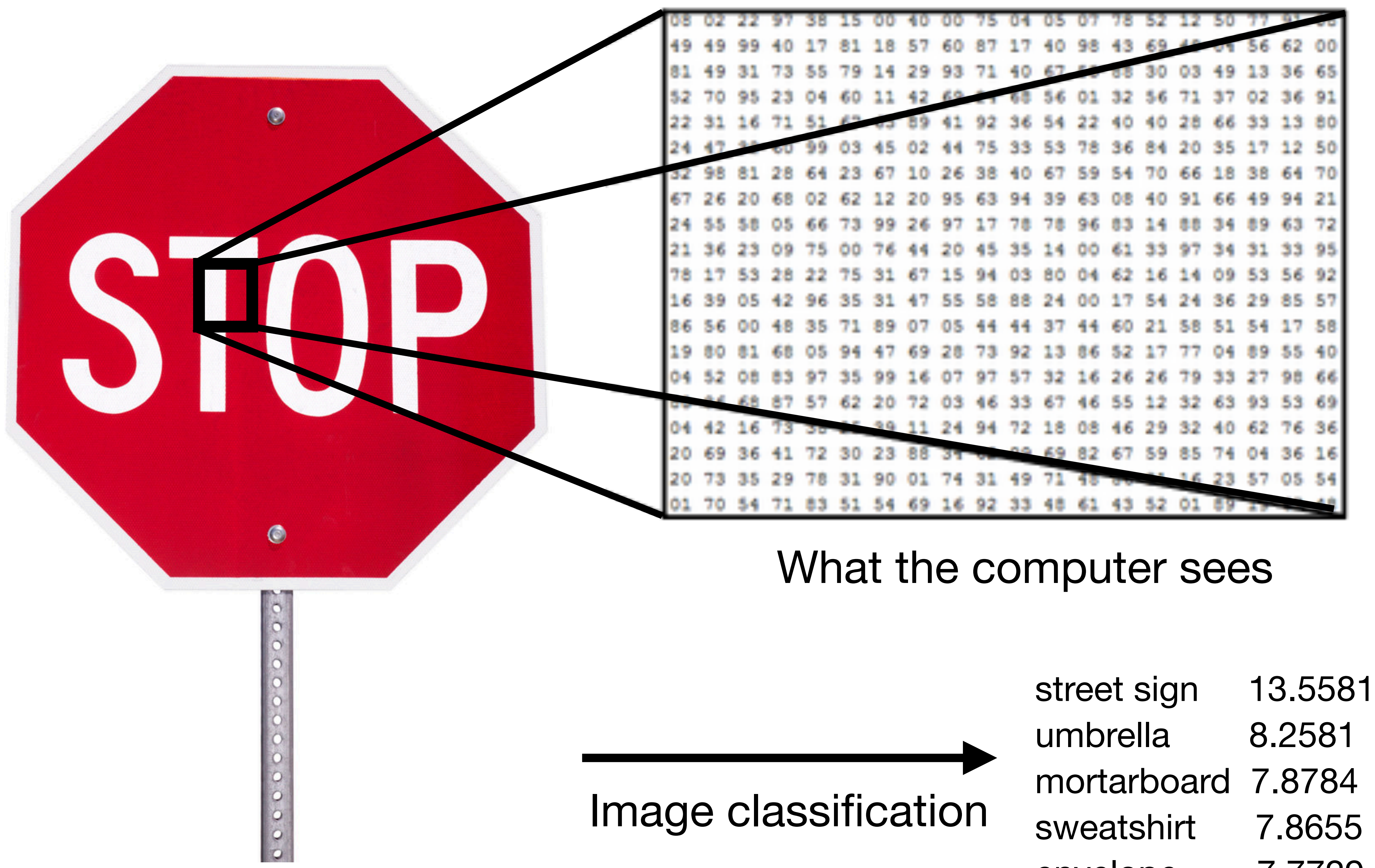
Multi-sensor

- Selection of sensors: LiDARs, Cameras, infrared,
- Calibration
- Synchronization

Contents

- Basics of autonomy
  - Example: self-driving cars
- Review of deep learning basics
  - Case study: traffic sign recognition
  - Training: backpropagation, stochastic gradient descent (SDG)
  - Structure design: Convolution, pooling, and dropout

Image data



What the computer sees

Image classification

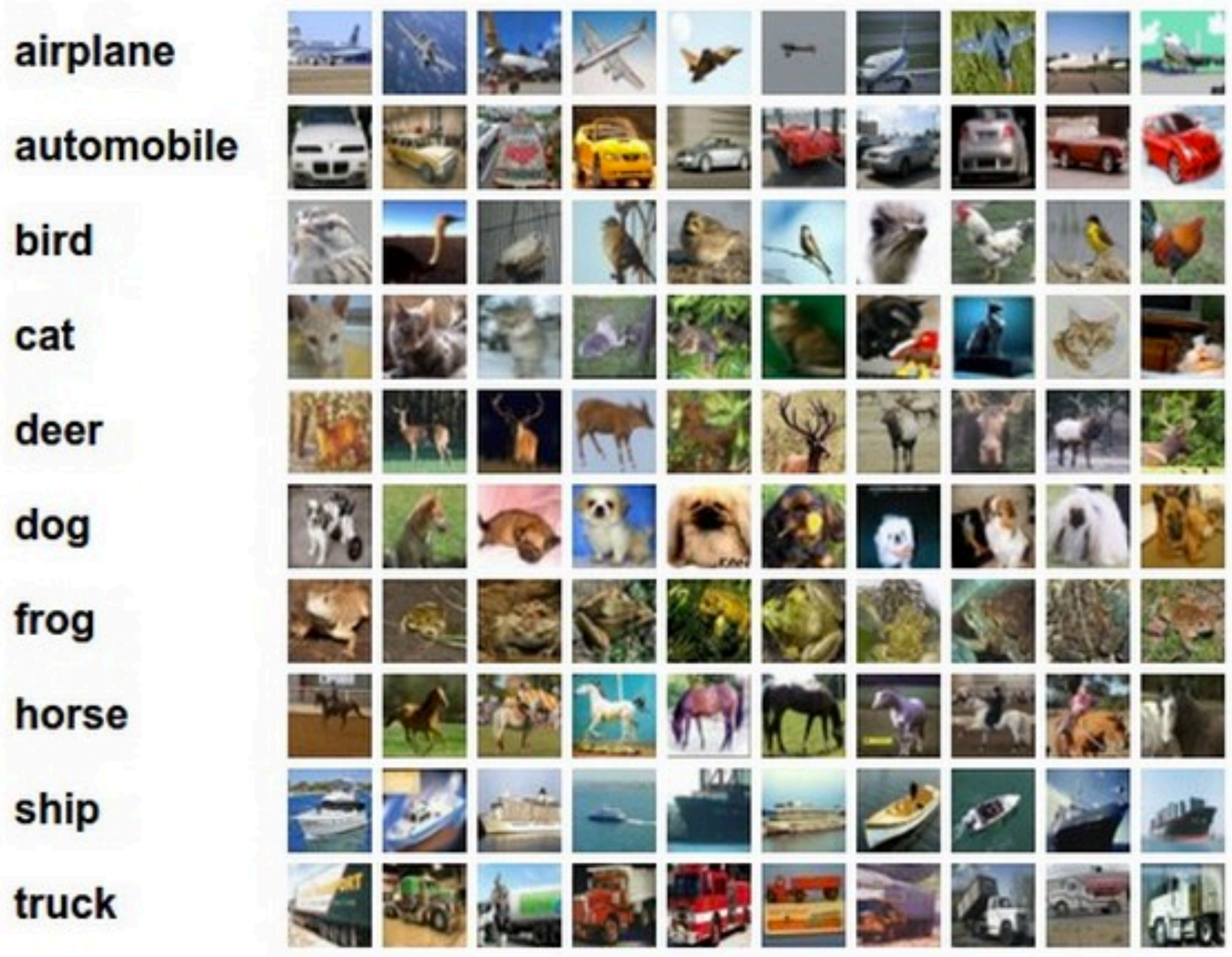
street sign	13.5581
umbrella	8.2581
mortarboard	7.8784
sweatshirt	7.8655
envelope	7.7729
…	

- Image is represented as one large 3-dimensional array of numbers
- The stop image has 248 x 400 pixels, so it has 248 x 400 x 3 = 297,600 numbers
- Each number is an integer ranging from 0 to 255.
- **Our task:**

predict the label *"street sign"* (y) *of* this 297,600-sized vector (x)

Image datasets

- We use **datasets** to train models to perform this task

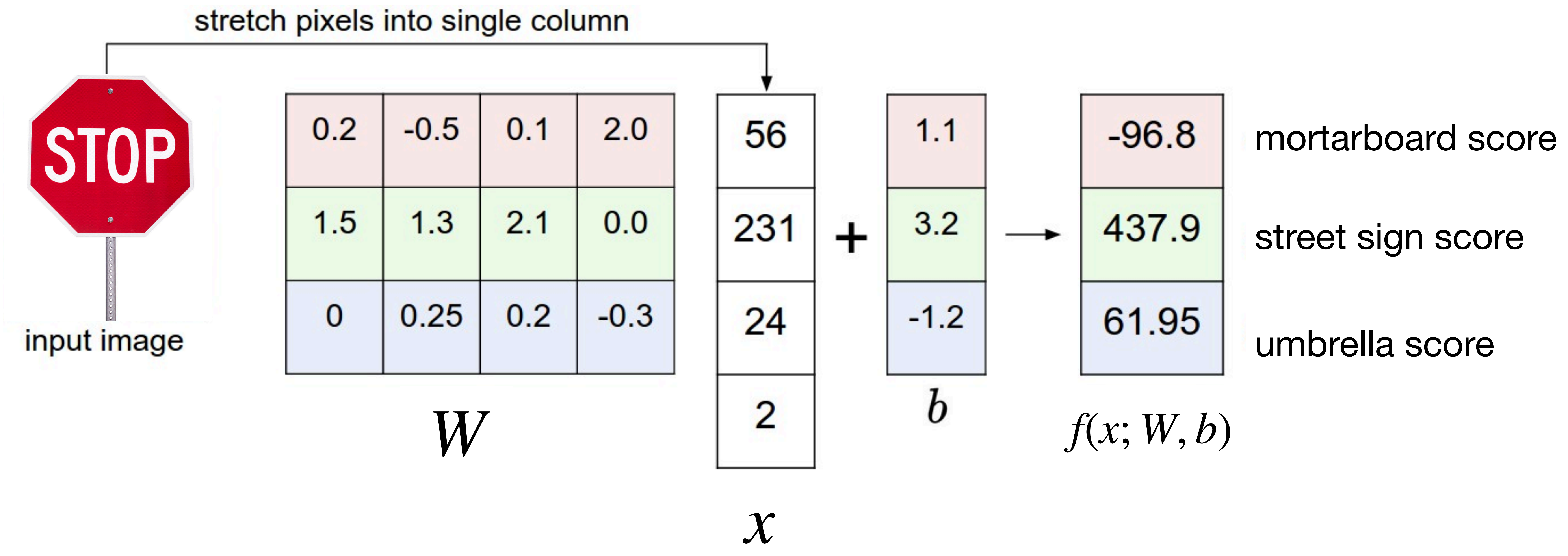


- But, our goal is to use the models to predict the labels of **unseen** data

Credits: https://cs231n.github.io

Training = Learning the model parameters

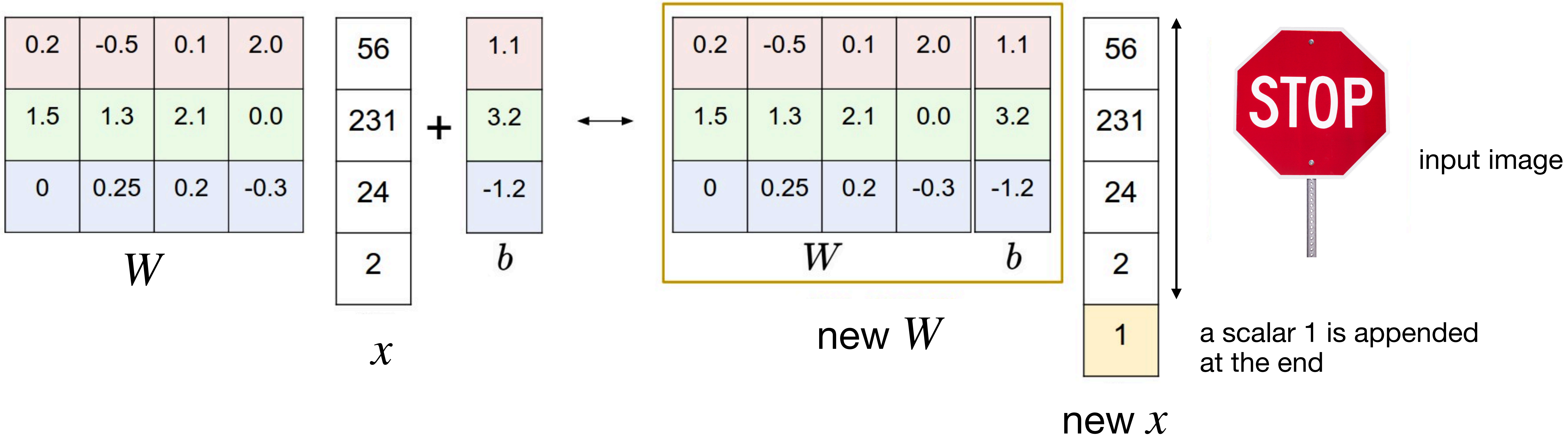
- Consider a linear model $f(x; W, b) = Wx + b$, where:
  - W, b :weight and bias parameters of the model
  - The parameters are obtained by solving optimization problem



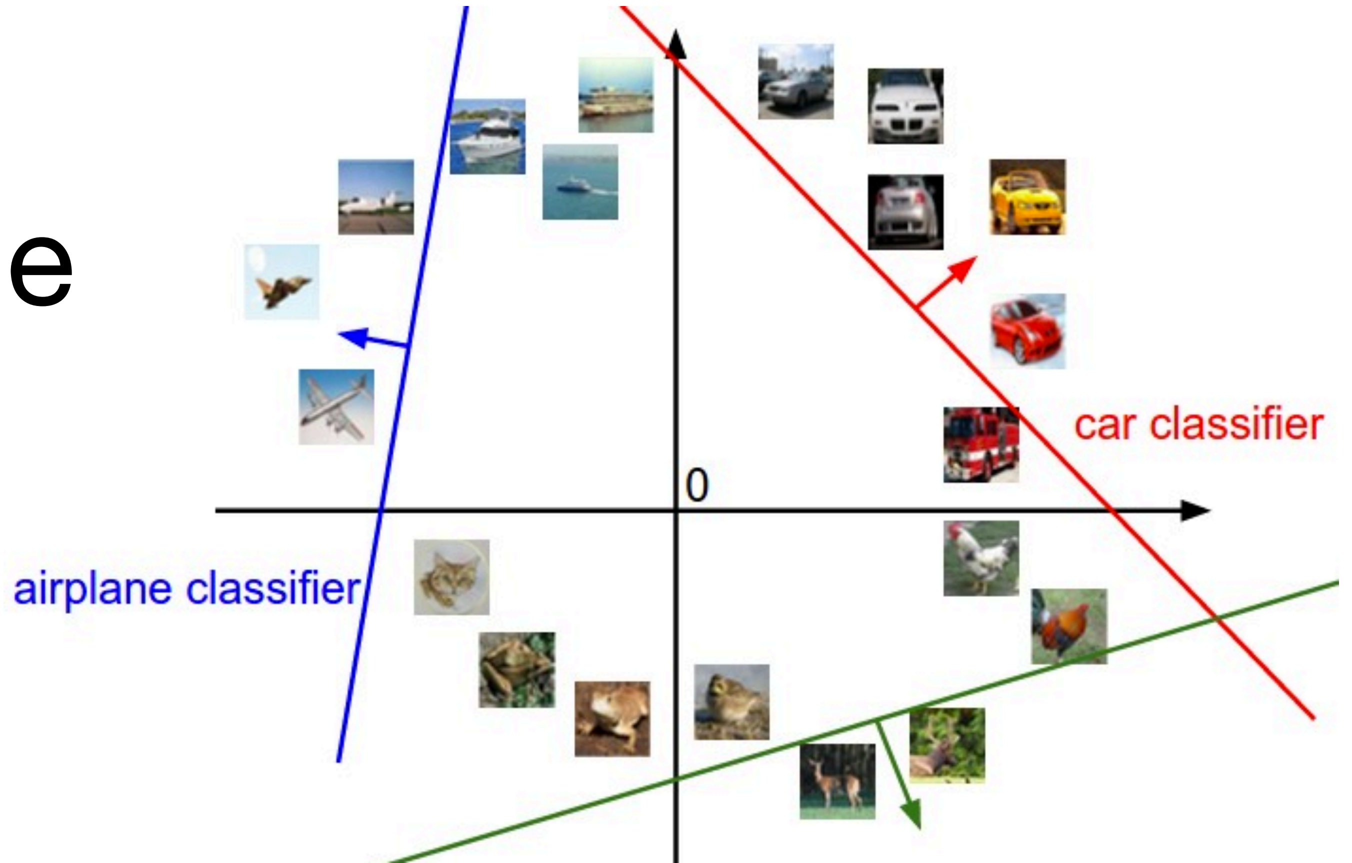
Credits: https://cs231n.github.io

Compact representation of parameters

- **Compact representation**: biases merged into weights



- The learned parameters determine the classifier boundary

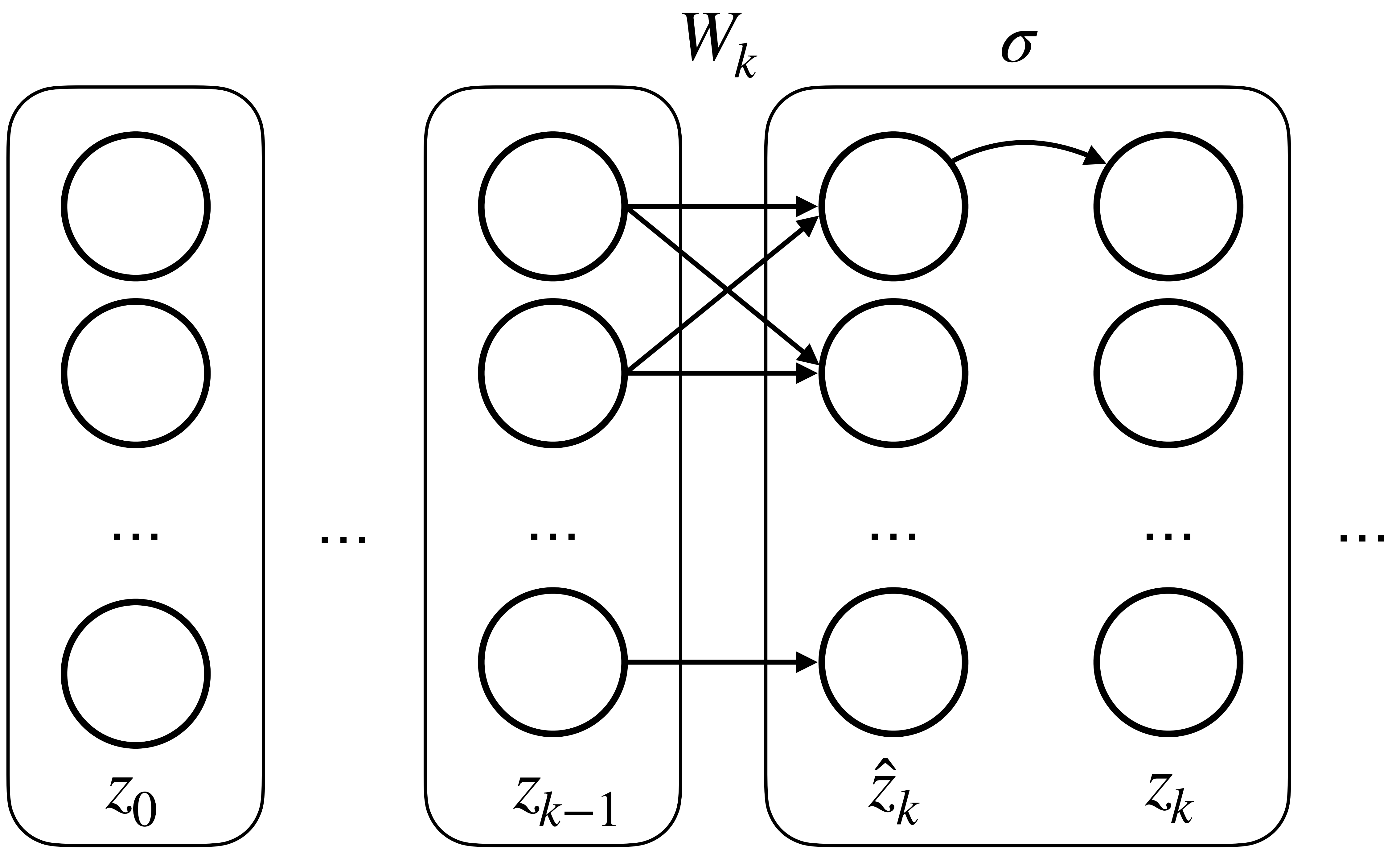


Credits: https://cs231n.github.io

Nonlinearities and deep models

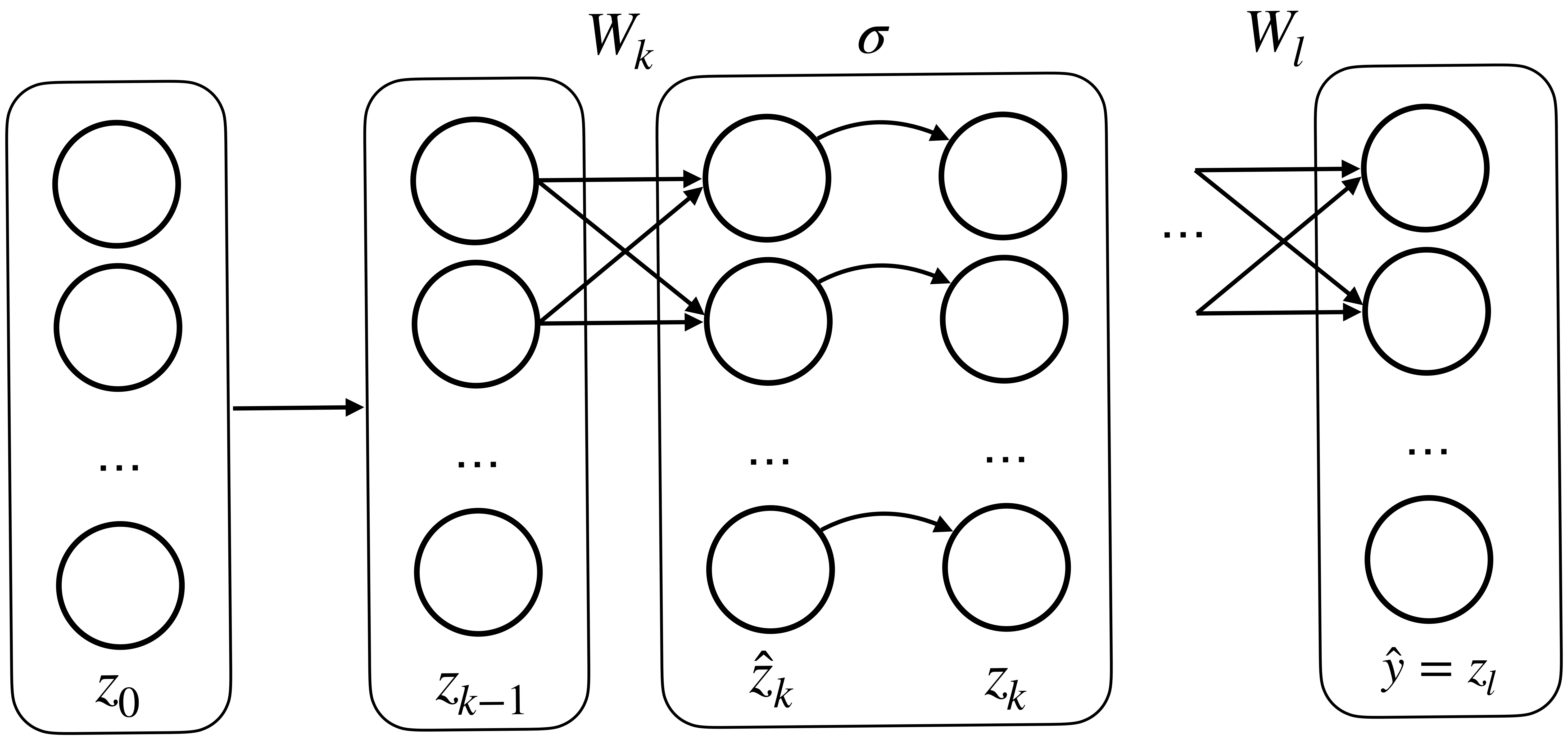
- Nonlinearities are needed by deep learning to deal with problems beyond classical machine learning methods, added by the activation function $\sigma(\cdot)$

- Example: Feedforward structure f with l layers:
    - Input: $z_0 = [x, 1]$
    - Pre-activation (logits): $\hat{z}_k = W_k z_{k-1}$
    - Post-activation: $z_k = \sigma(\hat{z}_k)$



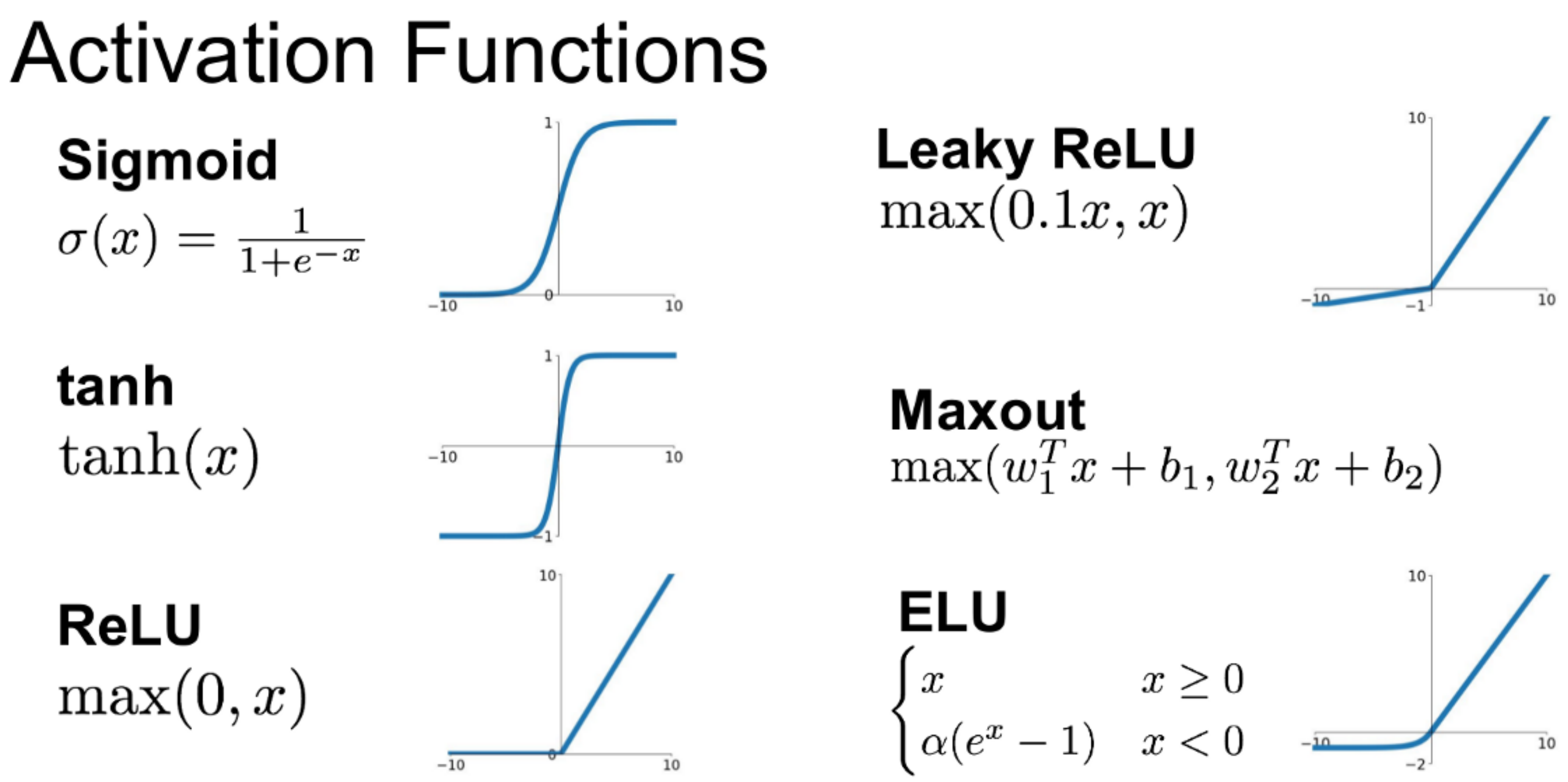
Nonlinearities and deep models

- Deep learning consists of multi-layered network with nonlinear activations



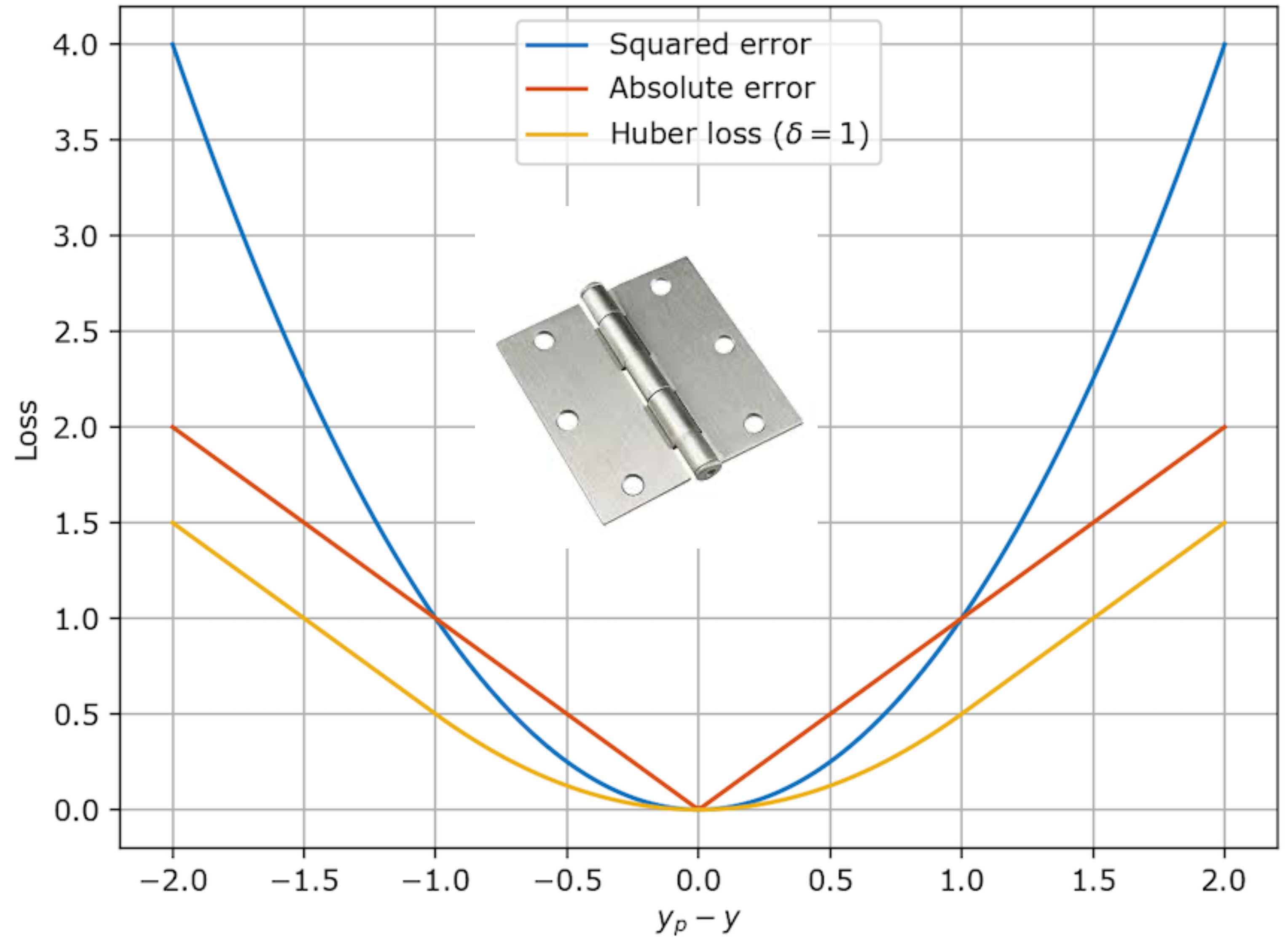
$$L(y, \hat{y}) = L(x, y; W)$$

$$\text{where } \hat{y} = f(x; W)$$



Credits: punch-us.com

Loss functions



- Loss functions for **regression**:
  - Mean Squared Error (MSE)
    $L(x, y; W) = \frac{1}{N}\sum_{i=1}^{N} (y_i - f(x_i; W))^2$
  - L1 loss or Mean Absolute Error (MAE): suitable when numerous outliers exist in the data
    $L(x, y; W) = \frac{1}{N}\sum_{i=1}^{N} |y_i - f(x_i; W)|$
  - Huber loss (with param. δ): mimics MSE (small δ) and MAE (large δ)

$$L_\delta(x, y; W) = \frac{1}{N}\sum_{i=1}^{N} L_i, L_i = \begin{cases} \frac{1}{2}(y_i - f(x_i; W))^2 & \text{for } |y_i - f(x_i; W)| \leq \delta \\ \delta |y_i - f(x_i; W)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

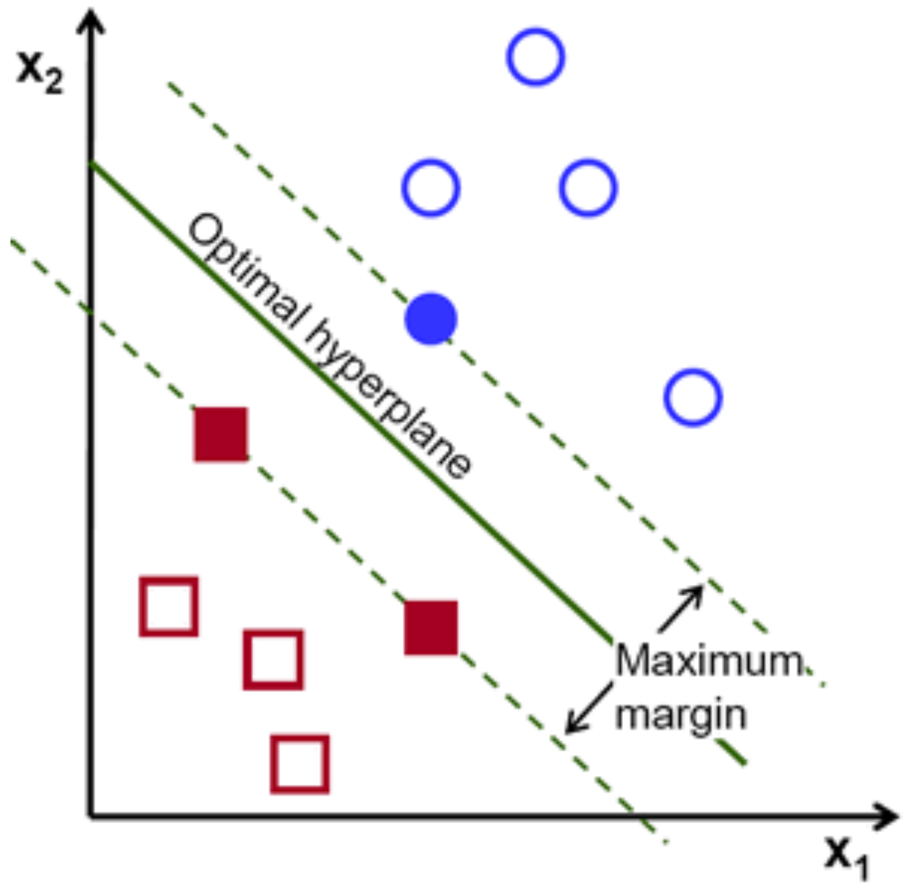
```
def Huber(f, y, delta=1.):
    return np.where(np.abs(y-f) < delta,.5*(y-f)**2 , delta*(np.abs(y-f)-0.5*delta))
```


https://www.evergreeninnovations.co/blog-machine-learning-loss-functions/

Loss functions

- Loss functions for **classification**:
  - **Cross entropy (softmax) loss:** suitable for model f whose output is a probability value between 0 and 1 (max likelihood of the correct choice)
    $L(x, y; W) = \sum_{i=1}^{N} -y_i \cdot \log f(x_i; W) - (1 - y_i) \cdot \log(1 - f(x_i; W))$
  - **Hinge loss:** penalizes both the wrong predictions and the right predictions that are closed to the margin. $y = \pm 1, f \in \mathbb{R}$. When f and y have the same sign (meaning f predicts the right class) and $|y| \geq 1$, the hinge loss is 0. When they have opposite signs, increases linearly with y, and similarly if $|y| < 1$, even if it has the same sign (correct prediction, but not by enough margin), which contra SVM)
    $L(x, y; W) = \sum_{i=1}^{N} \max\{0, 1 - f(x_i; W) \cdot y_i\}$

```
def CrossEntropy(f, y):
    if y == 1:
      return -log(f)
    else:
      return -log(1 - f)
```




```
def Hinge(f, y):
    return np.max(0, y - f*y)
```

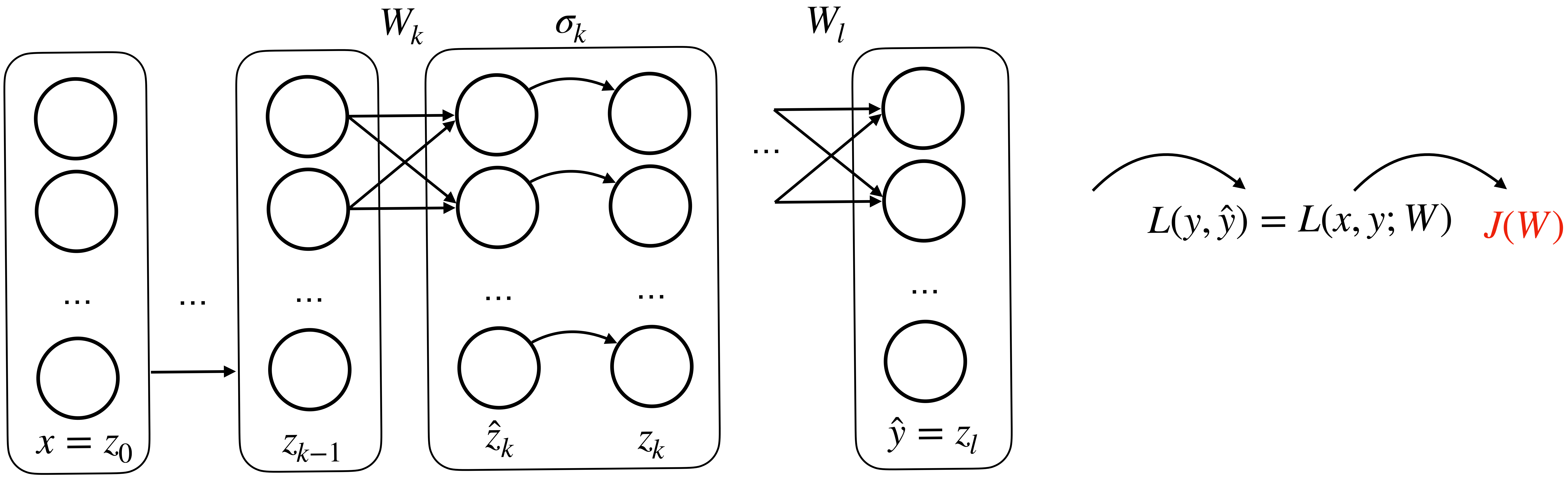
More info: lecture 2, 3
http://cs231n.stanford.edu

https://rohanvarma.me/Loss-Functions/

Training process: Forward pass

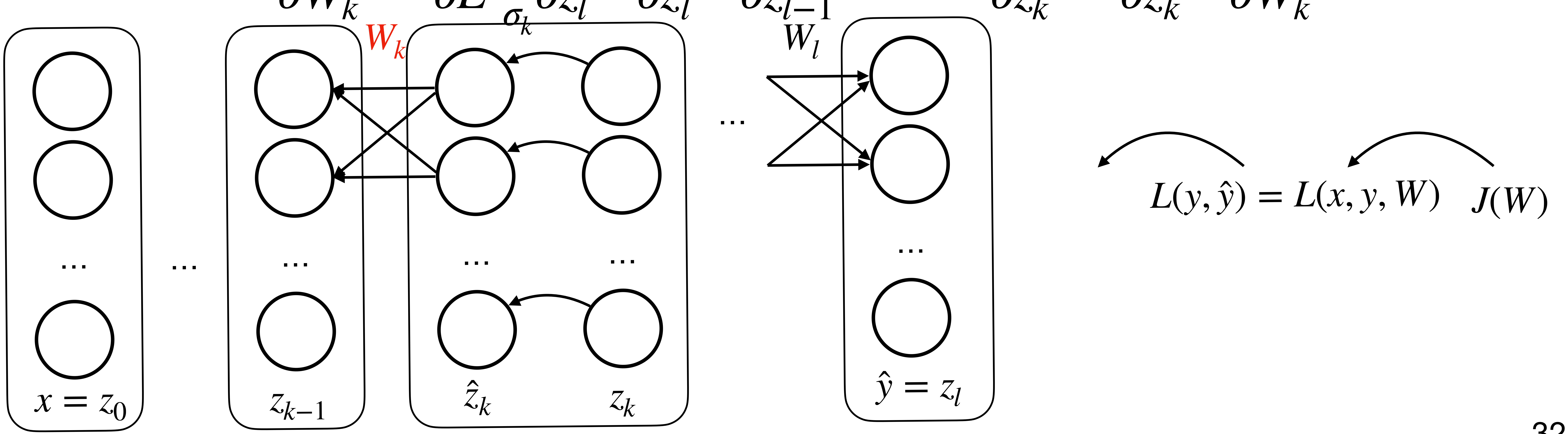
- The training minimizes the expected loss: $\min_W J(W) = \mathbb{E}_{p(x,y)}[L(x, y; W)]$

- **Forward pass**: obtaining the value of expected loss



Training process: Backpropagation

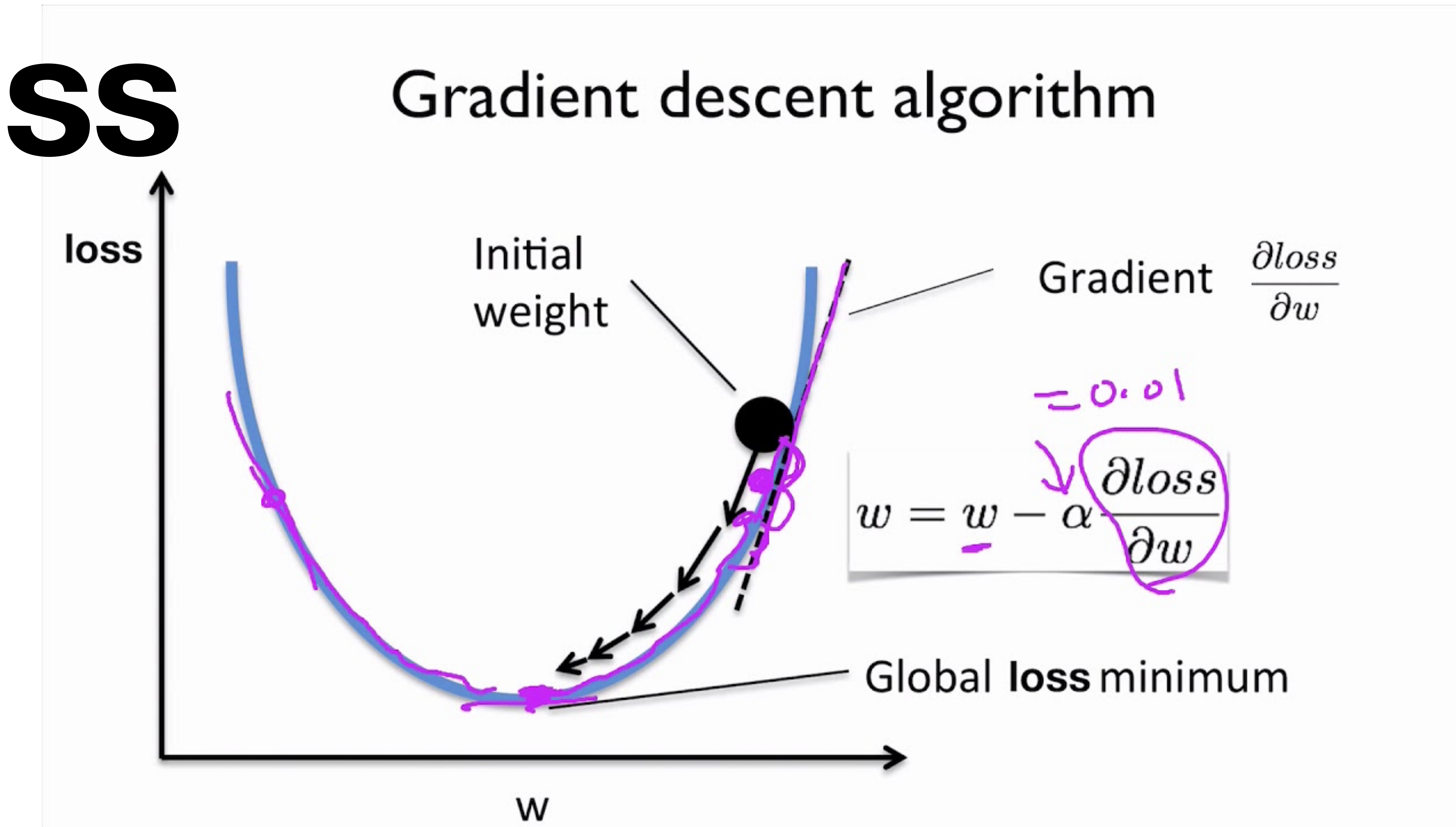
- To optimize W_k (weight at layer k), the solver needs gradient $\partial J / \partial W_k$
- **Backpropagation**: Efficiently computing gradients w.r.t. parameters
- Chain rule: $\frac{\partial J}{\partial W_k} = \frac{\partial J}{\partial L} \cdot \frac{\partial L}{\partial z_l} \cdot \frac{\partial z_l}{\partial \hat{z}_l} \cdot \frac{\partial \hat{z}_l}{\partial z_{l-1}} \cdot \ldots \cdot \frac{\partial \hat{z}_{k+1}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \hat{z}_k} \cdot \frac{\partial \hat{z}_k}{\partial W_k}$



Regularization

- The problem involves large space $W \in \mathscr{W}$ so we wish to obtain "well-behaved" solution
  - attained by adding penalty term $R(W)$ to the training objective
  - discourages learning too complex model, to avoid overfitting
- Some widely used regularization functions. Let W be a vector
  - L1 regularizer: $\|W\|_1 = \sum_i |W_i|$
  - L2 regularizer: $\|W\|_2^2 = \sum_i W_i^2$
  - Lp regularizer: $\|W\|_p^p = \sum_{i=1}^n |W_i|^p$
  - L∞ regularizer: $\|W\|_\infty = \max(|W_1|, \ldots, |W_n|)$
- Popular loss functions:
  - Ridge: $\sum_{i=1}^N (y_i - f(x_i; W))^2 + \lambda \|W\|_2^2$
  - Lasso (Least absolute **shrinkage** and selection operator): $\sum_{i=1}^N (y_i - f(x_i; W))^2 + \lambda \|W\|_1$

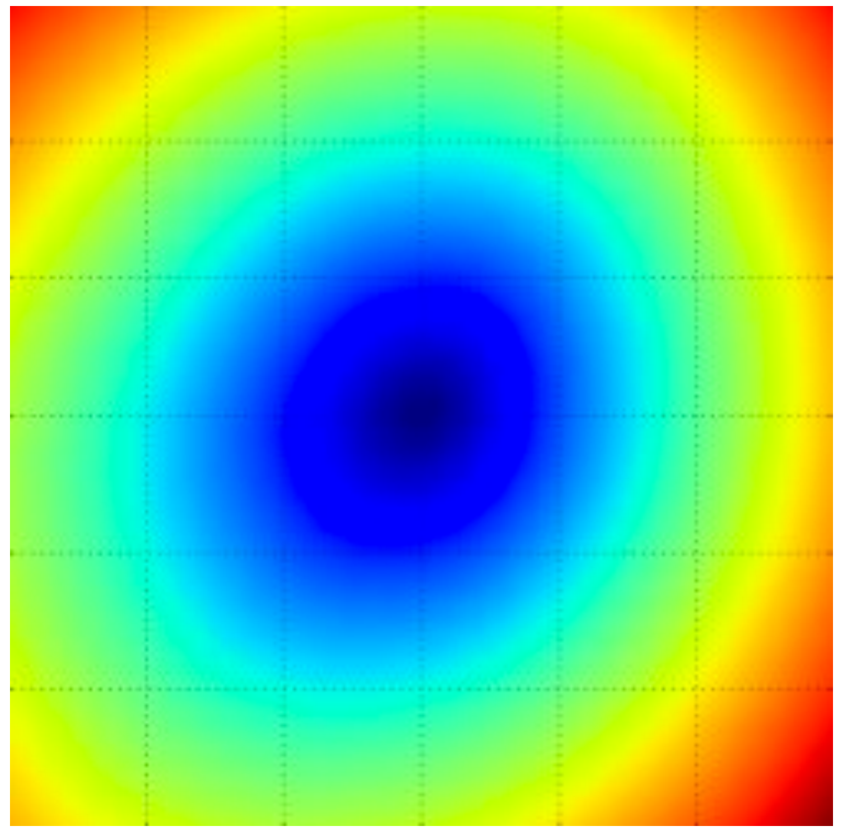
Minimizing the regularized loss



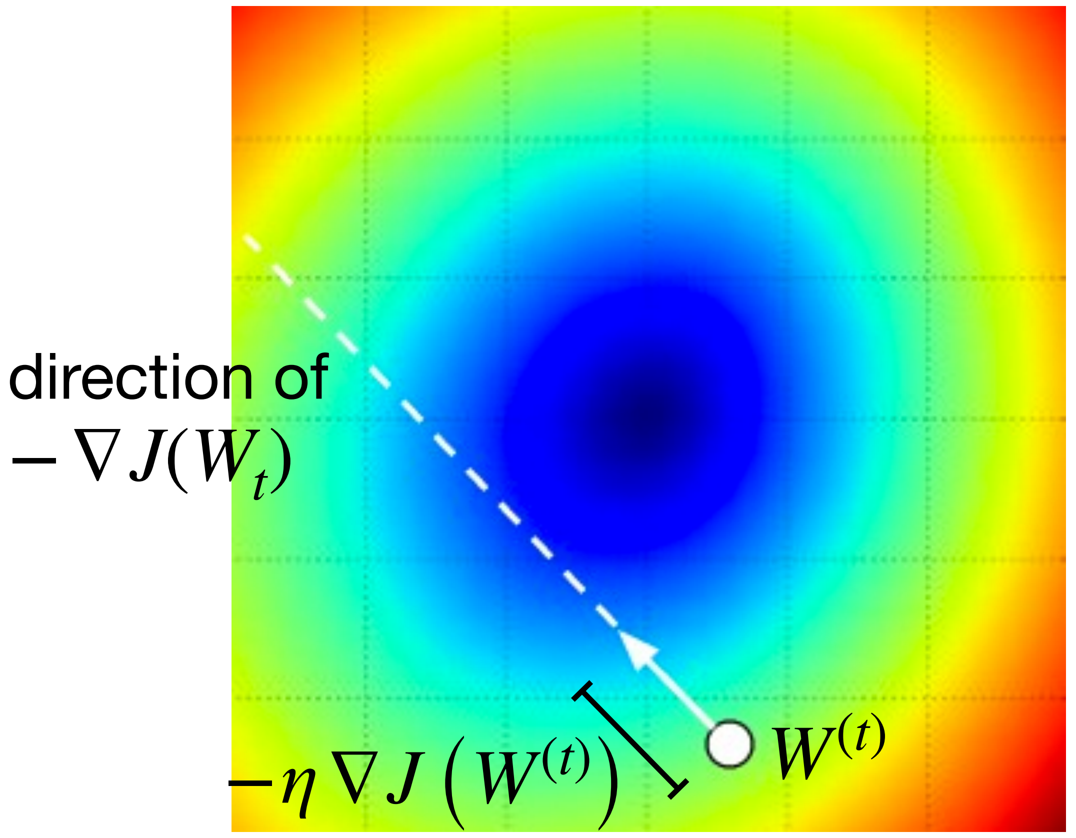
- $\min_W J(W) = \mathbb{E}\left[L(x, y; W)\right] + \lambda R(W)$
- In practice, use (stochastic) samples:

$$\min_W \hat{J}(W) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i; W) + \lambda R(W)$$

- **Gradient descent**: At iteration t, updates $W^{(t+1)} = W^{(t)} - \eta \nabla J\left(W^{(t)}\right)$ where η is the learning rate.
- ∇J is often approximated via sample $\hat{\nabla} J$, *i.e.,* Stochastic Gradient Descent (SGD)



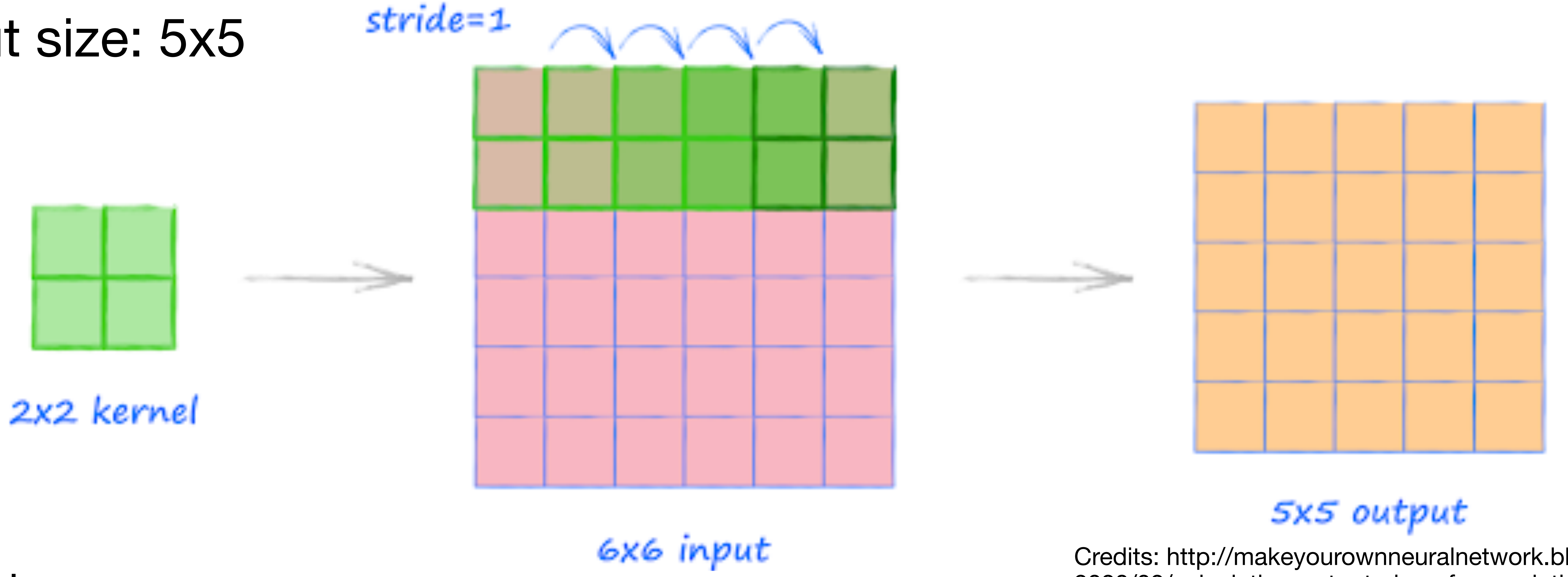
$J(W)$



Credits: https://cs231n.github.io
https://www.youtube.com/watch?v=b4Vyma9wPHo

Convolution operations

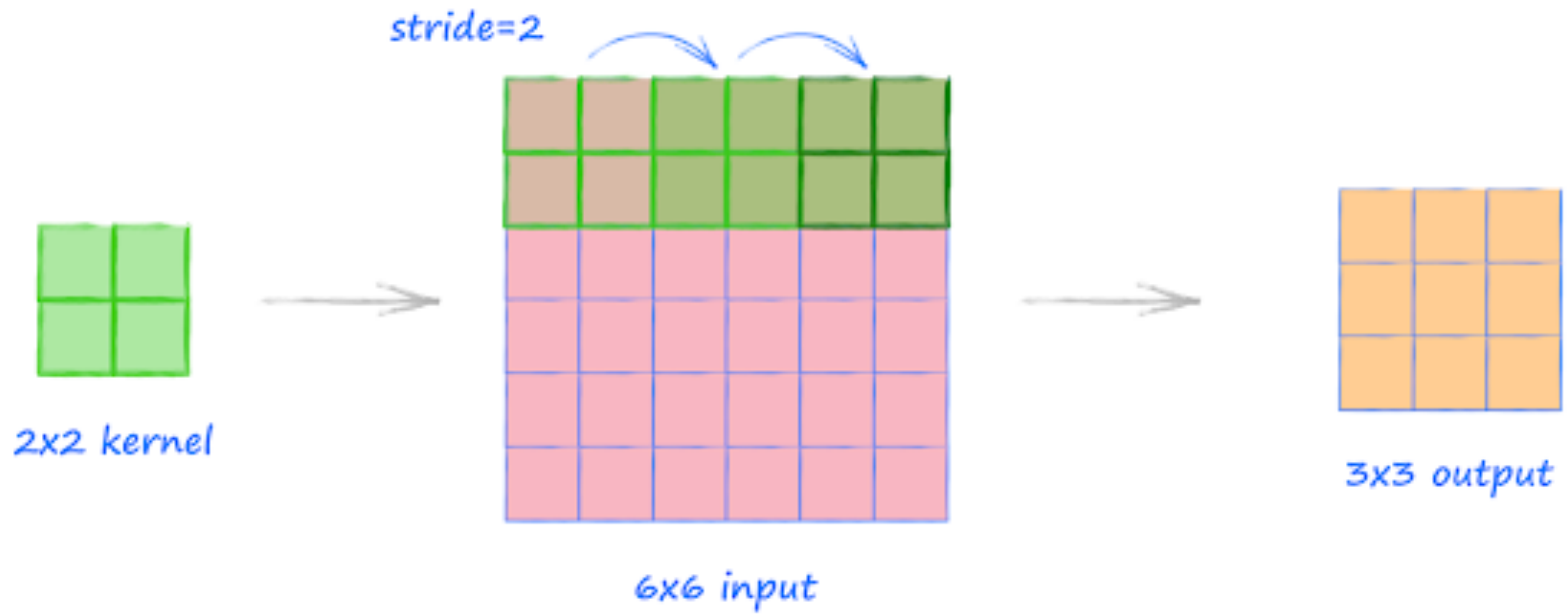
- Convolution is the sliding of a kernel over an input data (multiply the corresponding numbers and sum the results up)
- The sliding scale is called stride.
- Example: 2x2 kernel applied to 6x6 input with stride 1
- Output size: 5x5



Credits: http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html

Convolution operations

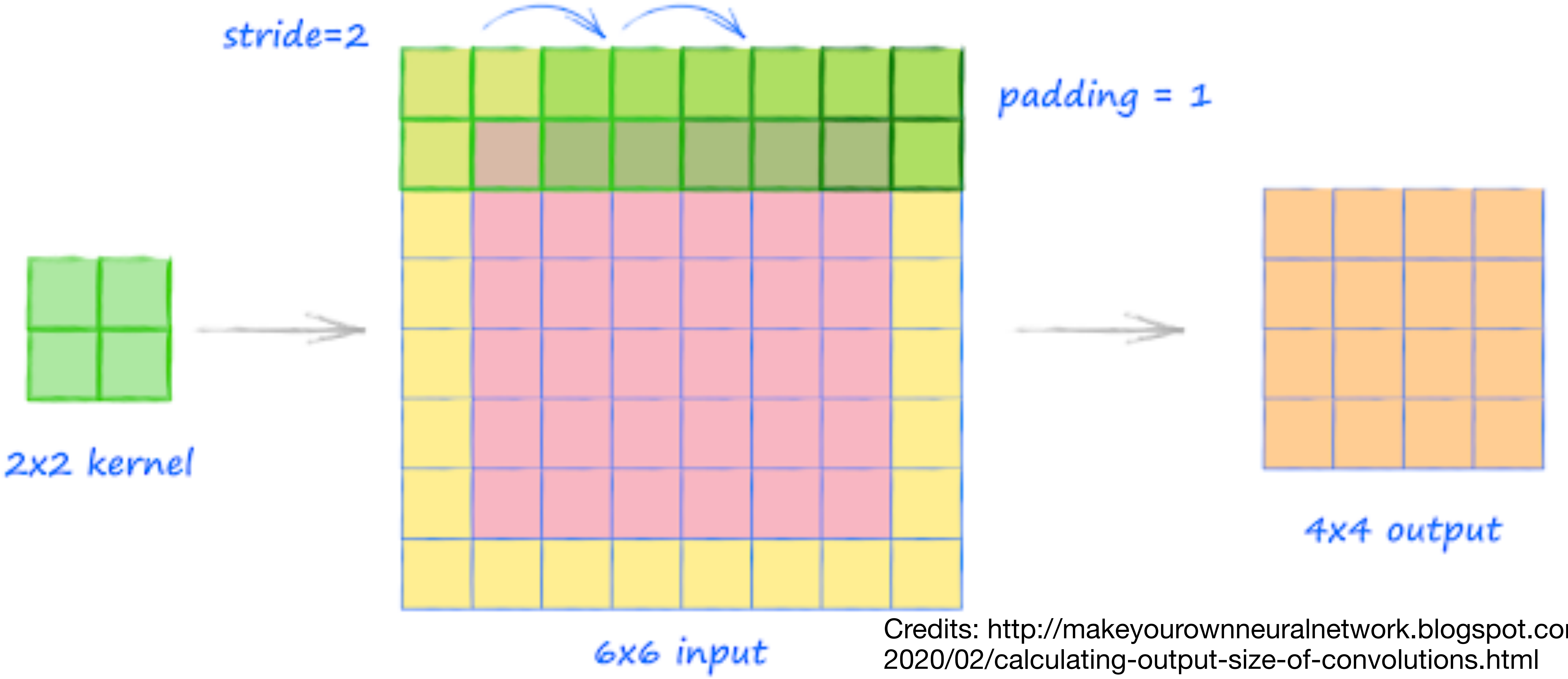
- Example: 2x2 kernel applied to 6x6 input with stride 2
- Output size: 3x3



Credits: http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html

Convolution operations

- We can control the output size by adding padding
- Example: 2x2 kernel applied to 6x6 input with stride 2 and padding 1
- Output size: 4x4



Credits: http://makeyourownneuralnetwork.blogspot.com/2020/02/calculating-output-size-of-convolutions.html

Convolution operations

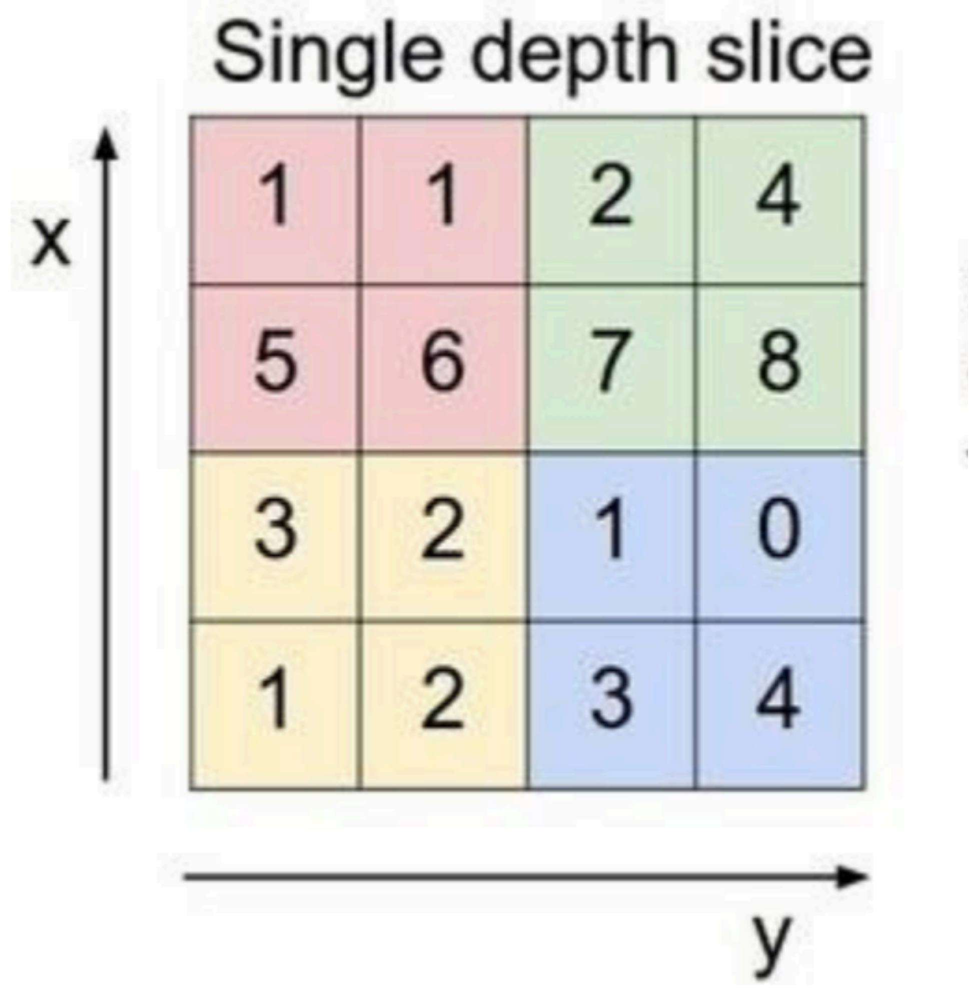
- Determining output size

$$\text{output size} = \left\lfloor \frac{(\text{input size}) + 2*\text{padding} - (\text{kernel size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

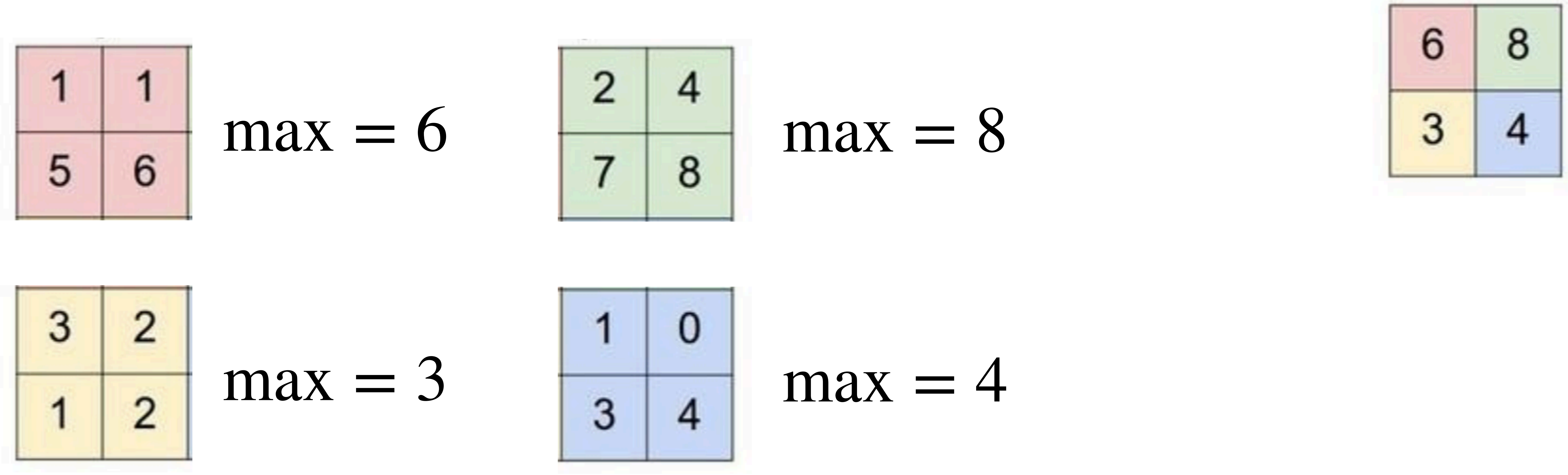
- Example: 2x2 kernel applied to 6x6 input with stride 2 and padding 1
  Output size = $n' = \left\lfloor \dfrac{6 + 2(1) - (2-1) - 1}{2} + 1 \right\rfloor = \left\lfloor \dfrac{6}{2} + 1 \right\rfloor = 4$

Credits: CMU 10701

Pooling



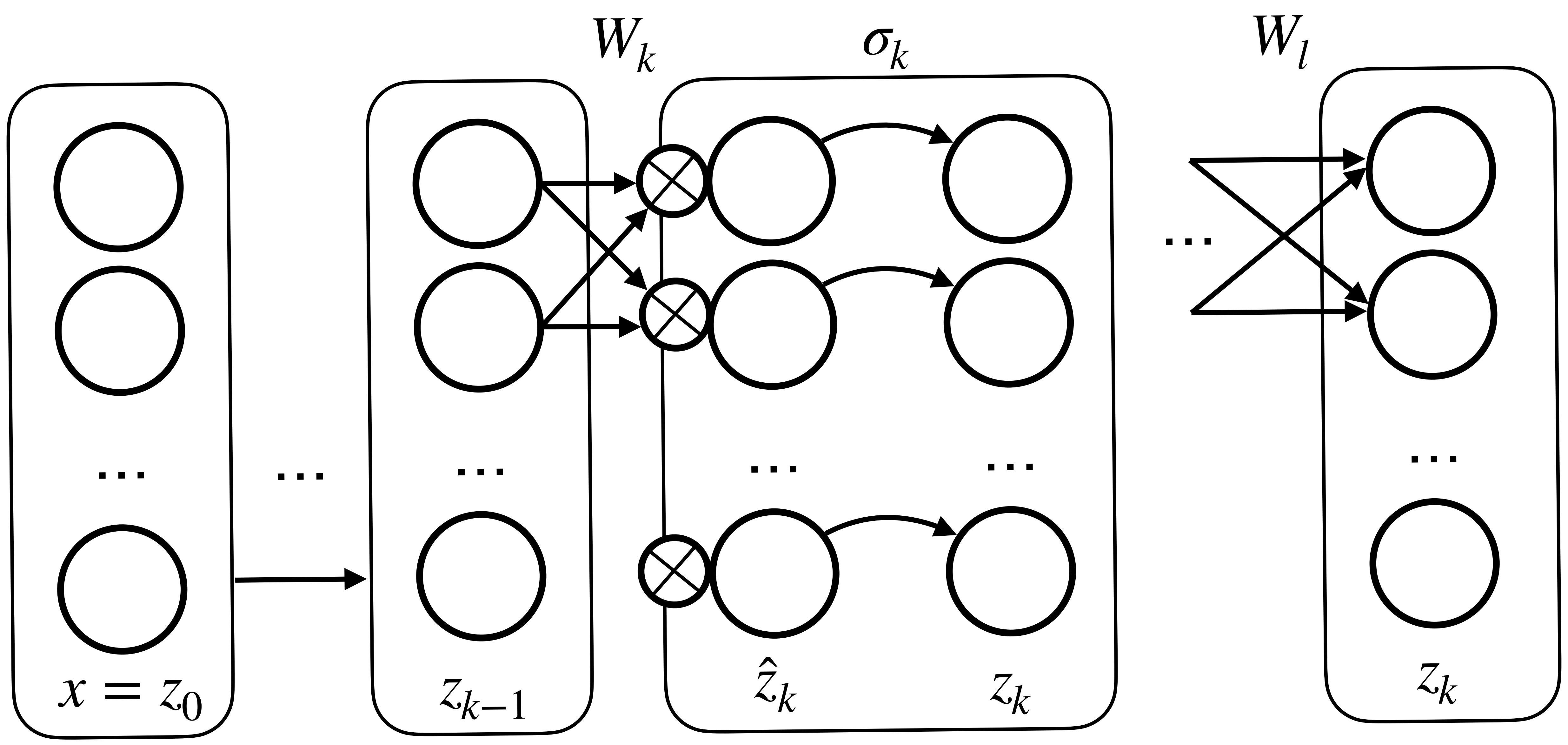
Max pooling with filter 2x2 and stride 2



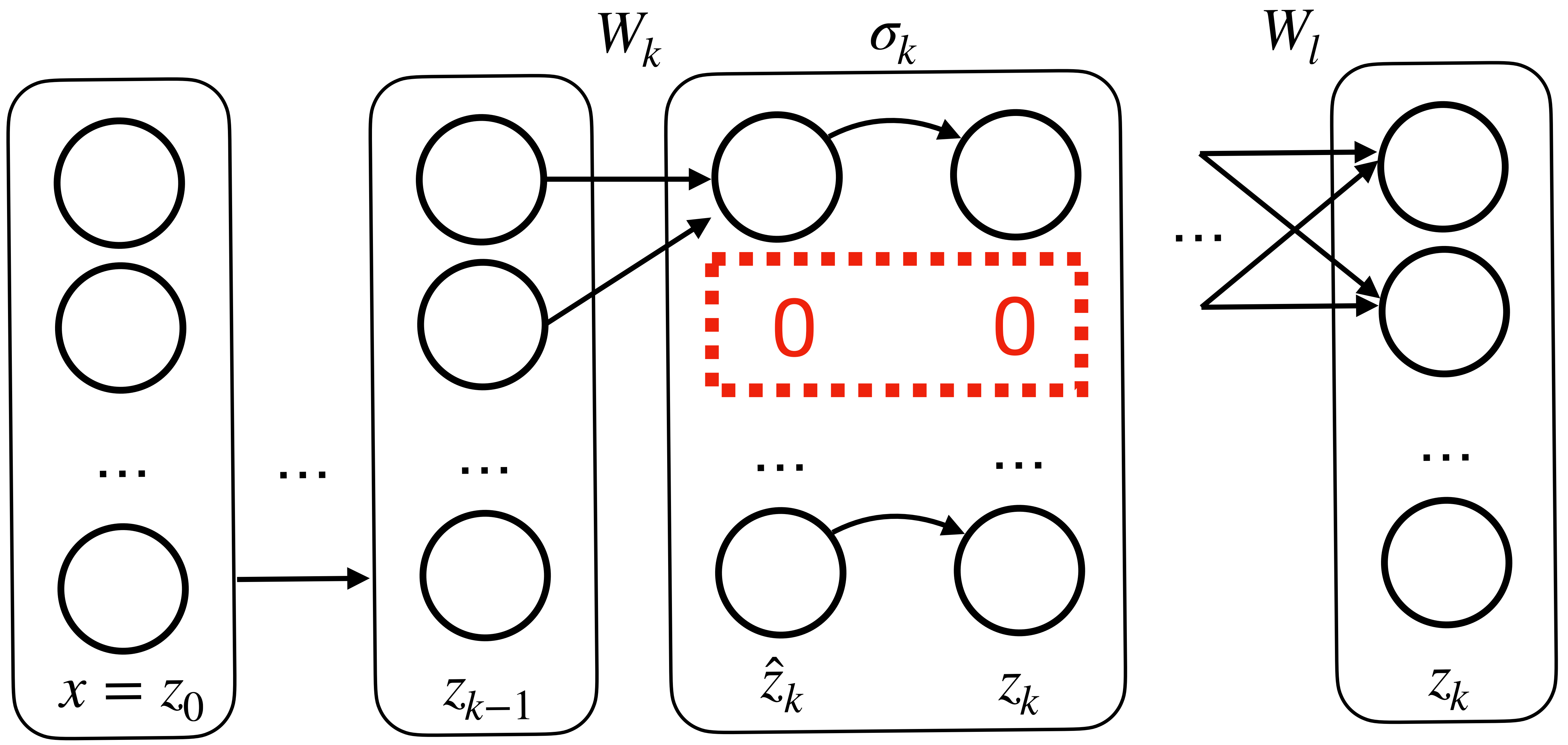
Credits: CMU 10701, Baeldung

Dropout

Original model



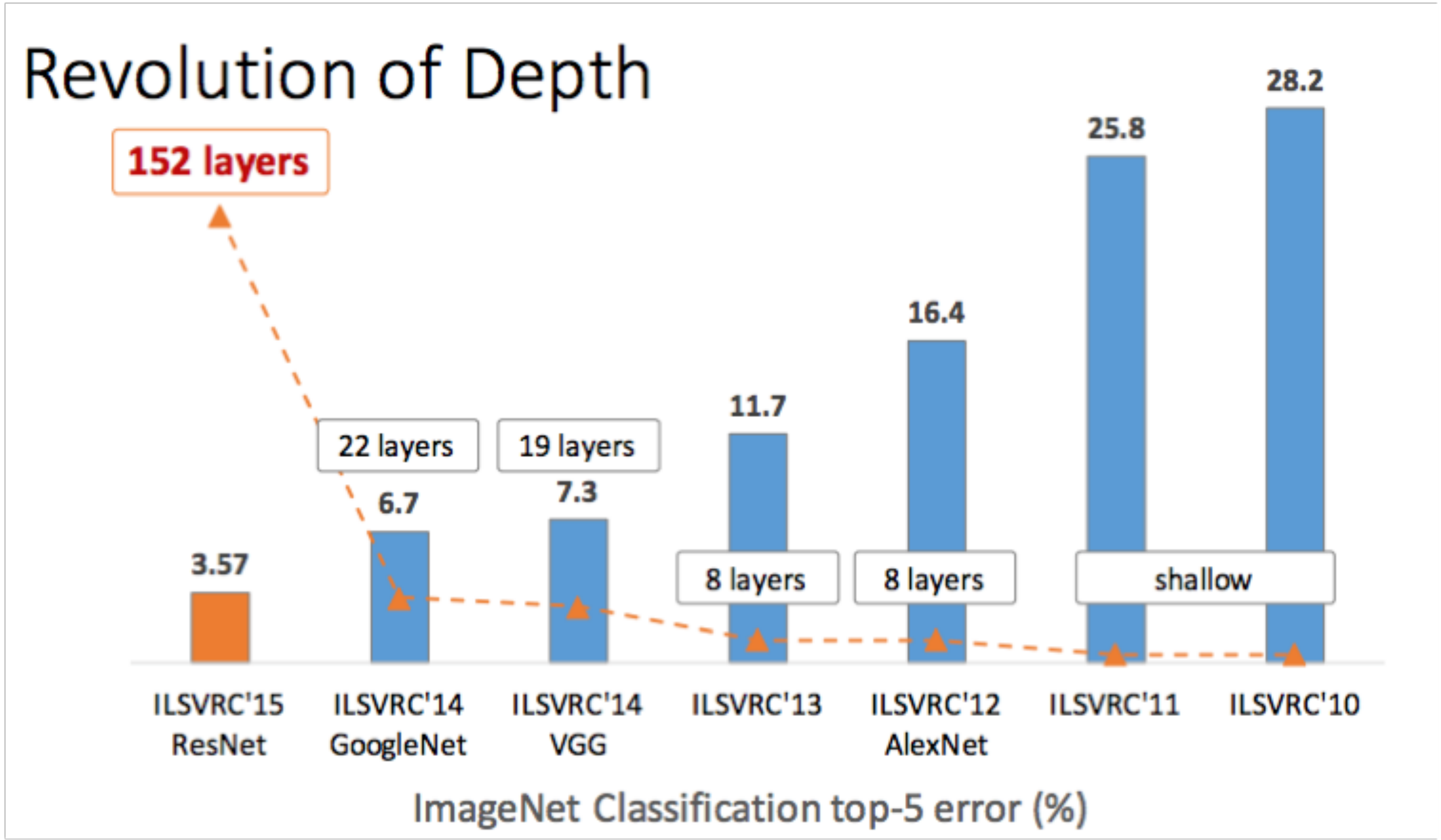
Dropout



Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; JMLR, 15(56):1929–1958, 2014.

Design deep learning structures

- AlexNet
- VGGNet
- ResNet
- Inception (GoogLeNet)



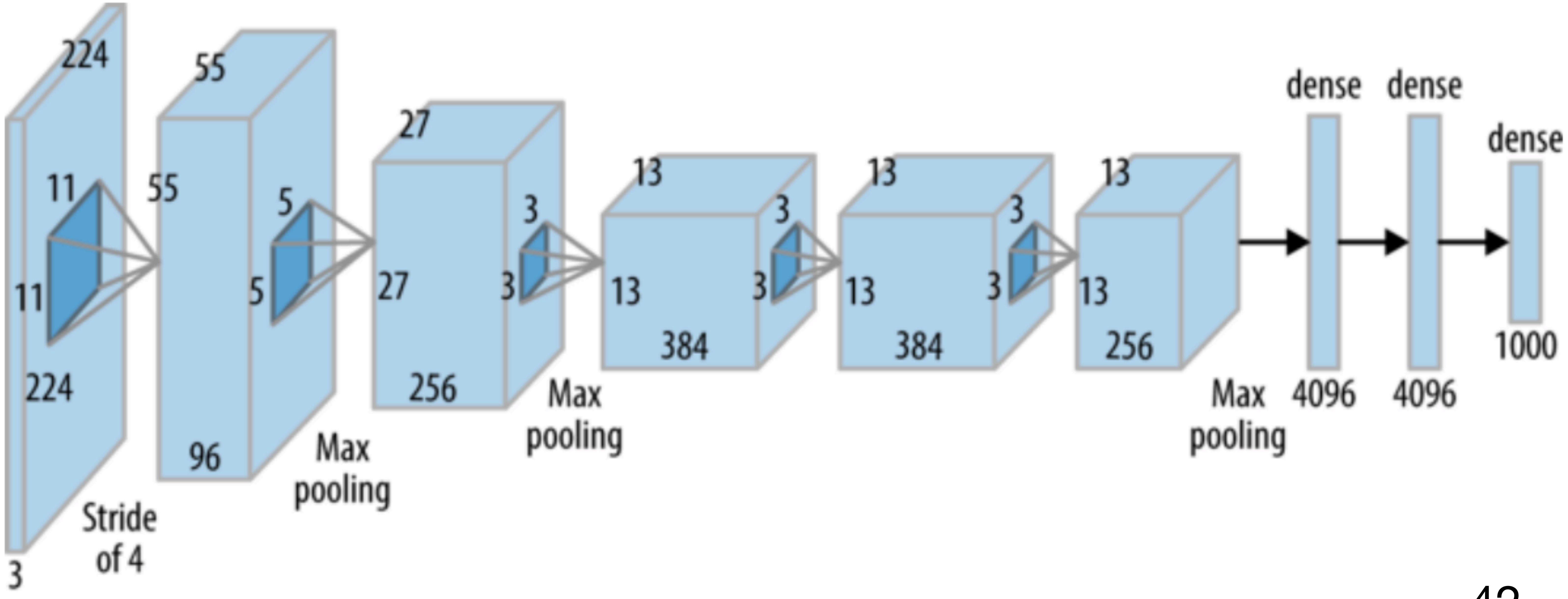
https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

AlexNet

- This was one of the first Deep convolutional networks to achieve considerable accuracy on the 2012 ImageNet challenge

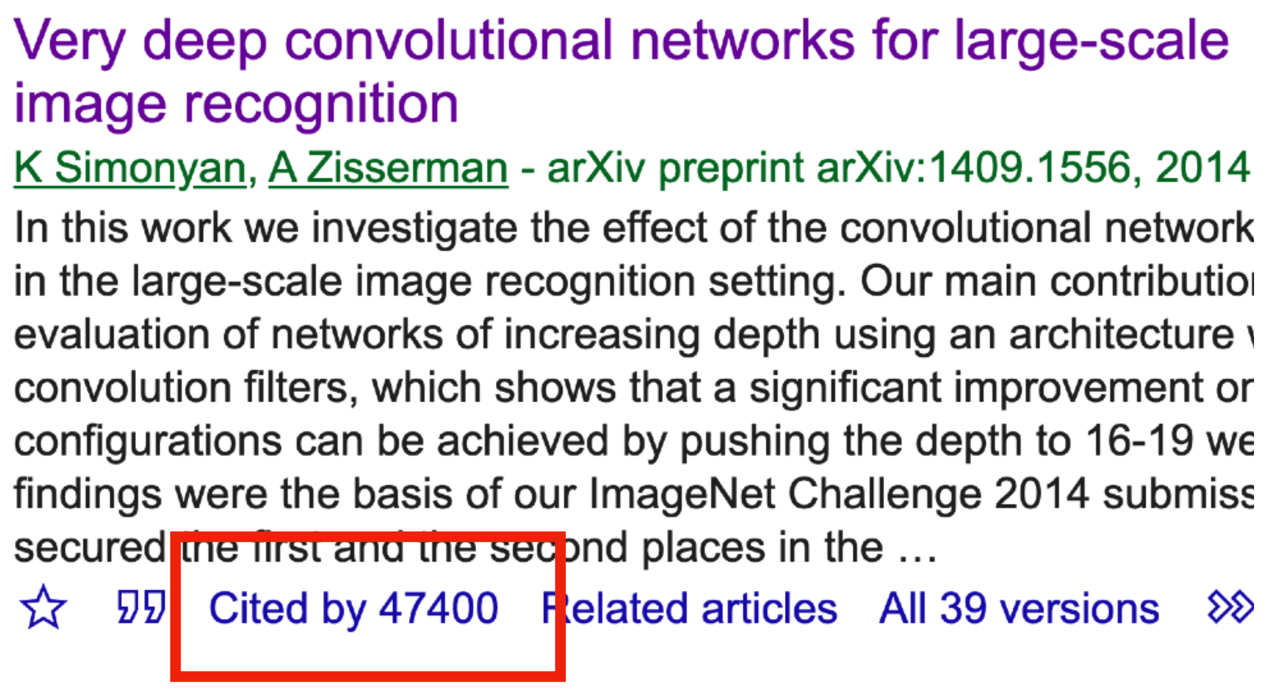
- With an accuracy of 84.7% as compared to the second-best with an accuracy of 73.8%.

AlexNet Network - Structural Details														
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
						Total							**62,378,344**

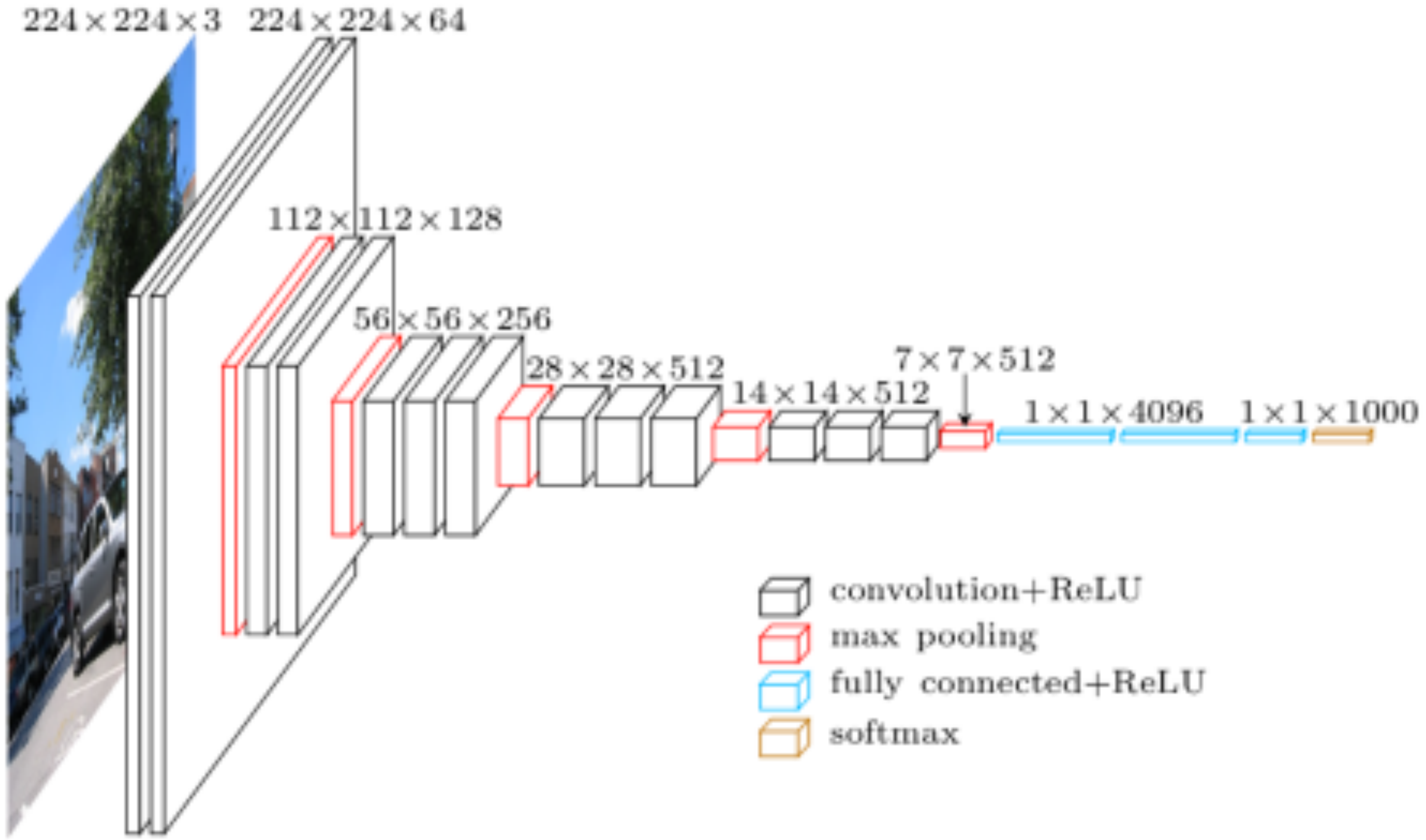


https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

VGGNet



- Multiple variants of VGGNet (VGG16, VGG19, etc.) which differ only in the total number of layers in the network.

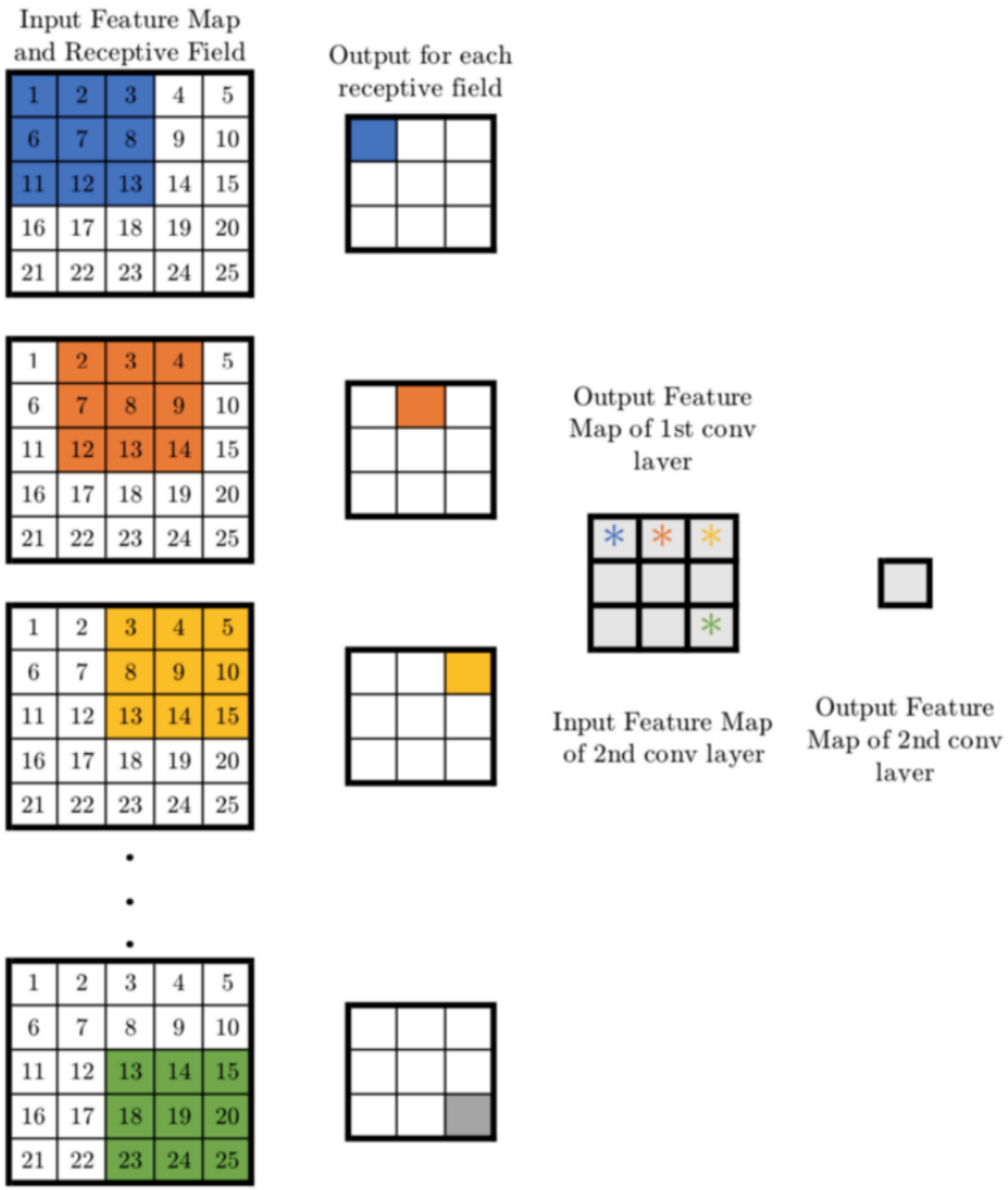


VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
							Total						138,423,208

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

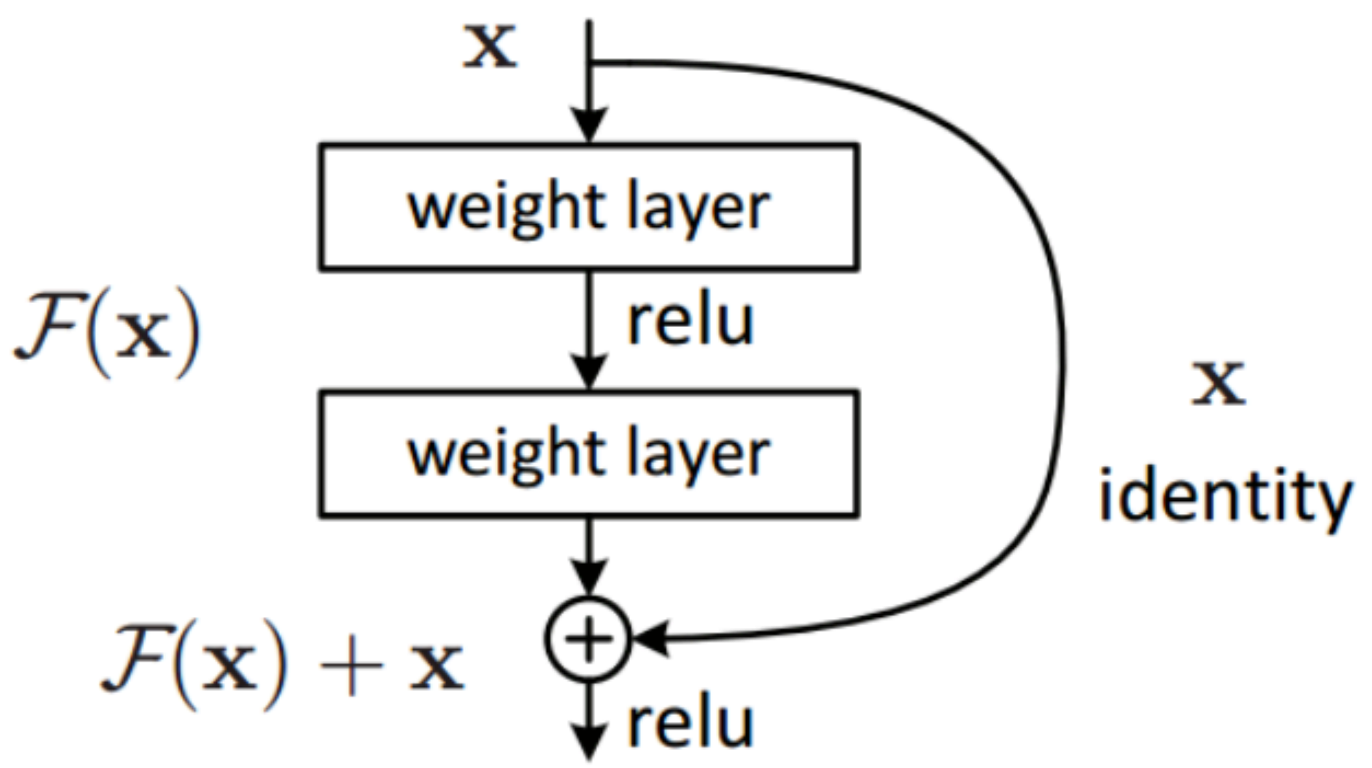
VGGNet

- VGGNet was born out of the need to reduce the # of parameters in the CONV layers and improve on training time.
- How?
  - All the variable size convolutional kernels used in Alexnet (11x11, 5x5, 3x3) can be replicated by making use of multiple 3x3 kernels as building blocks
  - For a 5x5 conv layer filter, the number of variables is 25. However, two conv layers of kernel size 3x3 have a total of 3x3x2=18 variables (a reduction of 28%).

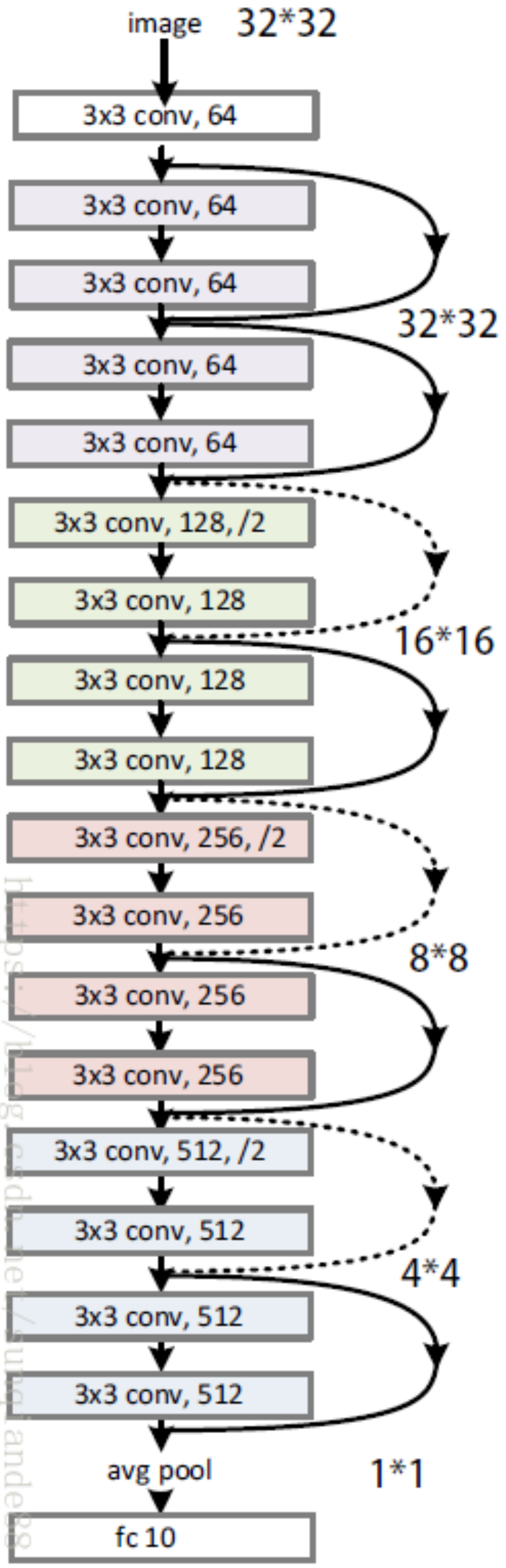


https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

ResNet



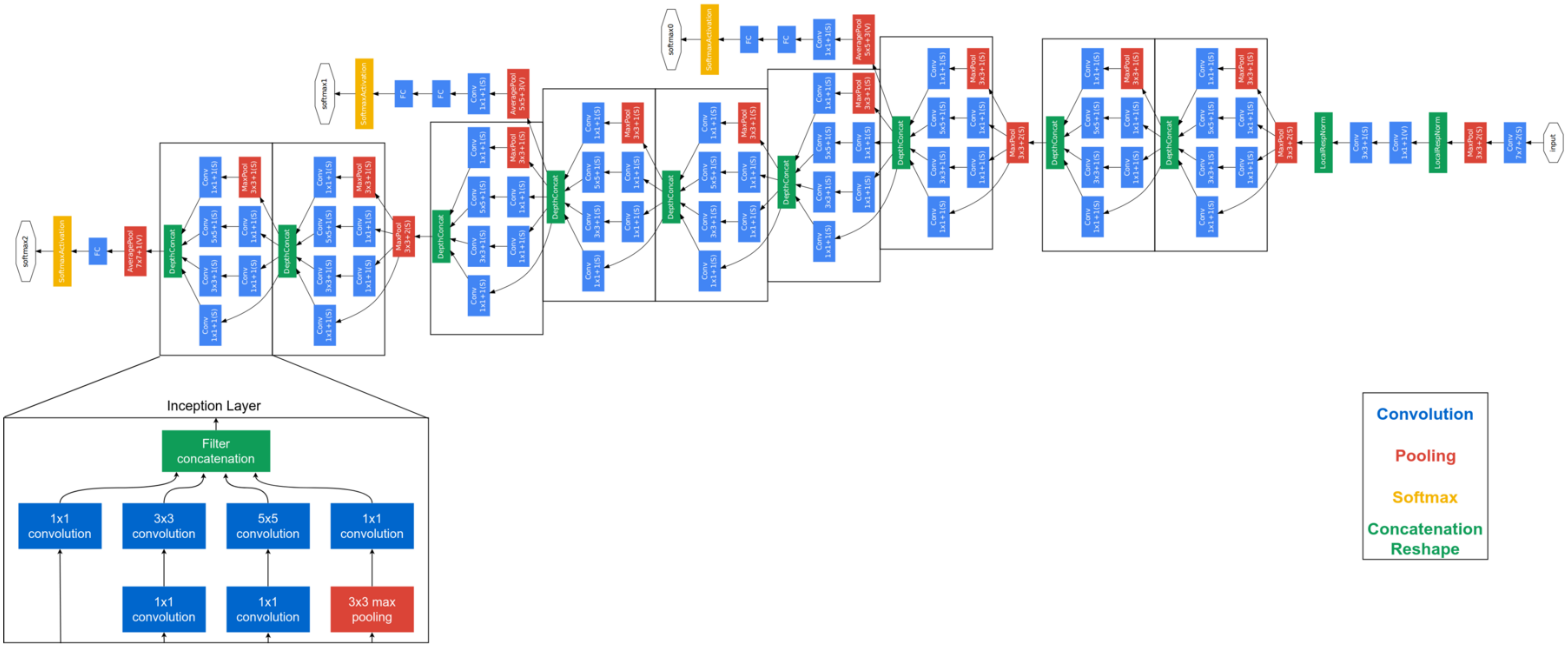
- Vanishing gradient problem
  - As we make the CNN deeper, the derivative when back-propagating to the initial layers becomes almost insignificant in value.
  - ResNet addresses this network by introducing 'shortcut connections'
  - Multiple versions of ResNetXX architectures where 'XX' denotes the number of layers. The most commonly used ones are ResNet50 and ResNet101. CNN started to get deeper and deeper, since the vanishing gradient problem was taken care of.



https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

ResNet18 - Structural Details														
#	**Input Image**			**output**			**Layer**	**Stride**	**Pad**	**Kernel**		**in**	**out**	**Param**
1	227	227	3	112	112	64	conv1	2	1	7	7	3	64	9472
	112	112	64	56	56	64	maxpool	2	0.5	3	3	64	64	0
2	56	56	64	56	56	64	conv2-1	1	1	3	3	64	64	36928
3	56	56	64	56	56	64	conv2-2	1	1	3	3	64	64	36928
4	56	56	64	56	56	64	conv2-3	1	1	3	3	64	64	36928
5	56	56	64	56	56	64	conv2-4	1	1	3	3	64	64	36928
6	56	56	64	28	28	128	conv3-1	2	0.5	3	3	64	128	73856
7	28	28	128	28	28	128	conv3-2	1	1	3	3	128	128	147584
8	28	28	128	28	28	128	conv3-3	1	1	3	3	128	128	147584
9	28	28	128	28	28	128	conv3-4	1	1	3	3	128	128	147584
10	28	28	128	14	14	256	conv4-1	2	0.5	3	3	128	256	295168
11	14	14	256	14	14	256	conv4-2	1	1	3	3	256	256	590080
12	14	14	256	14	14	256	conv4-3	1	1	3	3	256	256	590080
13	14	14	256	14	14	256	conv4-4	1	1	3	3	256	256	590080
14	14	14	256	7	7	512	conv5-1	2	0.5	3	3	256	512	1180160
15	7	7	512	7	7	512	conv5-2	1	1	3	3	512	512	2359808
16	7	7	512	7	7	512	conv5-3	1	1	3	3	512	512	2359808
17	7	7	512	7	7	512	conv5-4	1	1	3	3	512	512	2359808
	7	7	512	1	1	512	avg pool	7	0	7	7	512	512	0
18	1	1	512	1	1	1000	fc					512	1000	513000
							Total							**11,511,784**

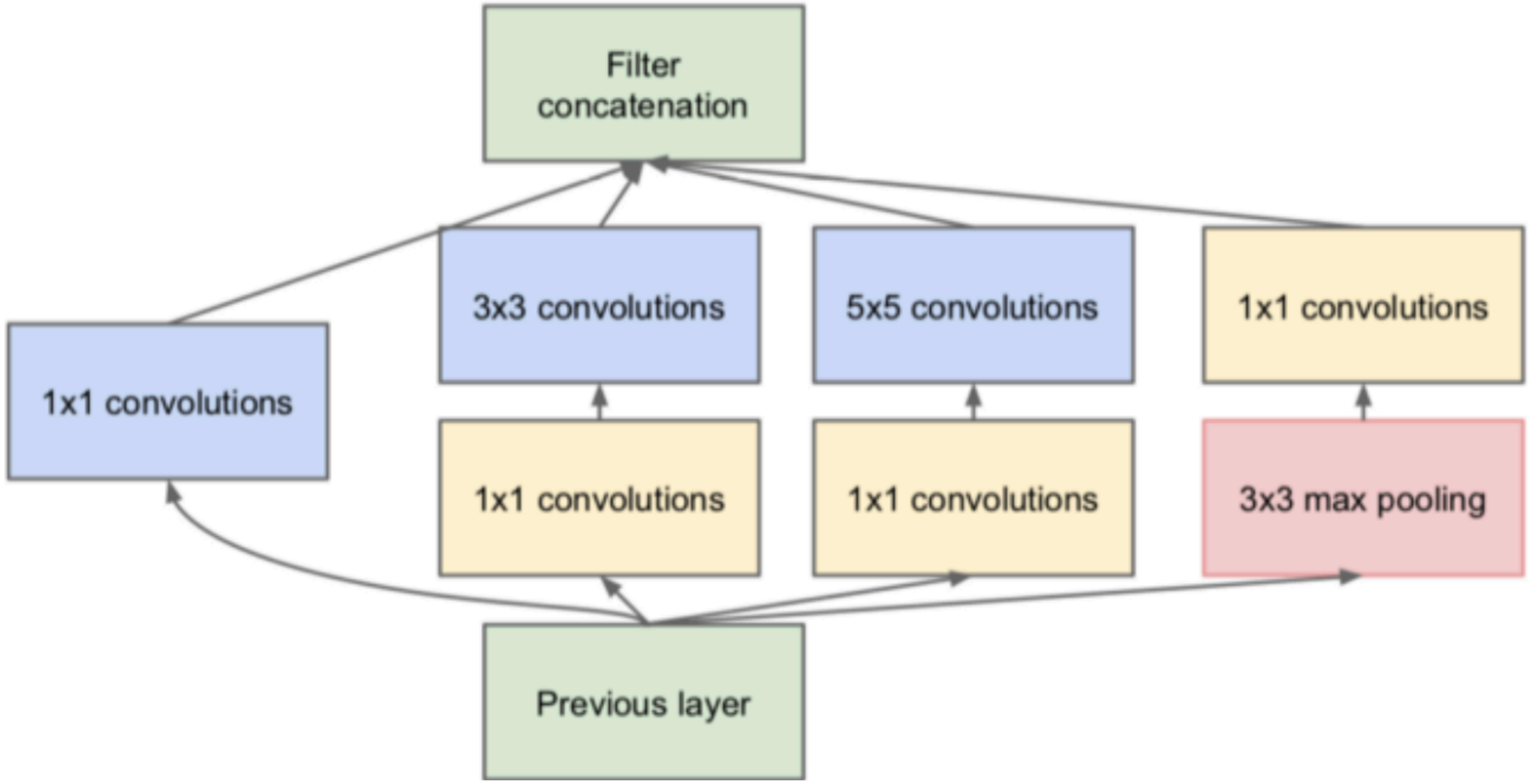
Inception/GoogLeNet (Inception v-1)



Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Inception/GoogLeNet (Inception v-1)

- Deciding on a fixed kernel size is rather difficult.
  - Lager kernels are preferred for more global features that are distributed over a large area of the image
  - Smaller kernels provide good results in detecting area-specific features that are distributed across the image frame.
- For effective recognition of such a variable-sized feature, we need kernels of different sizes.
  - Instead of simply going deeper in terms of the number of layers, it goes wider. Multiple kernels of different sizes are implemented within the same layer.



The 1x1 conv blocks shown in yellow are used for depth reduction

https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

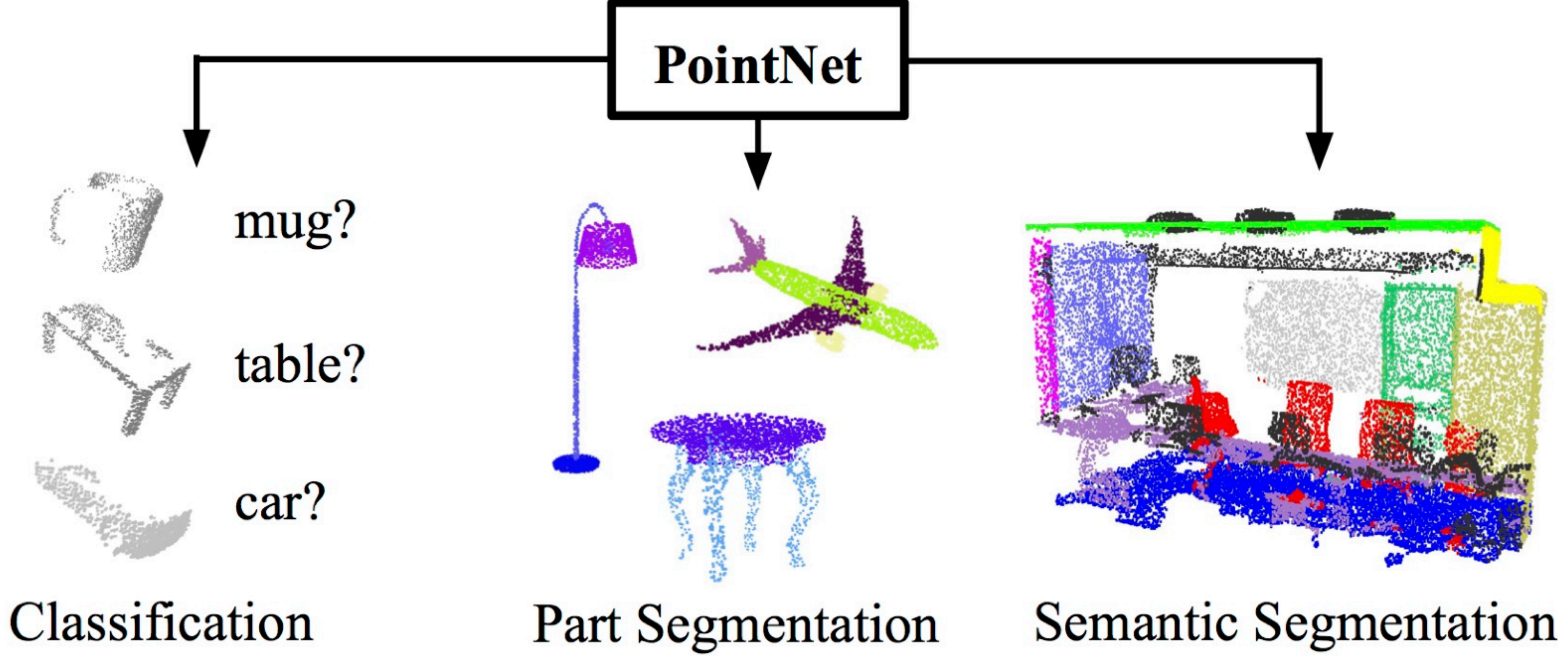
Design deep learning structures

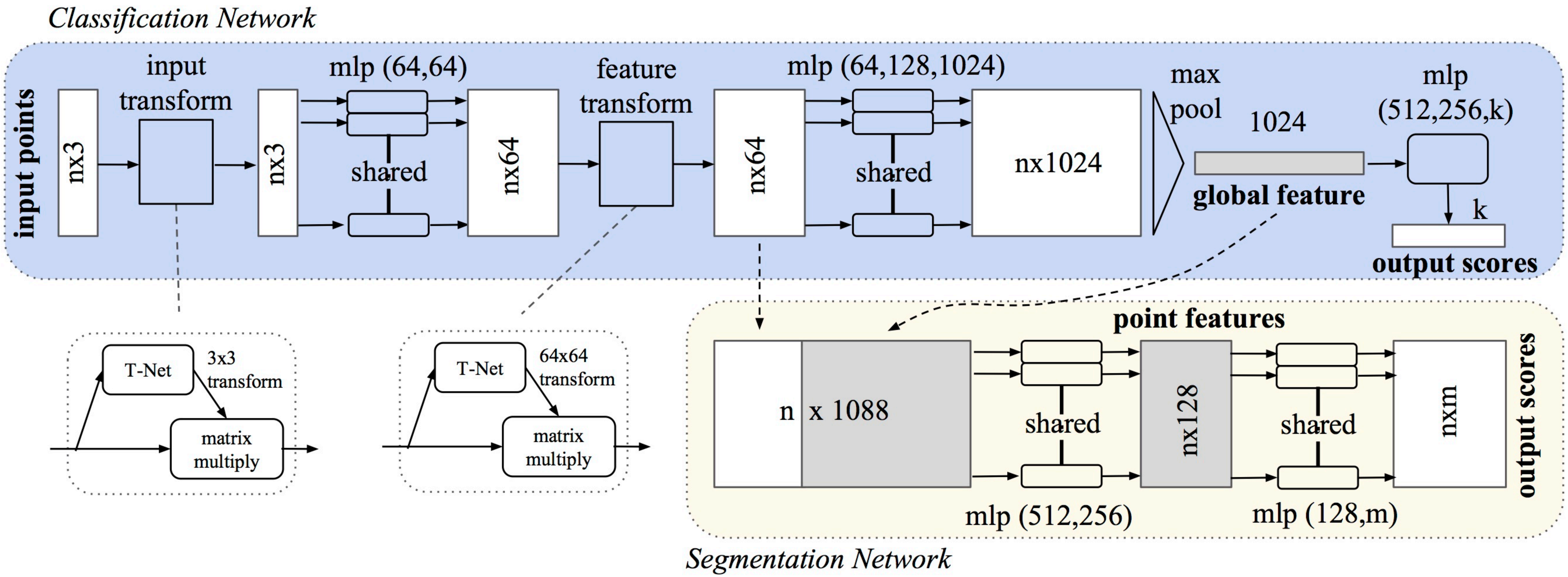
- AlexNet
- VGGNet
- ResNet
- Inception (GoogLeNet)

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

Pointnet (3D point cloud)





Summary

- Deep learning basics (automatic feature extraction)
  - Classification task and basic neural network architecture
  - Training of neural network
  - More complex deep learning models

Worth Reading

- **Deep learning basics**
  Stanford CS231. CNN for image recognition. https://cs231n.github.io