

AUTONOMOUS DRIVING COURSE

ULTRASONIC, RADAR AND LIDAR SENSORS

Bunciu Elena

CONTENTS

CONTENTS

Slide structure

1. Ultrasonic sensors (USS)

- Physical principles of operation of ultrasonic sensors
- Applications and System architecture
- Tracking algorithms

2. Radar sensors

- Sensor model
- Principles of operation (SRR, MRR, LRR)

3. Lidar sensors

- Principles of operation

4. Software architecture

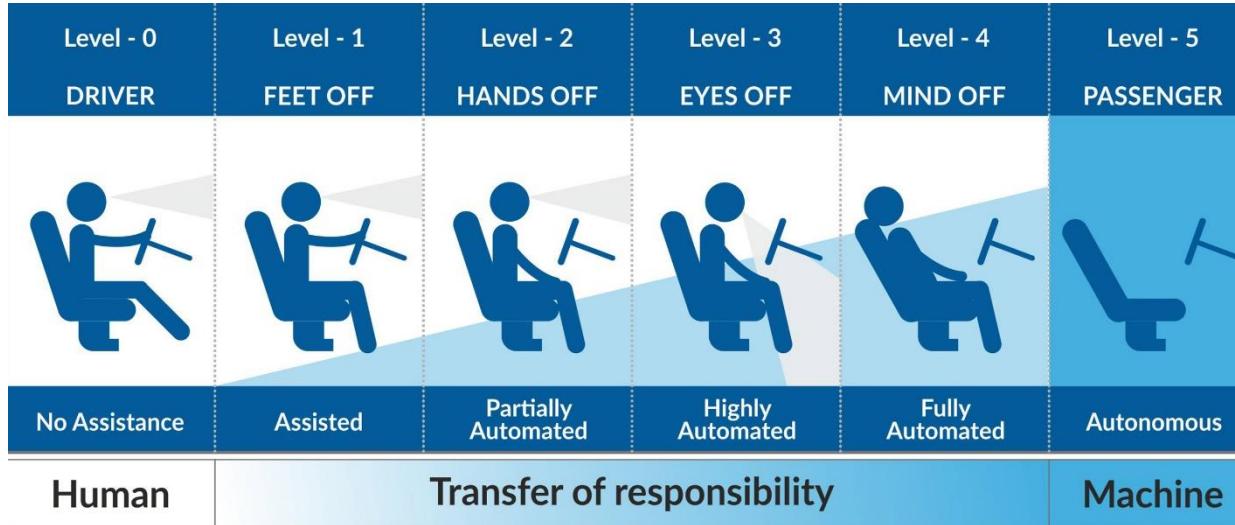
Autonomous Driving

Course Objectives

- What an ultrasonic sensor is?
- How does a sensor ultrasonic works?
- What functionality base on ultrasonic sensors we have?
- How a RADAR system works?
- What are the differences between USS and RADAR?
- How a LIDAR system works?
- What are the differences between LIDAR and RADAR?

Autonomous Driving

Levels of automation



ULTRASONIC SENSORS

Ultrasonic Sensors

Physical principles

- An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic waves.
- Ultrasonic waves are sounds waves with a higher frequency than the upper audible limit of human hearing
- Sound waves are mechanical waves that require a medium through which to propagate. Sound waves cannot travel through a vacuum

Ultrasonic Sensors

Physical principles

- Different materials have different acoustic properties
 - Varies the ability to transmit sound waves
 - Varies the ability to reflect sound at interfaces
- An ultrasonic sensor uses a transducer that evaluates targets by measuring the time between sending a signal and receiving an echo the distance of an object can be calculated
- High-frequency sound waves reflect from boundaries to produce distinct echo patterns.

Ultrasonic Sensors

Physical principles

- Ultrasonic sensors can convert electrical energy into acoustic wave and vice versa.
- The transducer of the sensor sends the ultrasonic waves and receives the reflected wave (echo) from the object in the near proximity.
- The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

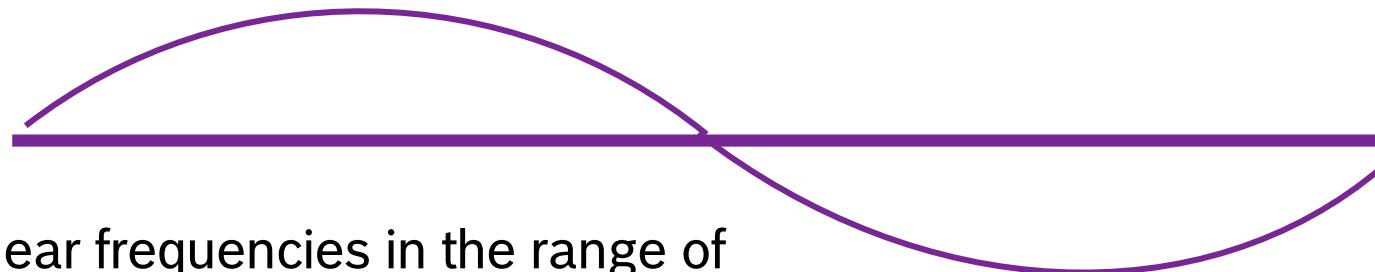
Ultrasonic Sensors

Physical principles

Frequency

The number of cycles completed per second.

1 cycle per second is called Hertz (Hz)



- Humans hear frequencies in the range of
infrasound<20Hz – 20kHz<ultrasounds
 - Sound above the level of human hearing is called **ultrasound**
- Bats => 10kHz – 200kHz
- Dolphins => 75kHz – 150kHz

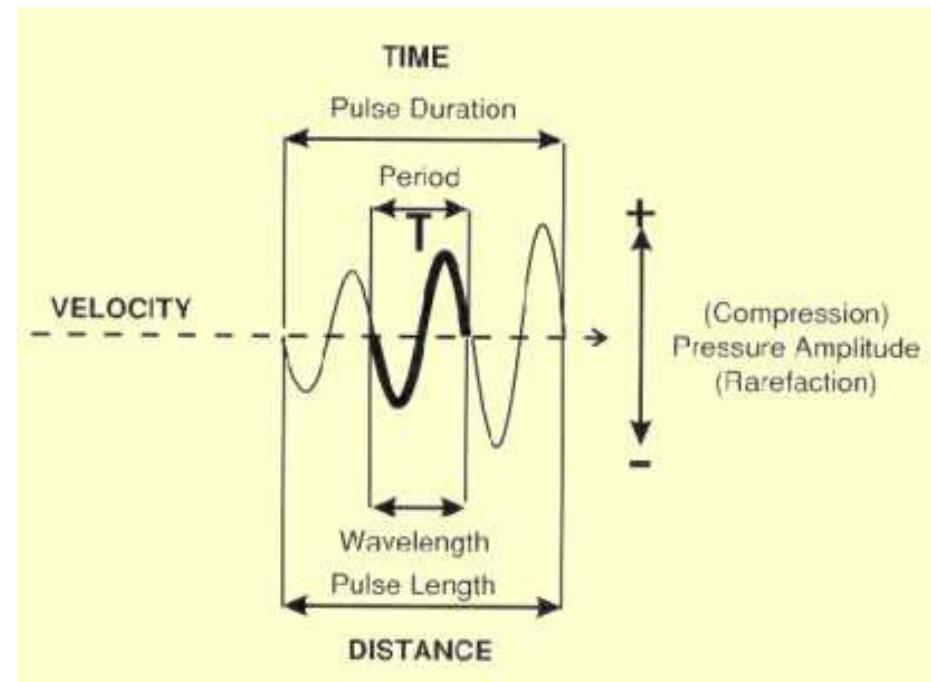
Ultrasonic Sensors

Characteristic parameters

- The frequency of sound is determined by its source.

$$\lambda = v \cdot T \leftrightarrow \lambda = \frac{v}{f}$$

- If the frequency increases then the wave length must decrease as they are inversely proportional to each other.



<https://www.slideshare.net/RakeshCa2/ultrasound-physics-39841154>

Ultrasonic Sensors

Bosch sensors

USS1



USS2



USS3



USS4



USS5



USS6



1995

1996

1998

2004

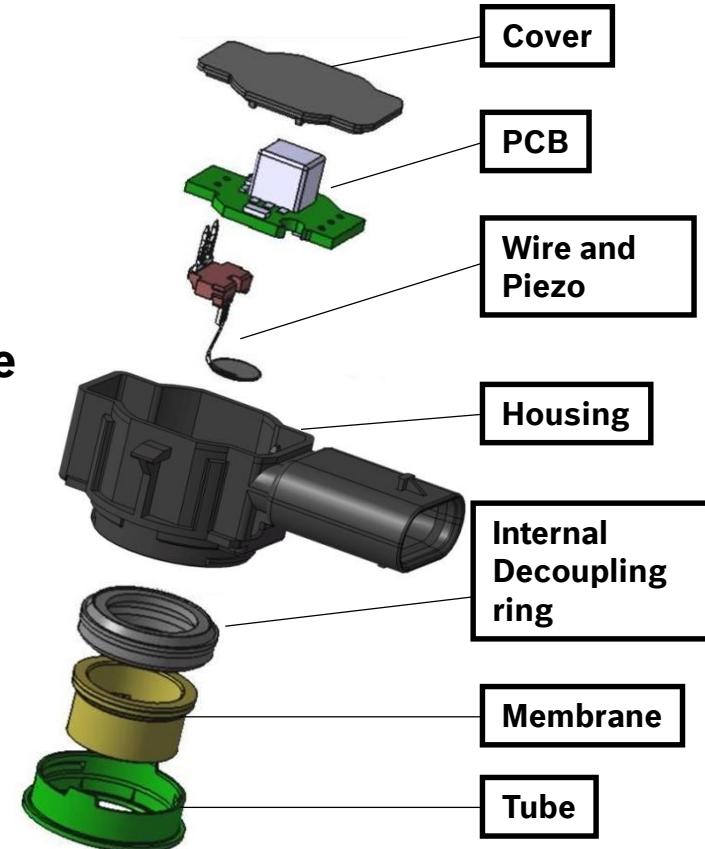
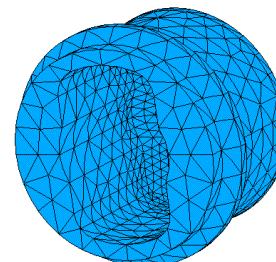
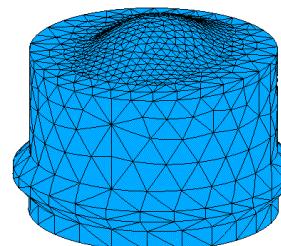
2011

2016/2017

Ultrasonic Sensors

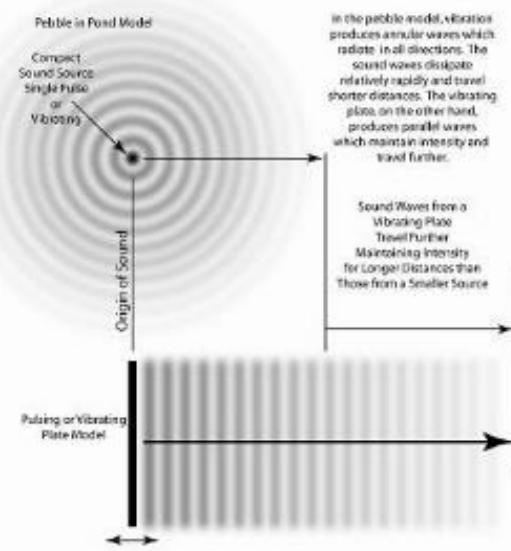
Structure

- ▶ Maximal distance up to 6.5 m
- ▶ Compact sensors for different connectors
- ▶ Wide range in installation in different cars
- ▶ Different surfaces of membrane
- ▶ **Add-on parts handling like license plate holder, trailer hitch, spare wheel etc... → laboratory**

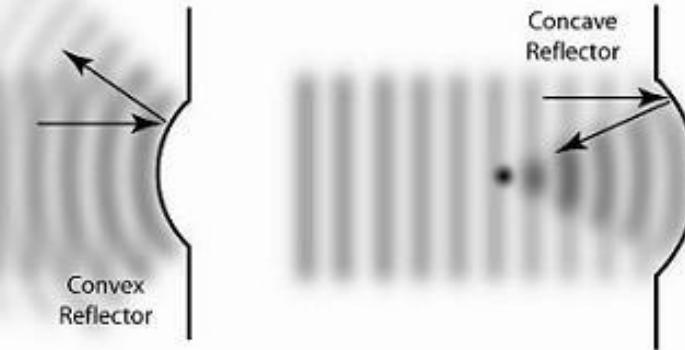


Ultrasonic Sensors

Physical principles



Sound Waves Reflecting from Convex and Concave Surfaces



Sound waves generated by a vibrating piezoelectric plate travel further distances maintaining intensity than those generated by a smaller vibrating source.

Sound waves can be concentrated and deviate much the same as light waves from convex and concave surfaces.

<http://www.ctgclean.com/tech-blog/ultrasonics-understanding-sound-waves-1>

Ultrasonic Sensors

Physical principles

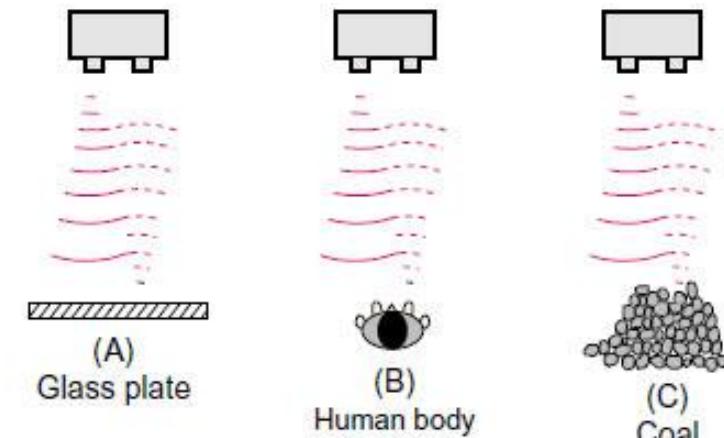
Types and shapes of detection objects (reflective type)

Detected objects can be classified as follows:

- (A) Flat-surface objects such as fluids, boxes, plastic sheets, paper, and glass.
- (B) Cylindrical objects such as cans, bottles, and human bodies.
- (C) Powders and chunk-like objects such as minerals, rocks, coal, coke, and plastic.

The reflective efficiency varies depending on the shape of these objects. In the case of (A), the greatest amount of reflected waves return, however, this is strongly affected by the inclination of the object.

In the case of (B) and (C), stray reflections occur and the reflected sound is not uniform, however, the effect of inclination is small.



http://www.omron-ap.com/service_support/technical_guide/ultrasonic_sensor/index.asp

Ultrasonic Sensors

Physical principles of operation

Bosch USS Gen6 sensors

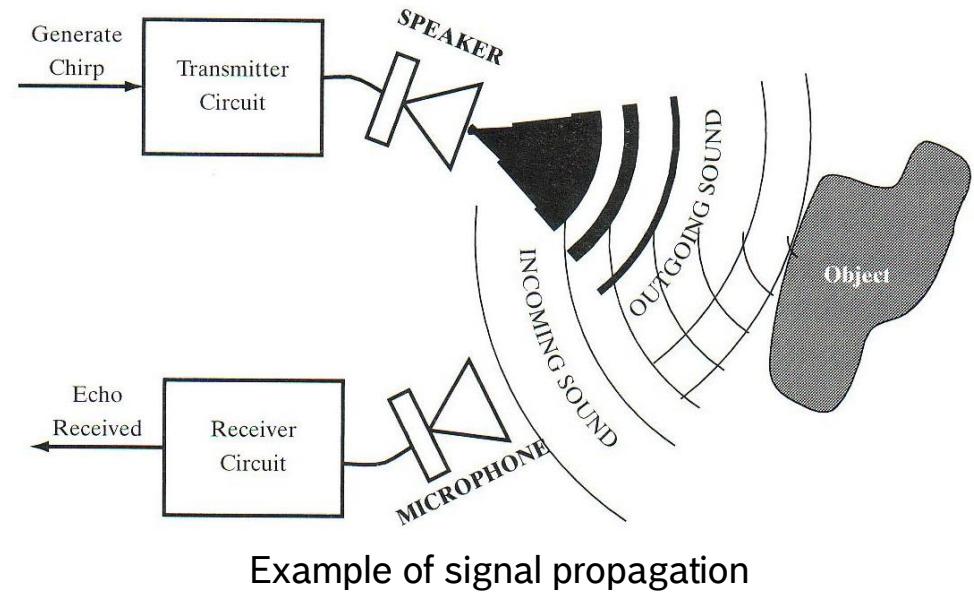
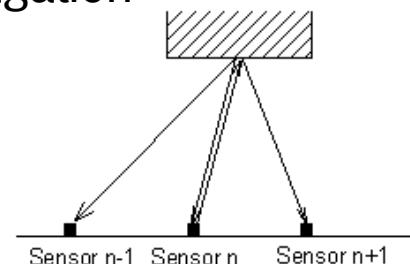
- 47 - 50 kHz Ultrasonic sound
- Induction with Piezo electronic
- $v = 340 \text{ m/s}$ – temperature dependent
- Distance calculation:

$$D = v * t$$

D = round-trip distance

v = speed of wave propagation

t = elapsed time



Example of signal propagation

<https://www.google.de/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUEwjKweKl9MTWAhXEZ1AKHbDKdkUQjhwlBQ&url=http%3A%2F%2Fslideplayer.com%2Fslide%2F6365487%2F&psig=AFQjCNEWpF3ekDkhMhBZPjgqk0tg7xzCAQ&ust=1506585314959443>

ULTRASONIC SENSORS

Applications and System architecture

Applications:

- ▶ Distance Measurement
- ▶ Level measurement
- ▶ Object detection
- ▶ Medical imaging (sonography)

Example:

<https://www.youtube.com/watch?v=9MlfxqAmQ9s>

Ultrasonic Sensors

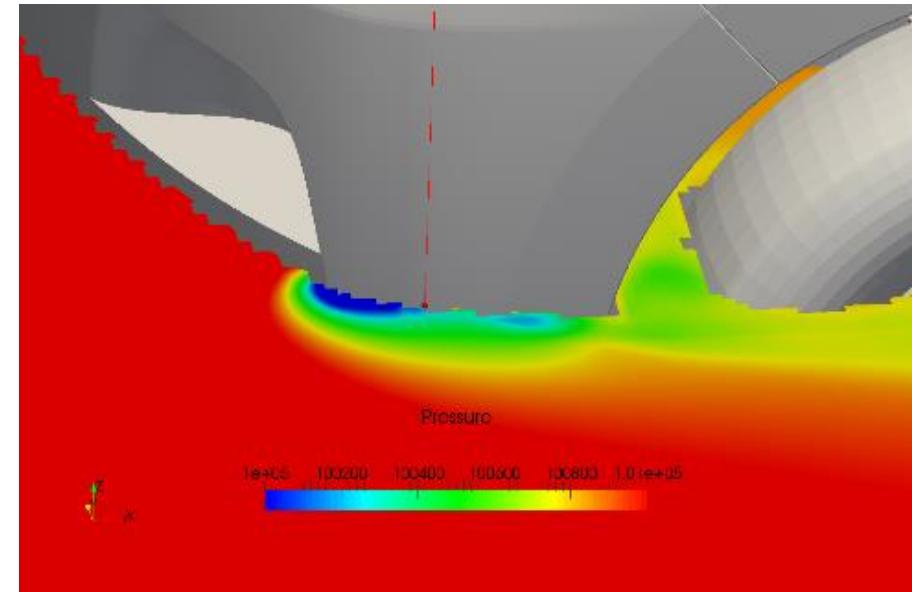
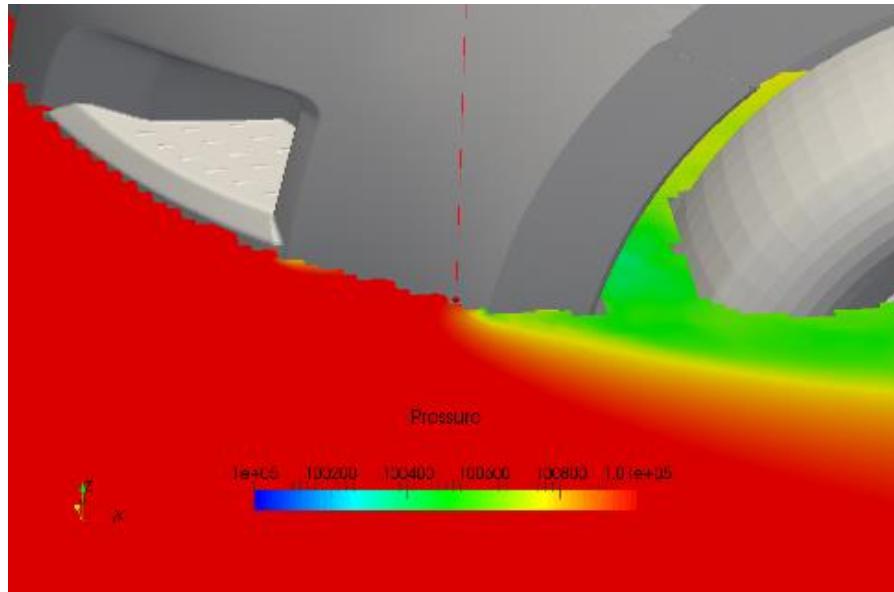
Environmental influences

- Air temperature (speed of sound changes by 0.17% per degree Kelvin)
- Humidity
- Air pressure
- External noise (tire noise, exhaust, as an example.)

Ultrasonic Sensors

Air currents

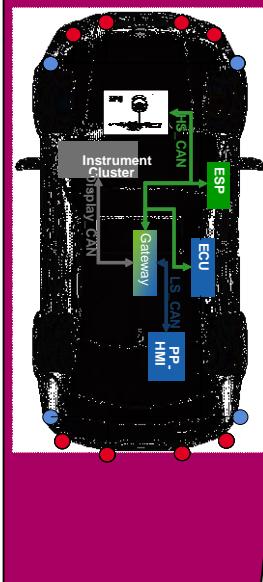
- Air currents (regular air currents (wind) have effect on ultrasonic measurement at speeds over (50-61.5 km/h). → aerodynamically simulation



Ultrasonic Sensors

Parking functions

Functions based on Ultrasonic Park Assist System Configuration



Side Distance Warning (SDW)

- Tracking of obstacles at the side of the car. Driver is warned dependent on steering angle



Park Steering Control (pPSC/cPSC/ dPSC)

- Automatic recognition of suitable parallel, cross, diagonal spaces
- Reverse parking in one or – if necessary – in multiple moves



Pull-out Control (POC)

- System supports pull-out situations by taking over all necessary steering moves when maneuvering forward and backward



Side View Assist (SVA)

- System warns the driver in case of vehicles bypassing or staying in the blind spot area.



Remote parking, Home zone park assist etc.

- No supervision by driver required. Robust surround sensing necessary.



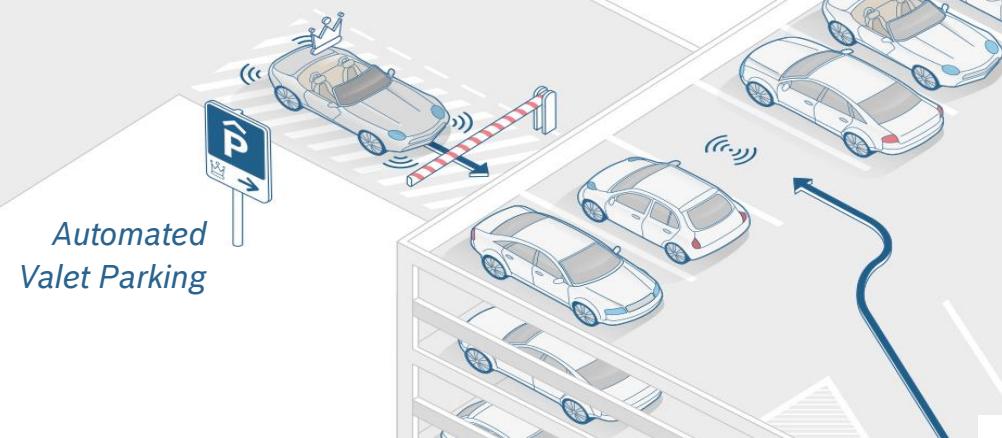
Ultrasonic Sensors Connected Parking

Connected **P**ARKING

offers car park operators* new business opportunities and suggestions for increasing the market value and range of services for drivers.



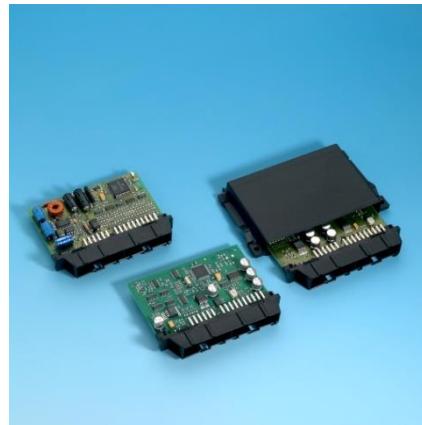
Community-based Parking



Automated
Valet Parking

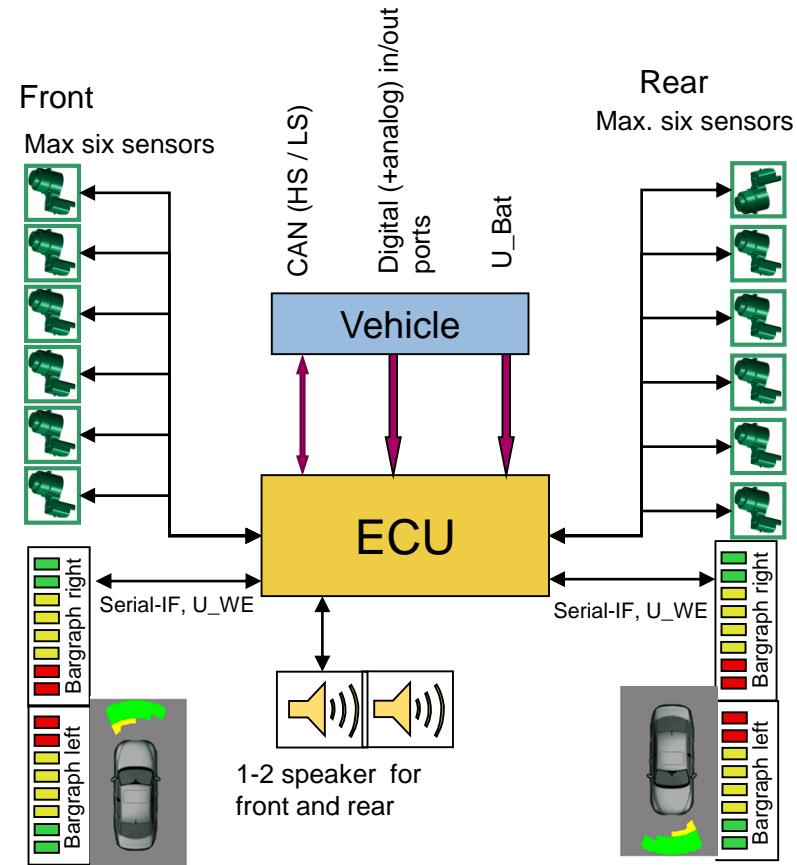
* Airports, hotels, shopping centres, towns etc.

Ultrasonic Sensors Electronic Control Unit



Ultrasonic Sensors

System architecture-example



Ultrasonic Sensors

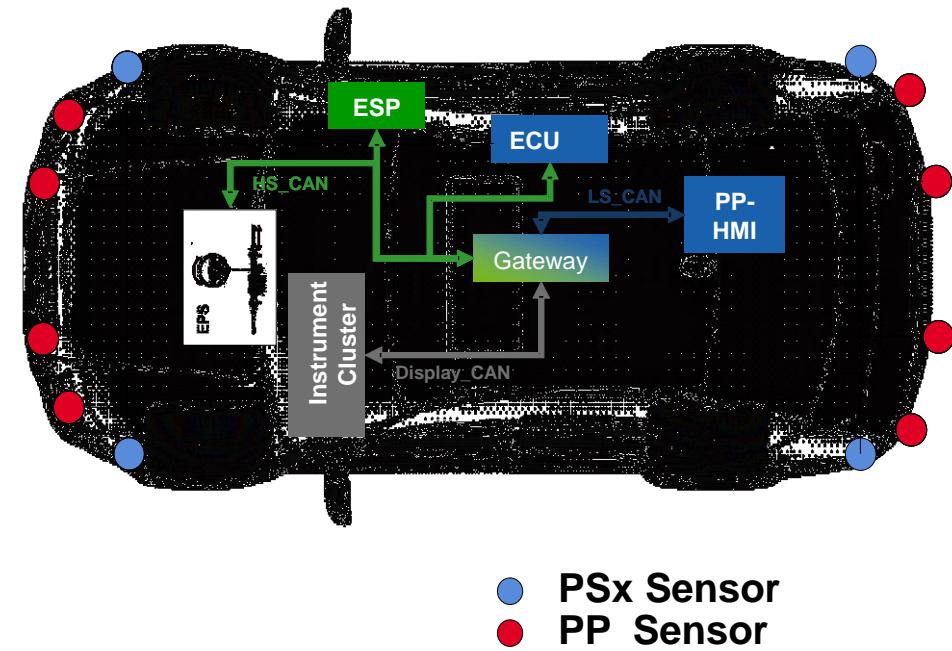
System configurations

Basic function

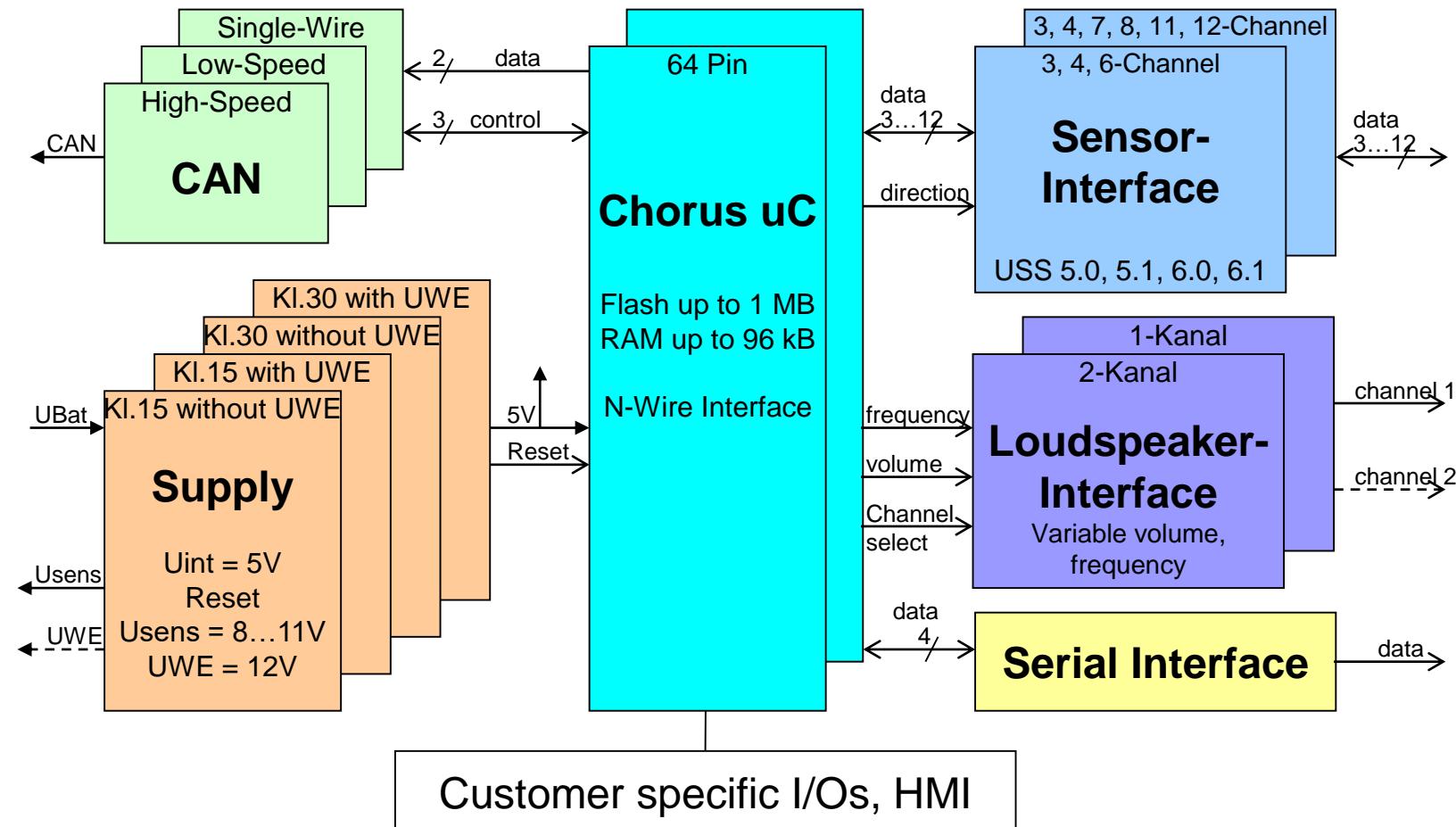
- Park Pilot (PP-PAS)
 - 4 Ch Rear PP
 - 8 Ch Front/Rear PP
 - 12 Ch Front/Rear PP

Advanced functions

- Park Steering Control (PSC - HFP)
- Pull Out Control (POC)
- Side View Assist (SVA-BSW)
- Side distance warning (SDW-FKP)

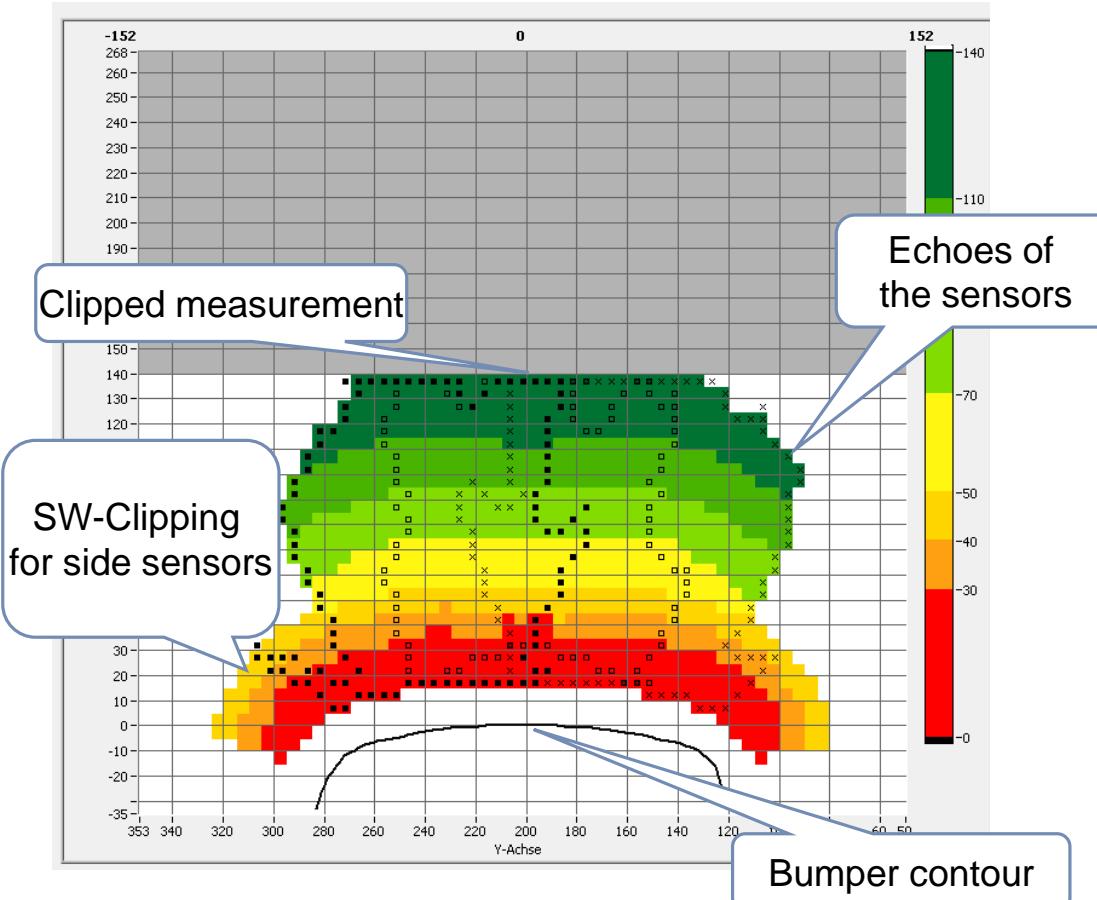


Ultrasonic Sensors ECU Block Diagram -Example



Ultrasonic Sensors

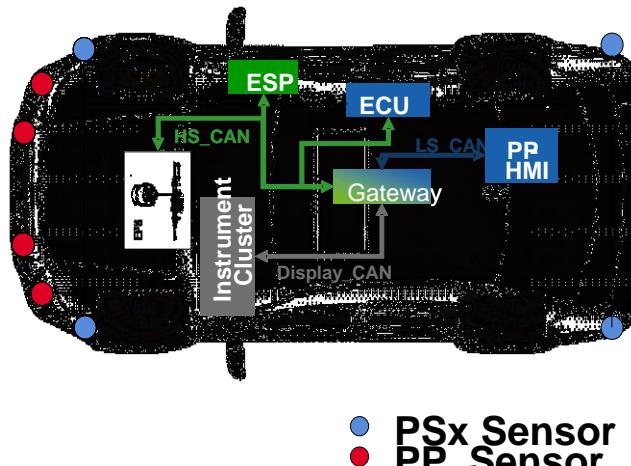
Basic function-Park Pilot



Ultrasonic Sensors Tracking Algorithms

What is a tracking algorithm?

- Is a software strategy for performance enhancement for radar, ultrasonic, sonar, etc. technologies.
- Provide the ability to calculate and predict the future position of static/moving obstacles, based on historical positions reported by the sensors.



Why do we need tracking algorithms in Autonomous Driving systems?

- Implementation of complex functions, using minimum number of sensors (ex: side protection of the car).
- Covering of unwanted holes and gaps in the field of view, generated by poor sensitivity of the sensors in certain zones, due to the multiple factors.
- Calculating and predicting of obstacles position, based on the car movement, speed, wheels direction, etc.

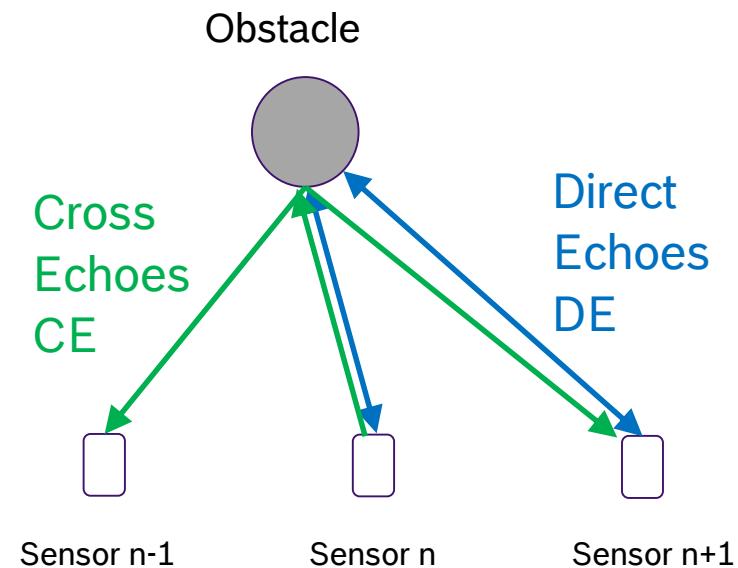
Ultrasonic Sensors Tracking Algorithms

Concept:

1. Using the triangulation algorithm, the position of the obstacle is detected and the distance by the car is provided by the sensors.

Direct Echoes and Cross Echoes

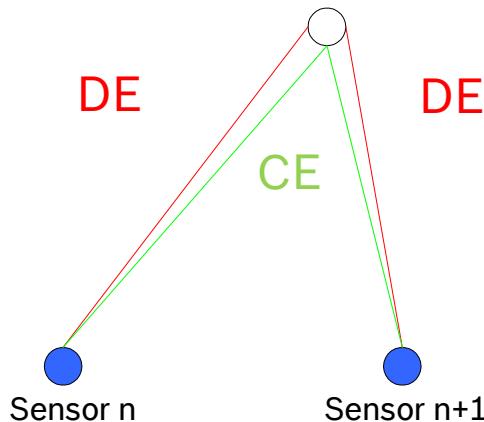
- Triangulation algorithm for object position.



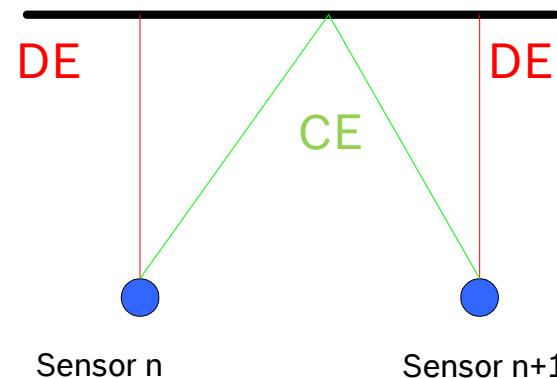
Ultrasonic Sensors Tracking Algorithms

Concept:

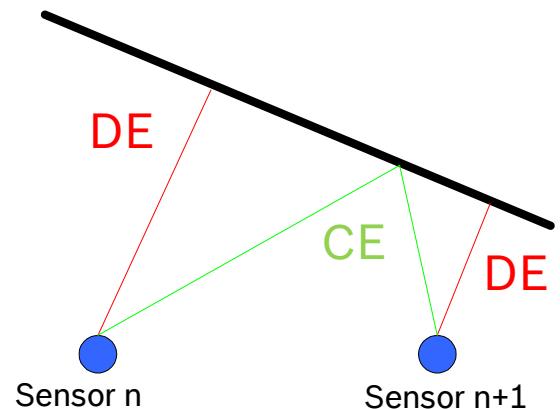
2. Detecting the type and the orientation of the obstacle using the information received from the sensors (DE and CE) and position of the obstacle (triangulation).
 - Including the obstacle in a category and, starting with this step, software will treat it as an **object**.



Ex: Round pole



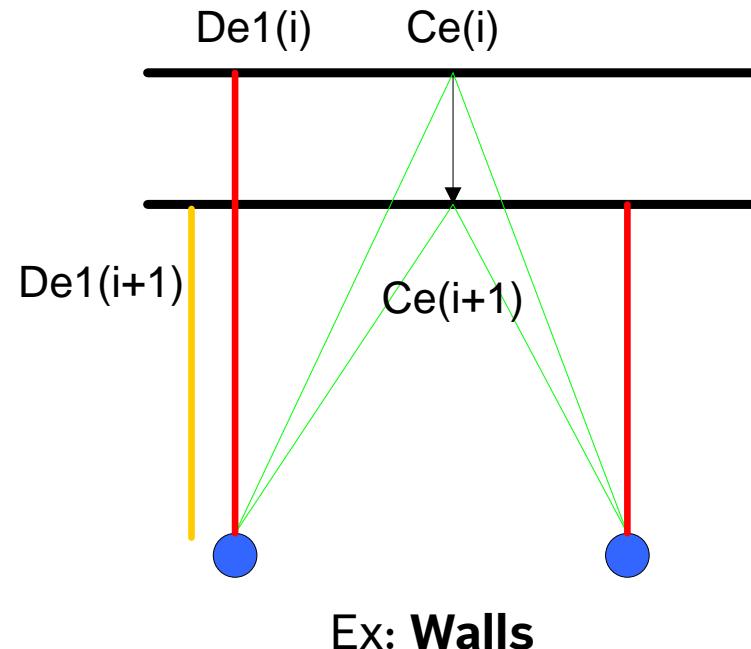
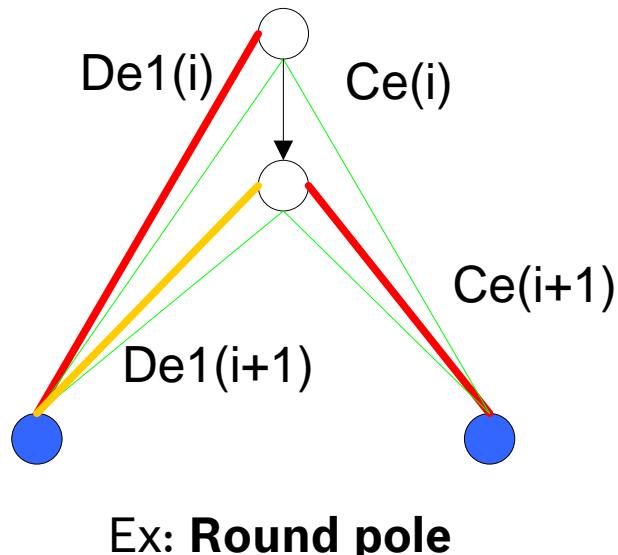
Ex: Walls



Ultrasonic Sensors Tracking Algorithms

Concept:

3. Predicting the next position of the objects, using the data received from sensors, position, type, speed of the car, wheel directions.

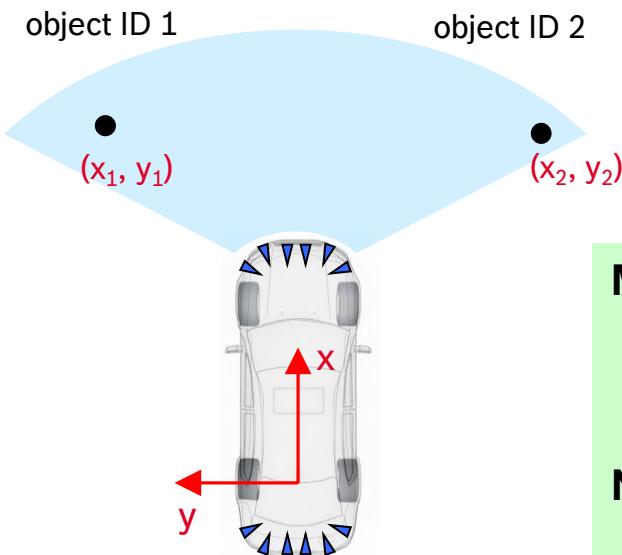


Ultrasonic Sensors Tracking Algorithms

Concept:

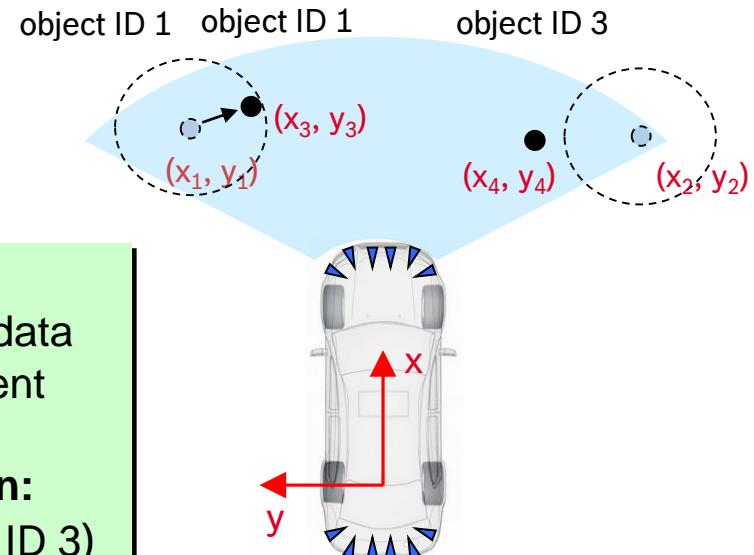
Initial situation

measurement time t_0



Tracking and matching

measurement time t_1



Matching decision:

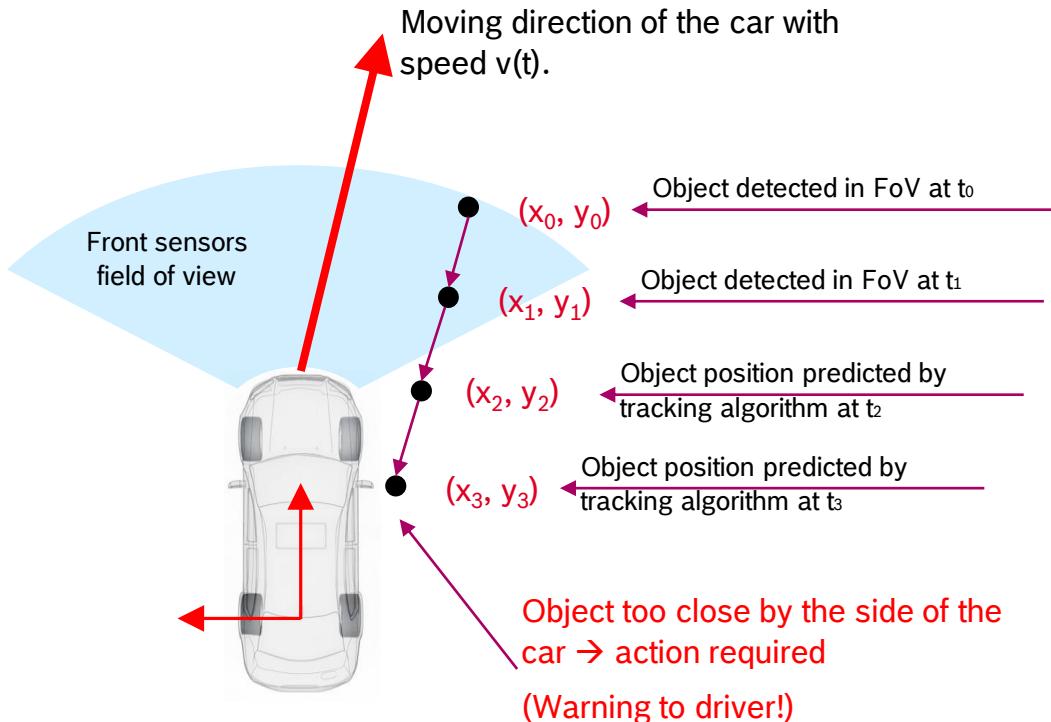
→ take-over position data
of new measurement
(object ID 1)

Non matching decision:

→ new object (object ID 3)

Ultrasonic Sensors Tracking Algorithms

Tracking of static obstacles (e.g. Side Distance Warning)

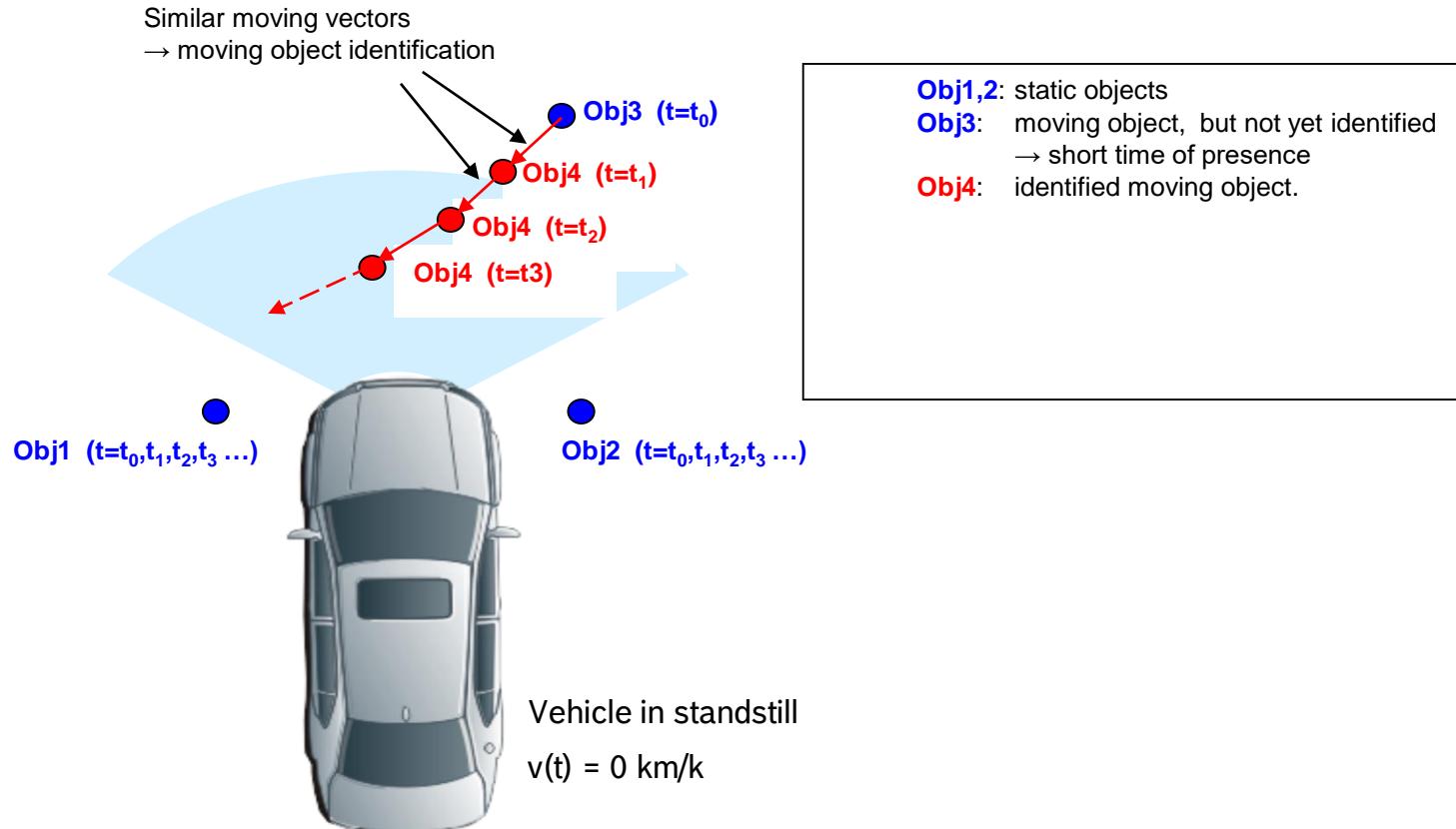


- Tracking the obstacles on the sides of the car.
- Driver is warned according with the position of the obstacle, wheel angle, speed.

Ultrasonic Sensors

Tracking Algorithms

Tracking of moving obstacles



RADAR SENSORS

Radar Sensors

The master of motion measurement

- The simplest function of radar is to tell you how far away an object is.
- To do this, the radar device emits a concentrated radio wave and listens for any **echo**.
- If there is an object in the path of the radio wave, it will reflect some of the electromagnetic energy, and the radio wave will bounce back to the radar device.

Doppler Shift



© 2000 HowStuffWorks

Radar Sensors

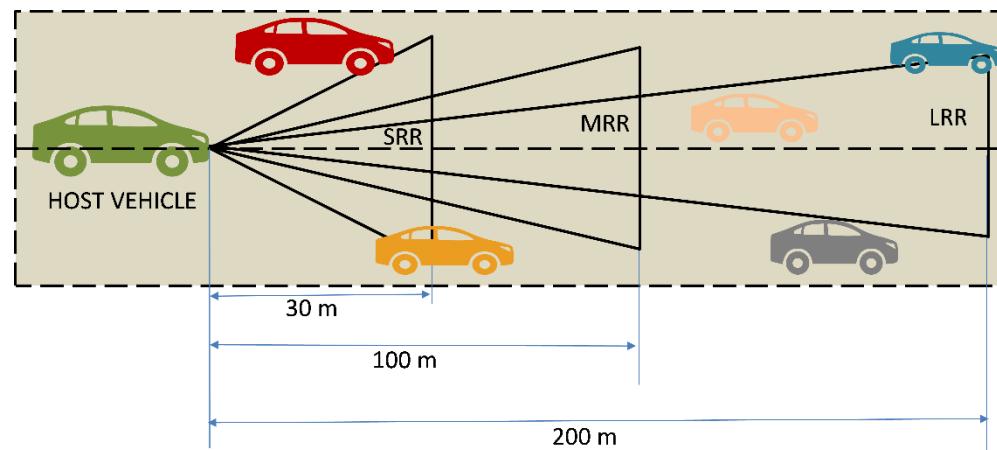
The master of motion measurement

- Radio waves move through the air at a constant speed, so the radar device can calculate how far away the object is based on how long it takes the radio signal to return.
- Radar can also be used to measure the speed of an object, due to a phenomenon called **Doppler shift**. Like sound waves, radio waves have a certain **frequency**, the number of oscillations per unit of time.
- <https://www.youtube.com/watch?v=VHR8AeZrlVc>

Radar Sensors

SRR, MRR, LRR

- **Long Range RADAR-** Long-range RADARs have the capability to detect objects situated in a wide geographical area, as they can easily cover a range of 10–200m.
- **Mid-Range RADAR-** Mid-range RADAR sensors operate at a range of 100-150m.
- **Short Range RADAR-** Short range RADAR is a technology which uses transceivers with the signal processing equipment in the vehicle and mounted behind the bumper.



Type	SRR	MRR	LRR
Frequency Band	76 -77 GHz	77- 79 GHz	77 – 81 GHz

<http://automotive.electronicspecifier.com/sensor/s/what-is-driving-the-automotive-lidar-and-radar-market>

Radar Sensors

Evolution of Bosch radar sensors



SOP: 2000

- Range: up to 150 m
- GaAs Oscillator (Gunn Diode)
- Opening Angle: 8°
- Dimensions (HxWxD) 124 x 91 x 97 mm
- Weight: 600 g



SOP: 2004

- Range: up to 200 m
- GaAs Oscillator (Gunn Diode)
- Opening Angle: 16°
- Dimensions (HxWxD) 73 x 70 x 60 mm
- Weight: 300 g



SOP: 2009

- Range: up to 250 m
- SiGe MMICs (bare chip)
- Opening Angle: 30°
- Dimensions (HxWxD) 77 x 74 x 58 mm
- Weight: 285 g



SOP: 2013

- Range: up to 160 m
- SiGe MMICs (packaged chip)
- Opening Angle: 45°
- Dimensions (HxWxD) 60 x 70 x 30 mm
- Weight: 200 g



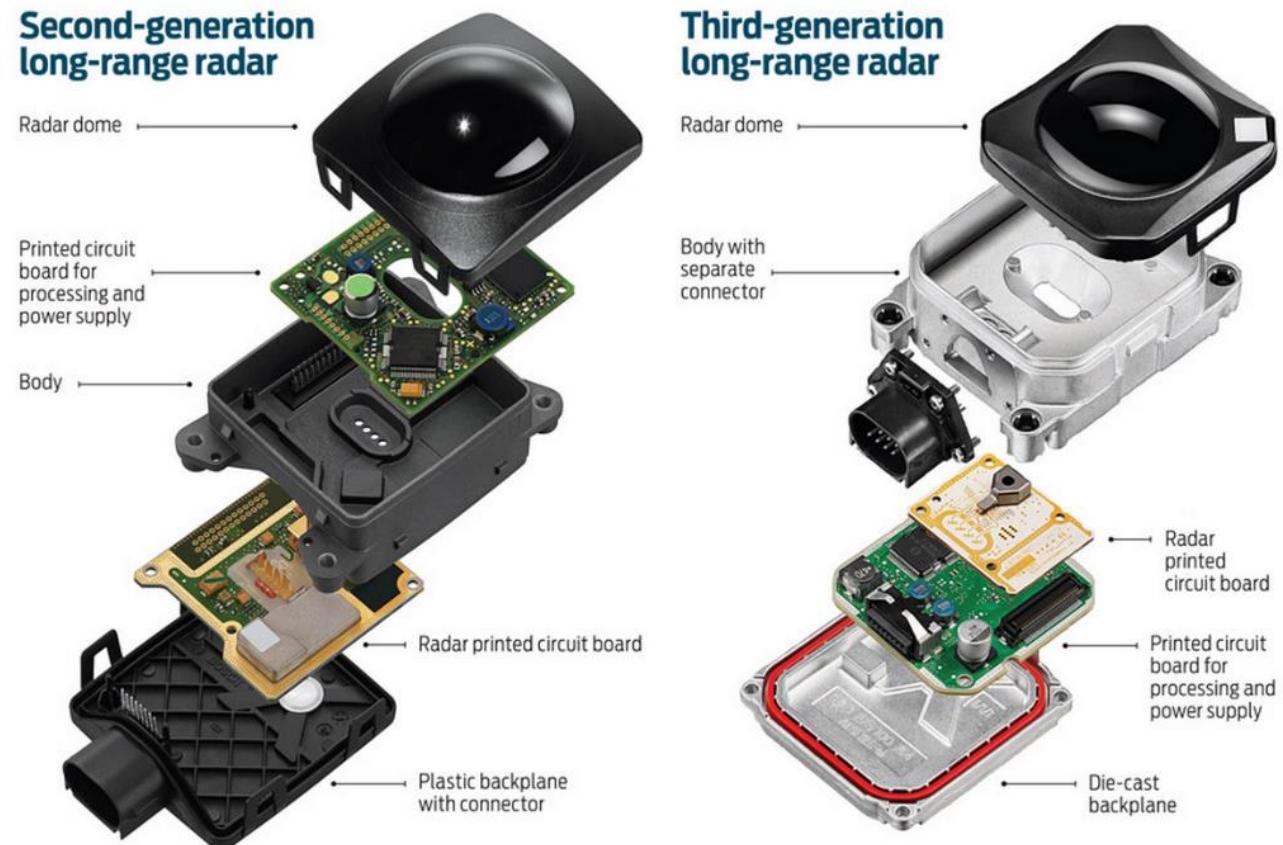
SOP: 2014

- Range: up to 100 m
- SiGe MMICs (packaged chip)
- Opening Angle: 150°
- Dimensions (HxWxD) 60 x 70 x 30 mm
- Weight: 190 g

Radar Sensors

Evolution of Bosch radar sensors

Bosch's latest long-range system greatly simplifies the radar's printed circuit board. Instead of handful of gallium arsenide chips to generate, amplify and detect the 77-gigahertz microwaves, the system uses just one or two (as shown) of Infineon's silicon germanium chips.



Radar Sensors

Application in automotive

- Adaptive Cruise Control (ACC/ACC Stop&Go)
- Predictive Collision Warning (PCW)
- Emergency Break Assist (EBA)
- Blind Spot Detection (BSD)
- Blind Spot Detection Extended (BSDE)
- Lane Change Assist (LCA)
- Rear cross traffic alert (rCTA)
- Front Cross Traffic Start Prevention (fCT-P)
- Cross Traffic Emergency Brake Assist (fCT-BA)
- Cross Traffic AEB (fCT-AEB)

Radar Sensors

RADAR vs Ultrasonic

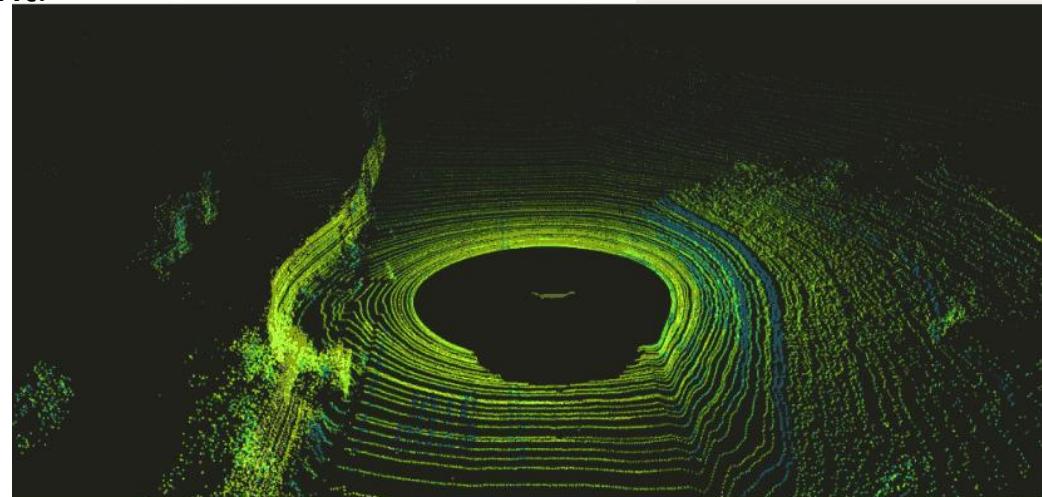
- Ultrasonic:
 - Smaller and flexible
 - excellent longevity.
 - are usually cheaper than radar.
 - More accurate close range surroundings is created
- RADAR
 - Environmental variables just don't affect radar measurements the way they do ultrasonic sensors.
 - More reliable
 - Longer distance
- Ultrasonic and radar sensors don't compete with each other, they complement each other.
Neither one is a one-size-fits-all level measurement solution.

LIDAR SENSORS

Lidar Sensors

The master of 3D mapping

- Lidar, short for light detection and ranging, is a technology that measures distance using laser light.
- The technology can scan more than 100 meters in all directions, generating a precise 3D map of the car's surroundings.
- This information is then used by car to make intelligent decisions about what to do next.
- The problem with lidar is that they generate a large amount of data and are still quite expensive for OEMs to cheaply implement.



<https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff>

Lidar Sensors

LIDAR Operational Theory

- Is based on the Time of Flight (ToF) method
- A pulse of light is emitted and the precise time is recorded.
- The reflection of that pulse is detected and the precise time is recorded.
- Using the constant speed of light, the delay can be converted into a “slant range” distance.
- Knowing the position and orientation of the sensor, the XYZ coordinate of the reflective surface can be calculated.

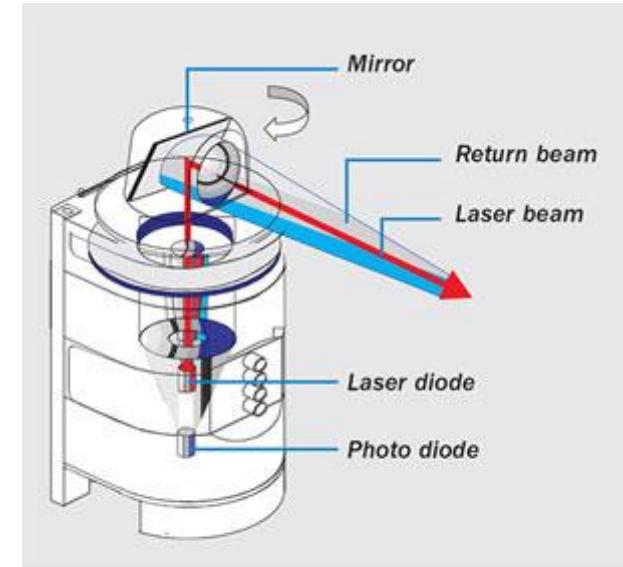


Illustration of LIDAR sensor demonstrating the time of flight principle. (Courtesy of SICK, Inc.)

Lidar Sensors

LIDAR Operational Theory

- ▶ In the automotive sector, laser pulses with a length of 3 to 20 nanoseconds are used for the ToF method whereas the shorter laser pulses provide a better accuracy.
- ▶ The tracking algorithms contained in the sensor's software are able to determine the position and shape, the velocity, yaw rate, heading direction and many other attributes of the objects. Therefore, the LIDAR sensors can be used as system for measuring distance and velocity.

$$Range = \sqrt{\frac{P \cdot A \cdot T_a \cdot T_o}{D_s \cdot P \cdot B}}$$

P = Laser Power

A = Rx Optics Area (lens or mirror)

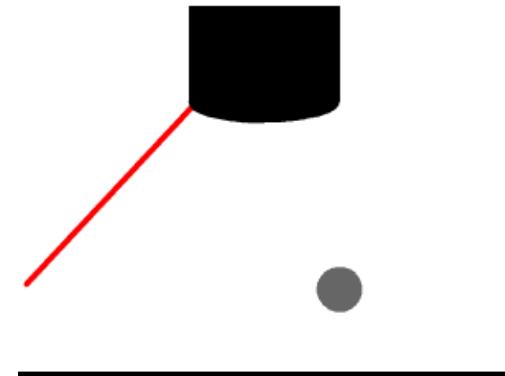
T_a = Transmittance of the atmosphere

T_o = Transmittance of the optics

D_s = Detector Sensitivity

B = Beam Divergence in Radians

ibeo LIDAR Sensor



Sensor View

<https://www.ibeo-as.com>

Lidar Sensors

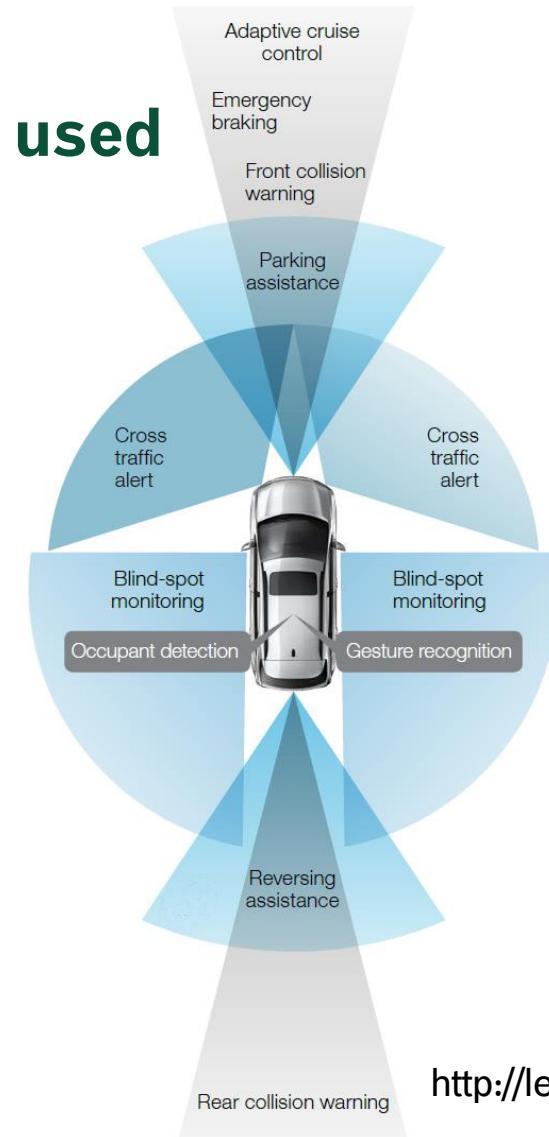
Stats

	Long range Automotive Lidar	Mid-range Automotive Lidar
Detection range: vehicles	Front-facing vehicles: 65 m Rear-facing vehicles: 150 m	Front-facing vehicles: 15 m Rear-facing vehicles: 40 m
Detection range: pedestrians	60 m	12 m
Field of view	20°	90°
Detection segment number	16	16
Operating temperature	-40° C to + 105° C	-40° C to + 105° C
Typical applications	Front/rear collision warning Adaptive cruise control Automatic emergency breaking Traffic jam assistance	Blind spot monitoring Cross-traffic alert Lane change assistance Parking assistance

Lidar Sensors

Applications where the sensor is best to be used

- Forward/rear collision warning
- Blind-spot monitoring
- Cross traffic alert
- Parking assistance
- Automatic emergency braking
- Adaptive cruise control
- Traffic jam assistance
- Gesture recognition
- Occupant detection



<http://leddartech.com>

LIGHT DETECTION AND RANGING (LiDAR)

Applications where the sensor is best to be used

- **Agriculture**
- LiDAR also can be used to help farmers determine which areas of their fields to apply costly fertilizer.
- **Archaeology**
- LiDAR has many applications in the field of archaeology including aiding in the planning of field campaigns, mapping features beneath forest canopy, and providing an overview of broad, continuous features that may be indistinguishable on the ground.
- **Autonomous vehicles**
- Autonomous vehicles use LiDAR for obstacle detection and avoidance to navigate safely through environments
- **Atmospheric remote sensing and meteorology**
- Atmospheric LiDAR is used to study atmospheric properties from the ground up to the top of the atmosphere. Such instruments have been used to study, among other, atmospheric gases, aerosols, clouds, and temperature.

Radar Sensors

LIDAR vs RADAR

- LIDAR:
 - More accurate for creating 3D objects
 - High density
 - Faster
 - Safer
- RADAR
 - Have much further range
 - Much Cheaper
 - Prettier
 - Unaffected by whether
 - Better adaptability

SOFTWARE ARCHITECTURE

Ultrasonic Sensors

Software architecture

- Real world systems are large and complex => Need to divide and conquer
- Software architecture is used for:
 - Understanding the big-picture of the system
 - Communication among stakeholders
 - Generate the project files structure
 - Integrating components
 - Project management, developing, testing, customer requirements tracking
 - Distributing work to teams for developing components
 - Planning and defining strategies for software testing
 - Create the system configuration and builders

Ultrasonic Sensors

Layered architecture

- The system is organized as a collection of layers based on abstraction levels
- Typically, components of the lowest-level layer interacts with the underlying OS and hardware
- A layer uses services offered by its adjacent low-level layer via API's
- Control-flow is from the top layer to the bottom layer
- At run-time, due to callbacks the control can flow from the bottom layer to the top layer

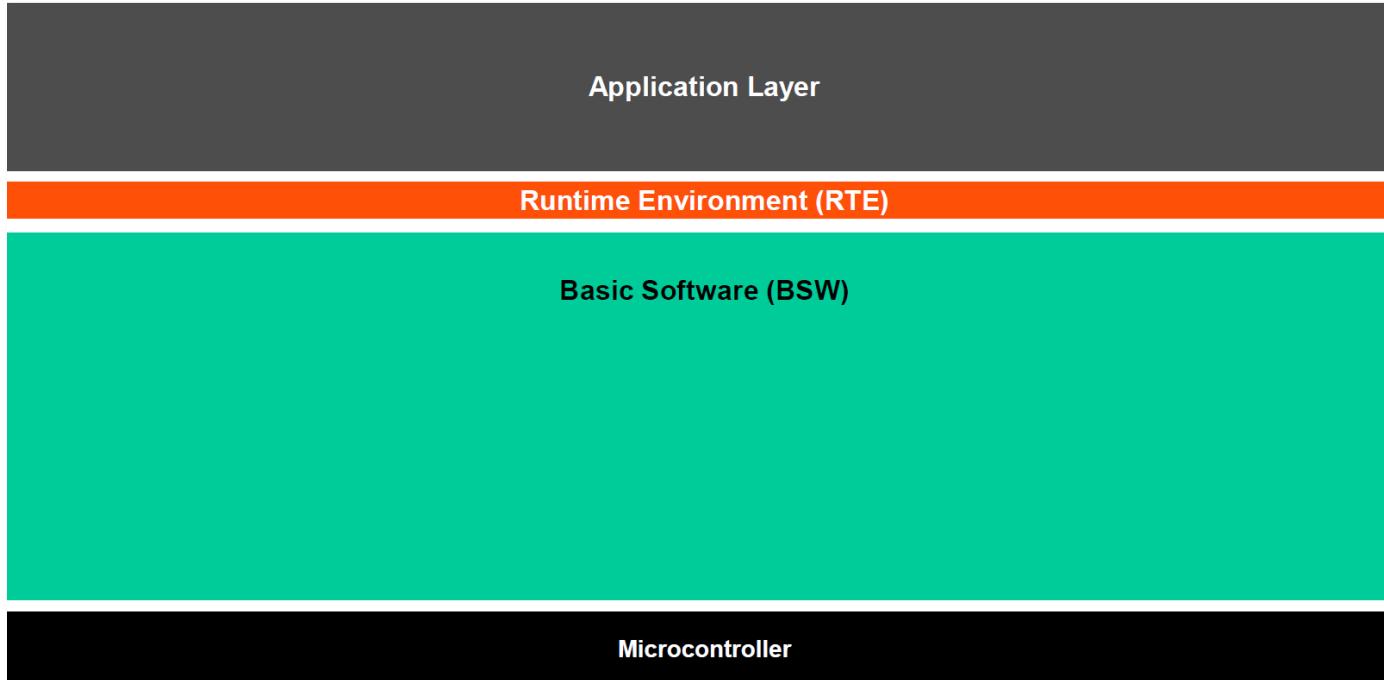
Benefits

- Enables incremental testing from the bottom to top layer
- Plug-out and Plug-in of a layer with a new layer conforming the same API (helpful in producing variants of a product)
- Helps to distribute the work to different teams (often different teams work on different layers)

Ultrasonic Sensors

Software architecture AUTOSAR

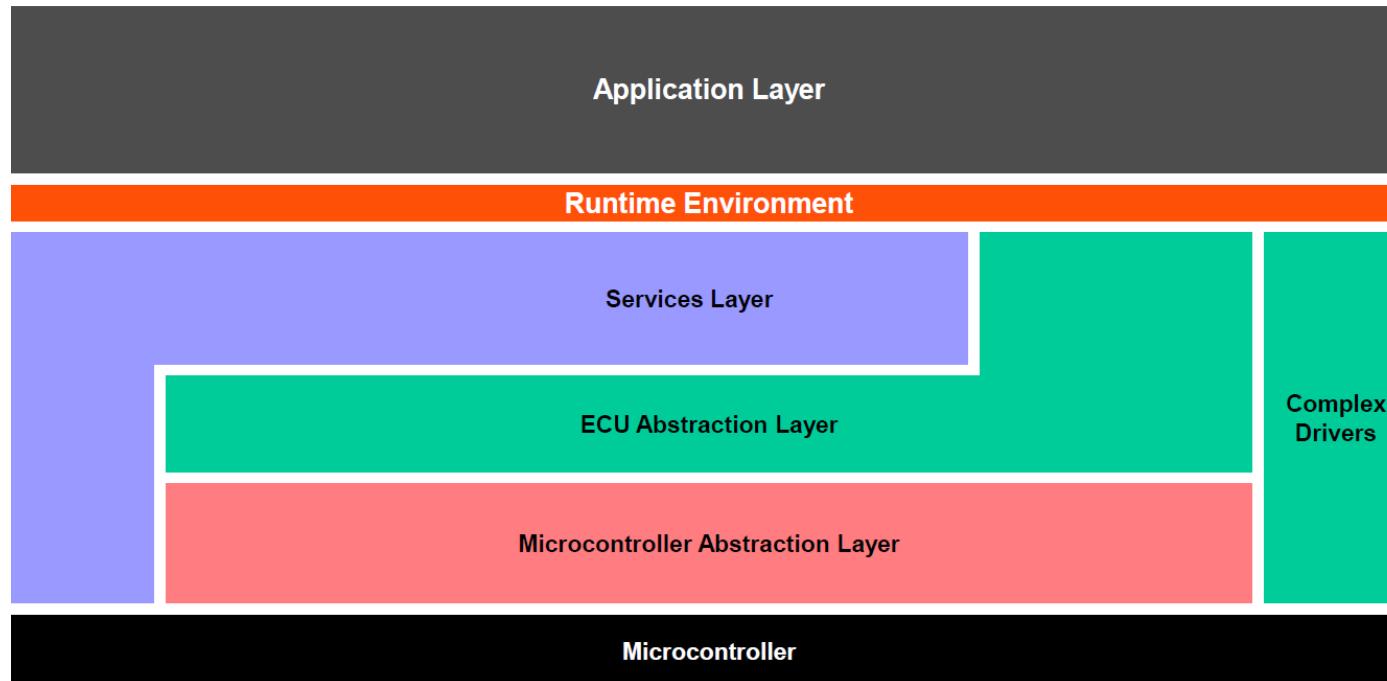
The AUTOSAR Architecture distinguishes on the highest abstraction level between three software layers: Application, Runtime Environment and Basic Software which run on Microcontroller.



Ultrasonic Sensors

Basic Software Layer

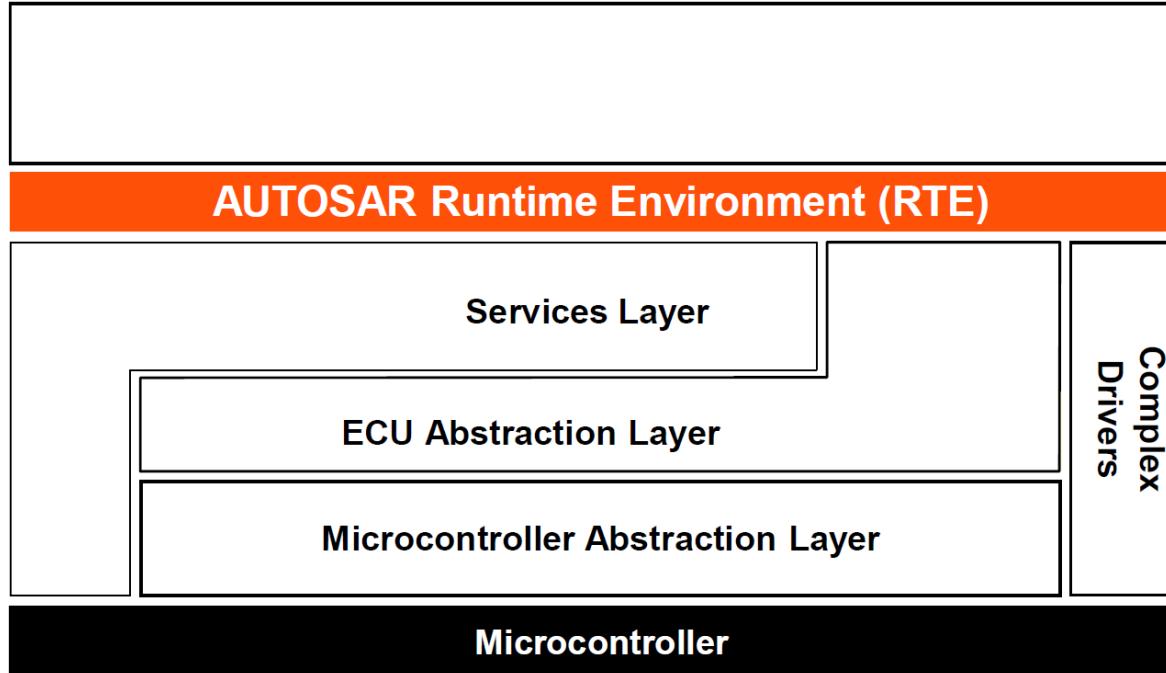
The Basic Software is further divided in the layers: Services, ECU Abstraction, Microcontroller Abstraction and Complex Drivers.



Ultrasonic Sensors Runtime Environment

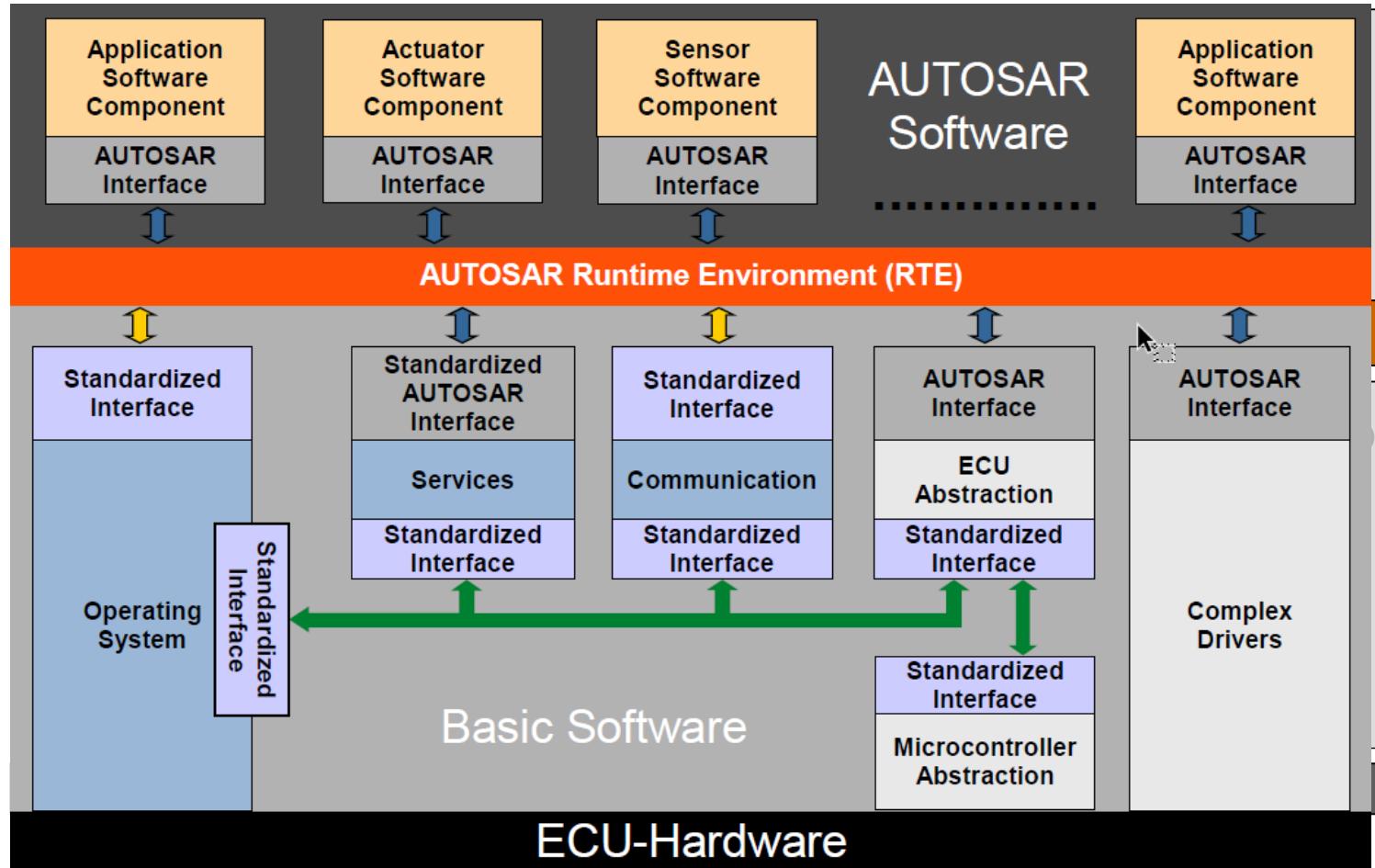
The RTE is a layer providing communication services to the Application software (software components).

Make AUTOSAR Software Components independent from the mapping to a specific ECU.



Ultrasonic Sensors

Top level-Application

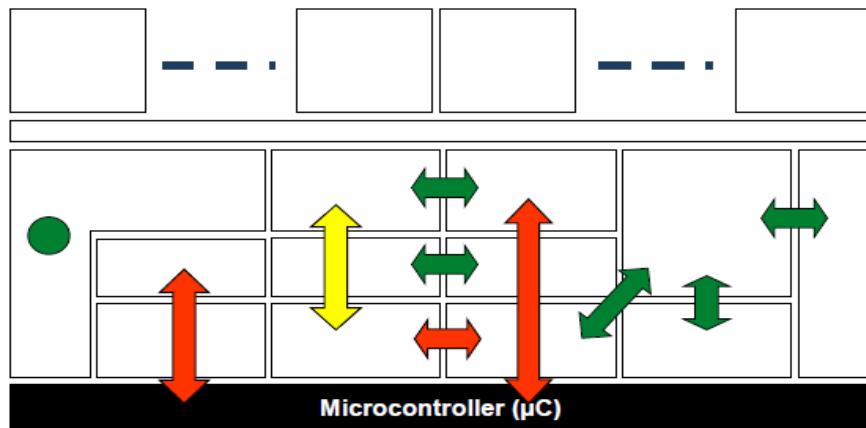


Ultrasonic Sensors

Top level-Application

Horizontal Interfaces

- ↔ Services Layer: horizontal interfaces are allowed
Example: Error Manager saves fault data using the NVRAM manager
- ↔ ECU Abstraction Layer: horizontal interfaces are allowed
- ↔ A complex driver may use selected other BSW modules
- ↔ µC Abstraction Layer: horizontal interfaces are not allowed. Exception: configurable notifications are allowed due to performance reasons.

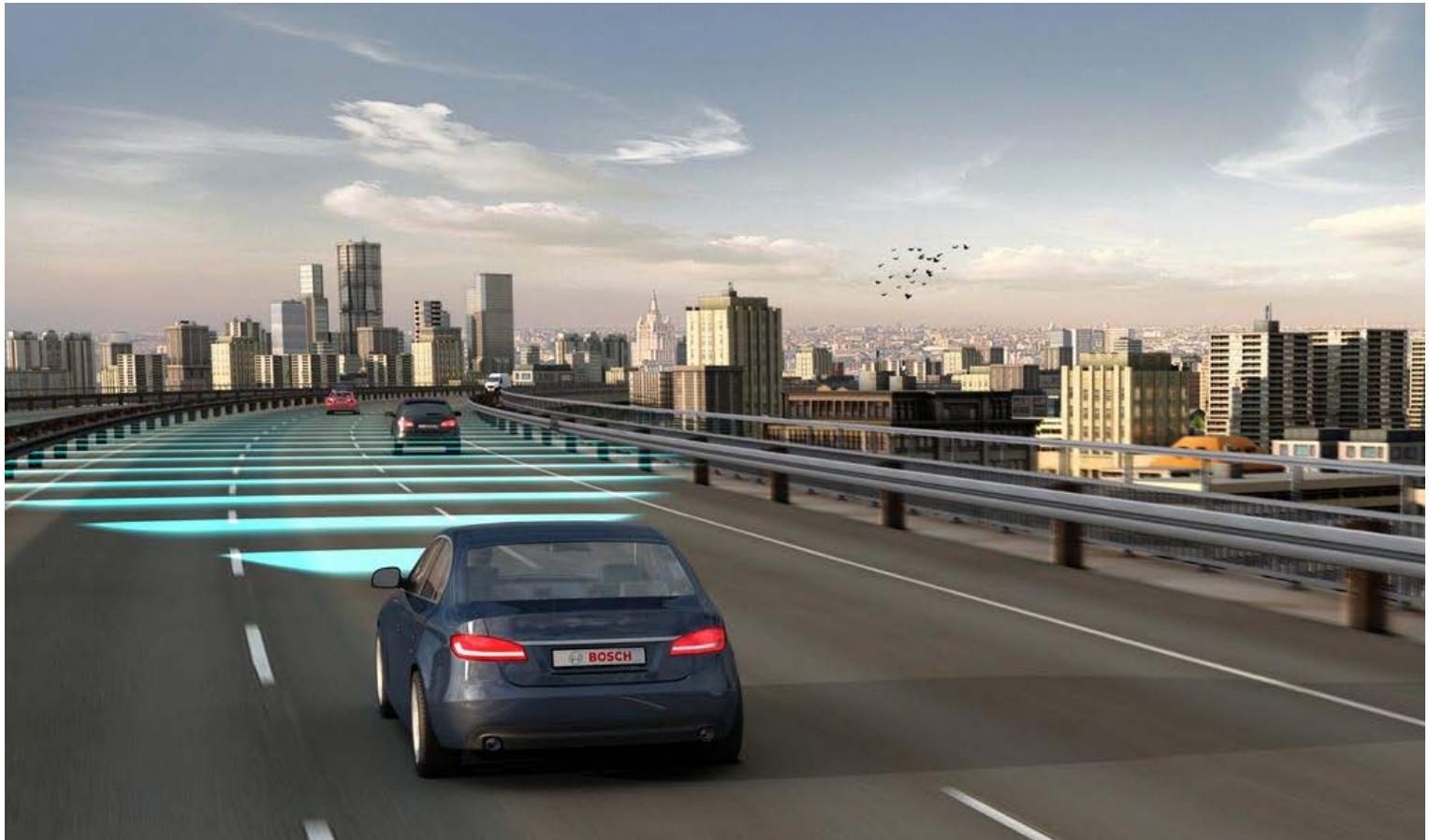


Vertical Interfaces

- ↔ One Layer may access all interfaces of the SW layer below
- ↑ Bypassing of one software layer should be avoided
- ↓ Bypassing of two or more software layers is not allowed
- ↓ Bypassing of the µC Abstraction Layer is not allowed
- ↔ A module may access a lower layer module of another layer group (e.g. SPI for external hardware)
- All layers may interact with system services.

THANK YOU FOR YOUR ATTENTION!

Radar Based Driver Assistance Systems



Radar Based Driver Assistance Systems

Agenda

► **Driver Assistance Systems**

- Basic Terms and Definitions
- Radar Basics
- Sensor Data Fusion
- SW Development

► **What can we achieve mainly with a radar sensor?**

- Adaptive Cruise Control
- Automatic Emergency Brake

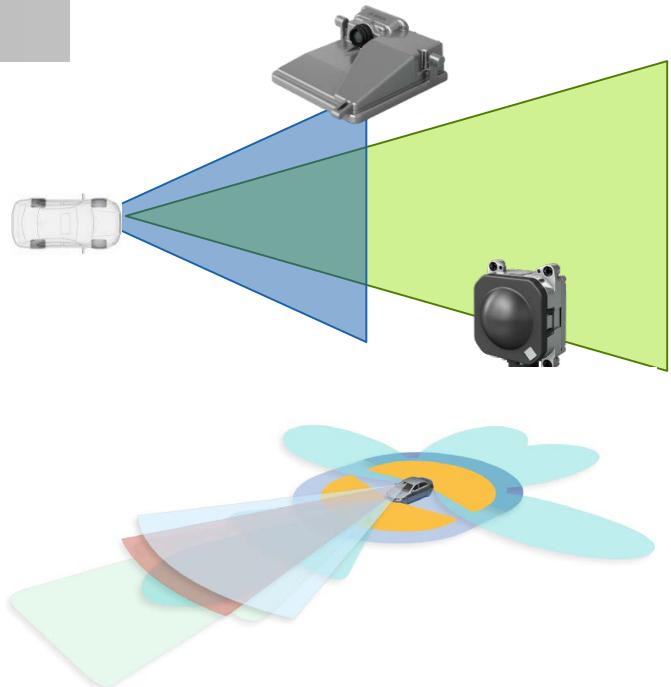
Radar Based Driver Assistance Systems

Driver Assistance Systems (DAS)

Basic terms

- Enhanced safety and driving comfort (Active safety)
- Accident-free driving
- Supports the driver at the best possible rate, especially in critical situations

- **Sensors** survey the surroundings and the interior of the vehicle
- **Control units** monitor and analyze the data of the sensors in real time



Goal:

- **reliable support** with validation by fusion of several sensors **to achieve injury, accident free and comfortable** driving

Radar Based Driver Assistance Systems

Use cases addressing end customers needs

Predictive safety

Predictive emergency braking



Evasion assistance



Lane keep assistance



Predictive pedestrian protection



Turn and crossing assistance



Driver comfort & information

Travel assistance



Driver monitoring



Light and sight assistance



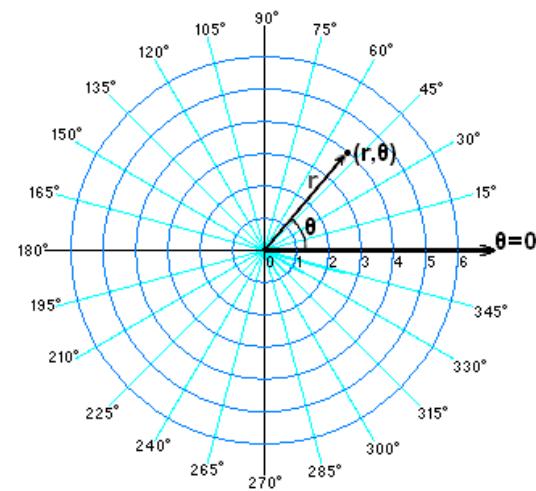
Park and maneuver assistance



Radar Based Driver Assistance Systems

What do we want to measure?

- ▶ Azimuth, range, [radial velocity]
- ▶ Traditionally the RADAR always uses a polar coordinate system
- ▶ Equivalent to a Cartesian coordinate system $[r, \Theta] \Leftrightarrow [x, y]$
- ▶ We only measure the radial velocity
 - ▶ the two components of the velocity vector in a Cartesian coordinate system can not be reconstructed (measured)

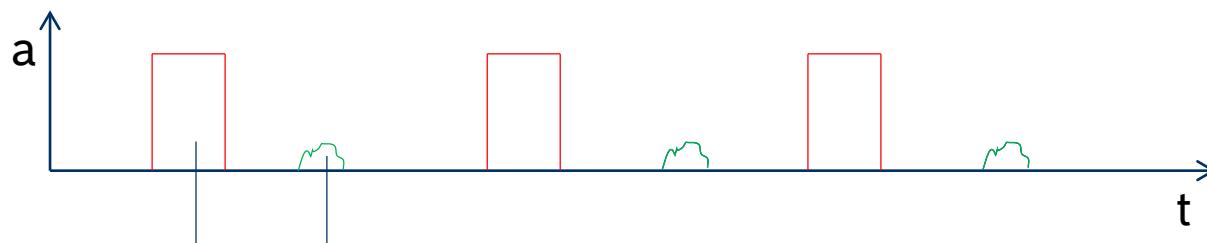


Radar Based Driver Assistance Systems

Distance Measurement Principle

- We measure the time elapsed between the transmitted pulse and the received echo

$$t_d = 2 \times \frac{D}{c} \Leftrightarrow D = t_d \times \frac{c}{2}$$



Radar Based Driver Assistance Systems

Radial Velocity Measurement Principle

$$f_d = \frac{2v_r f_{tx}}{c}$$

- ▶ f_{tx} = is the transmitters frequency
- ▶ c = is the speed of the light
- ▶ v_r = is the radial speed of the aim



© 2009 Christian Wolff

→ We know our transmit frequency, and the frequency we received from this we can measure the speed of the target object!

Radar Based Driver Assistance Systems

Radial Velocity Measurement Principle FMCW

$$f = \frac{c}{\lambda}$$

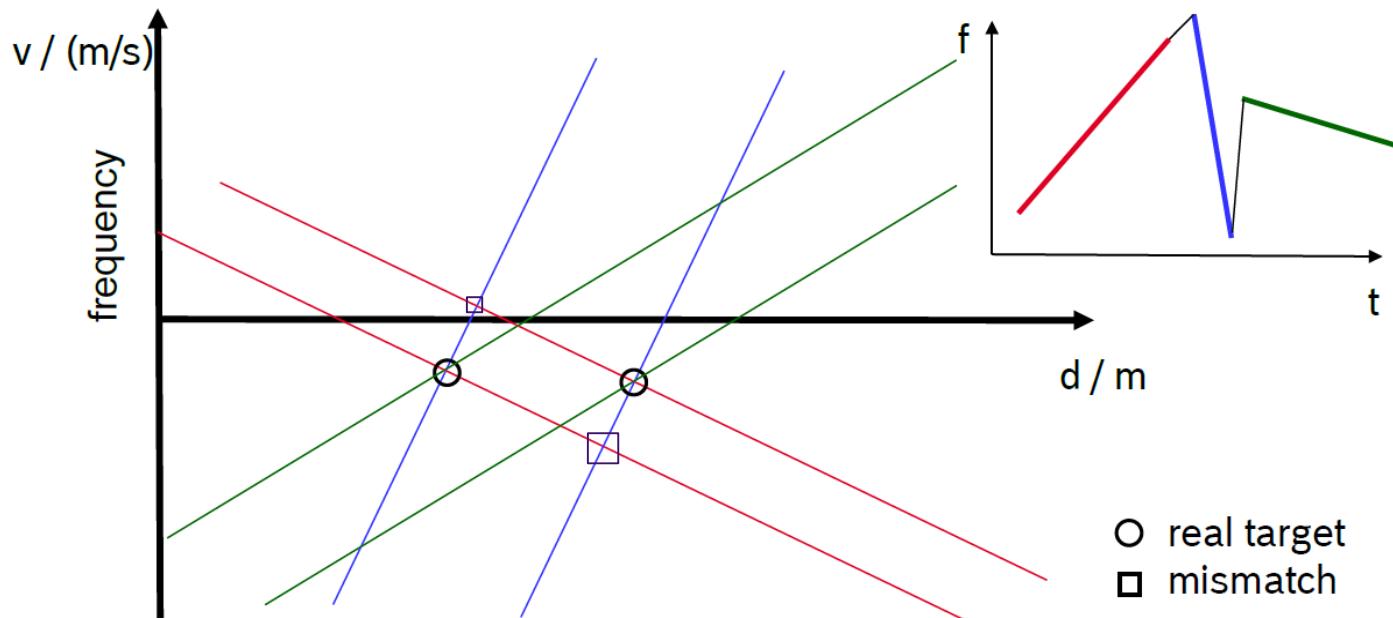
$$f_d = \frac{2D}{c} \cdot \frac{\Delta f}{\Delta t} + \frac{2 v_r f_{tx}}{c} = \frac{2D}{c} \cdot \frac{\Delta f}{\Delta t} + \frac{2v_r}{\lambda_{tx}}$$

- ▶ f_d = frequency difference of the emitted and received signals
- ▶ $\Delta f / \Delta t$ = variance of modulation in time
- ▶ v_r = is the radial speed of the aim
- ▶ λ_{tx} = frequency of emitted signal (77 GHz)

Radar Based Driver Assistance Systems

FMCW Principle

Frequency-Matching: 2 targets, 3 ramps



- Using three ramps, the method is capable of multi-target scenarios
- Using four ramps, ghost targets can be efficiently suppressed

Radar Based Driver Assistance Systems

Sensor Data Fusion

- ▶ Sensor Data Fusion consists of 3 elements:
 - ▶ Data fusion
 - ▶ Environment Model
 - ▶ Situation Interpretation
- ▶ Video / Radar / Navigation based joint architecture

Camera



- Lane markings
- Objects

Radar

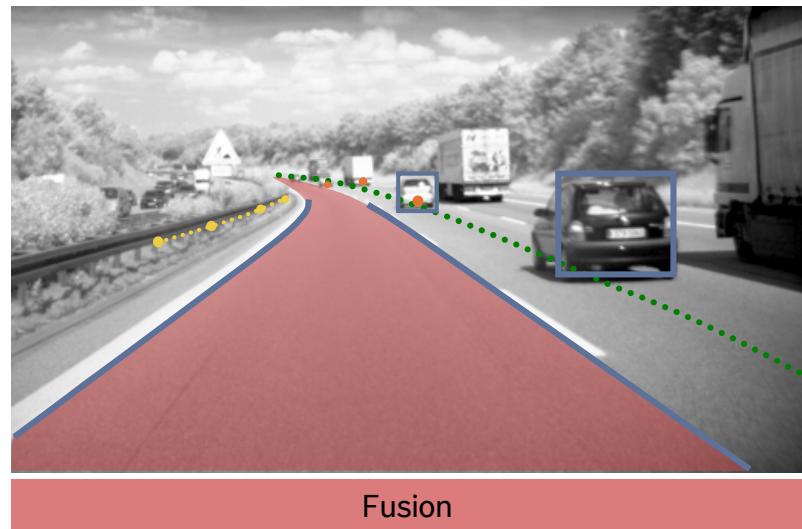


- Moving objects
- Stationary objects

Digital Map



- Roadway Attribute



Radar Based Driver Assistance Systems

ACC



Adaptive Cruise Control

Radar Based Driver Assistance Systems

Adaptive Cruise Control

► Goal

- Blue vehicle should always keep a secure distance to the yellow vehicle while keeping the set speed, or the speed of the yellow vehicle

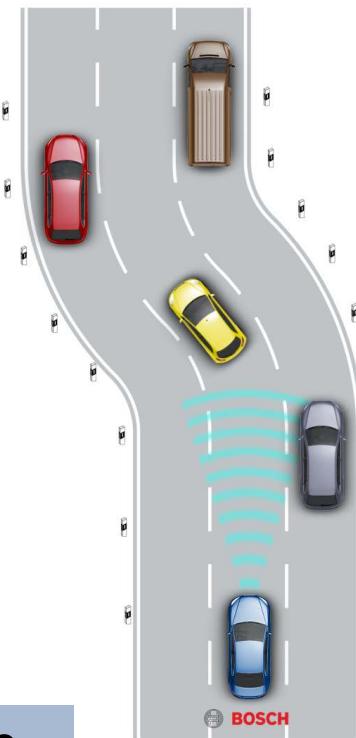
► Inputs

- Radar data
- Additional video data
- Ego car data

► Reaction

- Acceleration or deceleration within secure limits.

→ **Achieve comfortable driving through automatic longitudinal control**



Radar Based Driver Assistance Systems

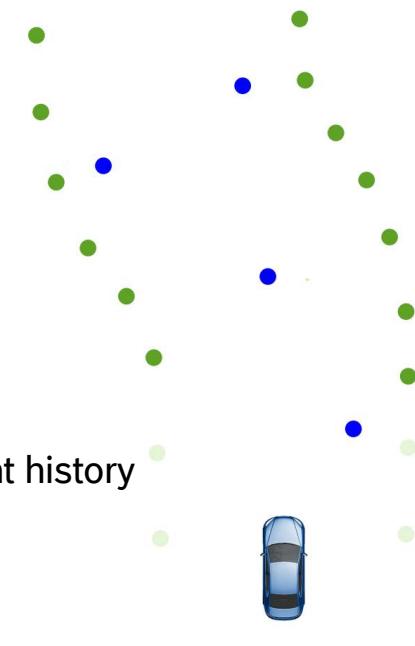
Radar points

- ▶ **Object types**

- ▶ Stationary objects
- ▶ Dynamic objects

- ▶ **Road estimation is based on**

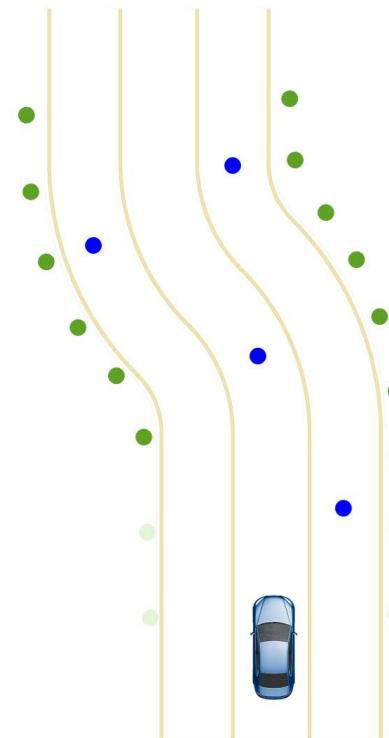
- ▶ the connection of stationary objects
- ▶ the tracking of moving objects(+ ego)
 - Creating lanes based on the object and ego movement history



Radar Based Driver Assistance Systems

Radar-Video Fusion

- ▶ Road estimation in the case of video fusion
 - ▶ Use of road markings (lines) from video based driver assistance systems
 - ▶ Object classification by video

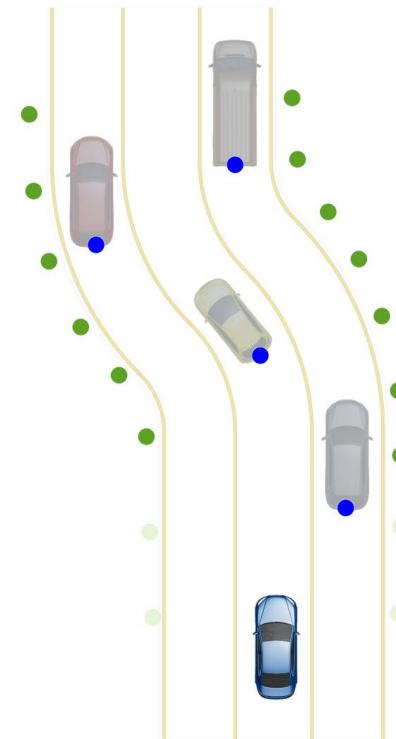


Radar Based Driver Assistance Systems

Object classification

- ▶ Classification of objects in both sensors:
 - ▶ Radar classification based on the behavior of the objects
 - ▶ Video classification based image features

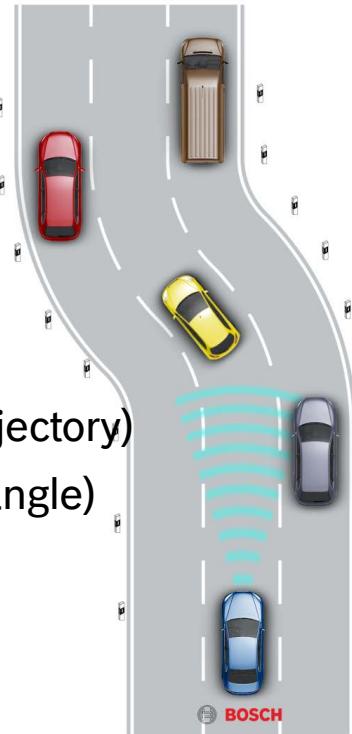
- ▶ Fusion of video based information and radar based information in one system in order to get reliable data



Radar Based Driver Assistance Systems

Parallel lanes

- ▶ Yellow vehicle remains the ACC target object because
 - ▶ Connection of stationary objects (reflector posts, guardrail)
 - ▶ Video line detection and lane recognition
 - ▶ Tracking of red and brown vehicle
 - ▶ the target object position, direction and trajectory
- ▶ The information coming from target object(position, trajectory)
- ▶ The information from the ego vehicle(yaw angle, s.w. angle)
- ▶ Road topology
 - >Indicates how the road ahead looks like



Radar Based Driver Assistance Systems



Automatic Emergency Braking

Radar Based Driver Assistance Systems

Automatic Emergency Breaking

► Goal

- Fast reaction to avoid collision (or mitigate a collision)

► Input

- Ego Motion
- Object type classification(ped., cyclist, etc.)
- Motion model for various object types
- Calculate time to collision
- Additional information from the driver
 - Driver monitoring to estimate the level of attention:
 - intention to brake, intention to evade

► Reaction

- Collision avoidance/mitigation with braking/steering
- **Achieve safer driving through automatic braking**



Radar Based Driver Assistance Systems

Pedestrian Protection

► Goal

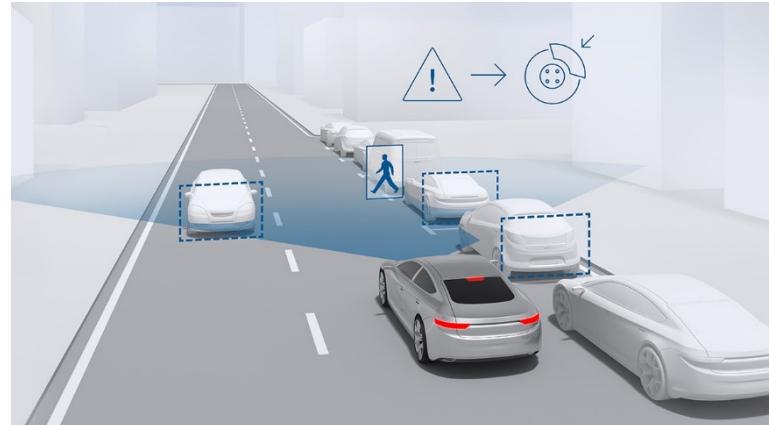
- Fast reaction to avoid collision

► Input

- Ego Motion
- Object type classification
- Micro Doppler information
- Calculate time to collision
- Trajectory overlap

► Reaction

- Collision avoidance/mitigation with braking/steering

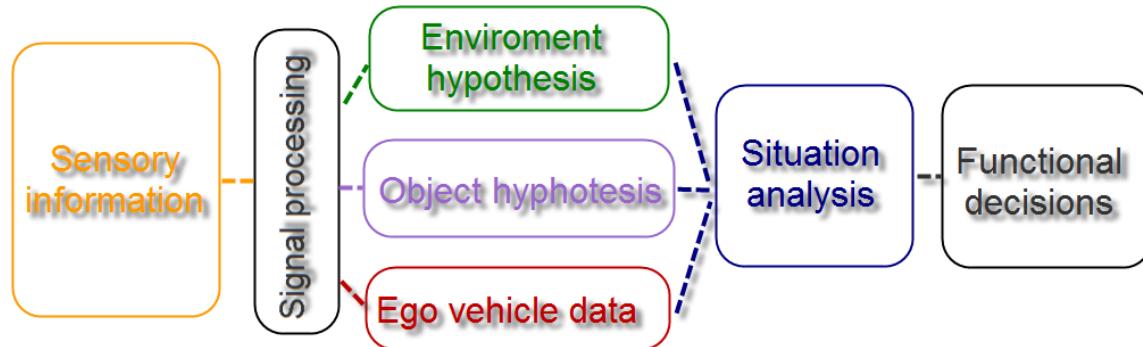


→ Achieve safe driving through automatic braking

Radar Based Driver Assistance Systems

System Approach

- ▶ **Data Fusion from different sources (e.g. Radar, Video, Ultrasonic)**
 - ▶ Objects, line, lane, road signs etc.
- ▶ **Environmental Hypothesis**
 - ▶ E.g.: Parallel Lanes, Object-Lane association
- ▶ **Situation Analysis**
 - ▶ Criticality of the situation, Driver Activity
- ▶ **Decision**
 - ▶ Warning, Partial Braking, Brake Support, Full Emergency Braking



Predictive Pedestrian Protection

www.bosch.com

This movie shows
proprietary Bosch
technology.

The vehicle shown was
chosen solely as a carrier
for demonstration purposes.



Radar Based Driver Assistance Systems



Thank you for your attention!

VIDEO SENSORS

Catalin Golban, Anca Foloba

CONTENTS

CONTENTS

Slide structure

1. Introduction to video sensors
2. What are images and how are they represented?
3. Calibration
4. Distortions
5. Projective geometry
6. 3D Reconstruction

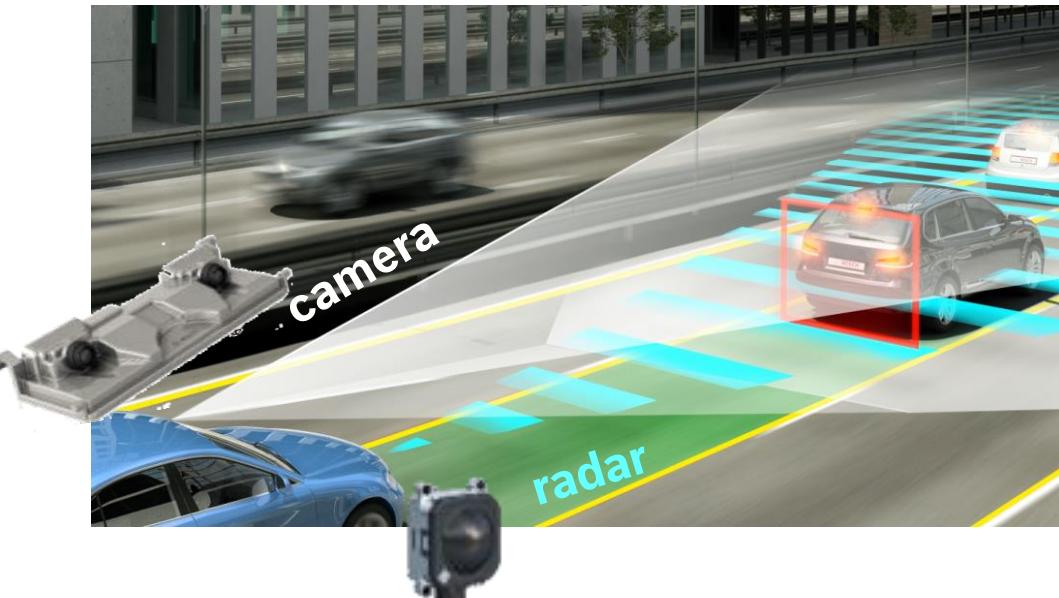
INTRODUCTION TO VIDEO SENSORS

Bosch Engineering Center Cluj

Automated driving activities



SOFTWARE ENGINEERING



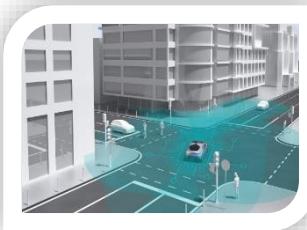
Radar Systems



Video Systems



Connectivity



Central processing unit



Ultrasonic Systems



Electric Power Steering

INTRODUCTION TO VIDEO SENSORS

What are we doing in Cluj?

- main responsibilities on video area
 - Software development and pre-development for mono and stereo video systems
 - Computer vision, image processing and machine learning algorithms development for driver assistance and automated driving



stereo
camera



mono
camera

INTRODUCTION TO VIDEO SENSORS

What algorithms are we developing in Cluj – some examples

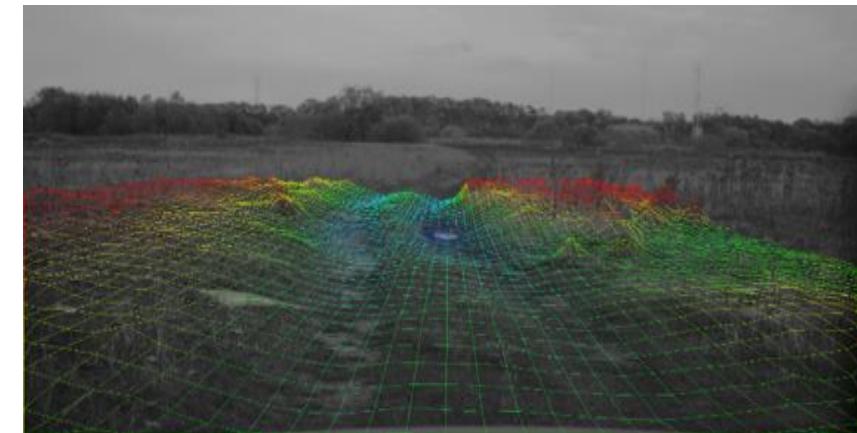
visual odometry



3D free-space

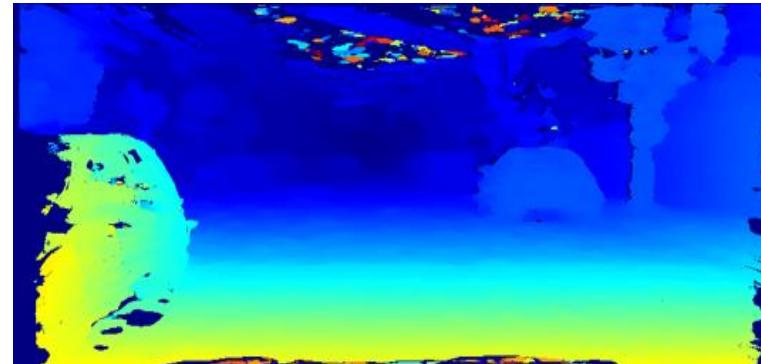
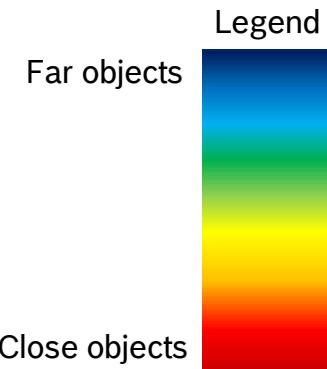


off-road

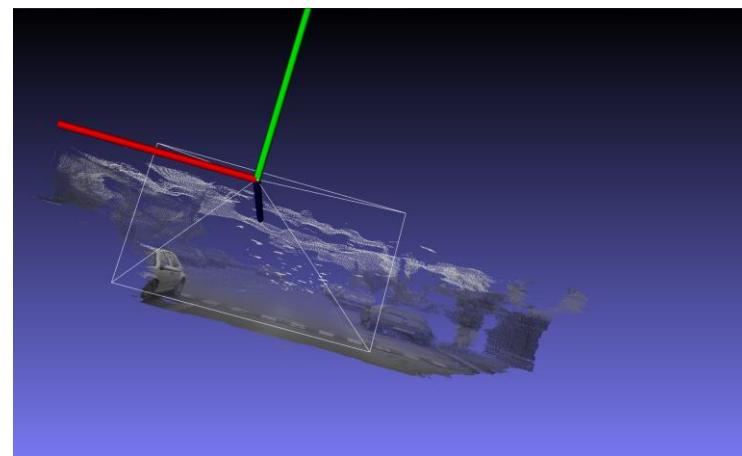


INTRODUCTION TO VIDEO SENSORS

Goal for today



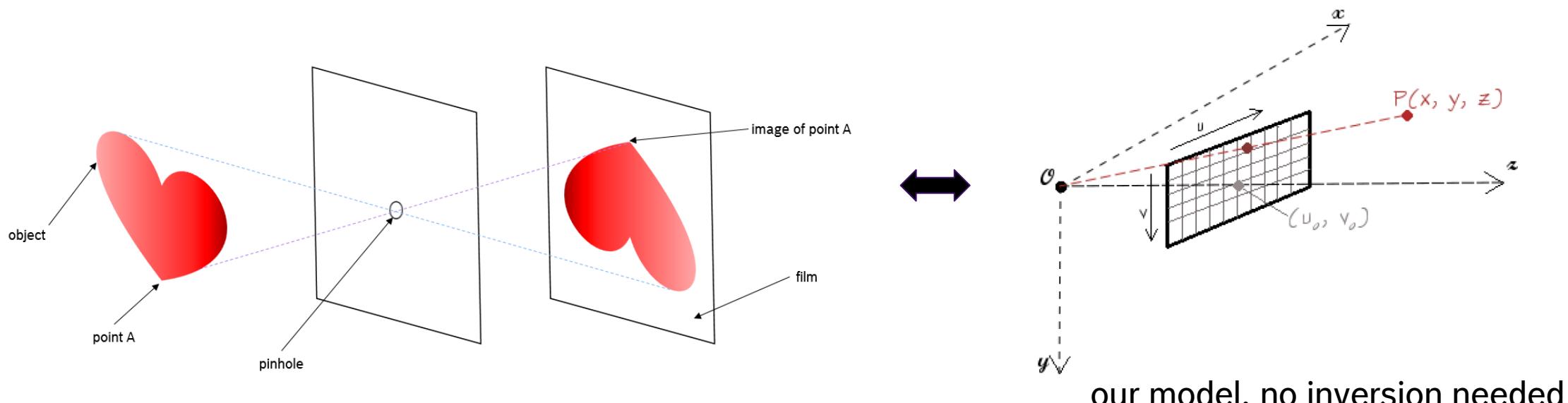
3D reconstruction



INTRODUCTION TO VIDEO SENSORS

Pinhole camera & geometric model

- pinhole camera model – the light passes through the pinhole and projects an inverted image → it has small amount of light



our model, no inversion needed

WHAT ARE IMAGES AND HOW ARE THEY REPRESENTED?

WHAT ARE IMAGES AND HOW ARE THEY REPRESENTED?

What is an image?

- image – depicts the visual perception which has the appearance of some object, person, landscape, etc.
- an image can be two-dimensional such as a photograph, or three-dimensional such as a hologram or statue
- moving image – a video



Hologram – Source – <https://www.technobuffalo.com/wp-content/uploads/2015/12/Hologram-Pyramid-1.jpeg>

WHAT ARE IMAGES AND HOW ARE THEY REPRESENTED?

How is an image represented?

Photograph

- photograph – image created using a camera
- the light is captured by the camera – it falls on a light-sensitive surface and then it is encoded in a digital format



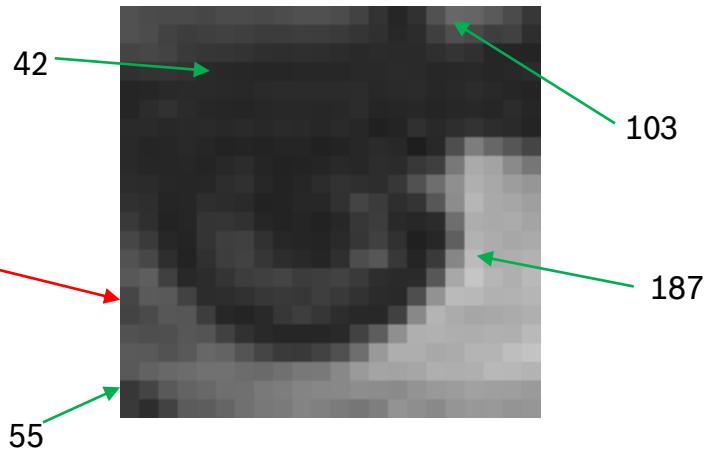
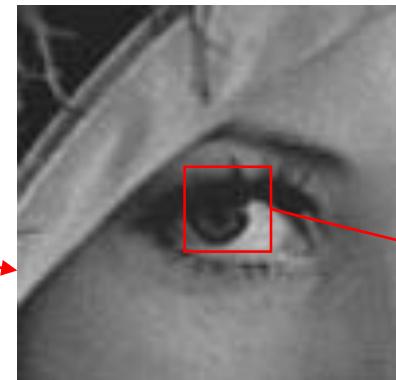
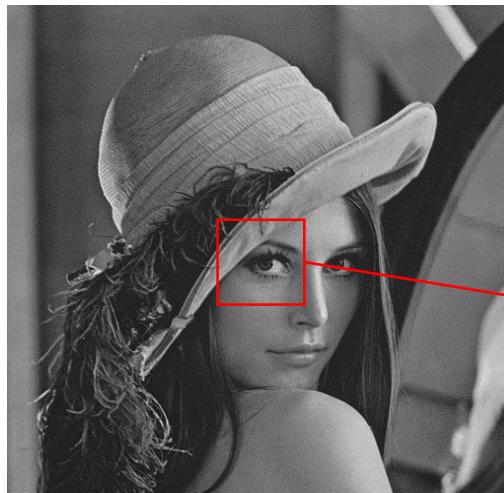
Photograph

WHAT ARE IMAGES AND HOW ARE THEY REPRESENTED?

How is an image represented?

Digital image

- numeric representation of an image
- represented as a vector or as a matrix
- each numerical value corresponds to a single pixel from the image



WHAT ARE IMAGES AND HOW ARE THEY REPRESENTED?

How is an image represented?

Digital image

- most common model to represent a color image is **RGB** (**R**ed, **G**reen, **B**lue) → each pixel is represented by three values – amount of red, green and blue
- this will use more amount of memory than gray-scale images



WHAT ARE IMAGES AND HOW ARE THEY REPRESENTED?

How is an image represented?

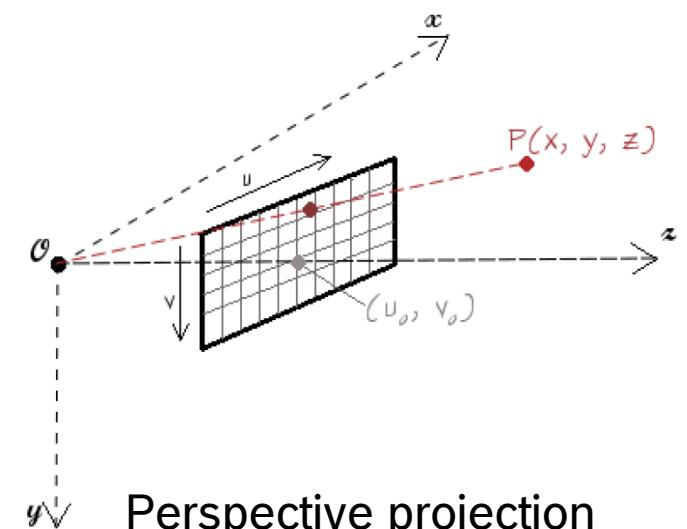


CALIBRATION

CALIBRATION

General concepts

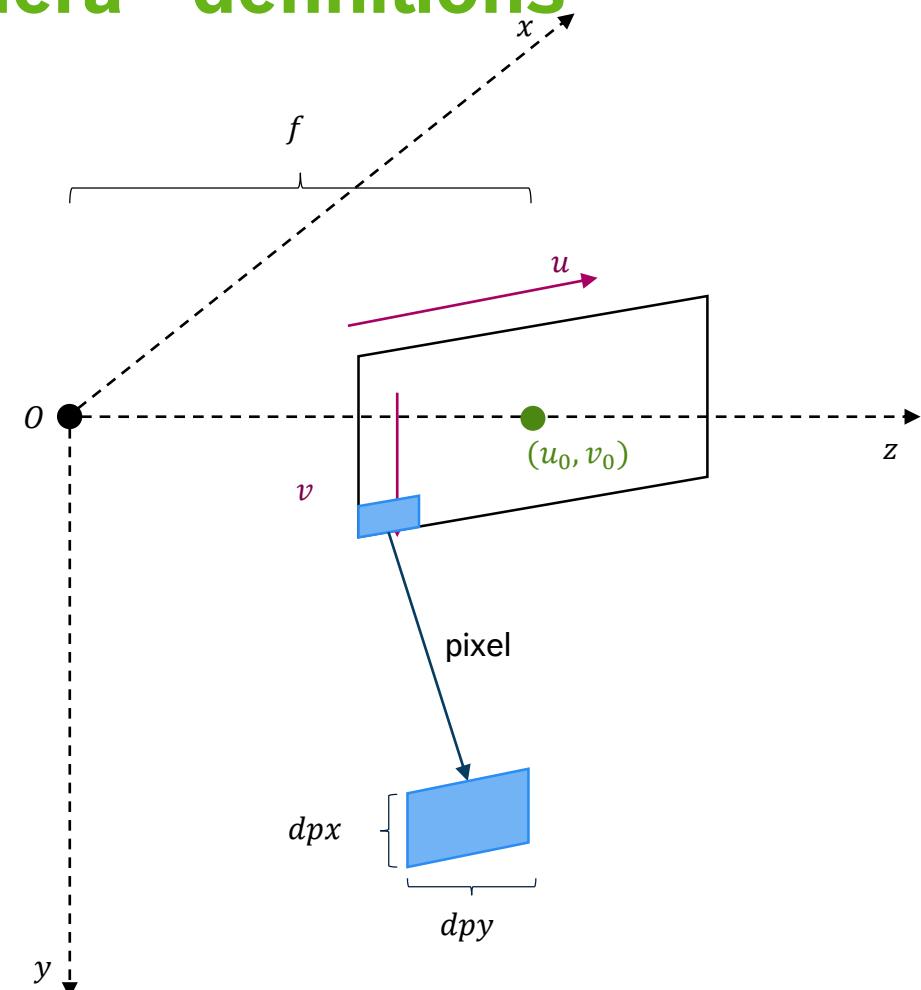
- **Computer vision** – getting 3D information from 2D image data → **the parameters of the model** used for transformation of the 3D points to 2D pixels must be known
- Calibration needed in order to obtain a good projection of the world in the image plane
 - **Intrinsic** parameters
 - **Extrinsic** parameters
- Calibration
 - **Static calibration** (happens before the actual usage of the video sensor)
 - Online (while driving the de-calibration is detected)
 - geometric model is not fixed, it changes in time => **online calibration**



CALIBRATION

Intrinsic parameters of the camera - definitions

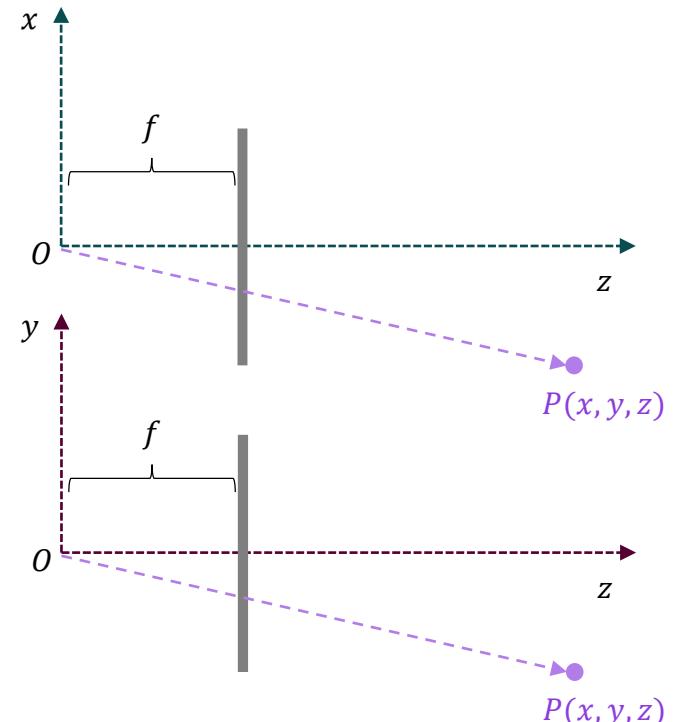
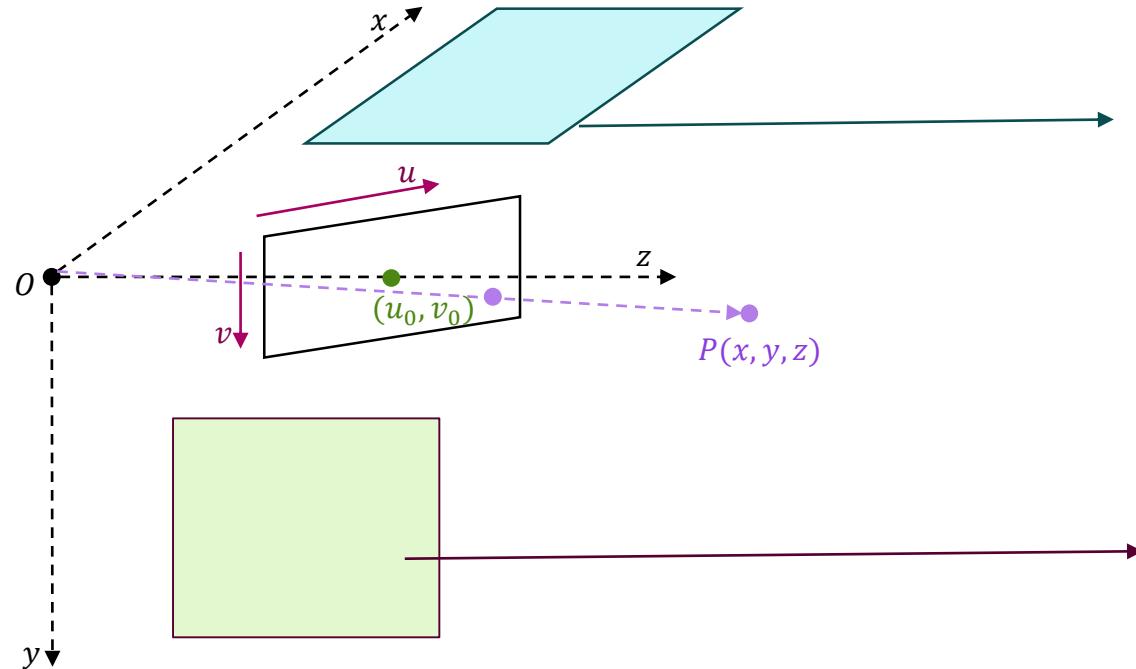
- O – camera center
- u_0, v_0 - principal point
- u, v – image coordinates
- f – focal length – distance from O to principal point
- dpx, dpy – size of pixels
- f_x, f_y - focal length in pixels units



3D RECONSTRUCTION

Intrinsic parameters of the camera – image plane

- the problem presented in the XOZ plane – compute the u coordinate for the 3D point
- the problem presented in the YOZ plane – compute the v coordinate for the 3D point



3D RECONSTRUCTION

Intrinsic parameters of the camera – image plane

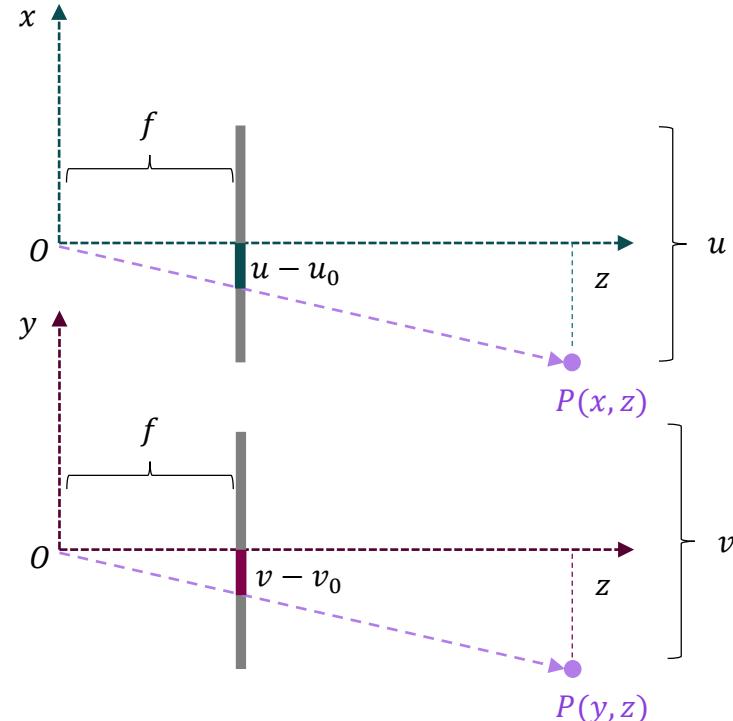
- the u and v can be obtained using similar triangles

$$\frac{(u - u_0) \cdot dpx}{f} = \frac{x}{z}$$

$$\frac{(u - u_0)}{f_x} = \frac{x}{z}$$

$$\frac{(v - v_0) \cdot dpy}{f} = \frac{y}{z}$$

$$\frac{(v - v_0)}{f_y} = \frac{y}{z}$$



CALIBRATION

Intrinsic parameters of the camera – matrix form

Intrinsic parameters

- there are 5 intrinsic parameters, which describes the internal geometry and optical characteristics of the camera – focal length, image sensor format and principal point

- it can be modelled using the following matrix $K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$, where:

- $f_x = f \cdot dpx$ and $f_y = f \cdot dpy$ represent the focal length in term of pixels, with dpx and dpy being the scale factors and f the focal length
- u_0 and v_0 represent the principal point
- Matrix based projection formula (equivalent to the one from the previous slide):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

CALIBRATION

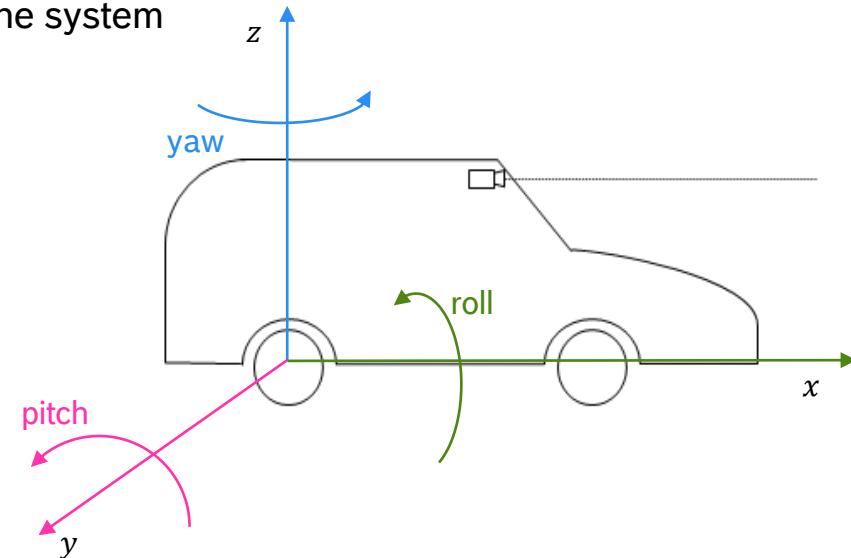
Parameters of the camera

Extrinsic parameters

- represent position and orientation of the camera in the car (world) coordinate system
 - Convention – use the center of the rear axis
- R matrix for each axis and T represents the translation vector applied on the system

- they can be written in the following form $[R|T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$
- there are three rotations applied in 3D space – **pitch**, **roll** and **yaw**

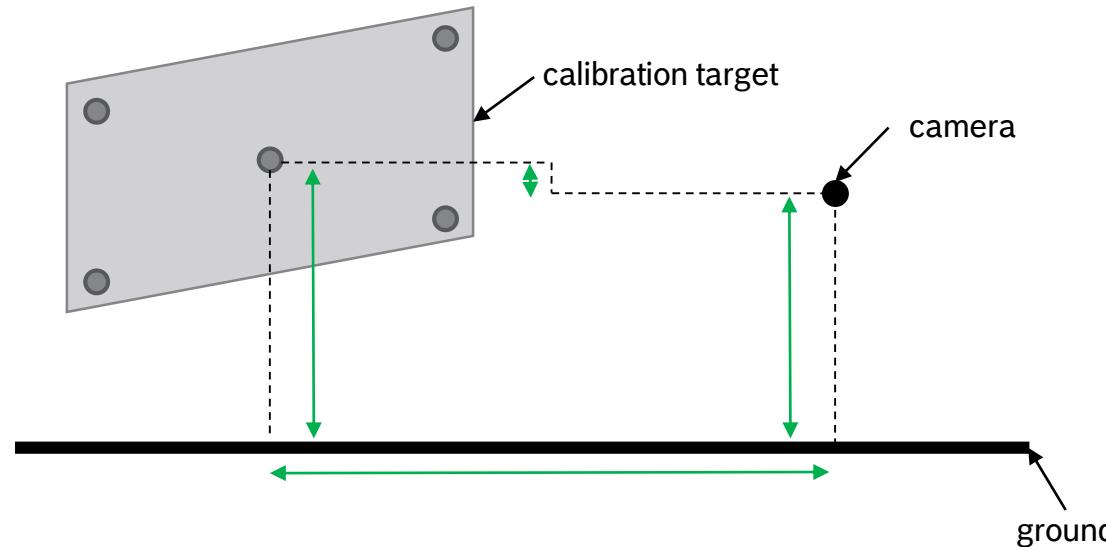
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \cdot \left(R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \right) = K \cdot [R|T] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



CALIBRATION

Static calibration

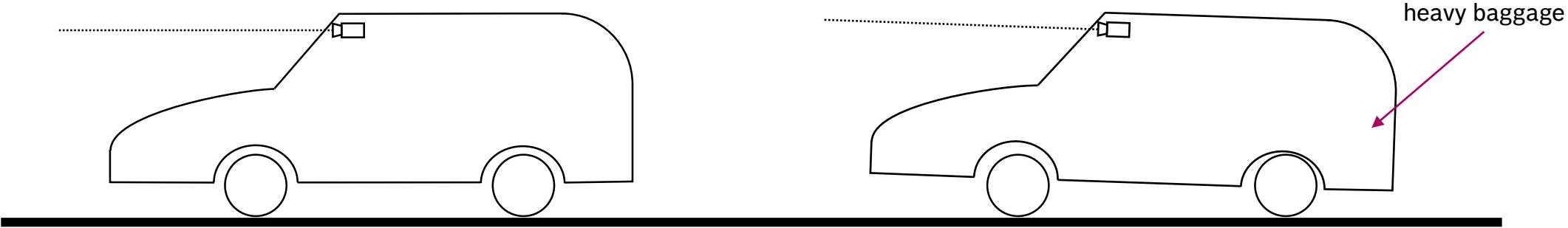
- align the camera towards a static calibration target with a specific visual pattern
- camera detects the calibration target in the image – sets the detected position in relation to the target's expected position – computes the intrinsic and extrinsic parameters



CALIBRATION

Online calibration

- starts from knowing the exact position of the camera with respect to the world system of coordinates
- detect a de-calibration of the camera as soon as possible since an initially determined calibration may change dynamically due to external influences (ex; heavy baggage)

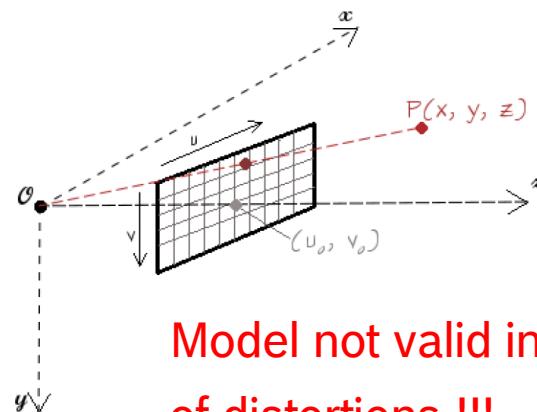


DISTORTION

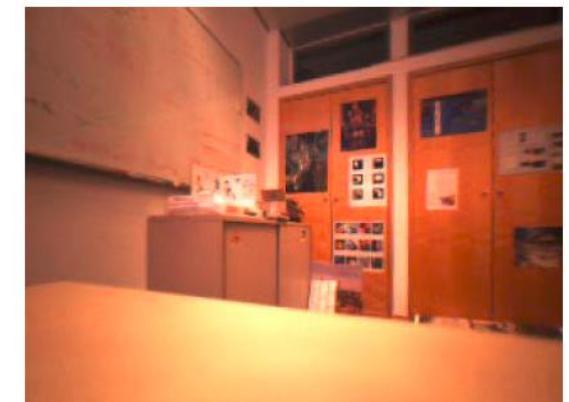
DISTORTION

General aspects

- distortion – alteration of the original image
- In the presence of distortions the perspective projection model does not hold



Model not valid in the presence
of distortions !!!



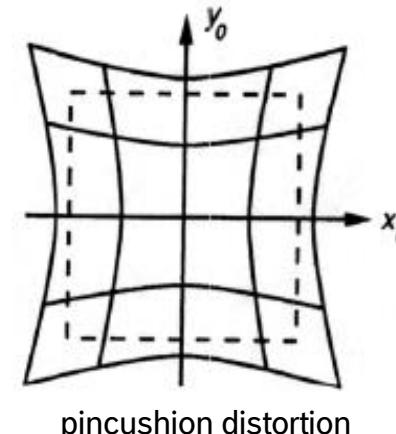
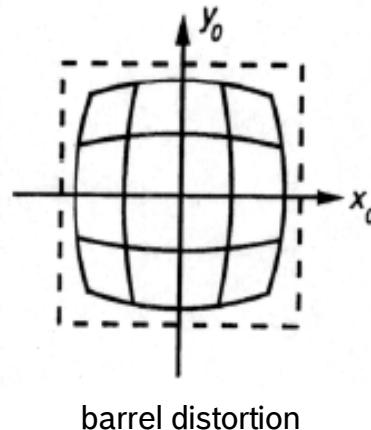
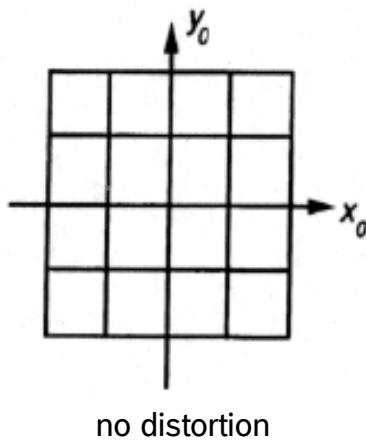
Radial Distortion – Source – Dynamic Vision, T. Schon

DISTORTION

Types of distortion

1. Radial distortion

- induced by the curve of the lens
- there are two types – barrel distortion or pincushion distortion

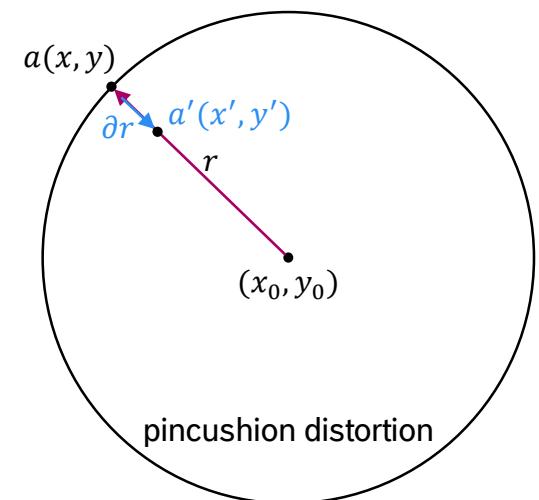
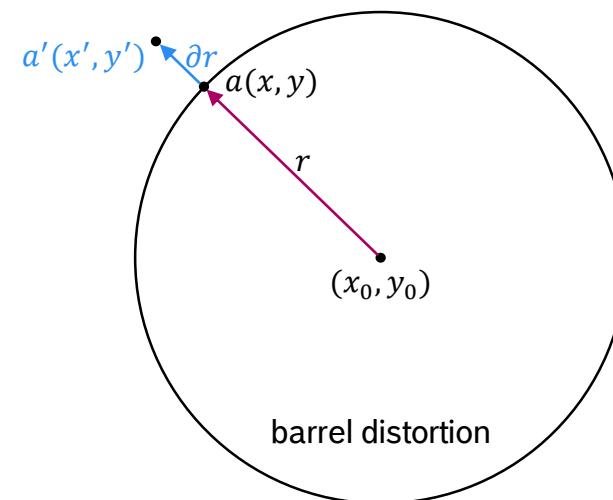
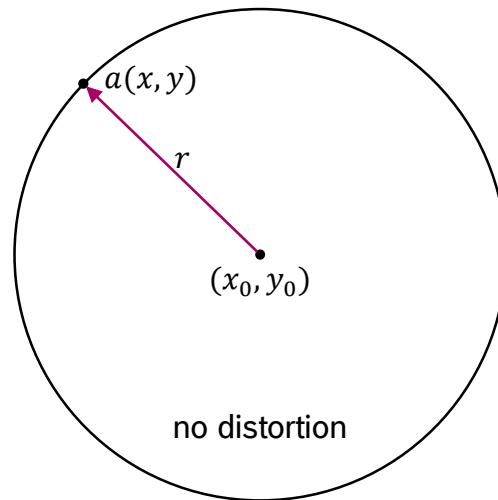


DISTORTION

Types of distortion

1. Radial distortion

- it can be modeled using the following equation: $\begin{bmatrix} x_{undistorted} - x_{distorted} \\ y_{undistorted} - y_{distorted} \end{bmatrix} = \begin{bmatrix} x \cdot (k_1 \cdot r^2 + k_2 \cdot r^4 + \dots) \\ y \cdot (k_1 \cdot r^2 + k_2 \cdot r^4 + \dots) \end{bmatrix}$, where $r^2 = x^2 + y^2$ and k_1, k_2, \dots are the coefficients of radial distortion

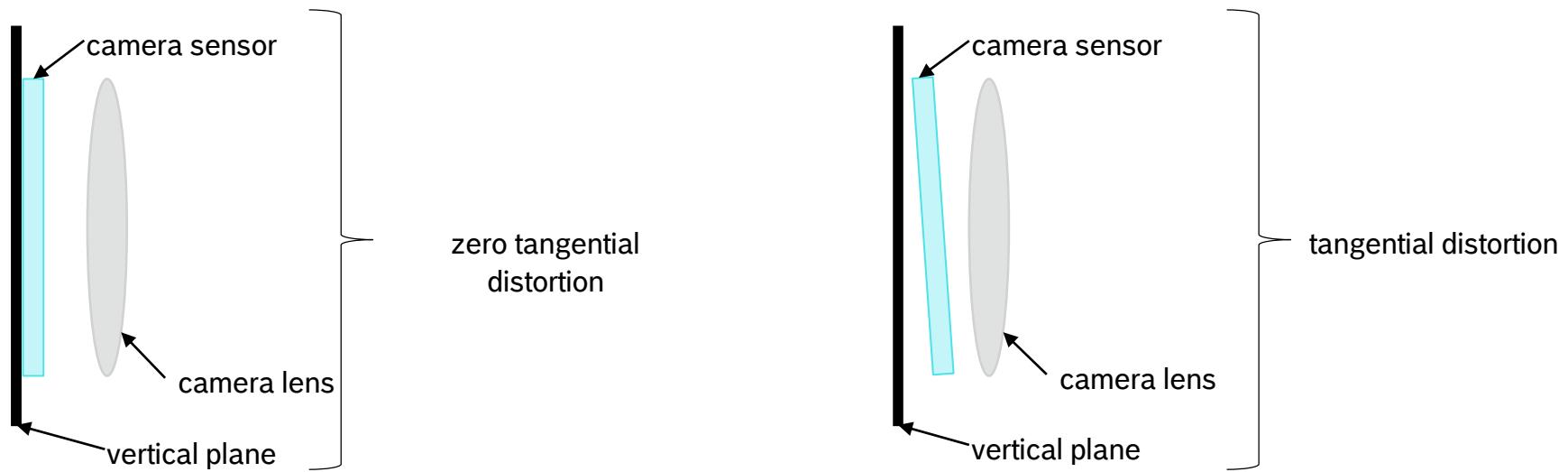


DISTORTION

Types of distortion

2. Tangential distortion

- induced by the misalignment of the center of camera lens and camera sensors

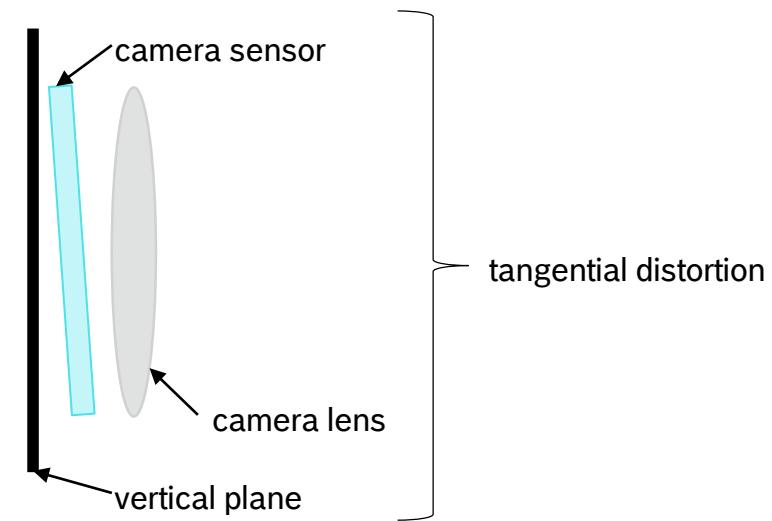
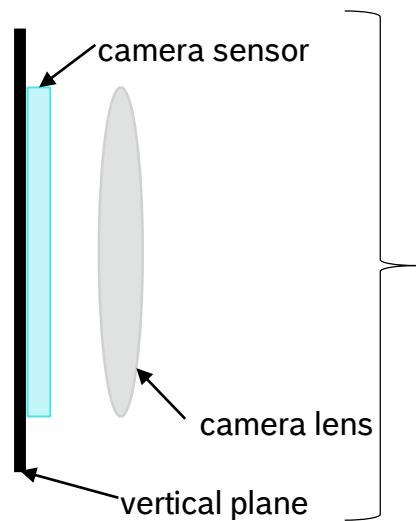


DISTORTION

Types of distortion

2. Tangential distortion

- it can be modeled using: $\begin{bmatrix} x_{undistorted} - x_{distorted} \\ y_{undistorted} - y_{distorted} \end{bmatrix} = \begin{bmatrix} 2 \cdot p_1 \cdot x \cdot y + p_2 \cdot (r^2 + 2 \cdot x^2) \\ p_1 \cdot (r^2 + 2 \cdot y^2) + 2 \cdot p_2 \cdot x \cdot y \end{bmatrix}$, where p_1, p_2 are the coefficients of tangential distortion

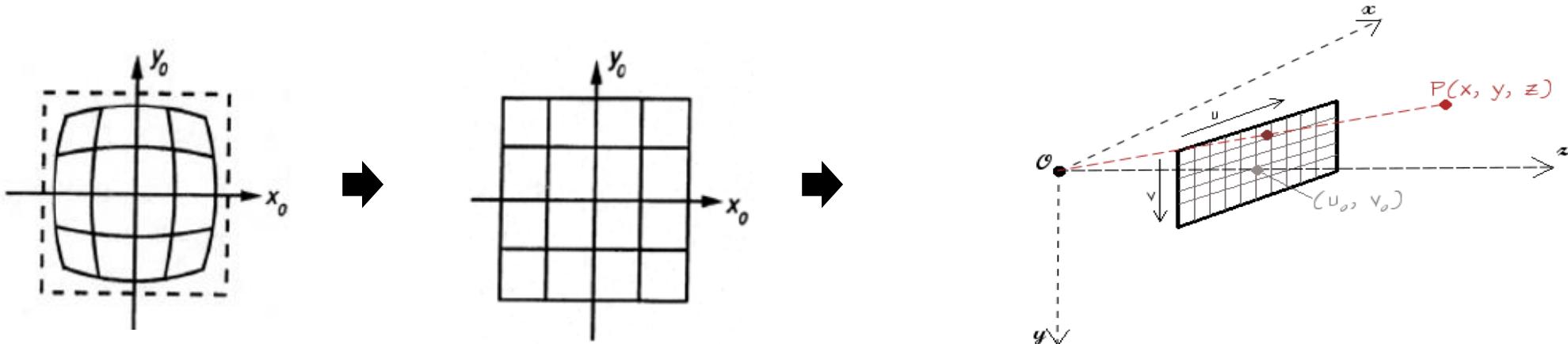


PROJECTIVE GEOMETRY

PROJECTIVE GEOMETRY

General aspects

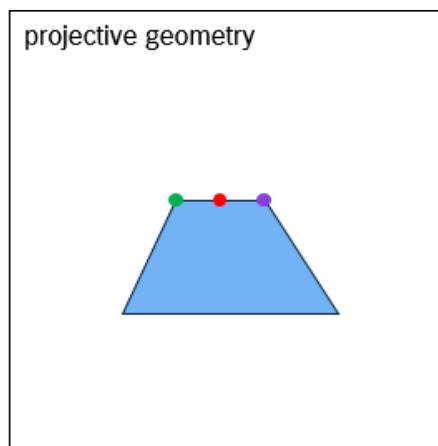
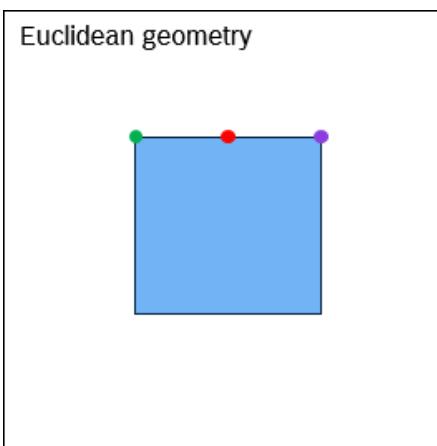
- a branch of geometry which deals with the properties of the objects that are invariant under projective transformations
- this can be applied after the elimination of the lens distortions (i.e. when the image formation model can be modeled based on a perspective projection)
- projective geometry is a very useful tool in computer vision



PROJECTIVE GEOMETRY

General aspects

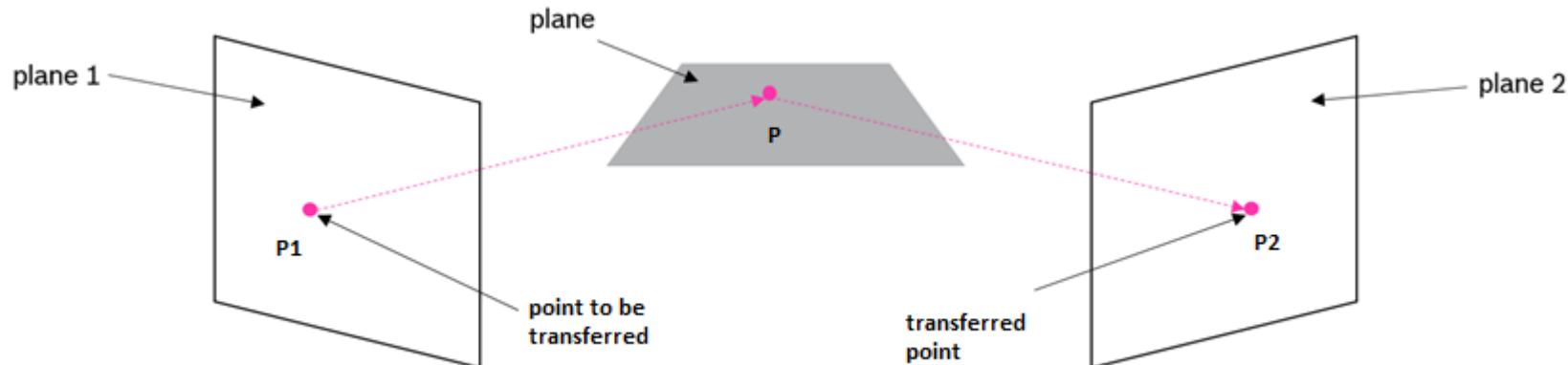
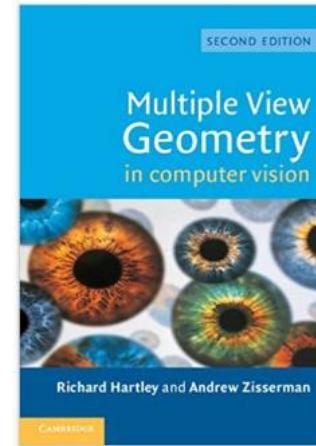
- difference between Euclidean and projective geometry:
 - Euclidean geometry – describe objects “as they are” (unchanged by rigid motions)
 - projective geometry – describe objects “as they appear”



PROJECTIVE GEOMETRY

References and example result from projective geometry

- <https://www.amazon.com/Multiple-View-Geometry-Computer-Vision/dp/0521540518>
- <http://robotics.stanford.edu/~birch/projective/>
- <http://robotics.stanford.edu/~birch/projective/projective.pdf>
- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>
- <http://mathworld.wolfram.com/ProjectiveGeometry.html>



H_{image}, H_1, H_2 – homography matrices

Transfer via plane
For details see the references

$$P_1 = H_1 \cdot P$$

$$P_2 = H_2 \cdot P$$

$$H_{image} = H_2 \cdot H_1^{-1}$$

$$P_2 = H_{image} \cdot P_1$$

3D RECONSTRUCTION

3D RECONSTRUCTION

General concepts

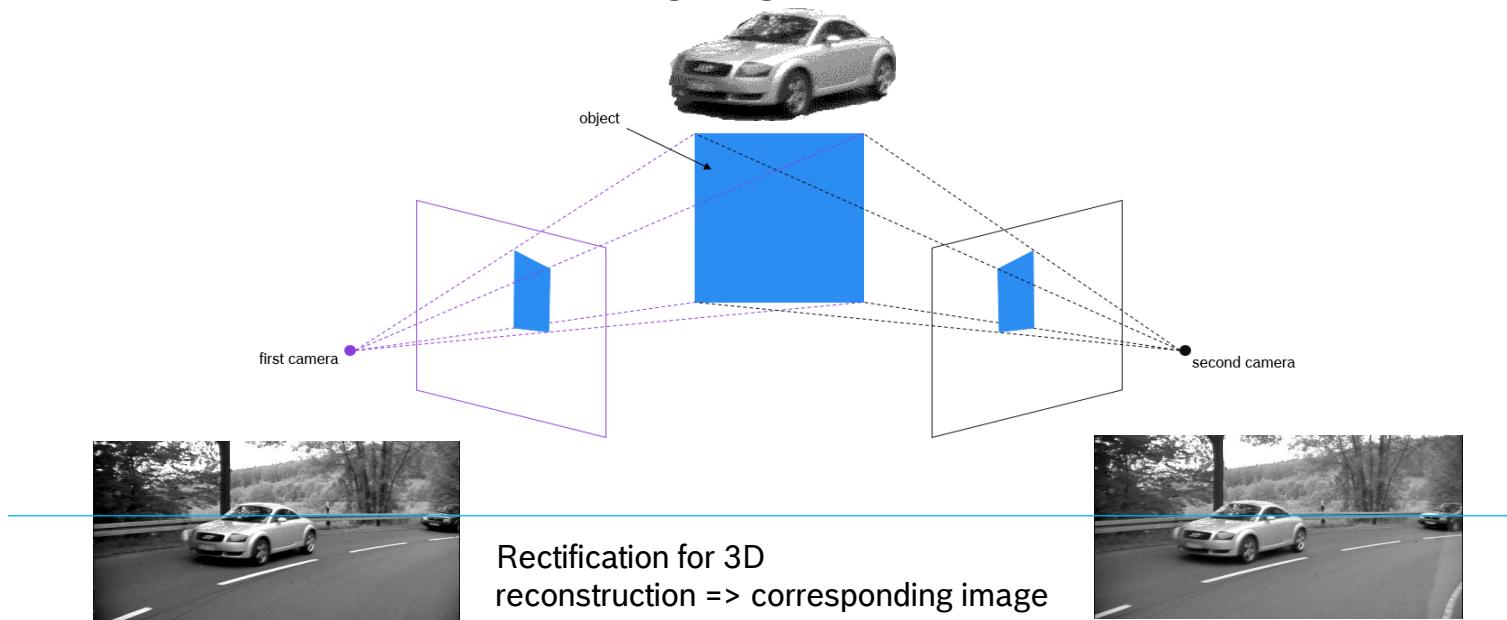


stereo
camera



mono
camera

- 3D reconstruction – the process of creating the 3D shape and position of real objects from images
- in computer vision for automated driving
 - using the stereo system - two cameras from different positions, targeting the same scene
 - using the mono system - same camera, targeting the same scene at different points in time



3D RECONSTRUCTION

Inverse problem

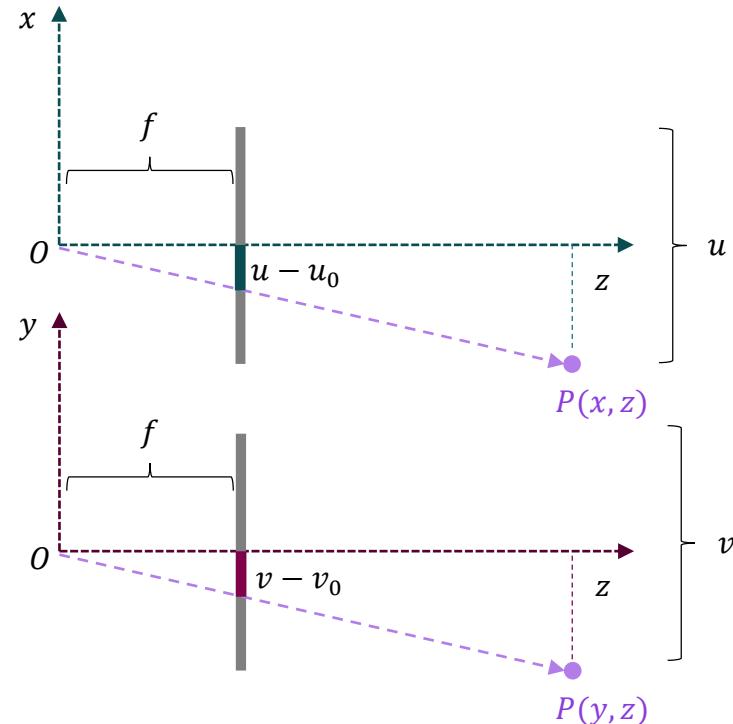
- compute x and y coordinates

$$\frac{(u - u_0)}{f_x} = \frac{x}{z}$$

$$\frac{(u - u_0) \cdot \textcolor{teal}{z}}{f_x} = x$$

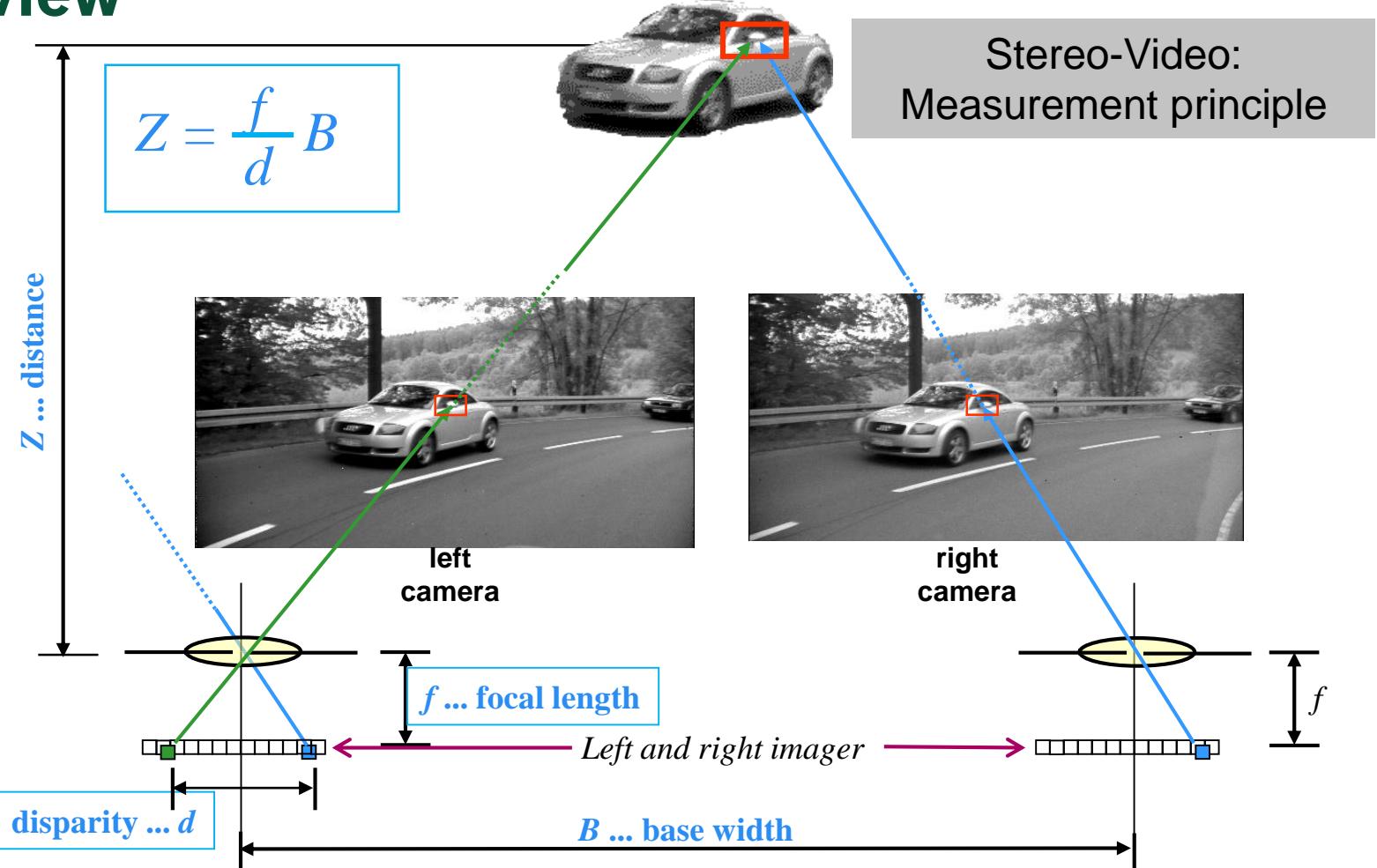
$$\frac{(v - v_0)}{f_y} = \frac{y}{z}$$

$$\frac{(v - v_0) \cdot \textcolor{violet}{z}}{f_y} = y$$



3D RECONSTRUCTION

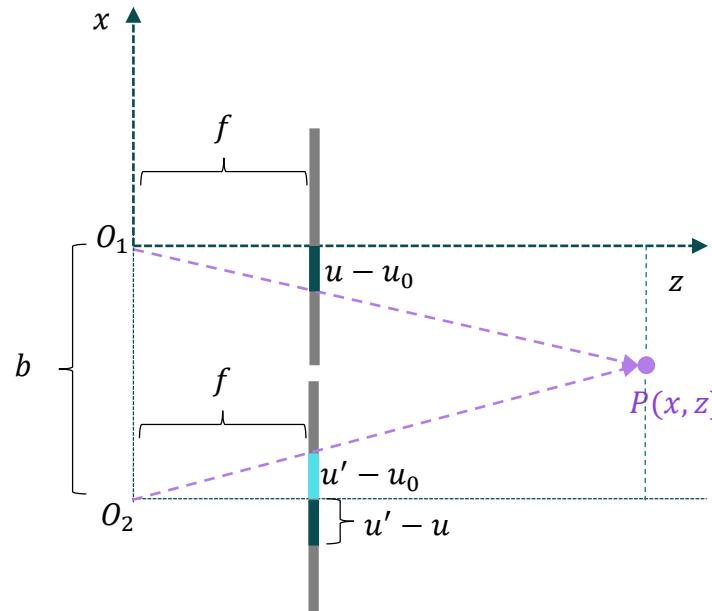
Disparity review



3D RECONSTRUCTION

Disparity

- used to determine the depth of an object using the stereo system – we assume that the two cameras are already rectified



$$\frac{(u - u_0)}{f_x} = \frac{x}{z}$$

$$\frac{(u' - u_0)}{f_x} = \frac{x + b}{z}$$

$$\frac{(u' - u)}{f_x} = \frac{b}{z}$$

$$\frac{f_x \cdot b}{u' - u} = z$$

$$\frac{f_x \cdot b}{d} = z$$

$d \rightarrow$ disparity \rightarrow displacement of the projection between the left & right image

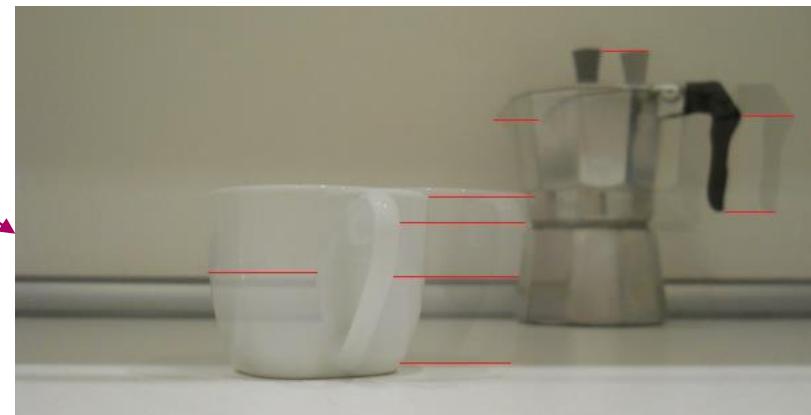
3D RECONSTRUCTION

Disparity



left image

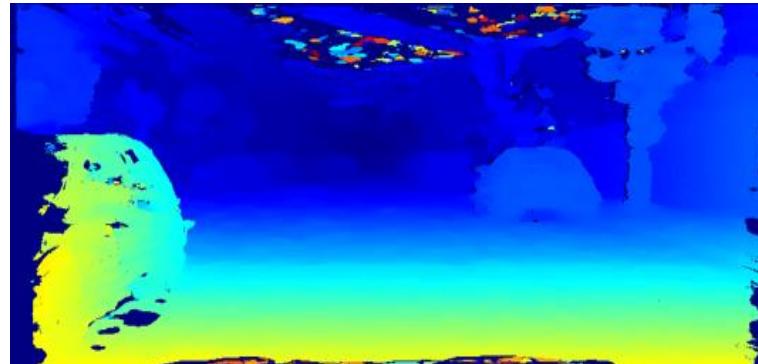
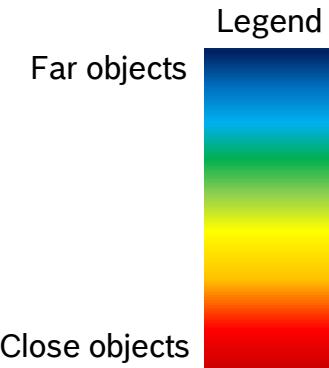
right image



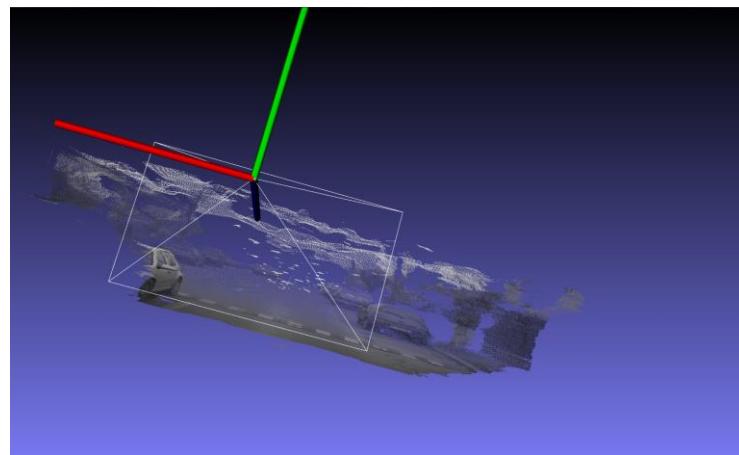
Disparity is small in far range
and big in close range

Disparity = displacement between left & right images

3D RECONSTRUCTION Disparity



3D reconstruction



Disparity is small in far range
and big in close range

Thank you for your attention!



VIDEO SENSORS II

Catalin Golban

CONTENTS

CONTENTS

Course structure

1. Remember from last course and goals for today
2. Theoretical refresh: vectors, dot product, cross product, coordinate systems, transformations
3. Essential matrix: scope, proof
4. Fundamental matrix
5. Fundamental matrix estimation
6. Conclusions

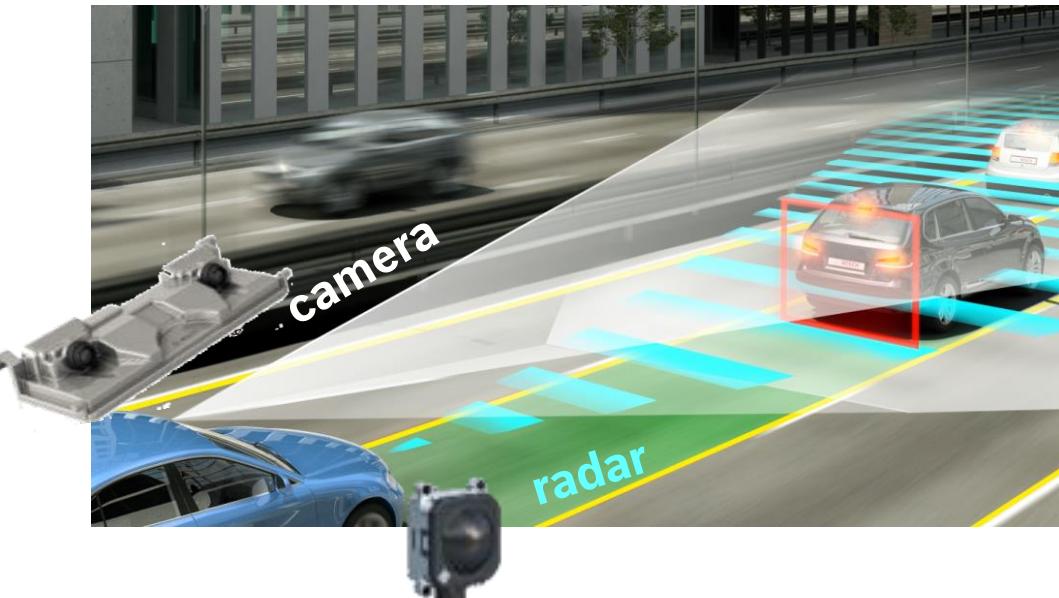
REVIEW OF THE LAST COURSE AND GOALS FOR TODAY

Bosch Engineering Center Cluj

Automated driving activities



SOFTWARE ENGINEERING



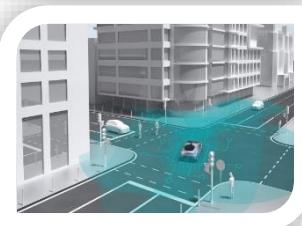
Radar Systems



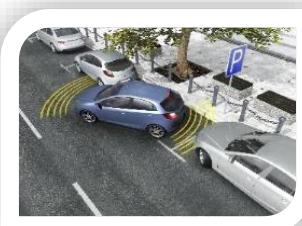
Video Systems



Connectivity



Central processing unit



Ultrasonic Systems



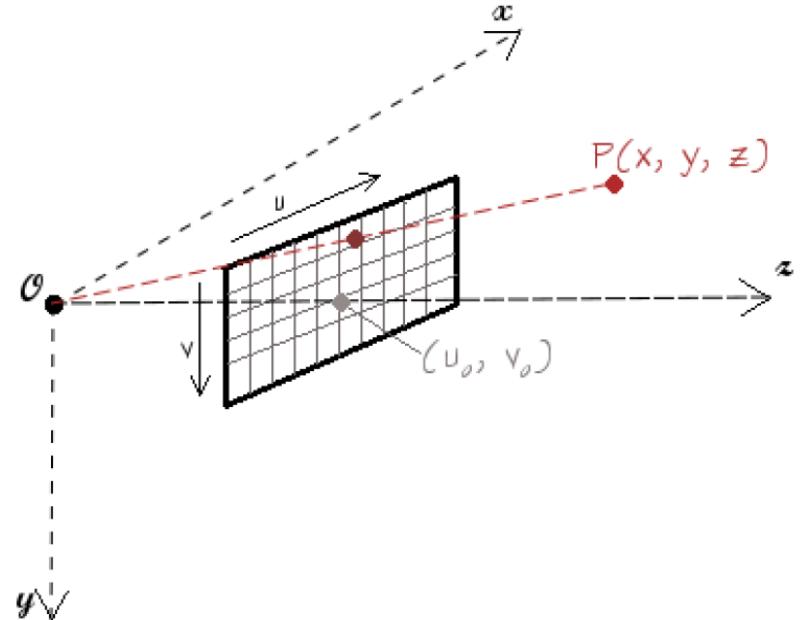
Electric Power Steering

Review of the last course and goals for today

Review - intrinsic parameters of the camera

- O – camera center
- u_0, v_0 - principal point
- u, v – image coordinates
- f – focal length – distance from O to principal point
- dpx, dpy – size of pixels
- f_x, f_y - focal length in pixels units

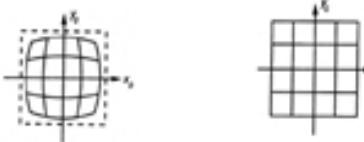
$$f_x = \frac{f}{dpx} \quad f_y = \frac{f}{dpy}$$



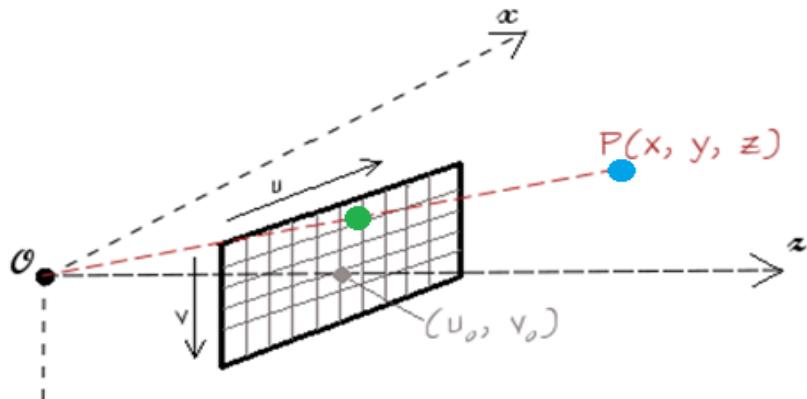
Review of the last course and goals for today

Review - computer vision and 3D perception

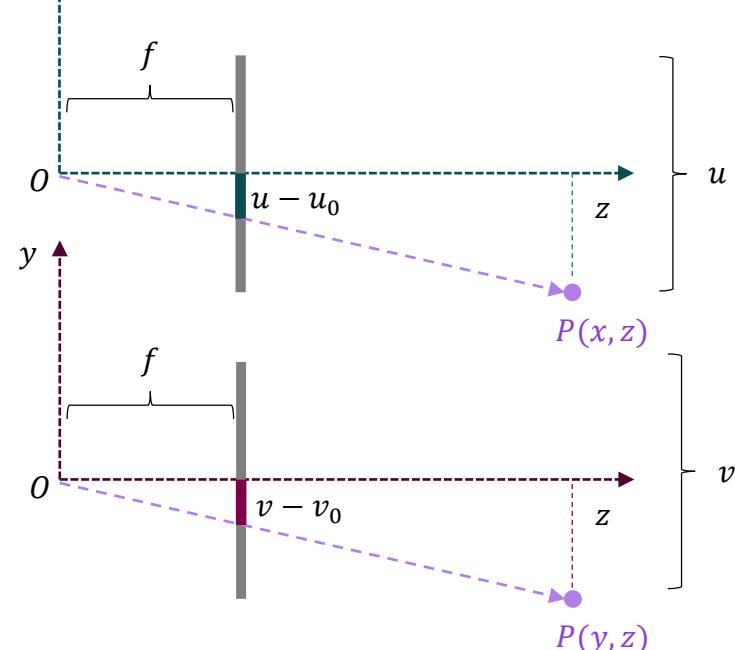
1



2



$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$



$$\frac{(v - v_0)}{f_y} = \frac{y}{z} \qquad \qquad \frac{(u - u_0)}{f_x} = \frac{x}{z}$$

Review of the last course and goals for today

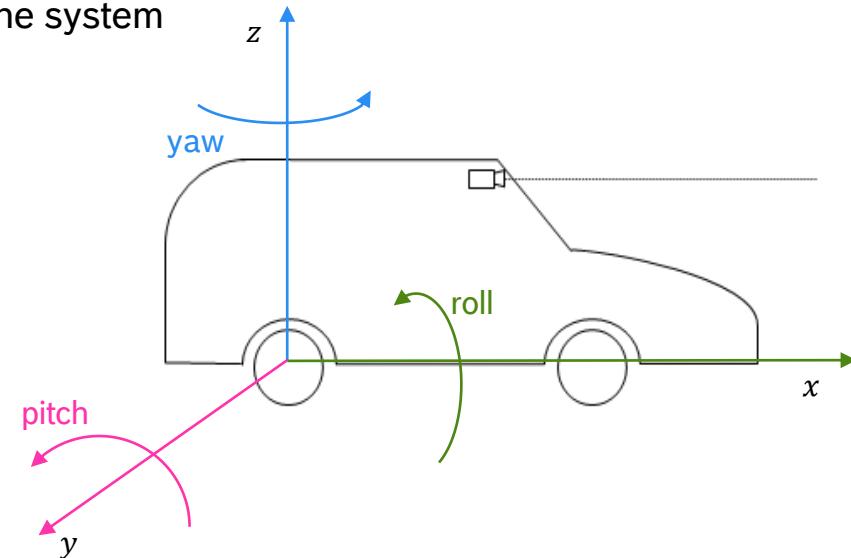
Review - parameters of the camera

Extrinsic parameters

- represent position and orientation of the camera in the car (world) coordinate system
 - Convention – use the center of the rear axis
- R matrix for each axis and T represents the translation vector applied on the system

- they can be written in the following form $[R|T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$
- there are three rotations applied in 3D space – **pitch**, **roll** and **yaw**

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K \cdot \left(R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \right) = K \cdot [R|T] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Review of the last course and goals for today

General concepts - review

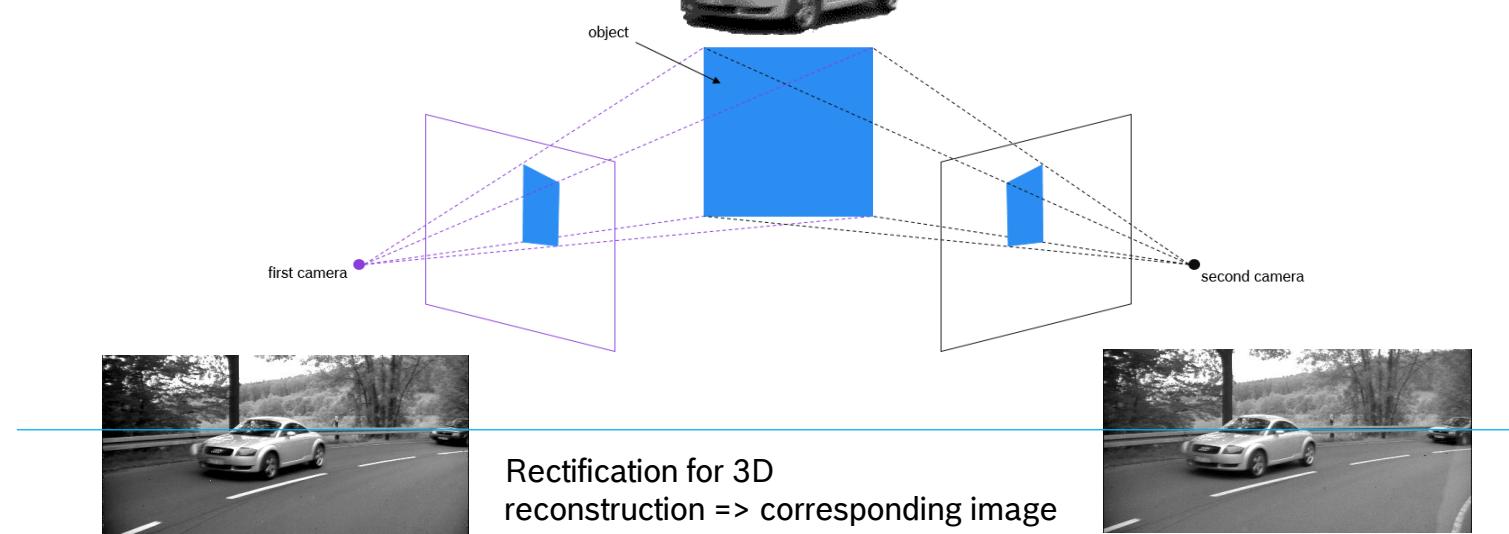
- 3D reconstruction – the process of creating the 3D shape and position of real objects from images
- in computer vision for automated driving
 - using the stereo system
 - using the mono system in time



stereo
camera

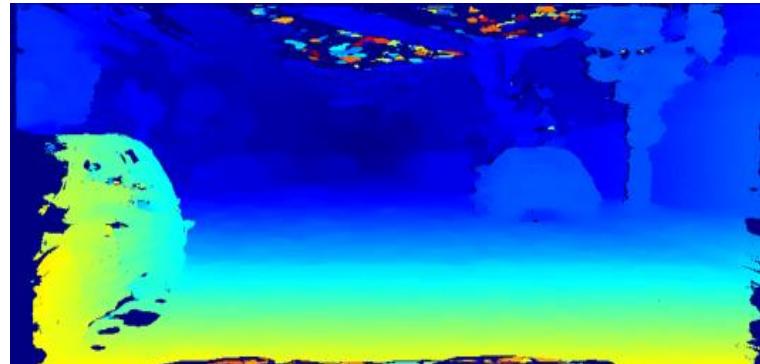
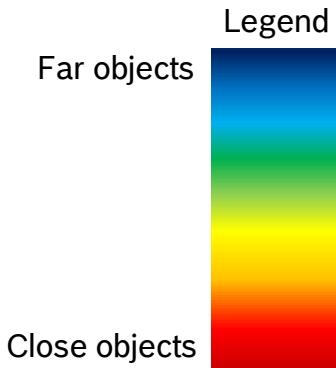


mono
camera



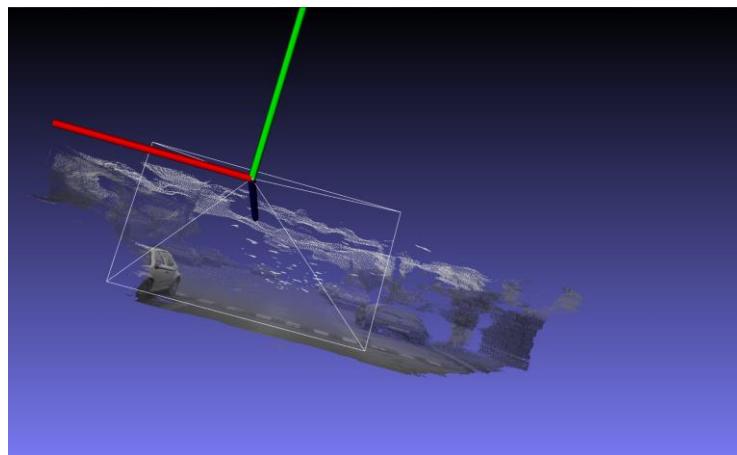
Review of the last course and goals for today

Review – disparity and 3D



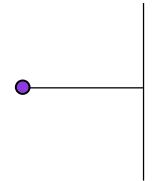
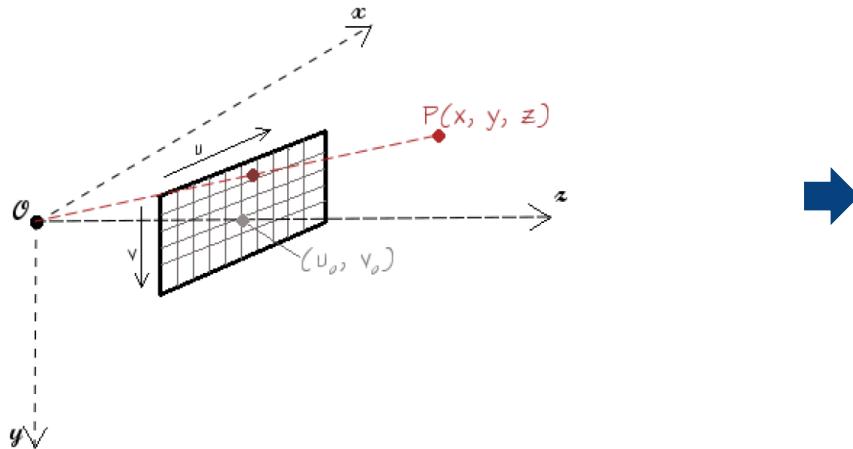
3D reconstruction

Disparity is small in far range
and big in close range

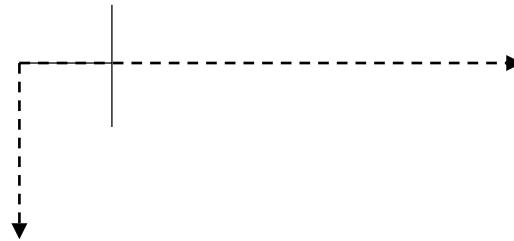


Review of the last course and goals for today

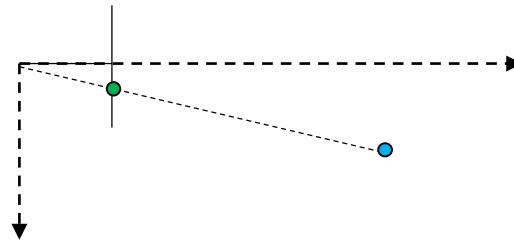
Simplified camera representations



Camera, magenta
is the optical center



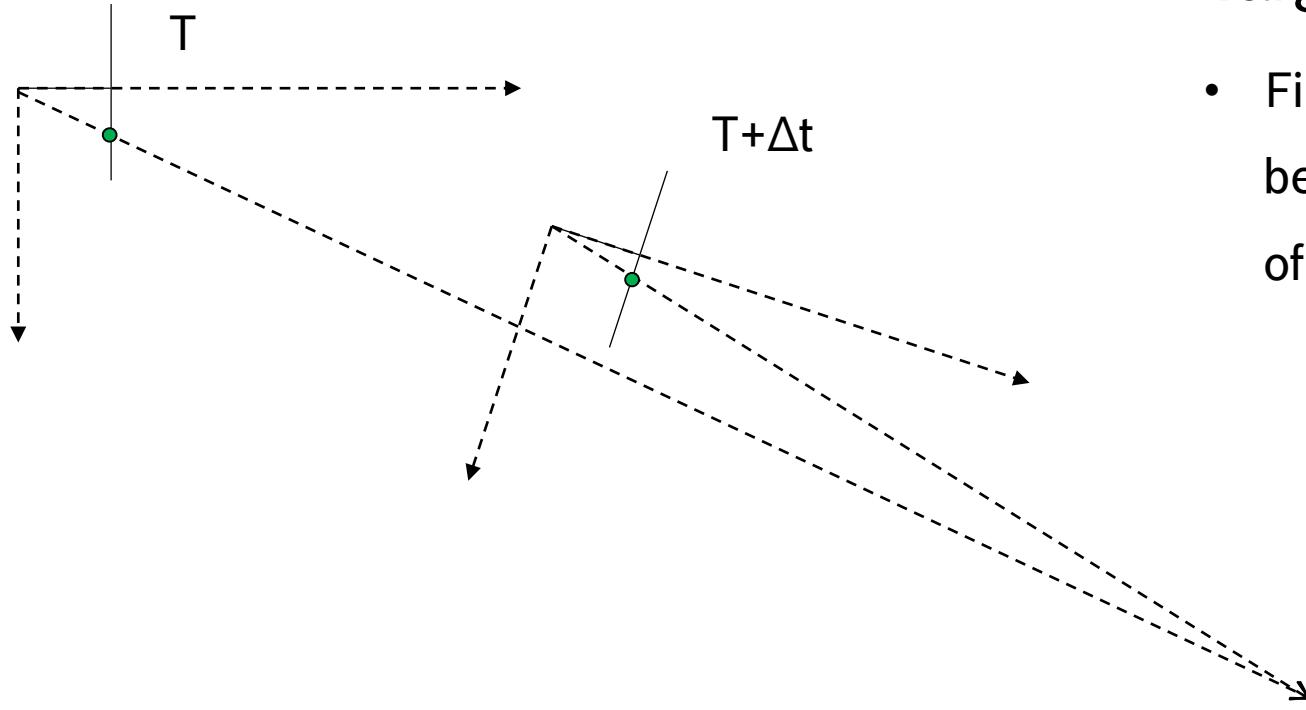
Camera with coordinate
system attached



Camera with coordinate
system attached
Blue dot is projected
in the green dot
in the image plane

Review of the last course and goals for today

Goal for today – geometry of a mono system

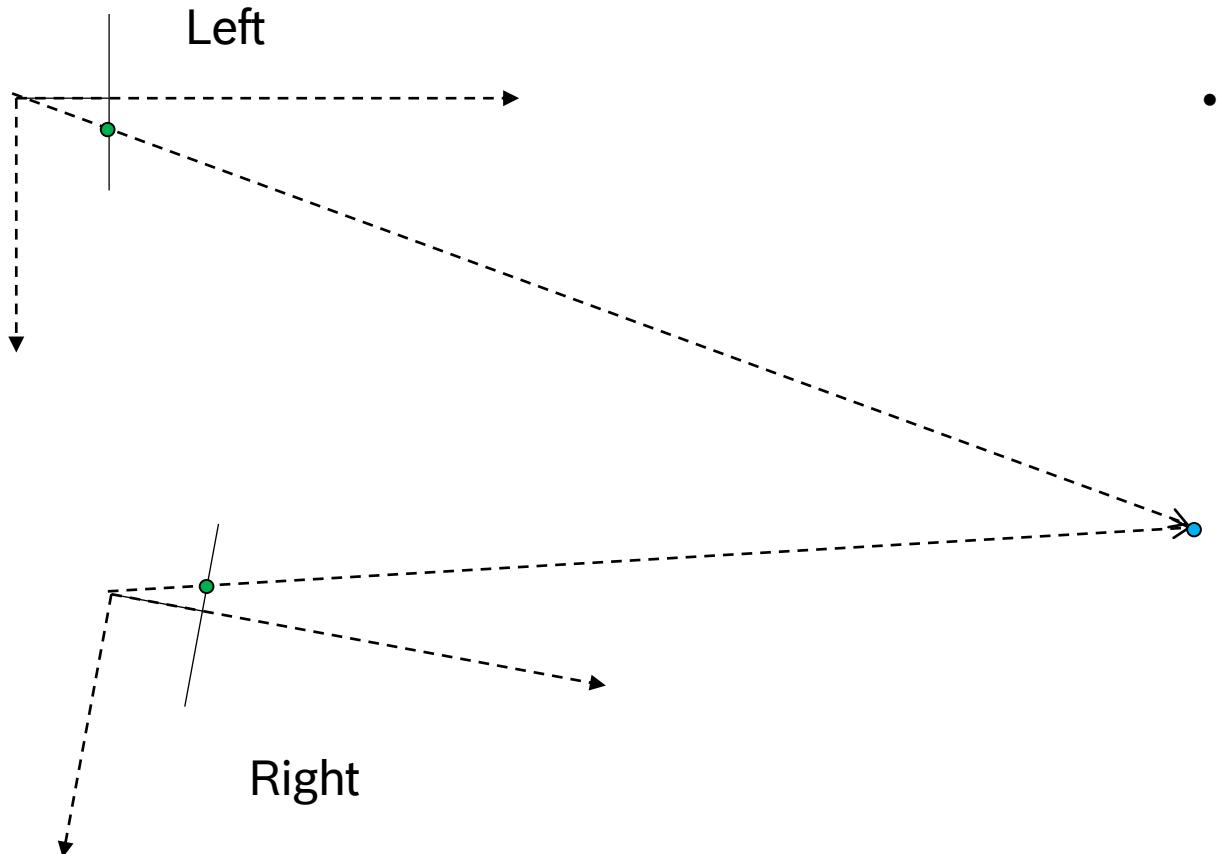


Target:

- Find the mathematical relationship between the two projections (green) of a given point in space (blue)?

Review of the last course and goals for today

Goal for today - geometry of a stereo system



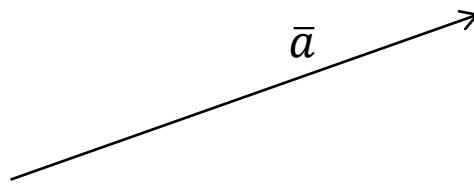
Target:

- Find the mathematical relationship between the two projections (green) of a given point in space (blue)?

THEORETICAL REFRESH

Theoretical refresh

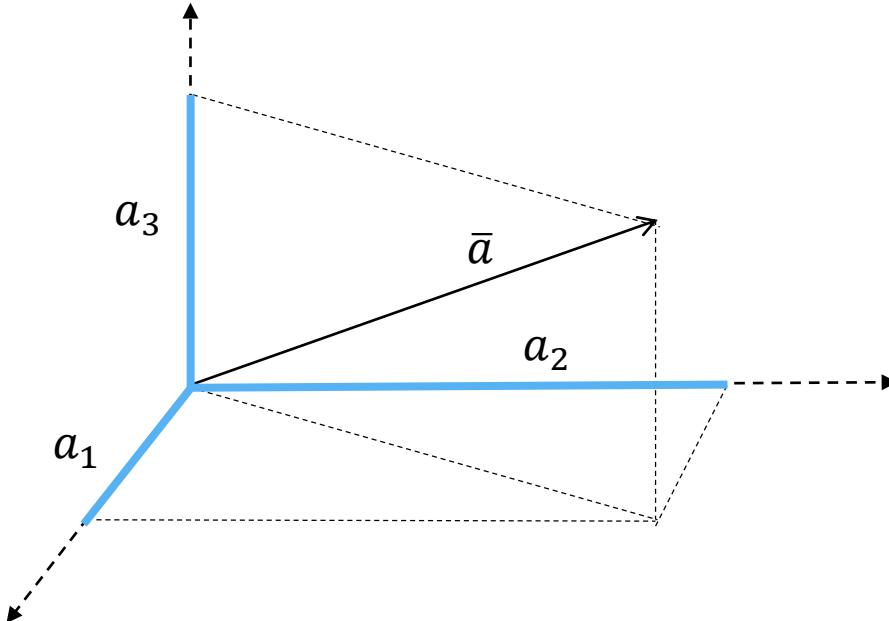
Vectors



- A quantity that has direction and magnitude
 - General theory about vectors and vector spaces you get from your linear algebra course
 - For computer vision we need 3-dimensional vector spaces, in the linear algebra course you studied n-dimensional vector spaces
 - Magnitude will be noted as $|\bar{a}|$

Theoretical refresh

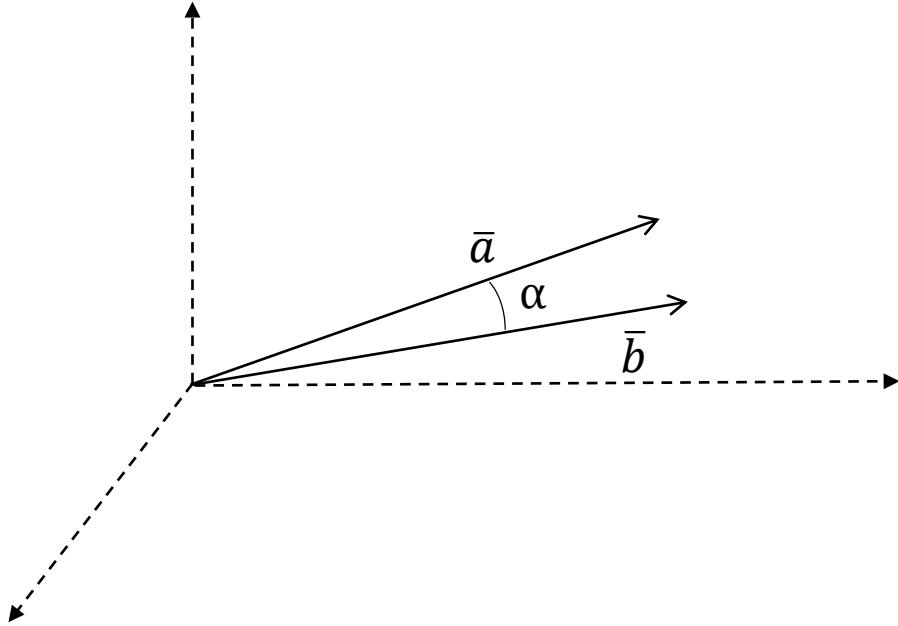
Vector's coordinates



- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
 - Coordinates are vector's projection on some orthogonal axes
 - $\bar{a} = (a_1 \ a_2 \ a_3)$ for 3-dimensional case
 - Based on axis unit vectors: $\bar{a} = a_1\bar{i} + a_2\bar{j} + a_3\bar{k}$
 - $\bar{a} = (a_1 \ a_2 \ \dots \ a_n)$ for n-dimensional case

Theoretical refresh

Dot product



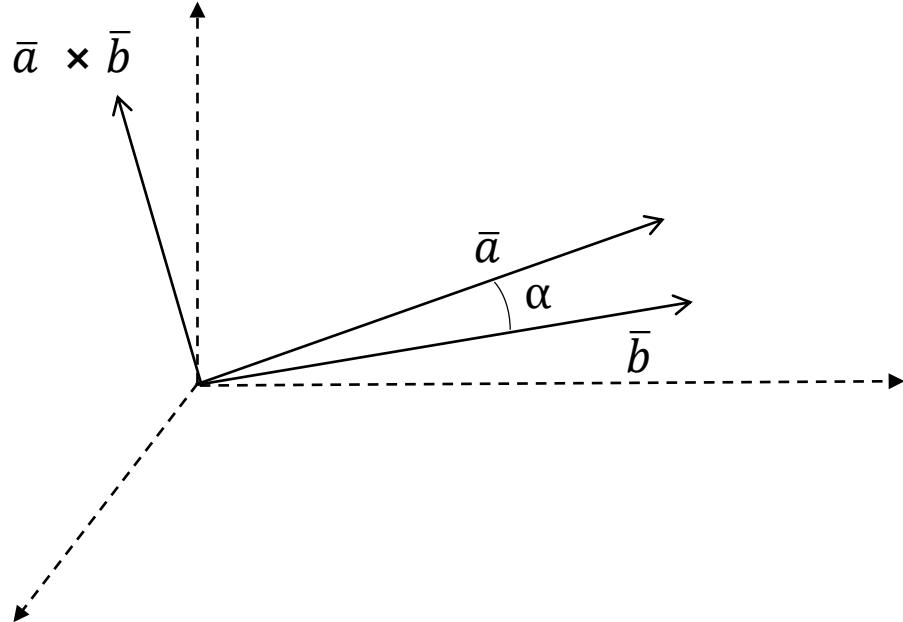
- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
 - $\bar{a} \cdot \bar{b} = |\bar{a}| |\bar{b}| \cos(\alpha)$
 - If $\bar{a} = (a_1 \ a_2 \ a_3)$ and $\bar{b} = (b_1 \ b_2 \ b_3)$ then

$$\bar{a} \cdot \bar{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}' \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = [a_1 \ a_2 \ a_3] \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- Remark
 - If vectors are perpendicular then the dot product is zero

Theoretical refresh

Cross product

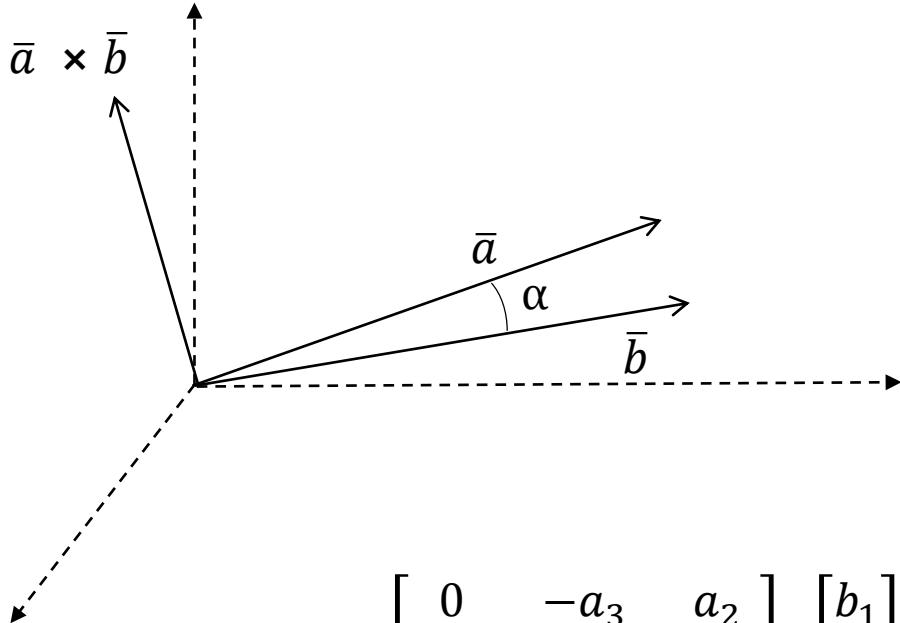


- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
 - Direction perpendicular to the two vectors
 - Magnitude $|\bar{a} \times \bar{b}| = |\bar{a}||\bar{b}|\sin(\alpha)$

$$|\bar{a} \times \bar{b}| = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

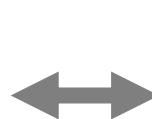
Theoretical refresh

Cross product matrix form



$$\bar{a} \times \bar{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \bar{a}_x \cdot \bar{b}$$

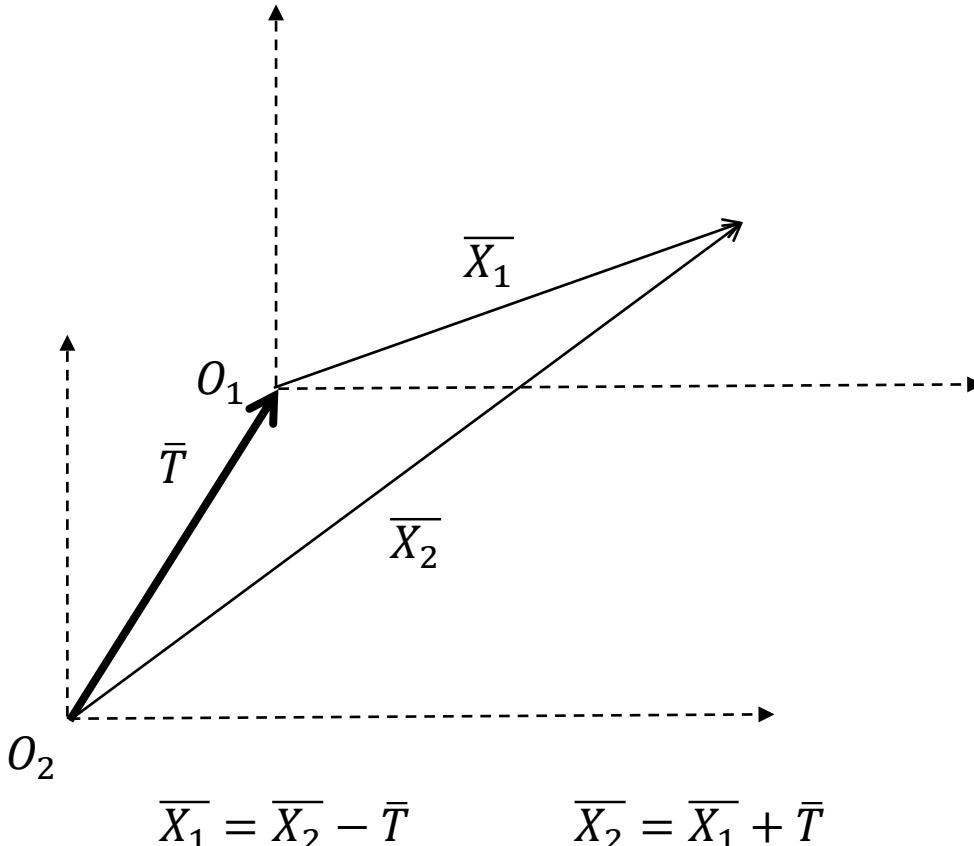
- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
- Cross product matrix form



$$|\bar{a} \times \bar{b}| = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

Theoretical refresh

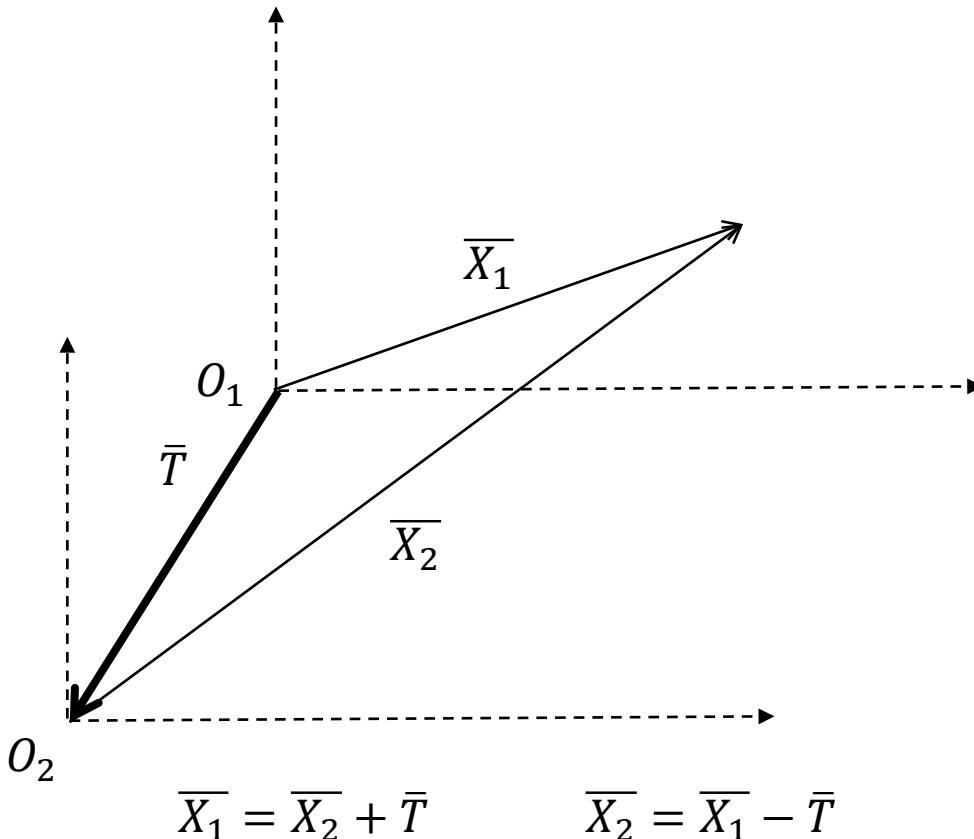
Translation



- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
- Cross product matrix form
- Translation
- \bar{T} represents the coordinates of the **first** coordinate system in the **second** coordinate system

Theoretical refresh

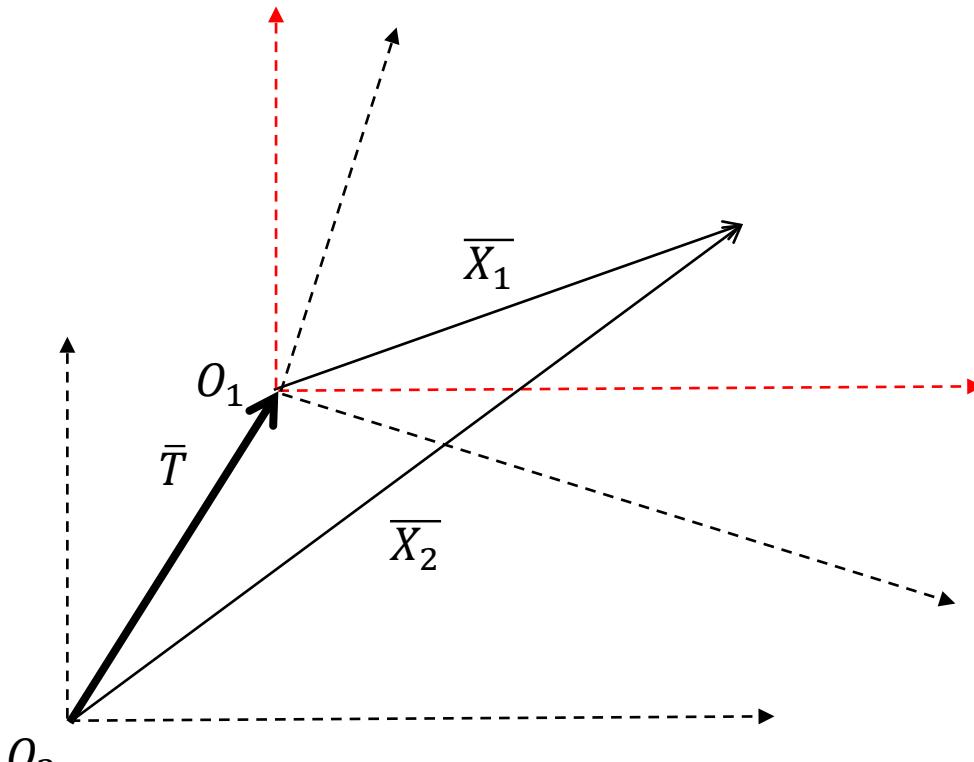
Translation



- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
- Cross product matrix form
- Translation
- \bar{T} represents the coordinates of the **second** coordinate system in the **first** coordinate system

Theoretical refresh

Rotation matrix

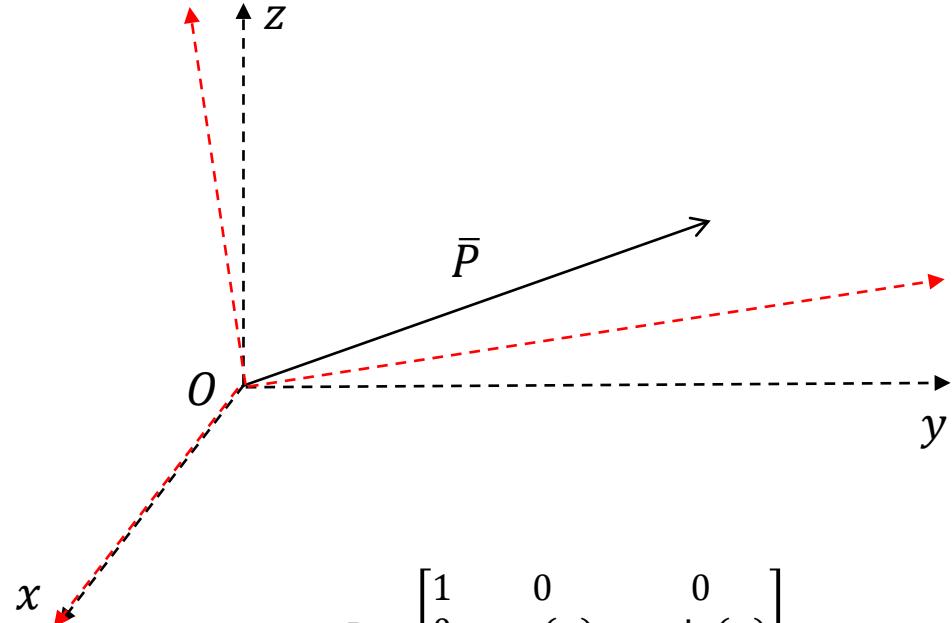


$$R \cdot \bar{X}_1 = \bar{X}_2 - \bar{T} \quad \bar{X}_2 = R \cdot \bar{X}_1 + \bar{T}$$

- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
- Cross product matrix form
- Translation
- Rotation
 - Rotation matrix $R \cdot R^t = I$
 - $R \cdot \bar{X}_1$ are the coordinates of \bar{X}_1 in the coordinate system with red axes

Theoretical refresh

Rotation around single axis



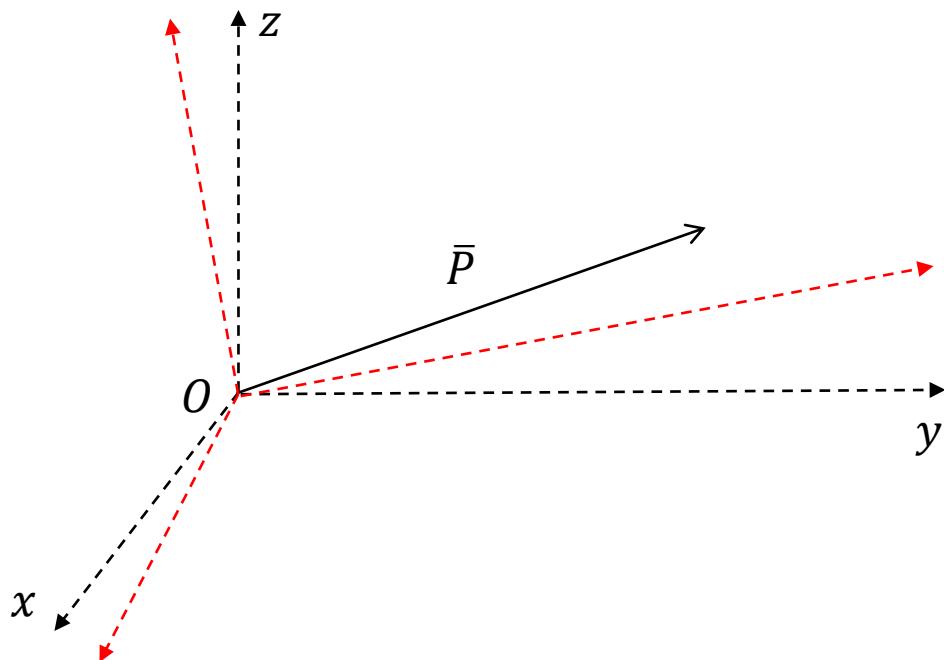
$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$R \cdot \bar{P} = R \cdot \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix} = \begin{bmatrix} x_P \\ \cos(\alpha)y_P - \sin(\alpha)z_P \\ \sin(\alpha)y_P + \cos(\alpha)z_P \end{bmatrix}$$

- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
- Cross product matrix form
- Translation
- Rotation
 - Rotation matrix $R \cdot R^t = I$
 - $R \cdot \bar{P}$ are the coordinates of the vector \bar{P} in the coordinate system with red axes

Theoretical refresh

General 3D rotation



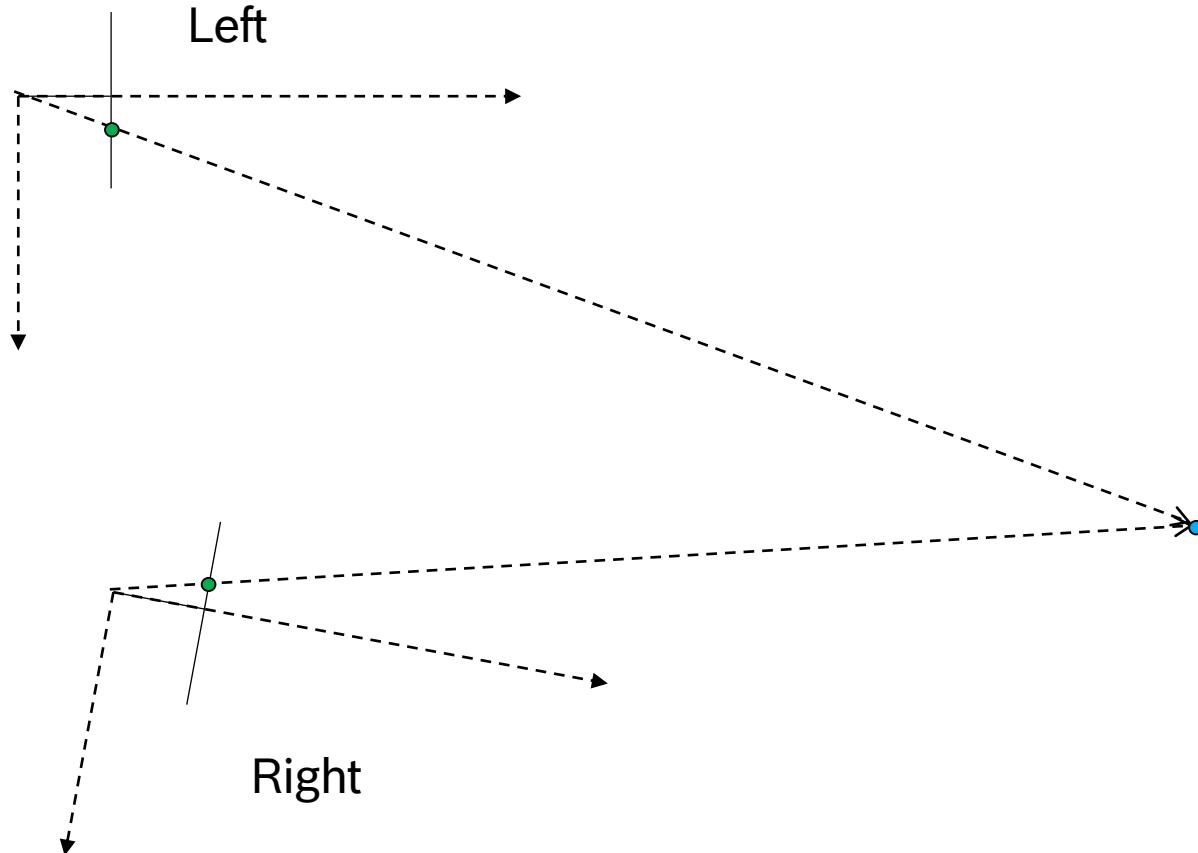
$$R = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

- A quantity that has direction and magnitude
- Has coordinates when represented in a certain coordinate system
- Dot product
- Cross product
- Cross product matrix form
- Translation
- Rotation
 - Rotation matrix $R \cdot R^t = I$
 - $R \cdot \bar{P}$ are the coordinates of the vector \bar{P} in the coordinate system with red axes

ESSENTIAL MATRIX

Essential matrix

Geometry of two cameras

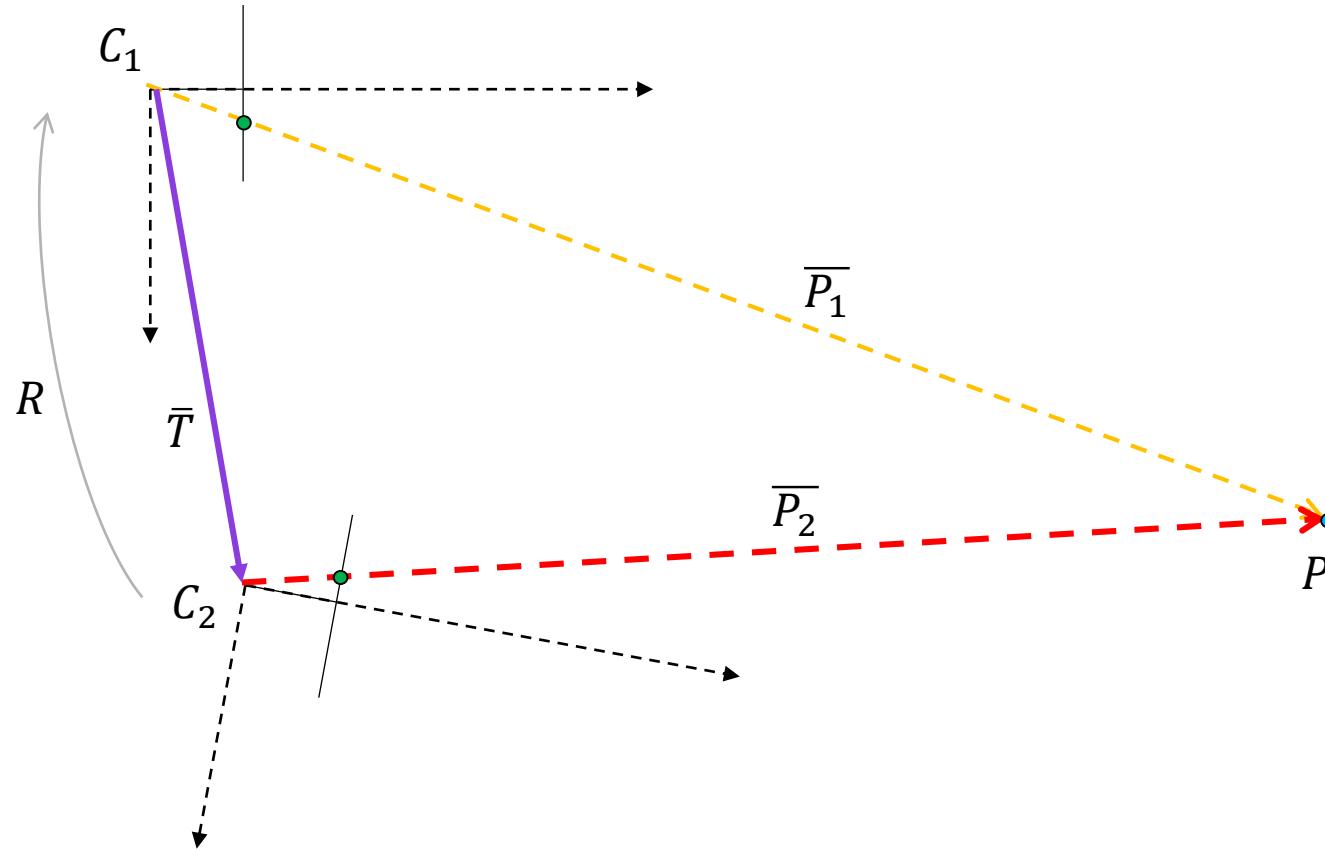


Target:

- Find the mathematical relationship between the two projections (green) of a given point in space (blue)?

Essential matrix

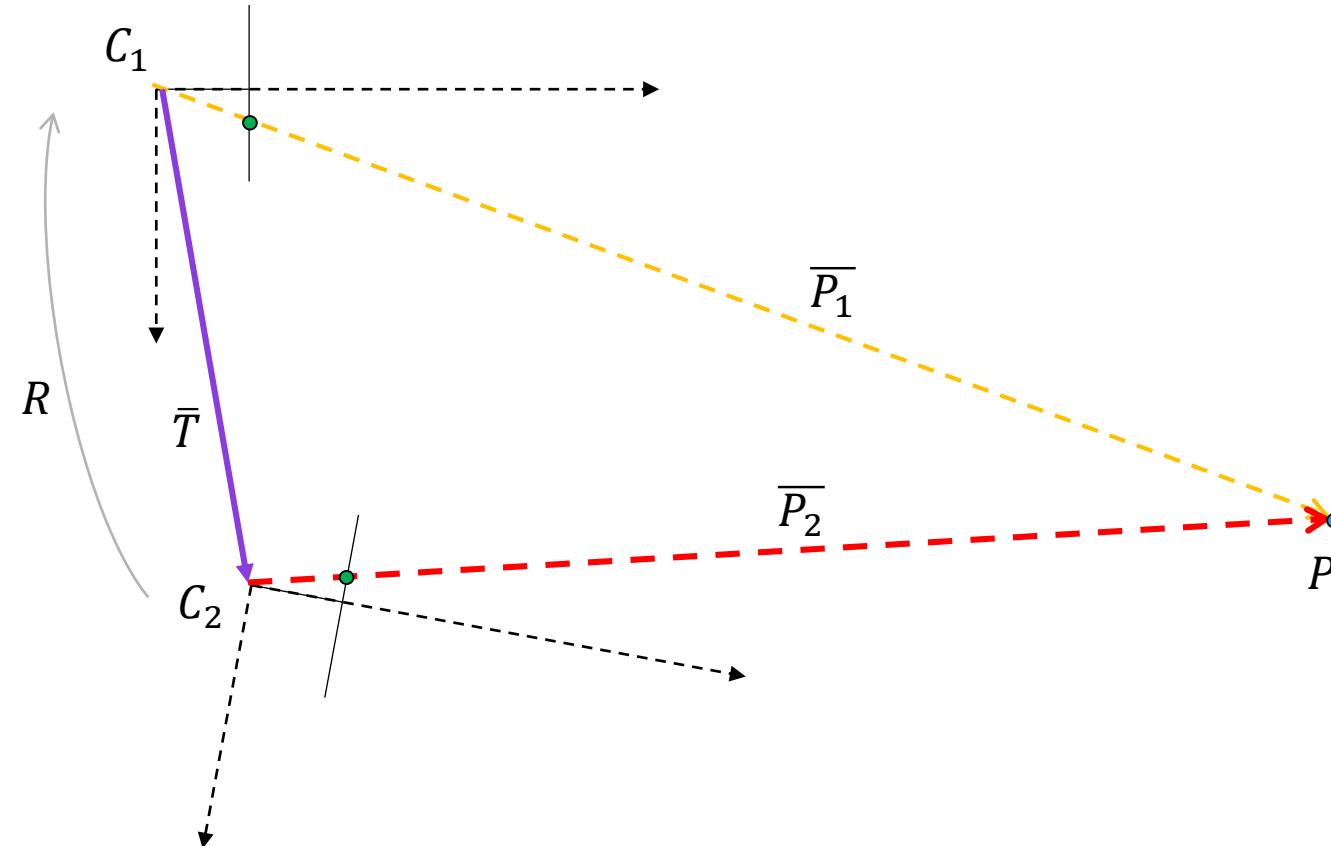
Geometry of two cameras



- Consider two projections (green) of a given point in space (blue)
- \bar{T} translation vector between the cameras (magenta)
- R rotation matrix from the C_2 coordinate system to C_1
- \overline{P}_1 column vector with coordinates of the point P in C_1 (orange)
- \overline{P}_2 column vector with coordinates of the point P in C_2 (red)

Essential matrix

Geometry of two cameras

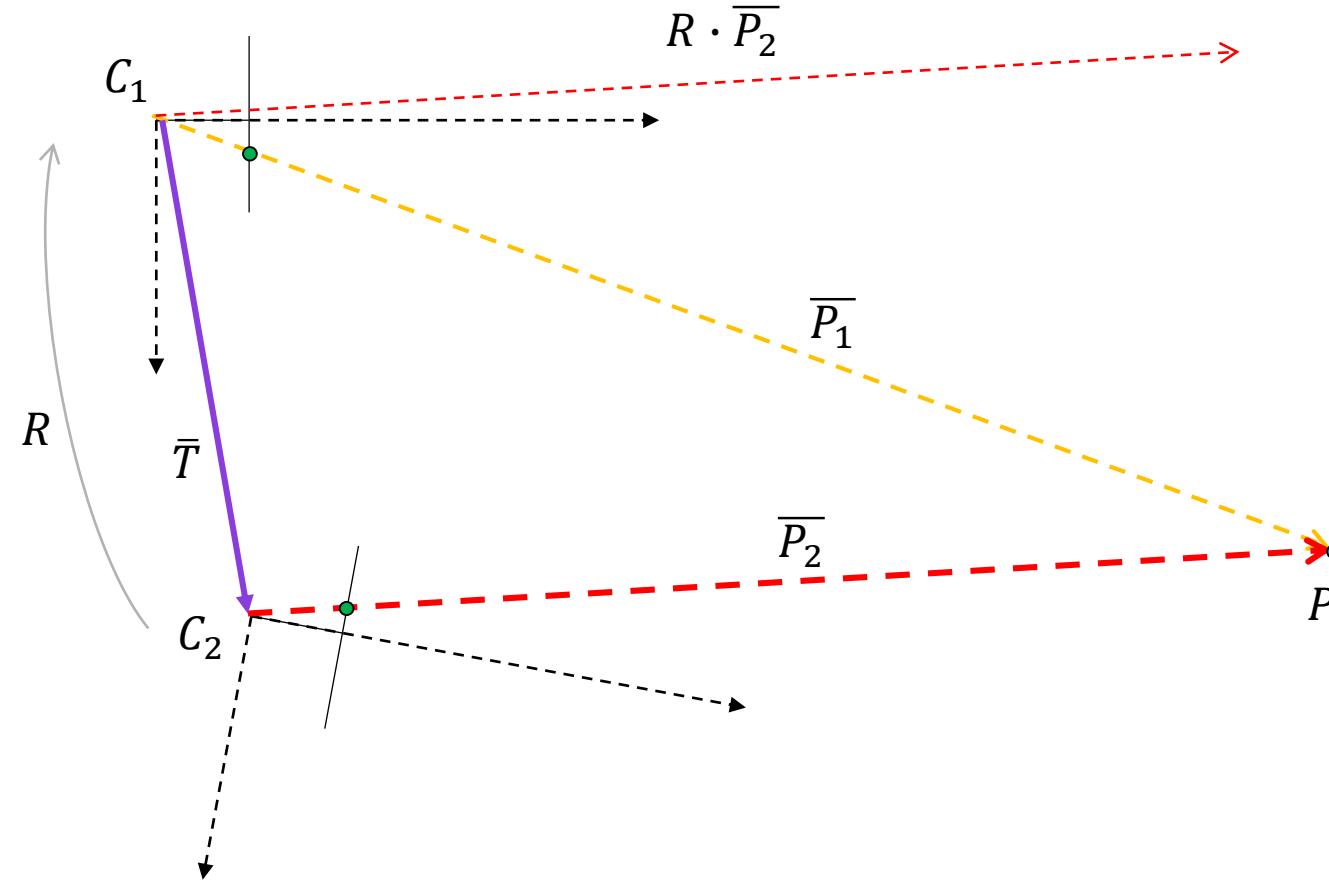


- Constraint – the magenta, orange and red vectors are co-planar
- Cross product of any two of them, dot product with the third is zero!
- Expressed in the C_1 the values of the three column vectors are:

$$\bar{T}, \overline{P_1}, R \cdot \overline{P_2}$$

Essential matrix

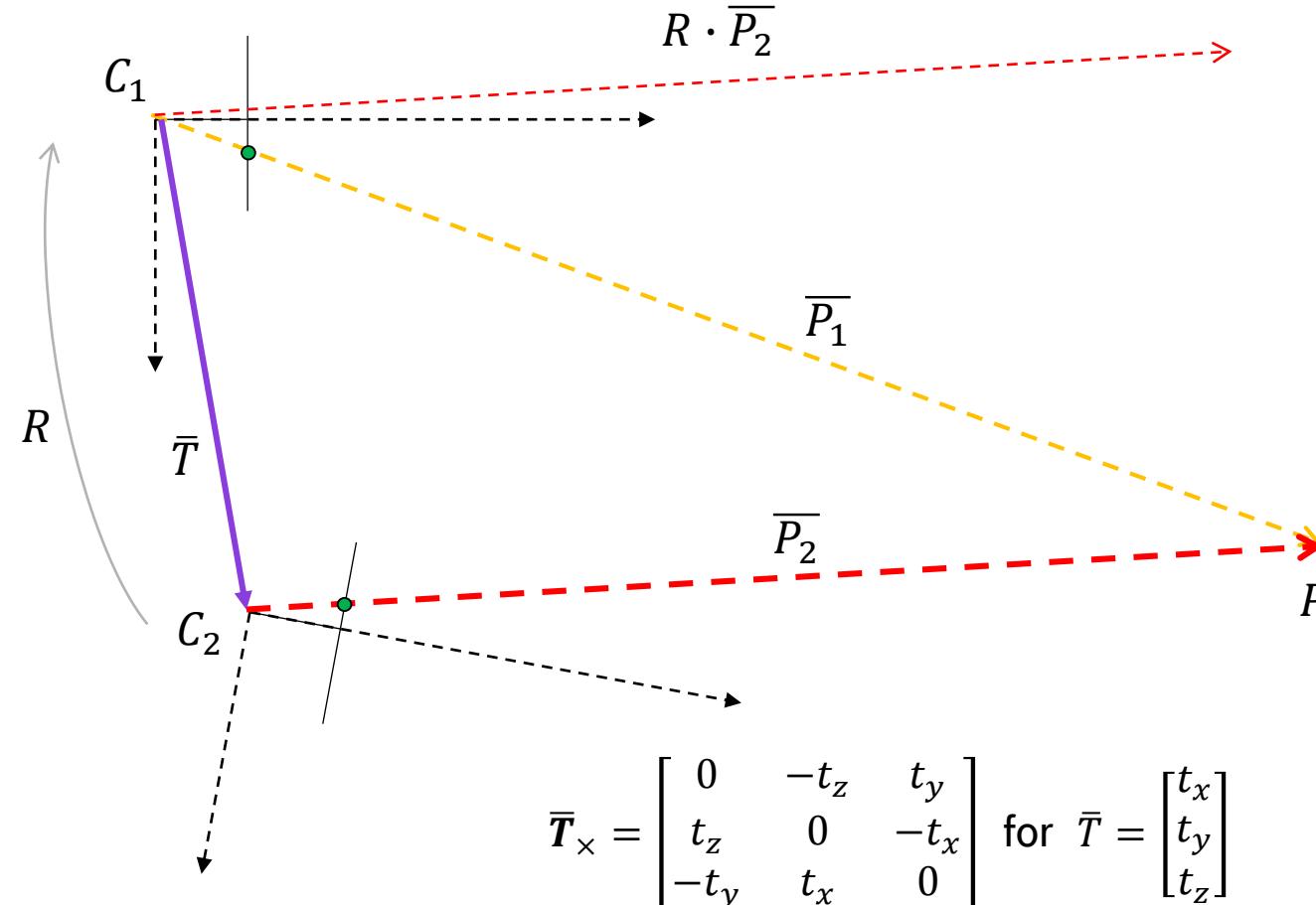
Geometry of two cameras



- Constraint – the magenta, orange and red vectors are co-planar
- **Cross product of any two of them, dot product with the third is zero!**
- Expressed in the C_1 the values of the three column vectors are:
$$\bar{T}, \bar{P}_1, R \cdot \bar{P}_2$$
$$\bar{P}_1 \cdot (\bar{T} \times R \cdot \bar{P}_2) = 0$$
- In matrix form (' means transpose):
$$\bar{P}_1' \cdot (\bar{T}_x \cdot R \cdot \bar{P}_2) = \bar{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \bar{P}_2 = 0$$

Essential matrix

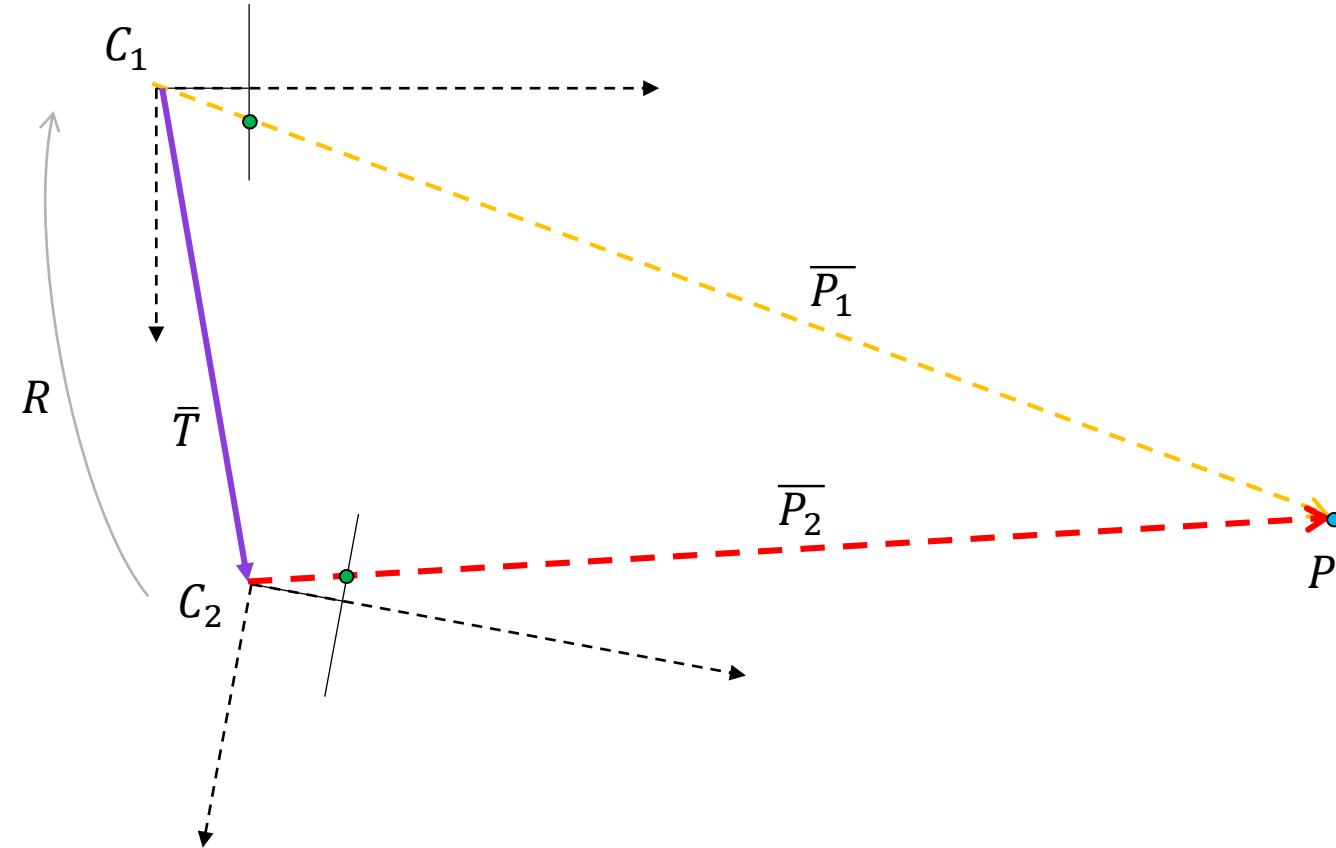
Geometry of two cameras



- Constraint – the magenta, orange and red vectors are co-planar
- Cross product of any two of them, dot product with the third is zero!
- In matrix form:
$$\overline{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \overline{P}_2 = 0$$
- $E = \bar{T}_x \cdot R$ is the **essential matrix** relating the two cameras

Essential matrix

Geometry of two cameras



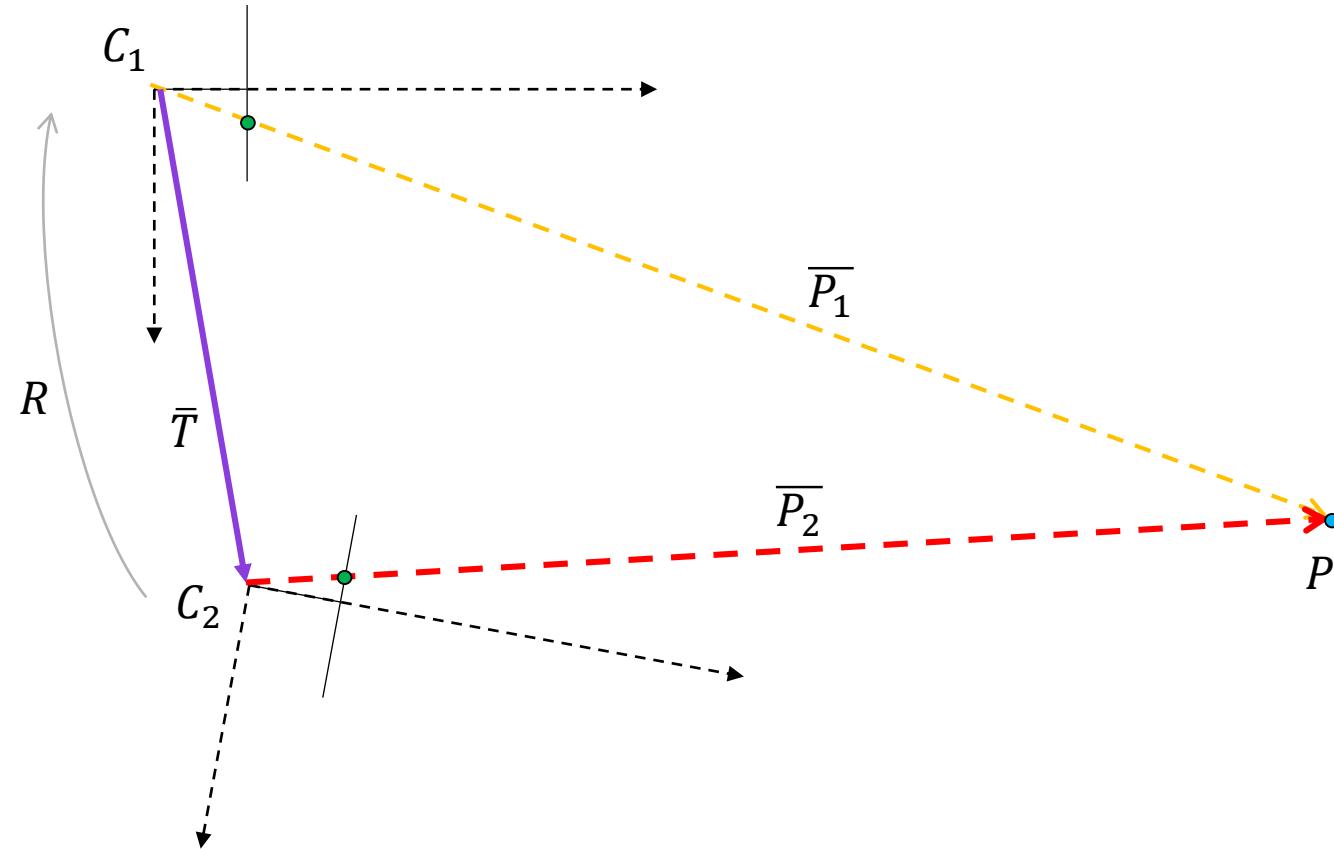
$$\overline{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \overline{P}_2 = 0$$

- $E = \bar{T}_x \cdot R$ essential matrix
- Remarks:
 - Equation still holds by multiplying \overline{P}_1 , \overline{P}_2 or \bar{T}_x with a scalar factor
 - Example: $k_1 \overline{P}_1' \cdot (k_2 \bar{T}_x \cdot R) \cdot k_3 \overline{P}_2 = 0$
- Thus \bar{T}_x has only 2 degrees of freedom (DoF). Let them be t_1 and t_2
- Essential matrix has 5 DoF:
 - $E(\alpha, \beta, \gamma, t_1, t_2)$
 - $\overline{P}_1' \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot \overline{P}_2 = 0$

FUNDAMENTAL MATRIX

Fundamental matrix

Geometry of two cameras



$$\overline{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \overline{P}_2 = 0$$

$E = \bar{T}_x \cdot R$ essential matrix

$$\overline{P}_1 = \begin{bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{bmatrix} \quad \overline{P}_2 = \begin{bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{bmatrix}$$

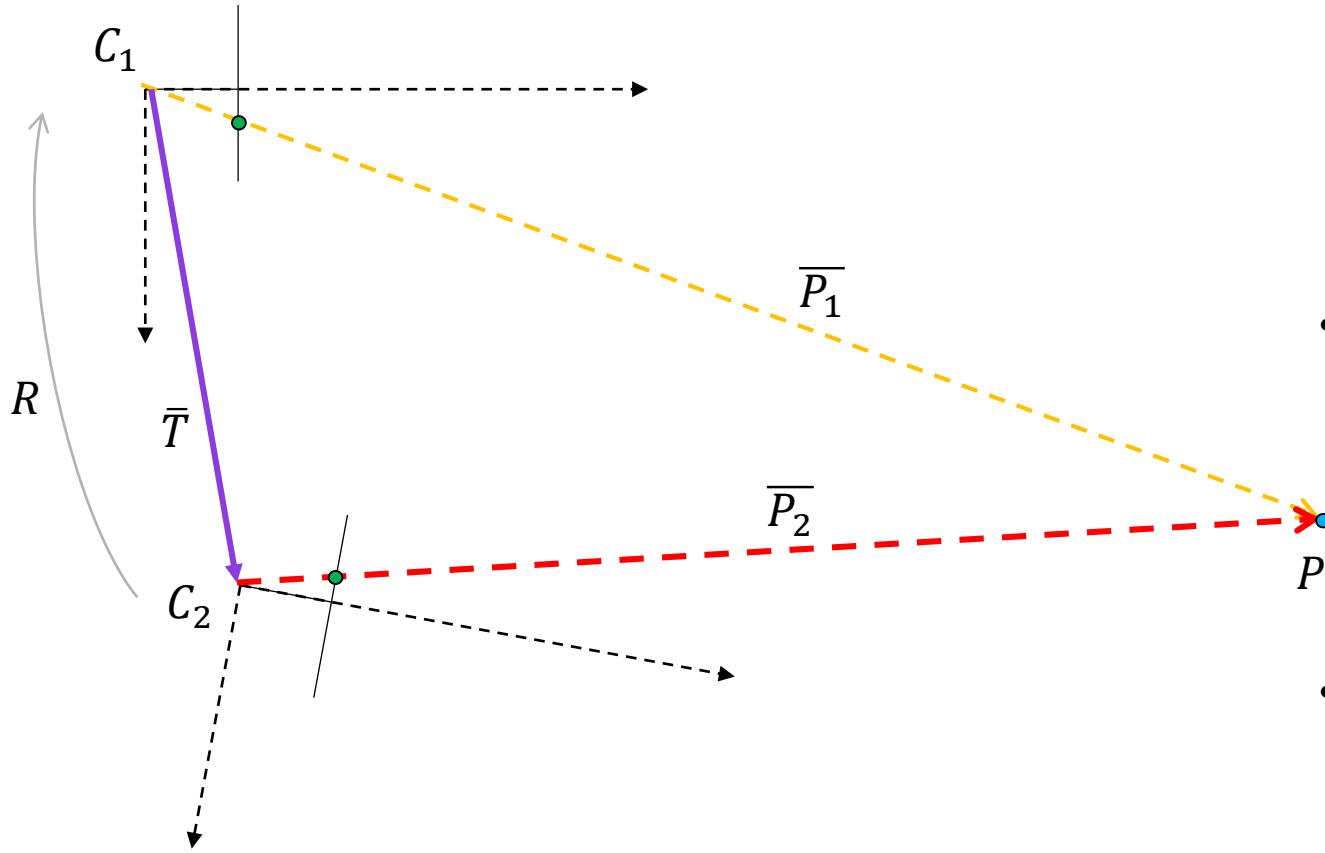
- Essential matrix constraint becomes:

$$\begin{bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{bmatrix}' \cdot (\bar{T}_x \cdot R) \cdot \begin{bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{bmatrix} = 0$$

- It can be divided with z_{P_1} and z_{P_2}

Fundamental matrix

Geometry of two cameras



$$\overline{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \overline{P}_2 = 0$$

$$\overline{P}_1 = \begin{bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{bmatrix} \quad \overline{P}_2 = \begin{bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{bmatrix}$$

- Essential matrix constraint becomes:

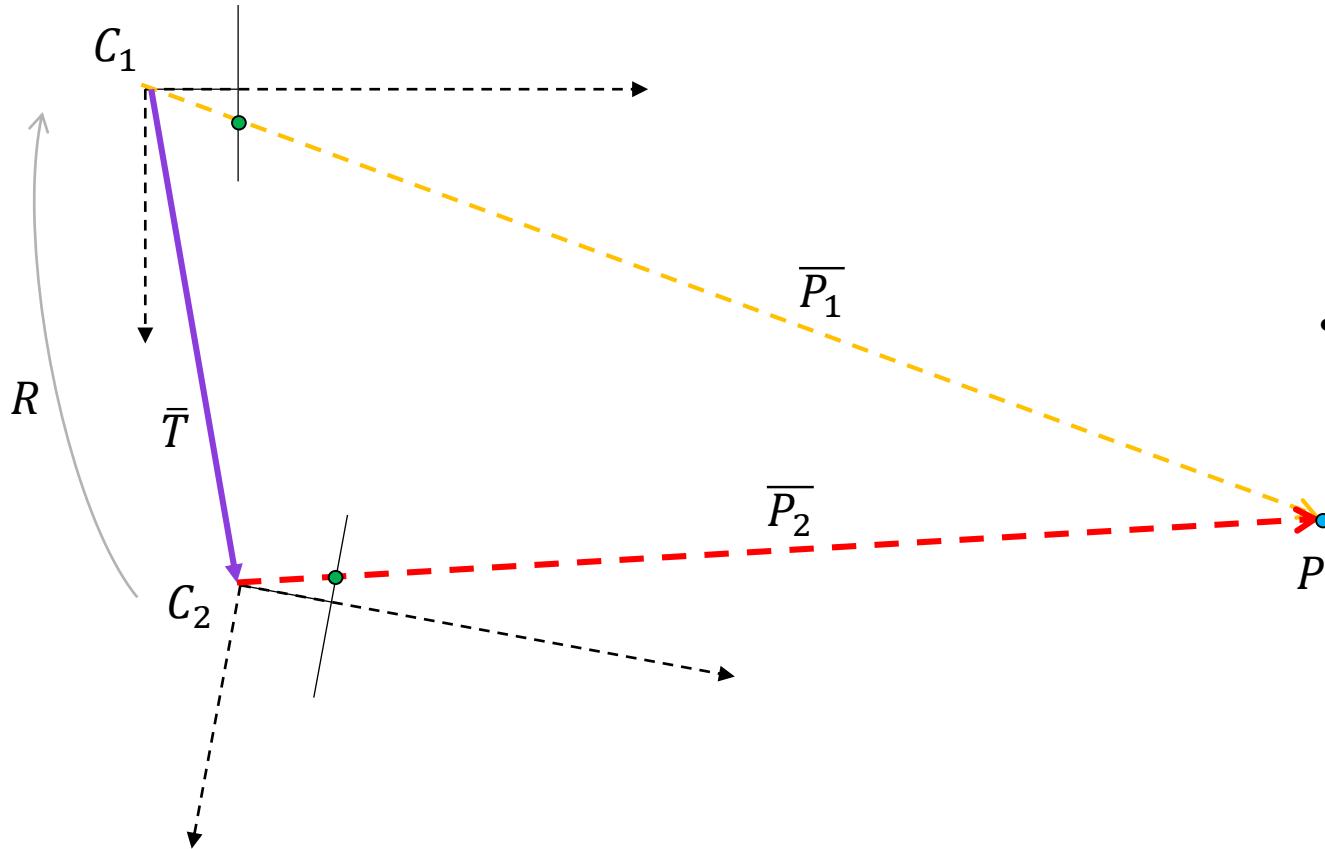
$$\begin{bmatrix} x_{P_1}/z_{P_1} \\ y_{P_1}/z_{P_1} \\ 1 \end{bmatrix}' \cdot (\bar{T}_x \cdot R) \cdot \begin{bmatrix} x_{P_2}/z_{P_2} \\ y_{P_2}/z_{P_2} \\ 1 \end{bmatrix} = 0$$

- But remember the projection equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \quad K^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

Fundamental matrix

Geometry of two cameras



$$\overline{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \overline{P}_2 = 0$$

$$\overline{P}_1 = \begin{bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{bmatrix} \quad \overline{P}_2 = \begin{bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{bmatrix}$$

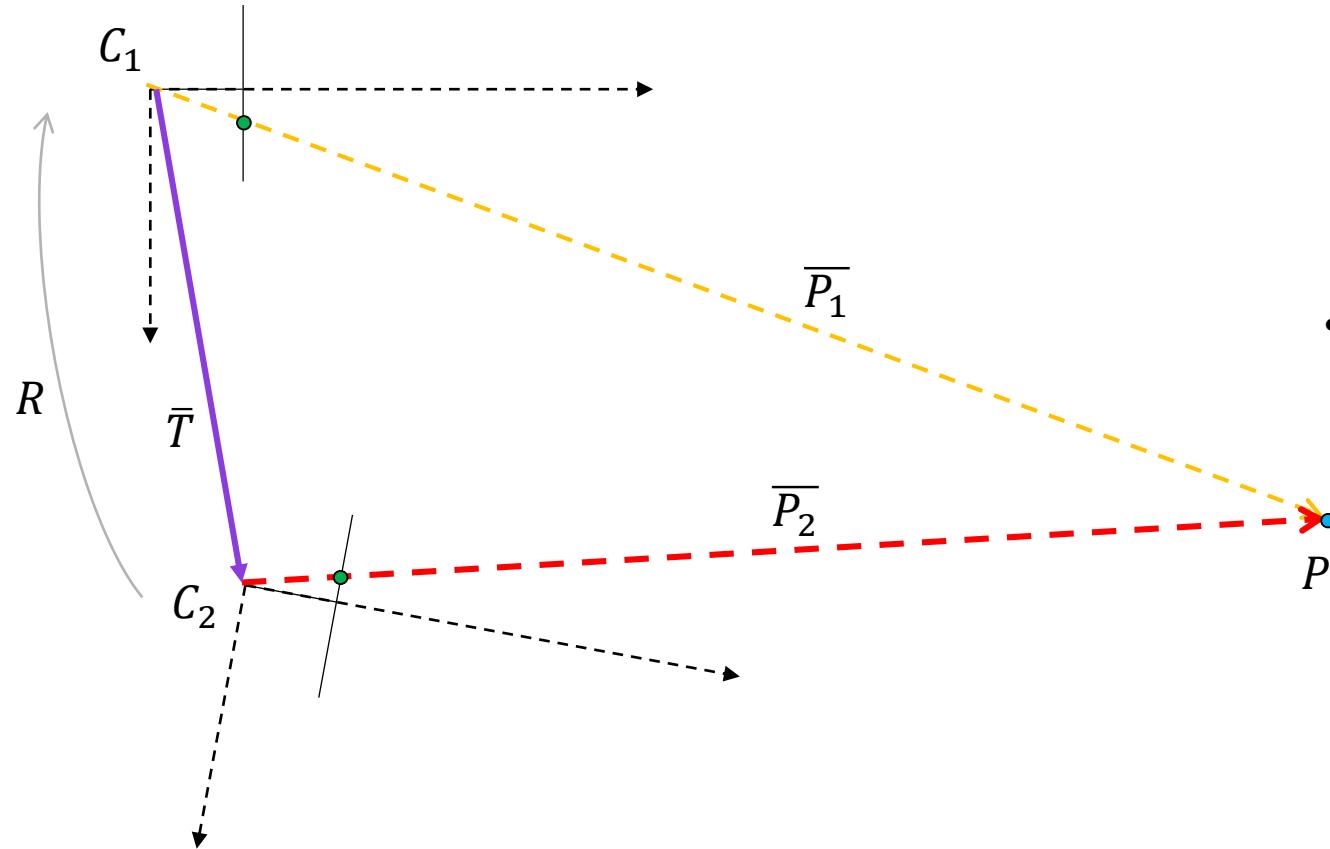
- Essential matrix constraint becomes:

$$\begin{bmatrix} x_{P_1}/z_{P_1} \\ y_{P_1}/z_{P_1} \\ 1 \end{bmatrix}' \cdot (\bar{T}_x \cdot R) \cdot \begin{bmatrix} x_{P_2}/z_{P_2} \\ y_{P_2}/z_{P_2} \\ 1 \end{bmatrix} = 0$$

$$\left(K_1^{-1} \cdot \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \right)' \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

Fundamental matrix

Geometry of two cameras



$$\overline{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \overline{P}_2 = 0$$

$$\overline{P}_1 = \begin{bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{bmatrix} \quad \overline{P}_2 = \begin{bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{bmatrix}$$

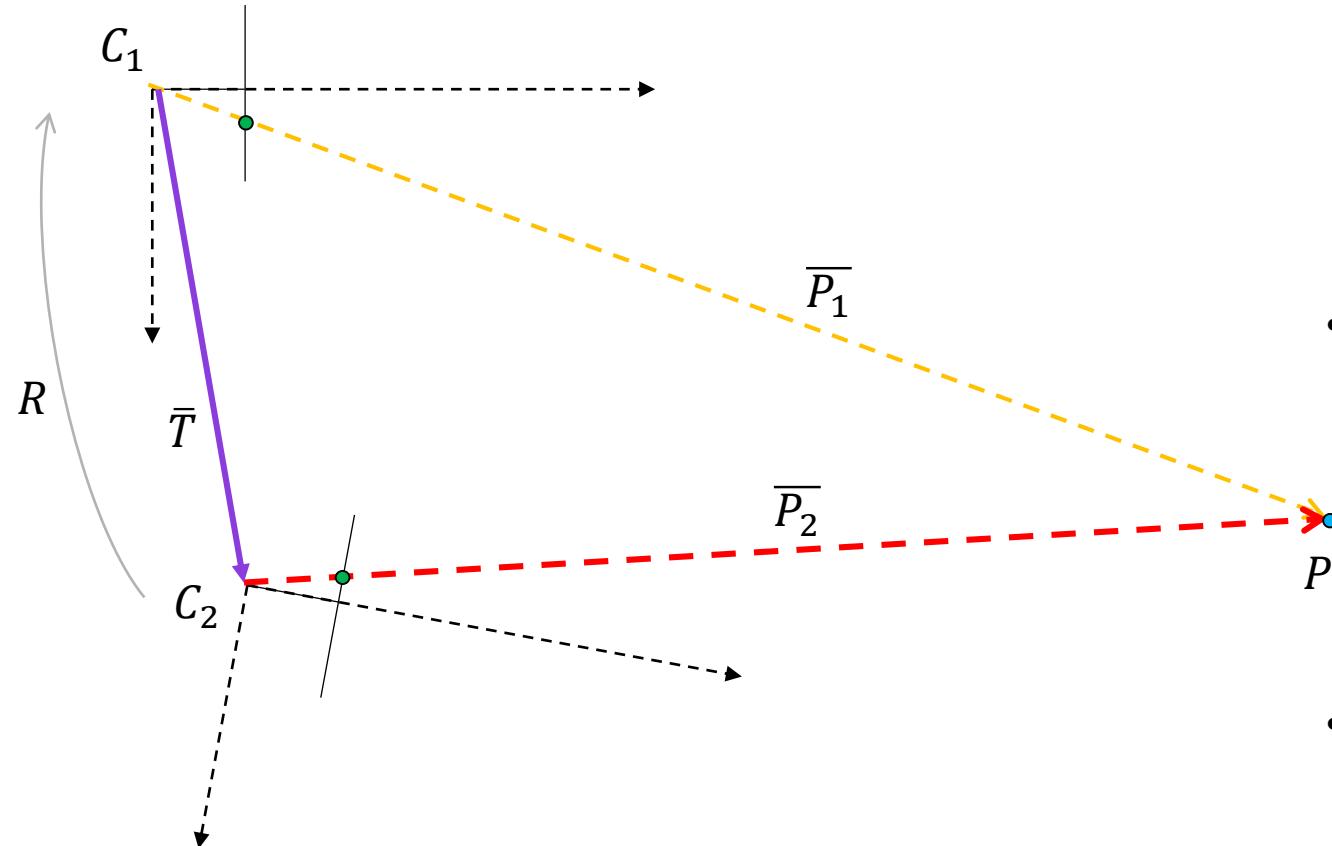
- Essential matrix constraint becomes:

$$\begin{bmatrix} x_{P_1}/z_{P_1} \\ y_{P_1}/z_{P_1} \\ 1 \end{bmatrix}' \cdot (\bar{T}_x \cdot R) \cdot \begin{bmatrix} x_{P_2}/z_{P_2} \\ y_{P_2}/z_{P_2} \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

Fundamental matrix

Geometry of two cameras



$$\bar{P}_1' \cdot (\bar{T}_x \cdot R) \cdot \bar{P}_2 = 0$$

$$\bar{P}_1 = \begin{bmatrix} x_{P_1} \\ y_{P_1} \\ z_{P_1} \end{bmatrix} \quad \bar{P}_2 = \begin{bmatrix} x_{P_2} \\ y_{P_2} \\ z_{P_2} \end{bmatrix}$$

- Fundamental matrix constraint:

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

- Fundamental relates pixel coordinates from different views of the same scene

Fundamental matrix

Intuition

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

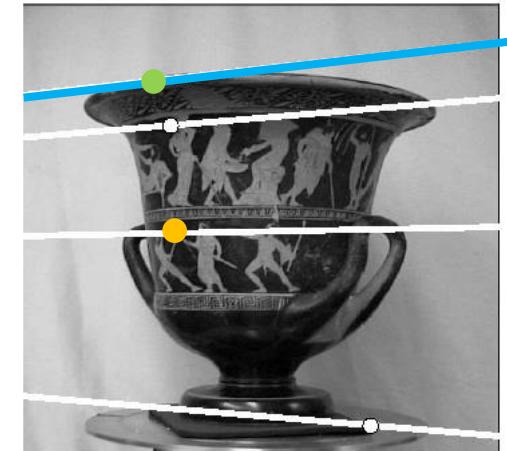
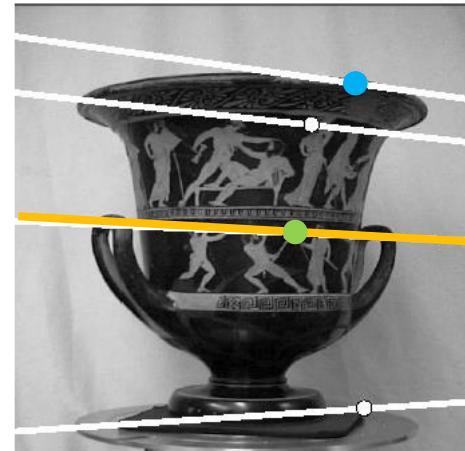
$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

If $\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot F = [a \quad b \quad c]$ then the constraint

becomes $[a \quad b \quad c] \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$

If $F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$ then the

constraint becomes $[u_1 \quad v_1 \quad 1] \cdot \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = 0$



- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>

Refresh line equation: $ax+by+c=0$

$$[a \quad b \quad c] \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Fundamental matrix

Intuition

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

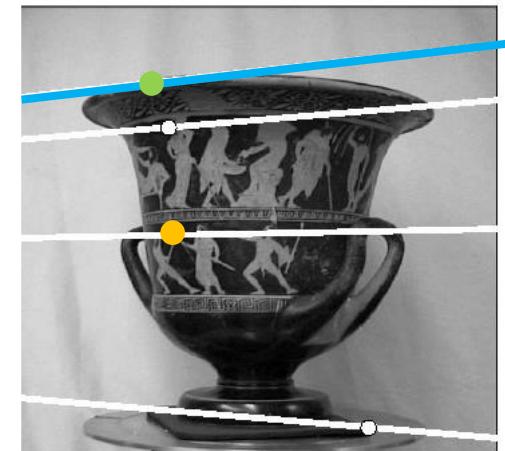
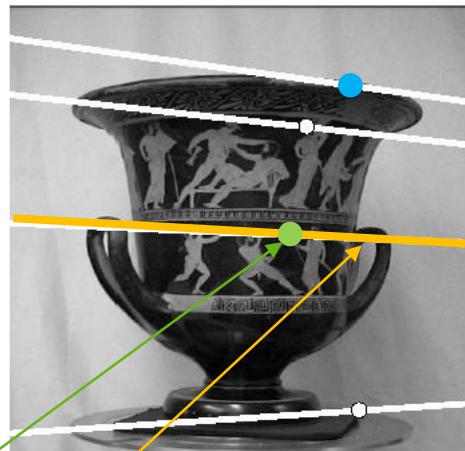
$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

If $\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot F = [a \quad b \quad c]$ then the constraint

$$\text{becomes } [a \quad b \quad c] \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

If $F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$ then the

constraint becomes $[u_1 \quad v_1 \quad 1] \cdot \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = 0$



- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>

Refresh line equation: $ax+by+c=0$

$$[a \quad b \quad c] \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Fundamental matrix

Intuition on rectified images

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

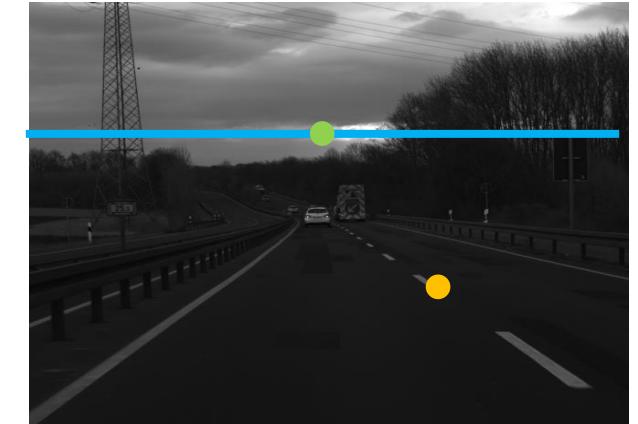
$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

If $\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot F = [a \quad b \quad c]$ then the constraint

$$\text{becomes } [a \quad b \quad c] \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

If $F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$ then the

$$\text{constraint becomes } [u_1 \quad v_1 \quad 1] \cdot \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = 0$$

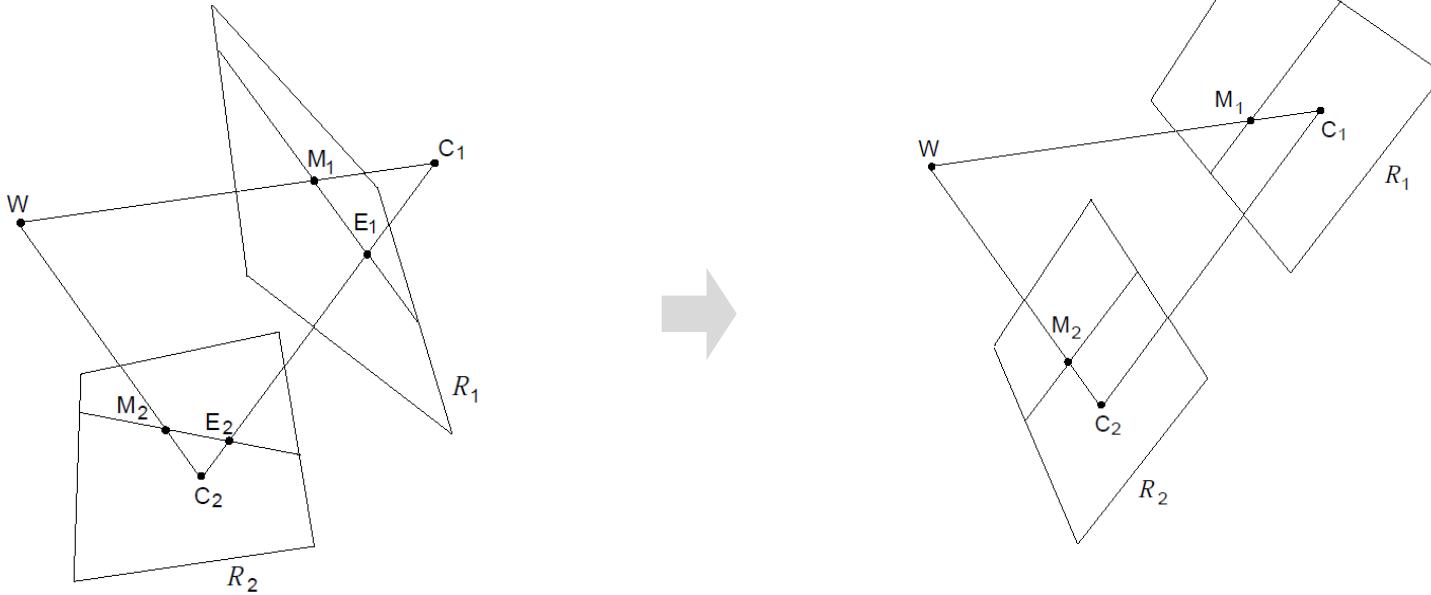


Refresh line equation: $ax+by+c=0$

$$[a \quad b \quad c] \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Fundamental matrix

Intuition on rectification



A compact algorithm for rectification of stereo pairs

Andrea Fusiello¹, Emanuele Trucco², Alessandro Verri³

¹ Dipartimento Scientifico e Tecnologico, Università di Verona, Ca' Vignal 2, Strada Le Grazie, 37134 Verona, Italy; e-mail: fusiello@sci.univr.it

² Heriot-Watt University, Department of Computing and Electrical Engineering, Edinburgh, UK

³ INFM, Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Genova, Italy

Received: 25 February 1999 / Accepted: 2 March 2000

ESSENTIAL MATRIX ESTIMATION

Essential matrix estimation

Mathematica model

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot (\bar{T}_x \cdot R) \cdot K_2^{-1} \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = 0$$

For **more points** we have:

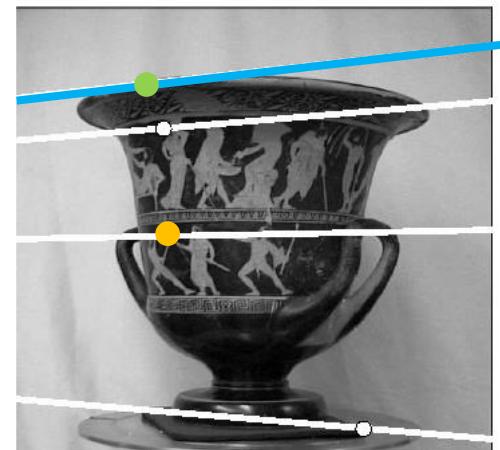
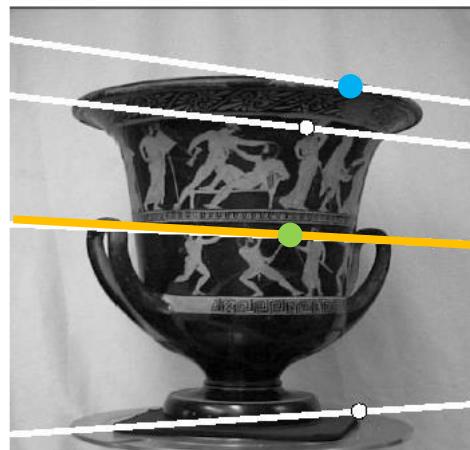
$$\begin{bmatrix} u_{1,i} \\ v_{1,i} \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot K_2^{-1} \cdot \begin{bmatrix} u_{2,i} \\ v_{2,i} \\ 1 \end{bmatrix} = 0$$

Let:

$$r_i(\alpha, \beta, \gamma, t_1, t_2)$$

$$= \begin{bmatrix} u_{1,i} \\ v_{1,i} \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot K_2^{-1} \cdot \begin{bmatrix} u_{2,i} \\ v_{2,i} \\ 1 \end{bmatrix}$$

$r_i(\alpha, \beta, \gamma, t_1, t_2)$ - means how good the essential matrix constraint is respected given that point correspondences have noise (i.e. how close is **green** point to **orange** line)



- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>

$$\text{If } F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} \text{ then the}$$

$$\text{constraint becomes } [u_1 \quad v_1 \quad 1] \cdot \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = 0$$

Essential matrix estimation

Mathematica model

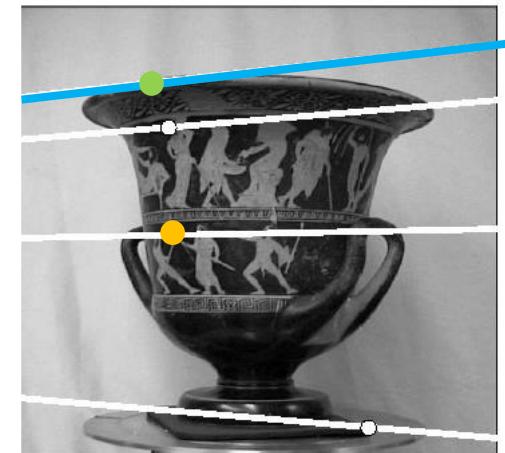
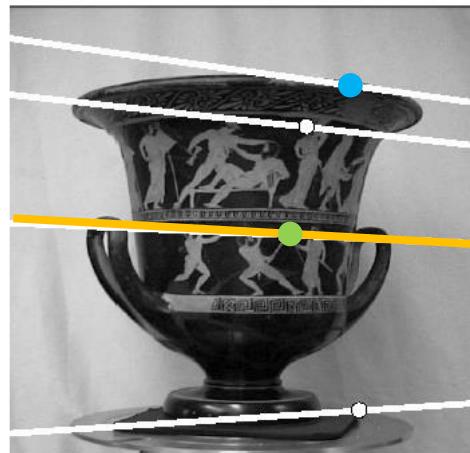
For **more points** we have:

$$\begin{bmatrix} u_{1,i} \\ v_{1,i} \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot K_2^{-1} \cdot \begin{bmatrix} u_{2,i} \\ v_{2,i} \\ 1 \end{bmatrix} = 0$$

Let:

$$r_i(\alpha, \beta, \gamma, t_1, t_2)$$

$$= \begin{bmatrix} u_{1,i} \\ v_{1,i} \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot K_2^{-1} \cdot \begin{bmatrix} u_{2,i} \\ v_{2,i} \\ 1 \end{bmatrix}$$



- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>

Find $\alpha, \beta, \gamma, t_1, t_2$ by doing:

$$\operatorname{argmin}_i \sum (r_i(\alpha, \beta, \gamma, t_1, t_2))^2$$

If $F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$ then the

constraint becomes $[u_1 \quad v_1 \quad 1] \cdot \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = 0$

Essential matrix estimation

Mathematica model

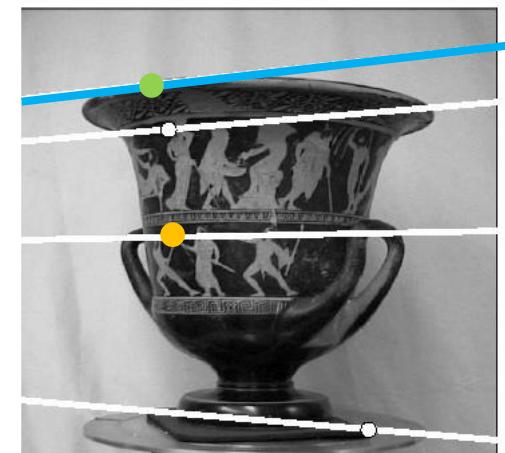
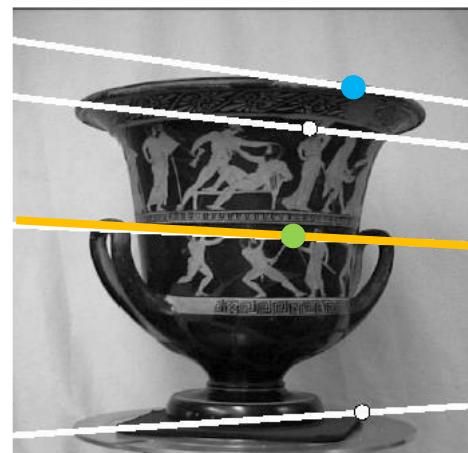
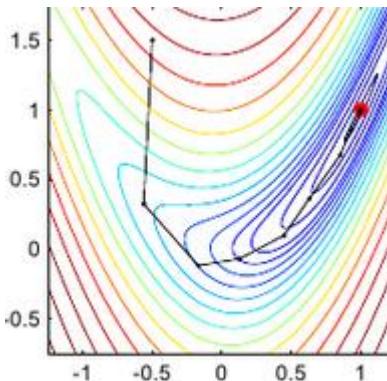
Find $\alpha, \beta, \gamma, t_1, t_2$ by doing:

$$\operatorname{argmin} \sum_i (r_i(\alpha, \beta, \gamma, t_1, t_2))^2$$

Gauss-Newton (see your numerical calculus course)

https://en.wikipedia.org/wiki/Gauss%20Newton_algorithm

Basic idea: find solution for the minimum iteratively



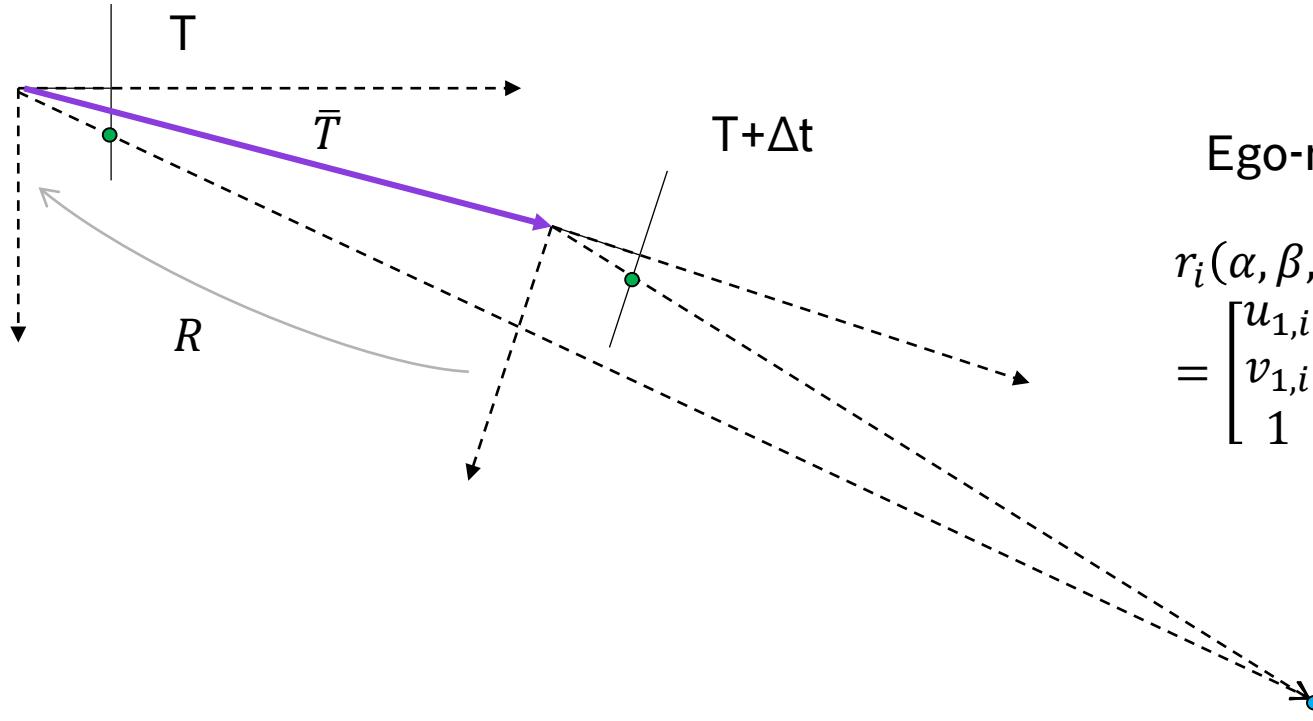
- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>

$$\text{If } F \cdot \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} \text{ then the}$$

$$\text{constraint becomes } [u_1 \quad v_1 \quad 1] \cdot \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = 0$$

Review of the last course and goals for today

Goal for today – geometry of a mono system



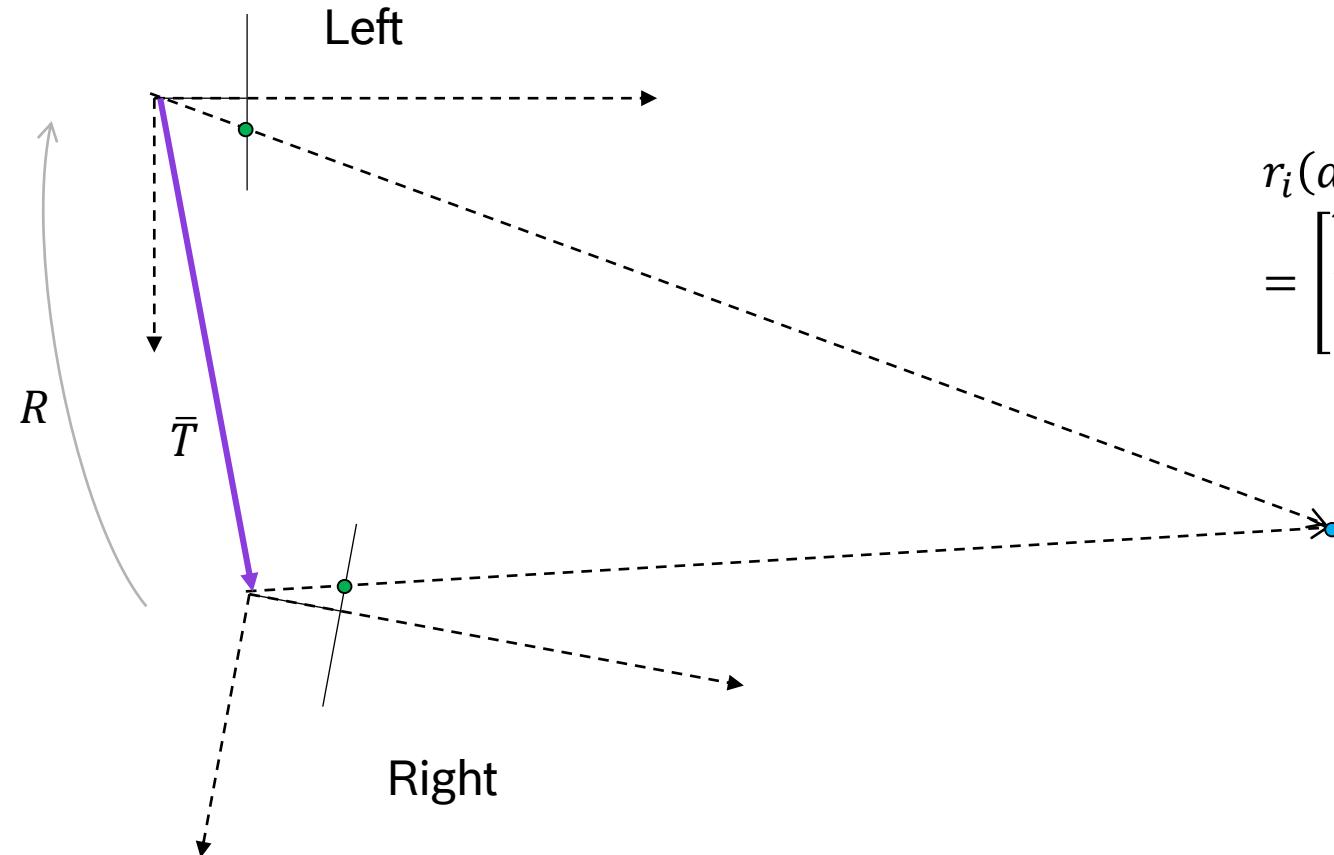
Ego-motion estimation for mono

$$r_i(\alpha, \beta, \gamma, t_1, t_2) = \begin{bmatrix} u_{1,i} \\ v_{1,i} \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot K_2^{-1} \cdot \begin{bmatrix} u_{2,i} \\ v_{2,i} \\ 1 \end{bmatrix}$$

$$\operatorname{argmin}_i \sum (r_i(\alpha, \beta, \gamma, t_1, t_2))^2$$

Review of the last course and goals for today

Goal for today - geometry of a stereo system



Relative pose estimation for stereo

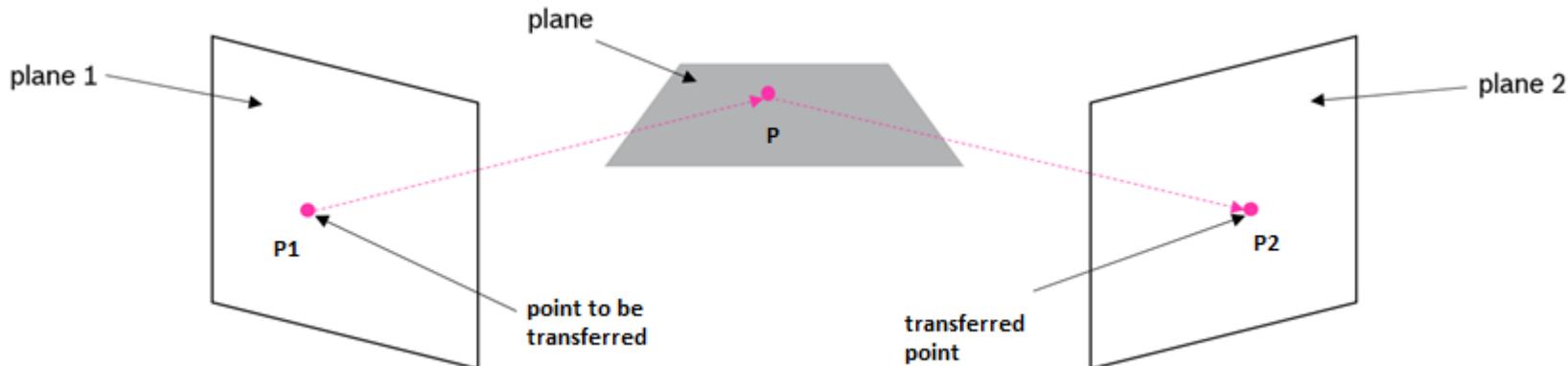
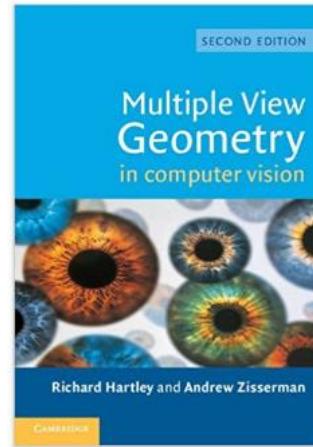
$$r_i(\alpha, \beta, \gamma, t_1, t_2) \\ = \begin{bmatrix} u_{1,i} \\ v_{1,i} \\ 1 \end{bmatrix}' \cdot K_1^{-T} \cdot E(\alpha, \beta, \gamma, t_1, t_2) \cdot K_2^{-1} \cdot \begin{bmatrix} u_{2,i} \\ v_{2,i} \\ 1 \end{bmatrix}$$

$$\operatorname{argmin}_i \sum (r_i(\alpha, \beta, \gamma, t_1, t_2))^2$$

PROJECTIVE GEOMETRY

References

- <https://www.amazon.com/Multiple-View-Geometry-Computer-Vision/dp/0521540518>
- <http://robotics.stanford.edu/~birch/projective/>
- <http://robotics.stanford.edu/~birch/projective/projective.pdf>
- <https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>
- <http://mathworld.wolfram.com/ProjectiveGeometry.html>



H_{image}, H_1, H_2 – homography matrices

Transfer via plane
For details see the references

$$P_1 = H_1 \cdot P$$

$$P_2 = H_2 \cdot P$$

$$H_{image} = H_2 \cdot H_1^{-1}$$

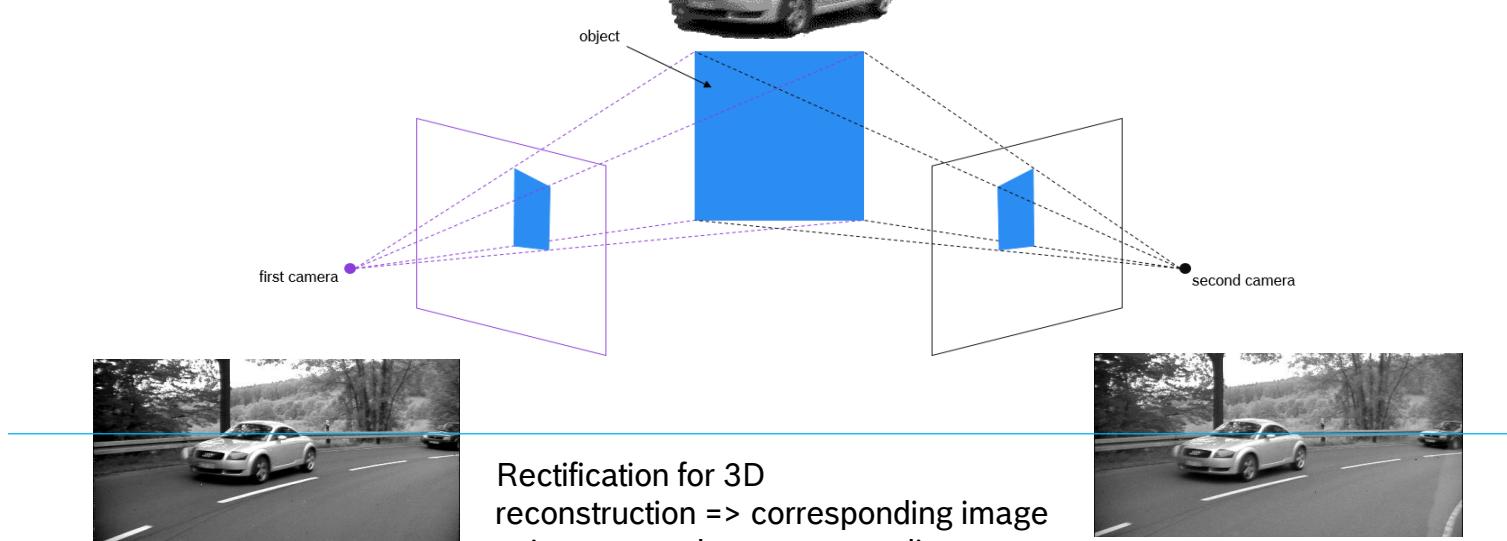
$$P_2 = H_{image} \cdot P_1$$

CONCLUSIONS

Conclusions

Conclusions

- Essential matrix as key part for
 - Rectification
 - Ego-motion estimation
- Same theory applicable for mono & stereo



Thank you for your attention!



THE PATH TO DEEP LEARNING

Ştefan Máthé

Outline

- ▶ Math Refresher: Probability Theory
- ▶ The Challenge
- ▶ Machine Learning
- ▶ The Path to Deep Learning

MATH REFRESHER: PROBABILITY THEORY

Quick Math Refresher

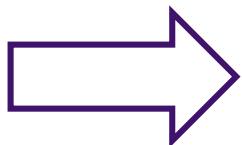
Uncertainty

- ▶ Sources of uncertainty
 - ▶ Stochastic universe
 - ▶ Incomplete observations
 - ▶ Incomplete modelling
- ▶ Approaches to measure uncertainty:
 - ▶ **Frequentist**: repeatable event outcomes
 - ▶ **Bayesian**: degree of belief
- ▶ **Random Variables**

Quick Math Refresher

Uncertainty

- ▶ Sources of uncertainty
 - ▶ Stochastic universe
 - ▶ Incomplete observations
 - ▶ Incomplete modelling
- ▶ Approaches to measure uncertainty:
 - ▶ **Frequentist**: repeatable event outcomes
 - ▶ **Bayesian**: degree of belief
- ▶ **Random Variables**



Governed by the same set of axioms

Quick Math Refresher

Probabilities

- ▶ Probability Distributions
 - ▶ Probability Mass Function (PMF) \Leftrightarrow discrete variables
 - ▶ Probability Distribution Function (PDF) \Leftrightarrow continuous variables
- ▶ Joint Probability Distribution: $p(x, y)$
- ▶ Prior Probability Distribution: $p(x)$
- ▶ Conditional Probability Distribution: $p(y|x) = \frac{p(x,y)}{p(x)}$

Quick Math Refresher

Common Probability Distributions

► Binary Variables:

► Bernoulli

► Multinoulli

► Multinomial

$$P(X = 1) = \theta$$

$$P(X = 0) = 1 - \theta$$

Bernoulli

$$P(X = i) = \theta_i, \forall i, 1 \leq i \leq k - 1$$

$$P(X = 0) = 1 - \sum_{i=1}^{k-1} \theta_i$$

Multinoulli

► Continuous Variables:

► Gaussian

► Dirac

► Exponential

► Laplace

$$p(X = x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

Gaussian (1-dimensional)

$$p(X = x) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} e^{\frac{(x-\mu)^\top \Sigma (x-\mu)}{2\sigma^2}}$$

Gaussian (n-dimensional)

$$p(X = x) = \delta(x - \mu)$$

$$\int_a^b \delta(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

Dirac (1-dimensional)

Quick Math Refresher

Expectation and KL Divergence

- ▶ Expectation: “average” value of a function $f(x)$ when x is drawn from $p(x)$
- ▶ For PDFs: $\mathbb{E}_{x \sim p(x)}[f(x)] = \int_x f(x)p(x)dx$
- ▶ For PMFs: $\mathbb{E}_{x \sim P(x)}[f(x)] = \sum_x f(x)P(x)dx$
- ▶ KL Divergence: “distance” between two probability distributions

$$D_{KL}[P||Q] = \mathbb{E}_{x \sim P(x)} \left[\log \frac{P(x)}{Q(x)} \right]$$

- ▶ Not a true distance (non-negative, but asymmetric and does not obey the triangle inequality)

THE CHALLENGE

The Challenge

Object Class Recognition: Easy or Hard?



Is this an image of a cat?

The Challenge

Object Class Recognition: Easy or Hard?



What's
the
catch?

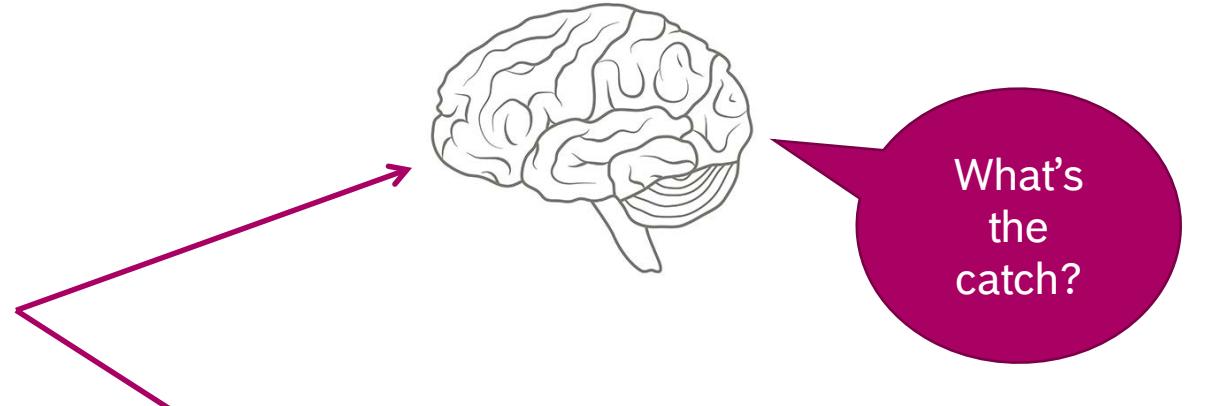
Is this an image of a cat?

The Challenge

Object Class Recognition: Easy or Hard?



Is this an image of a cat?



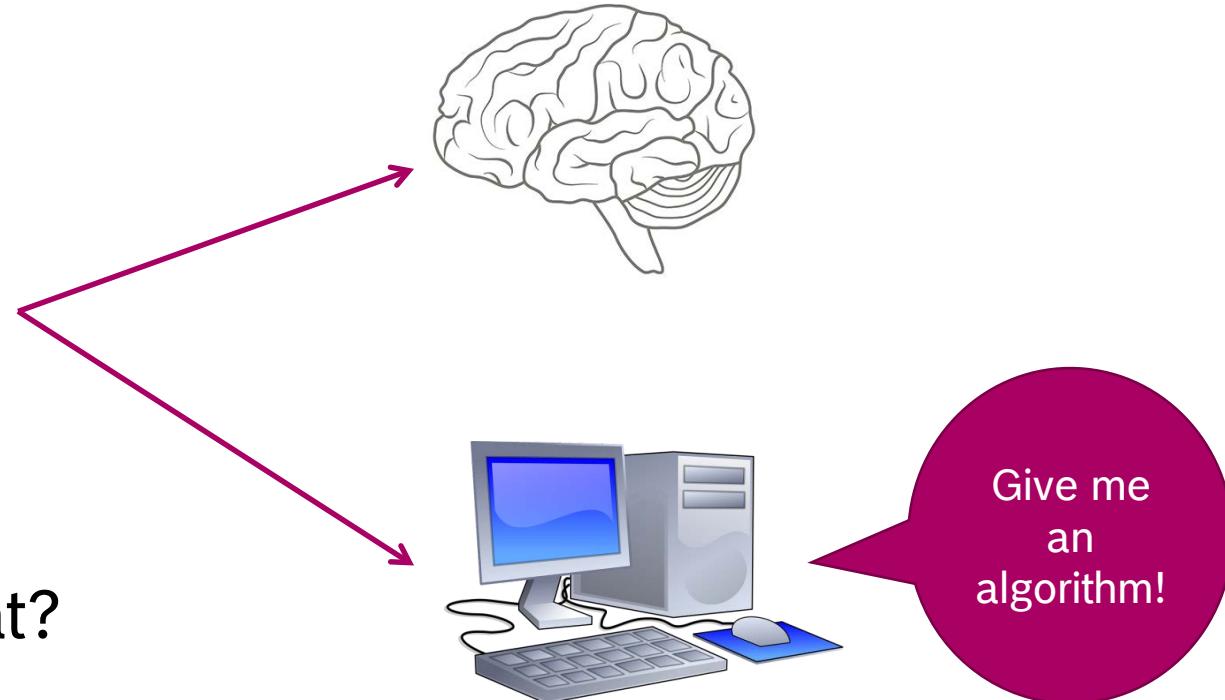
I don't
know. I
only see
pixels

The Challenge

Object Class Recognition: Easy or Hard?

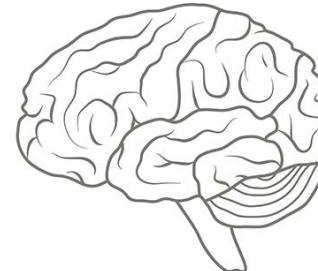
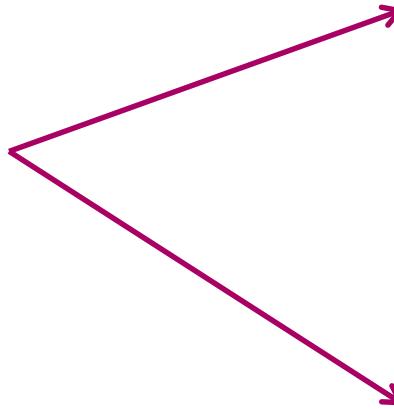


Is this an image of a cat?



The Challenge

Object Class Recognition: Easy or Hard?



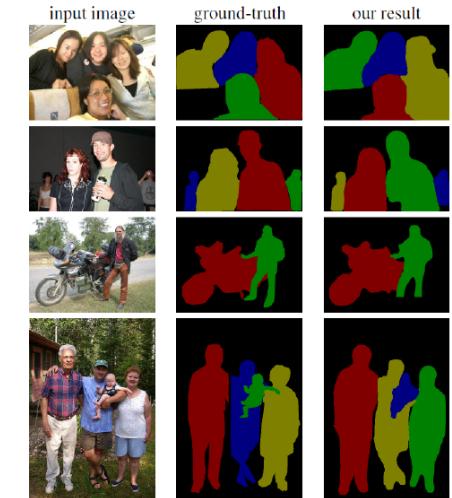
I don't have one.
I look for a furry
mammal with
long pointy ears,
etc.

Is this an image of a cat?

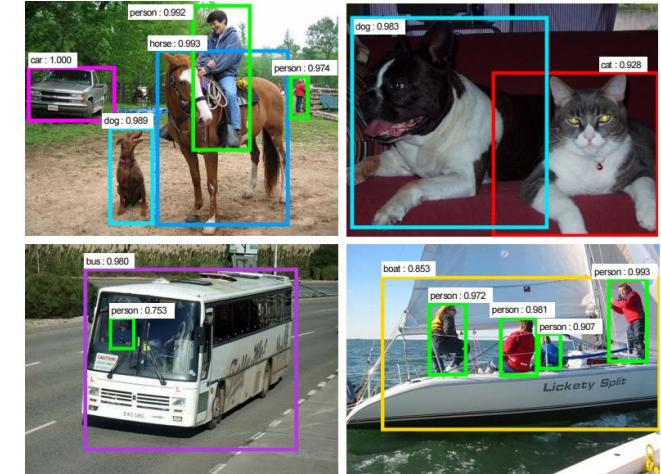
The Challenge Problems with Inconspicuous Solutions

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

handwritten digit recognition



instance level segmentation



object detection

The Challenge

Humans vs. Thinking Machines

World of Thinking Machines

World of Humans

	easy	hard
easy	(not interesting)	<ul style="list-style-type: none">• speech recognition• face recognition• car driving
hard	<ul style="list-style-type: none">• chess• go• question-answering (Quora)	<ul style="list-style-type: none">• plant identification• cancer diagnosis

The Challenge

Humans vs. Thinking Machines

World of Thinking Machines

World of Humans

	easy	hard
easy	(not interesting)	<ul style="list-style-type: none">• speech recognition• face recognition• car driving
hard	<ul style="list-style-type: none">• chess• go• question-answering (Quora)	<ul style="list-style-type: none">• plant identification• cancer diagnosis
 <ul style="list-style-type: none">• formal problem environments• rule-based inference		

The Challenge

Humans vs. Thinking Machines

World of Thinking Machines

World of Humans

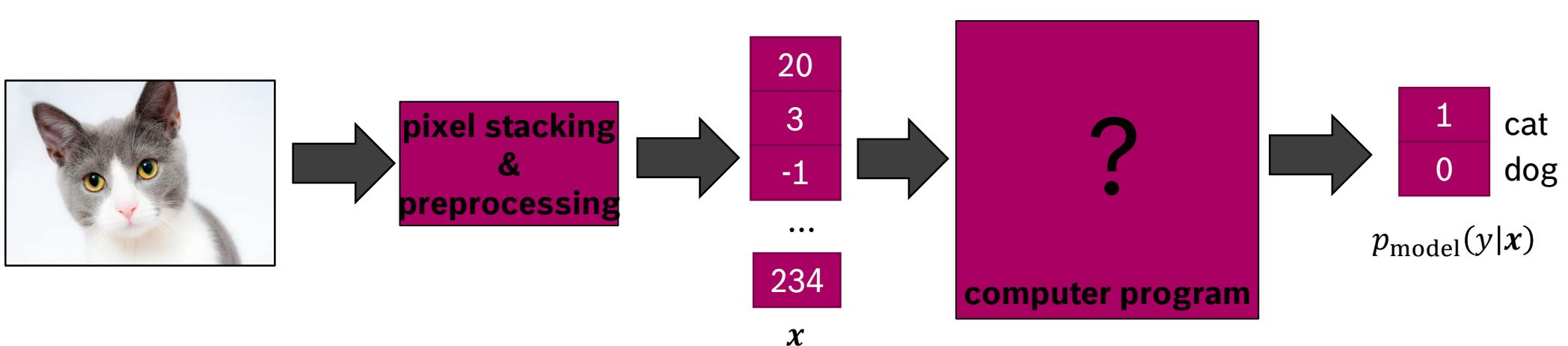
		easy	hard
		(not interesting)	• speech recognition • face recognition • car driving
easy	easy	• chess • go • question-answering (Quora)	• plant identification • cancer diagnosis
hard	hard	<ul style="list-style-type: none">formal problem environmentsrule-based inference	<ul style="list-style-type: none">real-world environmentsrequire informal knowledge

MACHINE LEARNING

Machine Learning

What do we want?

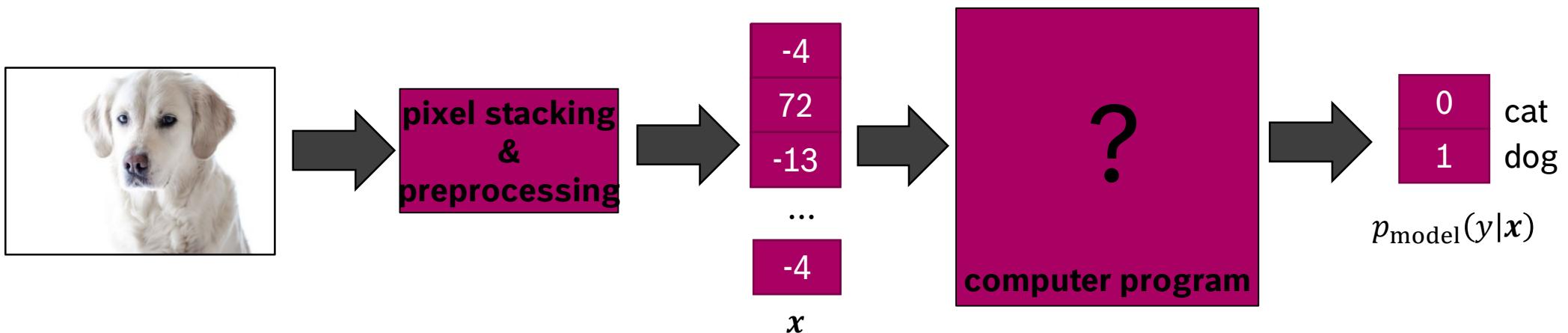
- A computer program
 - Input: a **feature** vector x obtained from the input image
 - Output: the probability $p_{\text{model}}(y|x)$ for each object class y



Machine Learning

What do we want?

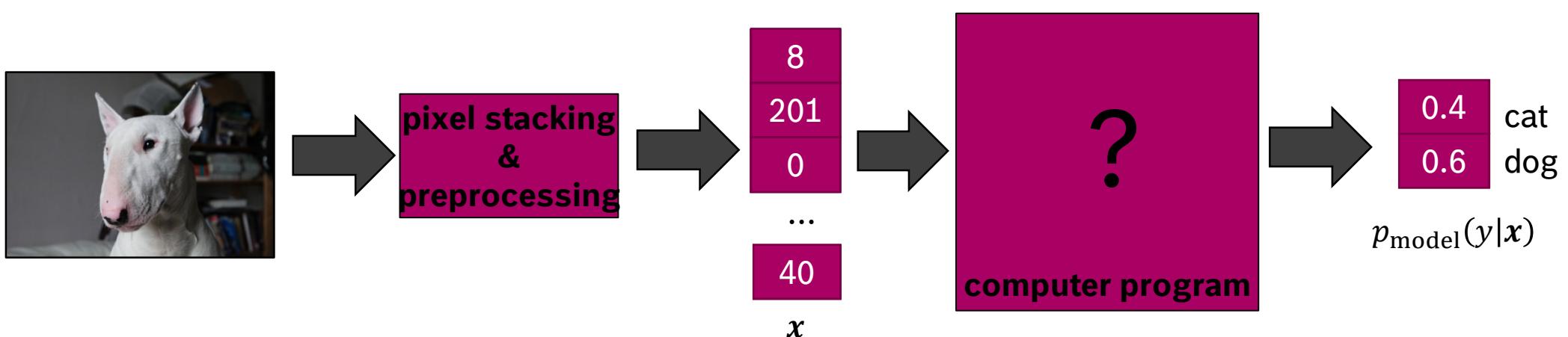
- A computer program
 - Input: a **feature** vector x obtained from the input image
 - Output: the probability $p_{\text{model}}(y|x)$ for each object class y



Machine Learning

What do we want?

- A computer program
 - Input: a **feature** vector x obtained from the input image
 - Output: the probability $p_{\text{model}}(y|x)$ for each object class y

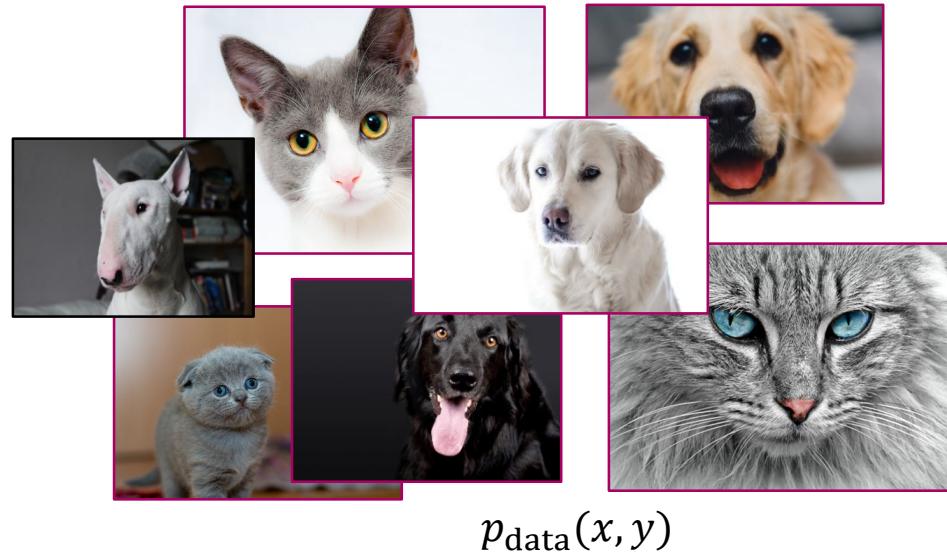


Machine Learning

What is a good solution?

- Let $p_{\text{data}}(x, y)$ be the **true distribution** over features and labels
- Program output has a high **expected likelihood** on the true distribution:

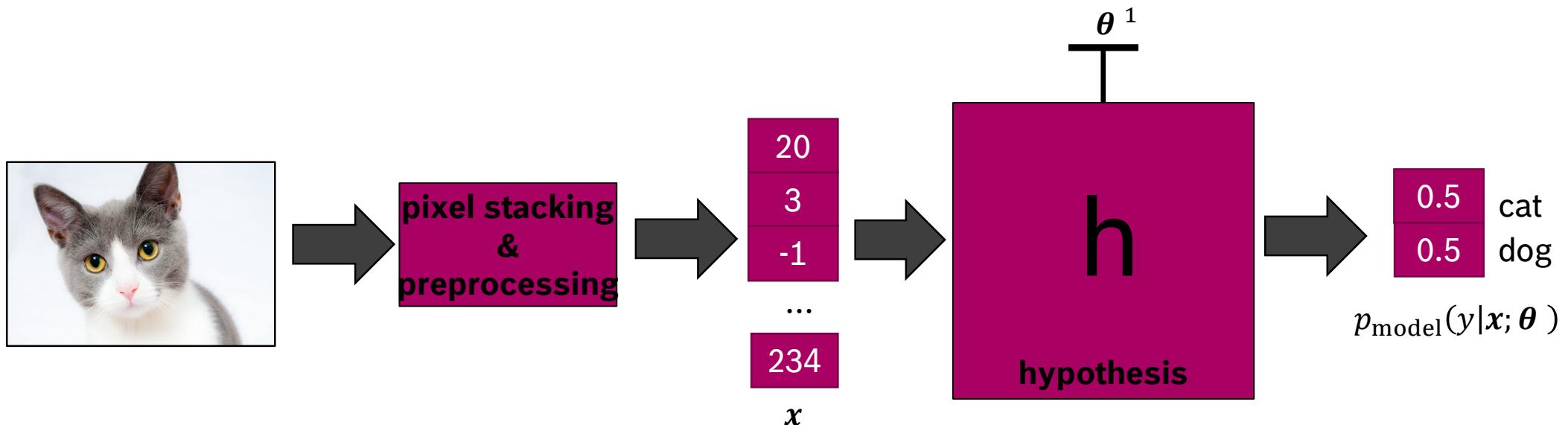
$$\mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x)]$$



Machine Learning

Parametric Hypothesis Space

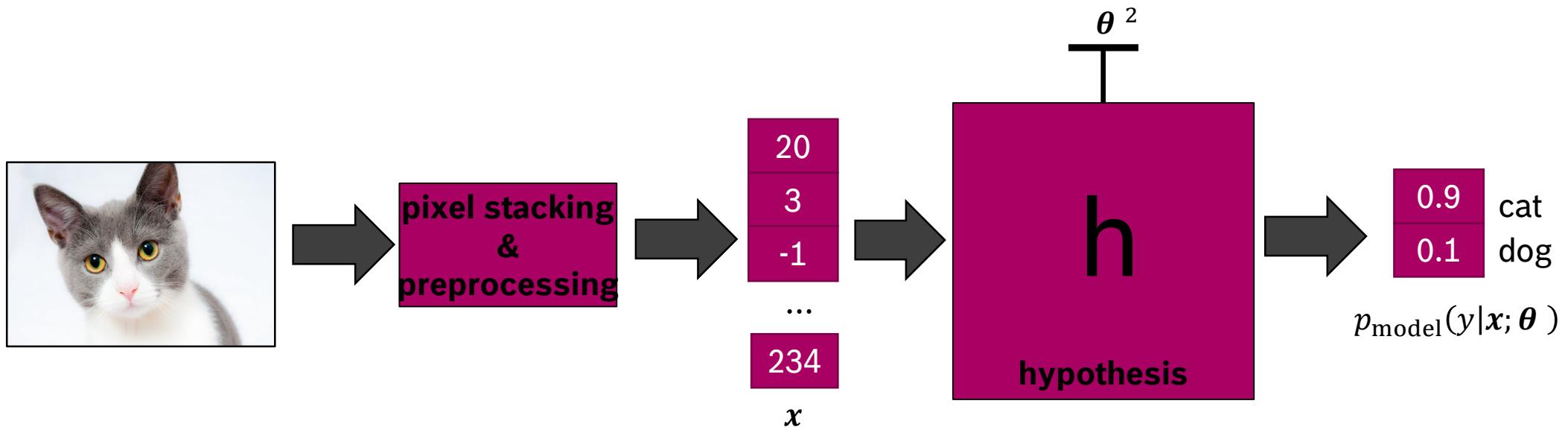
- Consider a whole set H of possible programs H , called **hypothesis space**
- Assume each hypothesis is uniquely characterized by a **parameter vector** θ
- The parameter vector changes the behavior of our program



Machine Learning

Parametric Hypothesis Space

- Consider a whole set H of possible programs H , called **hypothesis space**
- Assume each hypothesis is uniquely characterized by a **parameter vector** θ
- The parameter vector changes the behavior of our program

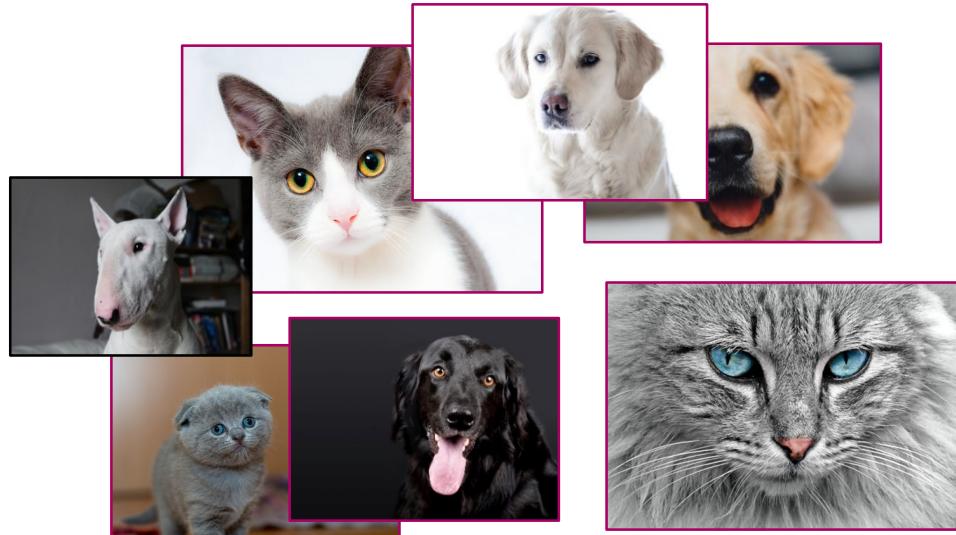


Machine Learning

The Ideal Setting

- Choose the hypothesis θ^* that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \theta)]$$

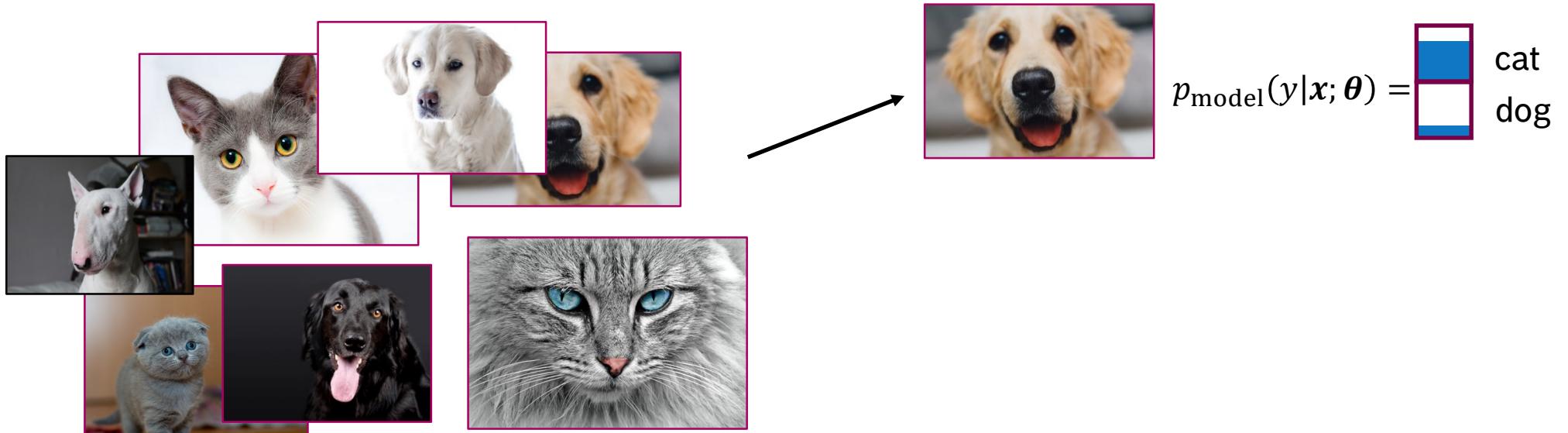


Machine Learning

The Ideal Setting

- Choose the hypothesis θ^* that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \theta)]$$

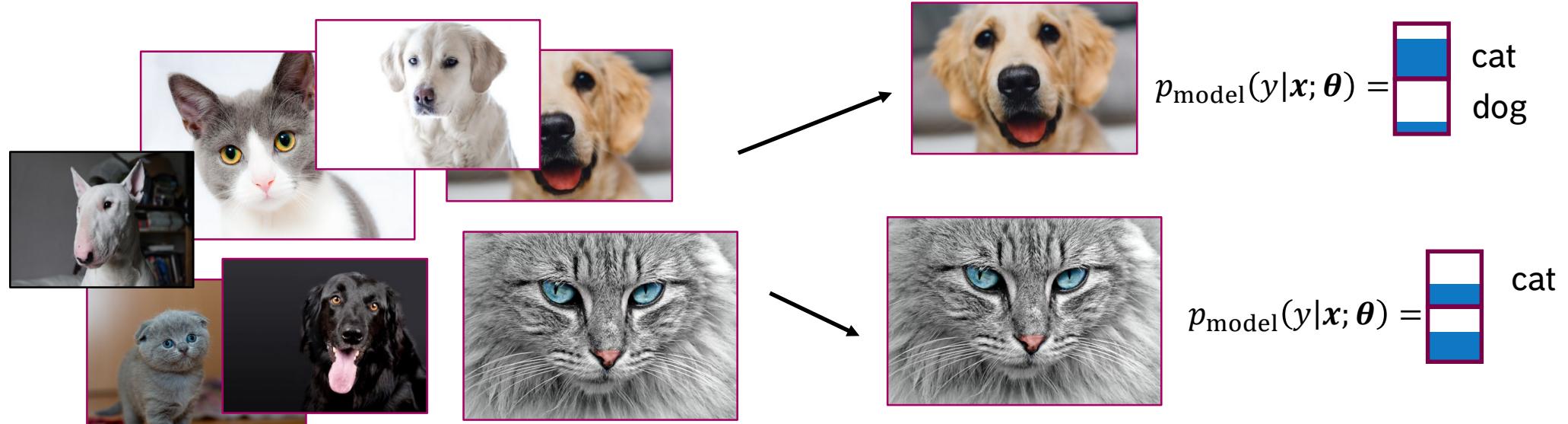


Machine Learning

The Ideal Setting

- Choose the hypothesis θ^* that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \theta)]$$

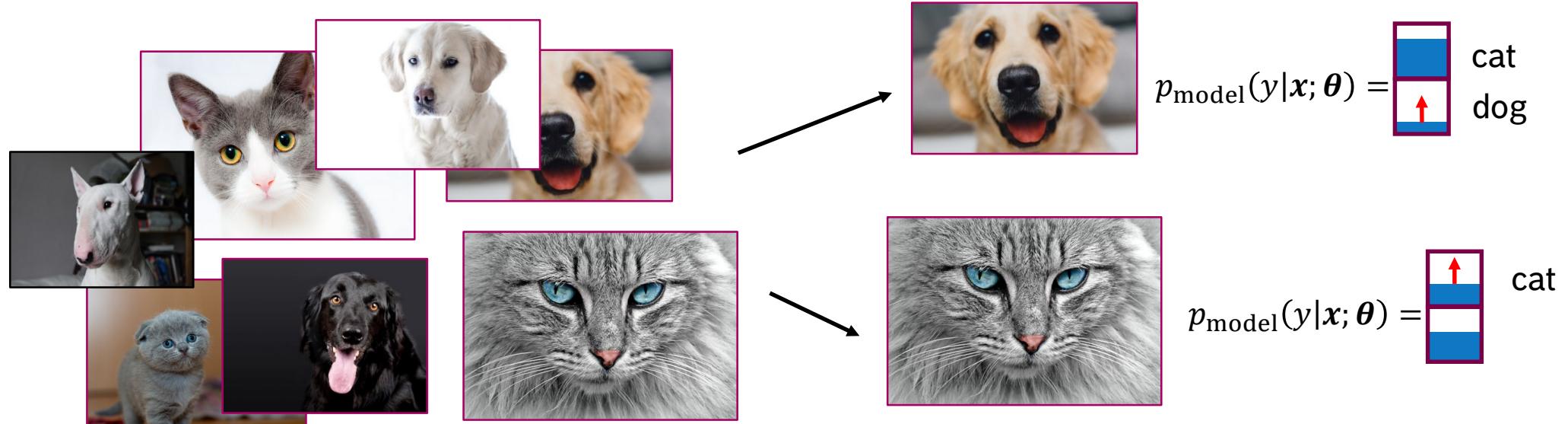


Machine Learning

The Ideal Setting

- Choose the hypothesis θ^* that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \theta)]$$

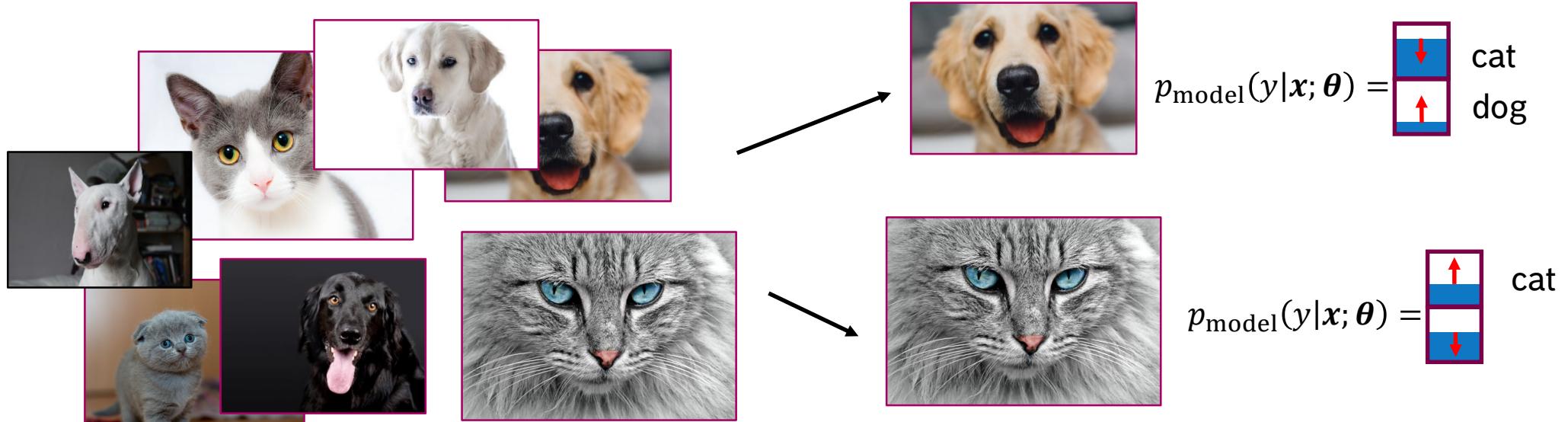


Machine Learning

The Ideal Setting

- Choose the hypothesis θ^* that works best in the real world
- Formally: maximize the expected log likelihood on the true distribution

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \theta)]$$



Machine Learning

Why Maximize the Log Likelihood?

- Numerical reasons: will come back to these later!
- Information theoretic reasons: same as minimizing the KL divergence between the predicted and ground truth posteriors

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}(x,y)} [\log p_{\text{model}}(y|x; \boldsymbol{\theta})]$$



Prove the
equivalence
as homework!



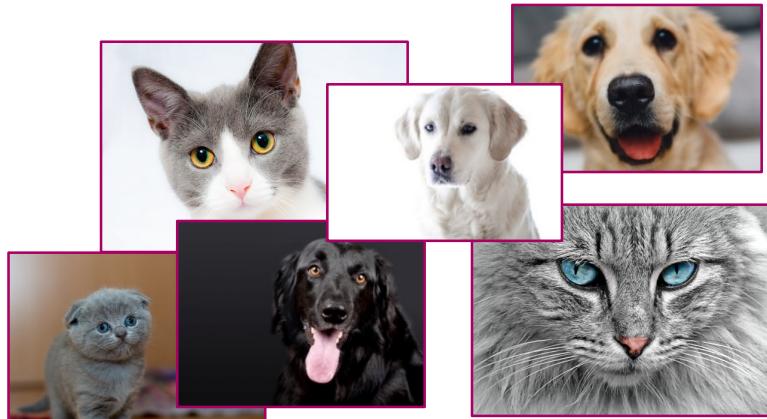
$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}(x)} [D_{\text{KL}}(p_{\text{data}}(y|x; \boldsymbol{\theta}) || p_{\text{model}}(y|x; \boldsymbol{\theta}))]$$

Machine Learning Expectation Meets Reality

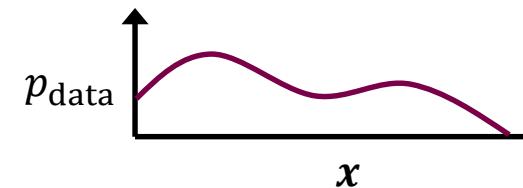
- But we do not have the true distribution!
- We only have a small **training set** drawn from the true distribution

$$x_i, y_i \sim p_{\text{data}}(x, y), i = \overline{1, n}$$

- The training set can be used to define the **empirical data distribution**:



$p_{\text{data}}(x, y)$



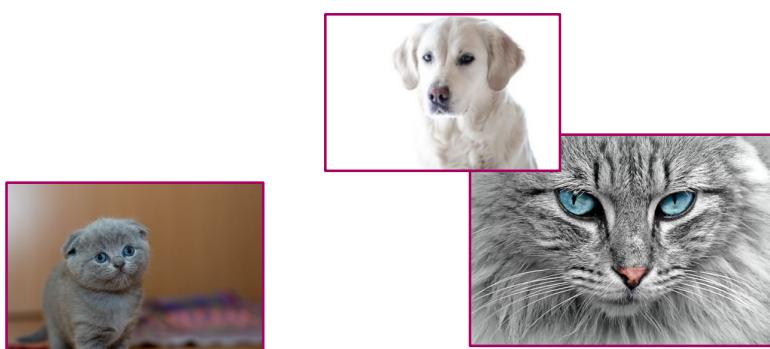
Machine Learning Expectation Meets Reality

- But we do not have the true distribution!
- We only have a small **training set** drawn from the true distribution

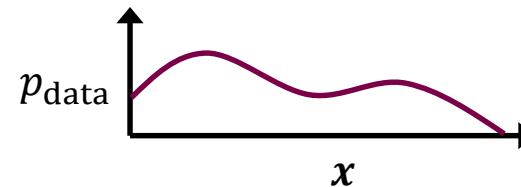
$$x_i, y_i \sim p_{\text{data}}(x, y), i = \overline{1, n}$$

- The training set can be used to define the **empirical data distribution**:

$$\hat{p}_{\text{data}}(x, y) = \sum_{i=1}^n \llbracket x = x_i \wedge y = y_i \rrbracket$$



$$\hat{p}_{\text{data}}(x, y)$$



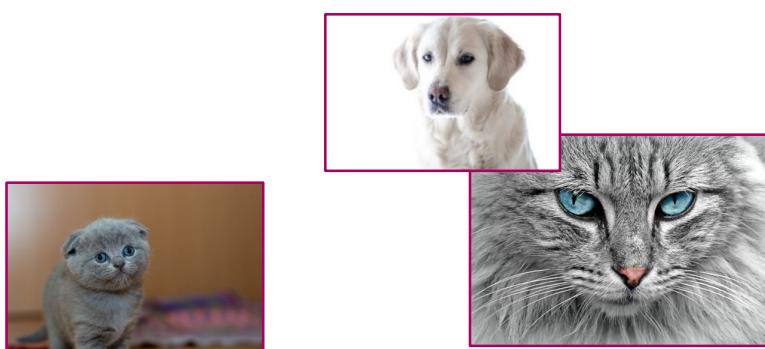
Machine Learning Expectation Meets Reality

- But we do not have the true distribution!
- We only have a small **training set** drawn from the true distribution

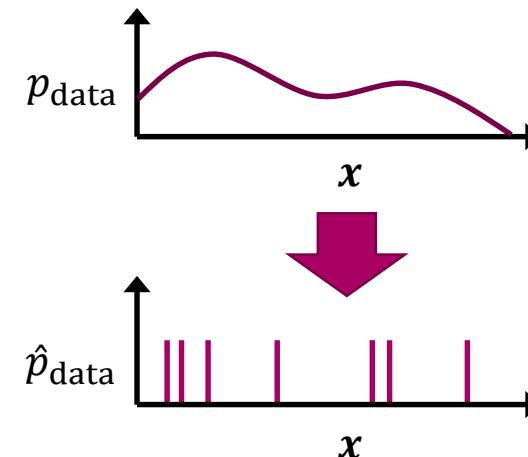
$$\mathbf{x}_i, y_i \sim p_{\text{data}}(\mathbf{x}, y), i = \overline{1, n}$$

- The training set can be used to define the **empirical data distribution**:

$$\hat{p}_{\text{data}}(\mathbf{x}, y) = \sum_{i=1}^n \llbracket \mathbf{x} = \mathbf{x}_i \wedge y = y_i \rrbracket$$



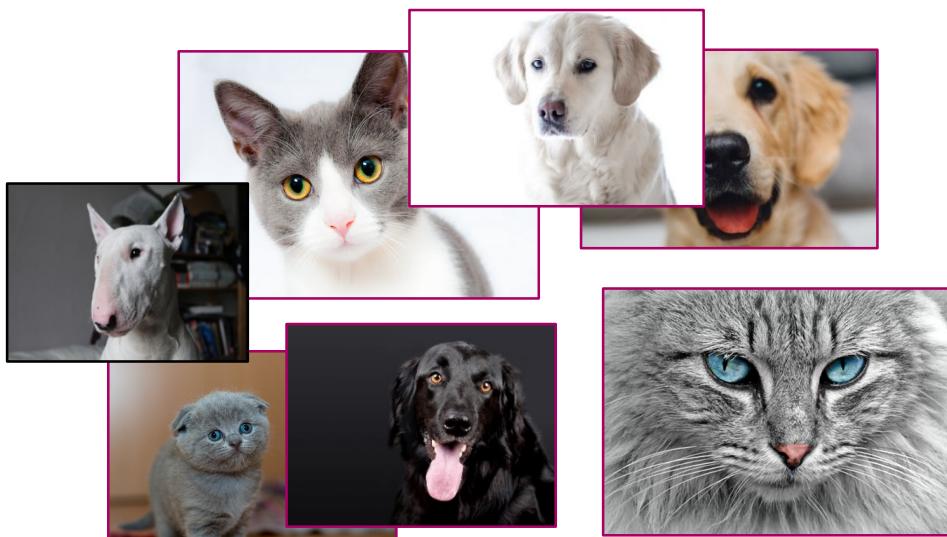
$$\hat{p}_{\text{data}}(\mathbf{x}, y)$$



Machine Learning

The Compromise

- Solution: maximize the likelihood of the empirical data distribution
- Why would this work?



$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \boldsymbol{\theta})]$$

$$p_{\text{data}}(x, y)$$

Machine Learning

The Compromise

- Solution: maximize the likelihood of the empirical data distribution
- Why would this work?



$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \boldsymbol{\theta})]$$

$$\hat{p}_{\text{data}}(x, y)$$

Machine Learning

The Compromise

- Solution: maximize the likelihood of the empirical data distribution
- Why would this work?



$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{x,y \sim p_{\text{data}}} [\log p_{\text{model}}(y|x; \boldsymbol{\theta})]$$



$$\boldsymbol{\theta}^{\text{ML}} = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^n \log p_{\text{model}}(y_i|x_i; \boldsymbol{\theta})$$

$\hat{p}_{\text{data}}(x, y)$

Machine Learning Justification: The IID Assumptions

- Look at the way we built our empirical distribution:

$$x_i, y_i \sim p_{\text{data}}(x, y), i = \overline{1, n}$$

- Assumption 1:
 - Data samples are drawn **independently** of each other
 - Why? Hypothesis program always processes inputs from scratch.
- Assumption 2:
 - Training samples were **identically distributed**, i.e. drawn from the true distribution
 - Why? No learning unless training set is representative of the true world
- Are these assumptions enough?

Machine Learning

Justification: Maximum Likelihood Consistency

- An estimator is **consistent** if it converges given enough training data ($n \rightarrow \infty$)
- ML is **consistent** if:
 1. The hypothesis set contains the true model. For classifiers:
$$\exists \boldsymbol{\theta}, \forall \mathbf{x}, \forall y, p_{\text{data}}(y|\mathbf{x}) = \log p_{\text{model}}(y|\mathbf{x}; \boldsymbol{\theta})$$
 2. Each hypothesis is uniquely identified by $\boldsymbol{\theta}$
- In practice we never have enough data!

Machine Learning

The Loss Function

- Can also see ML as minimizing the **negative log likelihood** (NLL):

$$\boldsymbol{\theta}^{\text{ML}} = \operatorname{argmin}_{\boldsymbol{\theta}} -\frac{1}{m} \sum_{i=1}^n \log p_{\text{model}}(y_i | \mathbf{x}_i; \boldsymbol{\theta})$$

- In general, we seek to minimize a **loss function** over the training set:

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

- NLL is a special case of loss function:

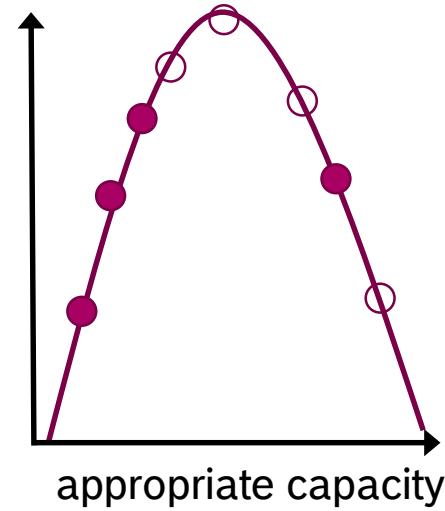
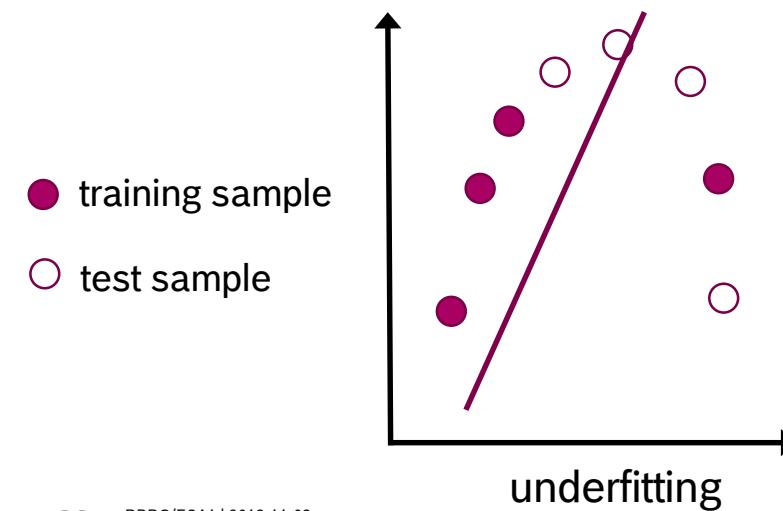
$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{m} \sum_{i=1}^n \log p_{\text{model}}(y_i | \mathbf{x}_i; \boldsymbol{\theta})$$

- The training loss is also called the **empirical risk**

Machine Learning

The Dangers of Failed Assumptions: Underfitting and Overfitting

- **Underfitting:** cannot find a model that fits the training data
↔ high loss on training data
- **Overfitting:** cannot find a model that generalizes on test (unseen) data
↔ low training loss, high test loss



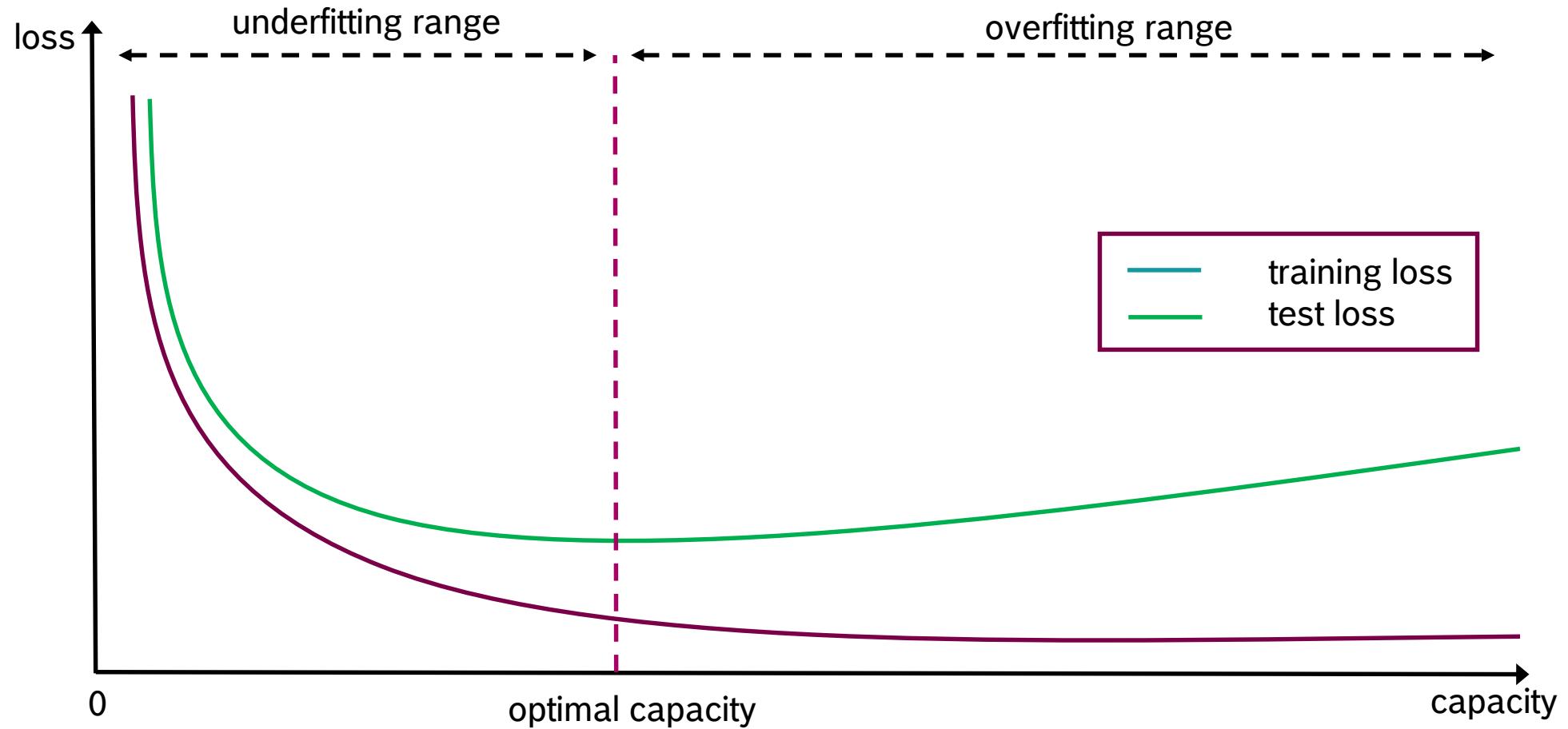
Machine Learning Model Capacity

- **(Effective) capacity** = ability of the model to fit a wide variety of functions
- **Representational capacity** = ability of the hypothesis set to fit a wide variety of functions
- Due to imperfect optimizers:

effective capacity \leq representational capacity

- Underfitting \Leftrightarrow effective capacity too low
- Overfitting \Leftrightarrow effective capacity too high
- Formal definitions of capacity exist: Vapnik-Chervonenkis Dimension, Rademacher Complexity

Machine Learning Capacity-Loss Relationship Curve



Machine Learning Regularization (Capacity Control)

- Need to control model capacity:
 - via the optimization process (do not look for the optimal solution)
 - optimize a different objective that also looks at the parameters
 - restrict hypothesis set
- All of these methods introduce an **inductive bias**:
 - Favor some hypotheses over others
 - Ockham's Razor principle: prefer simple explanations of the data
- There is no universally applicable bias:



The “no free lunch” theorem: Averaged over all data distributions, any two machine learning algorithms have the same accuracy (Wolpert, 1996)

Machine Learning Definition

- Machine learning can do much more than classification!

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience $E.$ ” (Mitchell, 1997)

Machine Learning

The Task (T)

Task	Input	Output	Example Problem
classification	feature vector	discrete category	object recognition
regression	feature vector	real-valued quantity	image quality assessment
transcription	feature vector	sequence of symbols	transcribe text from image
translation	sequence of symbols	sequence of symbols	translate from English to German
synthesis	sequence of symbols feature vector	real-valued signal	speech synthesis style transfer
denoising	real-valued signal	real-valued signal	image restoration

- The list above is far from exhaustive!

Machine Learning The Experience (E)



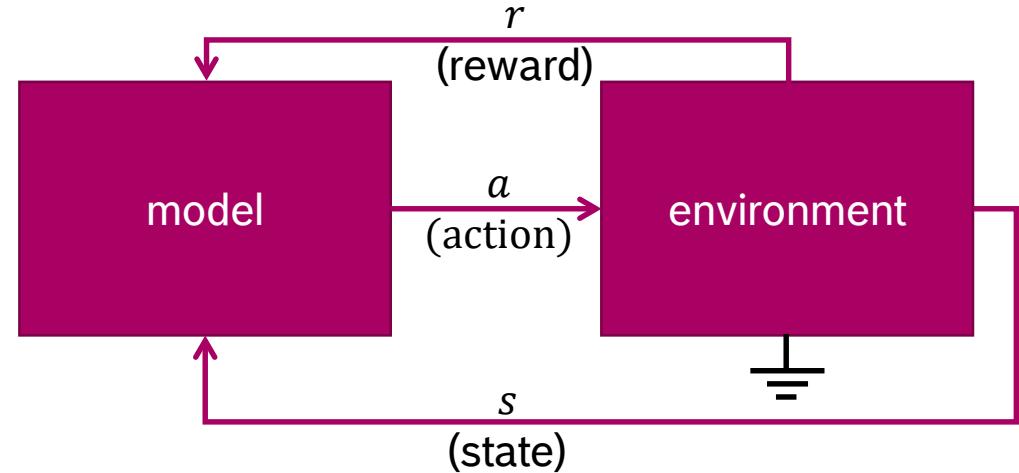
supervised learning



unsupervised learning



semi supervised learning
(some labels may be missing)



reinforcement learning

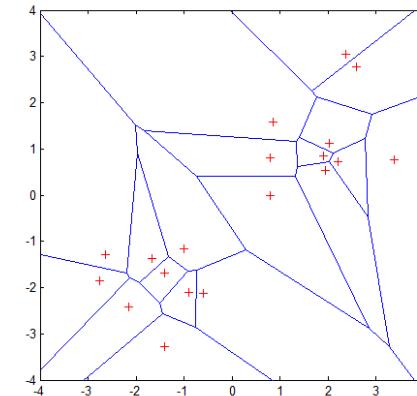
Machine Learning

The Performance Measure (P)

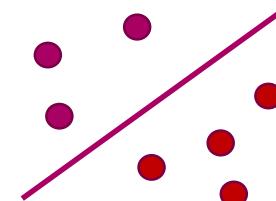
- Very much task dependent
- For non-interactive methods: loss measured over the test set
 - Negative log likelihood (cross entropy)
 - 0-1 loss
 - Euclidean Loss
 - Hinge Loss
 - etc.
- For reinforcement learning: expected reward
- Many losses correspond to probabilistic interpretations of model outputs (e.g. Euclidean Loss \Leftrightarrow Gaussian Output Distributions)

Machine Learning Model Examples

- Shallow models
 - Non-parametric:
 - k-Nearest Neighbor (kNN) (Fixed & Hodges, 1951)
 - Kernel Density Estimation (Parzen, 1956; Rosenblatt, 1962)
 - Parametric:
 - Linear Regression (Legendre, 1805)
 - Logistic Regression (David Cox, 1958)
 - ADALINE (Widrow & Hoff, 1960)
 - Support Vector Machines (SVM) (Vapnik & Chervonenkis, 1963)
 - Decision Trees (Morgan & Sonquist, 1963)
- Deep Models
 - Multilayer Perceptron (MLP) (Ivaknenko & Lapa, 1965)
 - Neocognitron (Fukushima, 1980)
 - Lenet-5 (Lecun et al., 1998)
 - AlexNet (Krizhevsky et al., 2012)
 - VGG (Simonyan and Zisserman, 2014)
 - ResNet (He et al., 2015)
 - DETR (Carion et al. 2020)



kNN decision surface



SVM decision surface

Machine Learning Recap

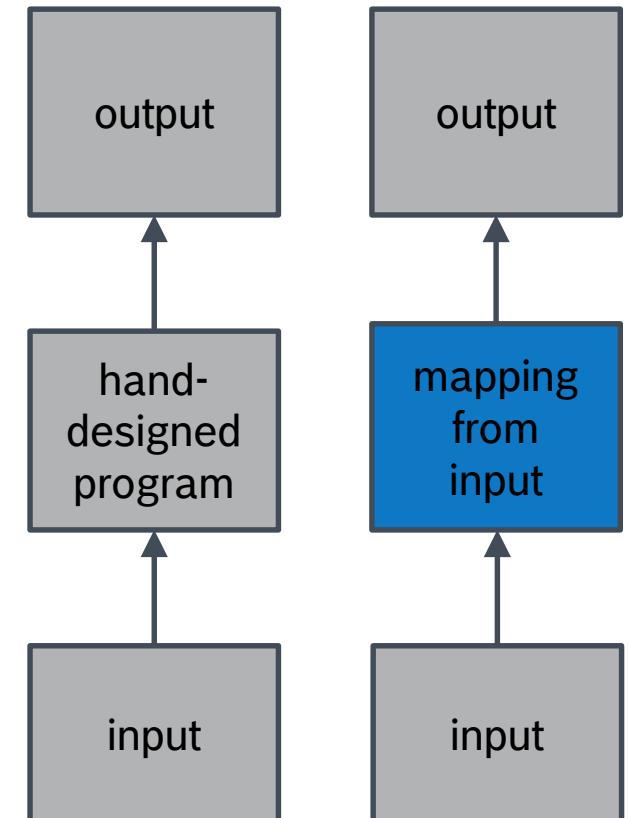
- Reformulated the inference problem as a machine learning problem
- Select the best hypothesis such that:
 - Fits the training data (seen)
 - Generalizes to the real world data (unseen)
- These are conflicting goals!
- The true challenges:
 - Controlling model capacity (underfitting, overfitting)
 - Introducing the right inductive bias
 - Finding the hypothesis that fits the data (optimization problem)

THE PATH TO DEEP LEARNING

The Path to Deep Learning

Approaches to AI

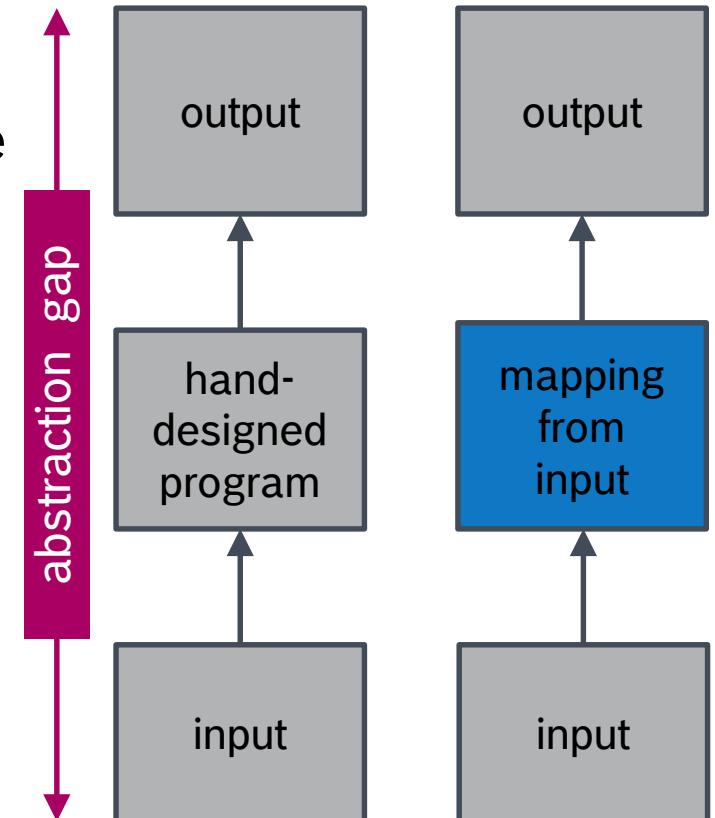
- Rule Based Approaches
 - Hard-code informal knowledge in a formal language
 - Not very successful 😞
- Machine Learning Approaches
 - Acquire knowledge automatically from real-world data
 - Most promising to date 😊
 - But wait! What kind of knowledge? Can we learn a mapping from input to output?
 - We need to cover a large **abstraction gap!**



The Path to Deep Learning

Approaches to AI

- Rule Based Approaches
 - Hard-code informal knowledge in a formal language
 - Not very successful 😞
- Machine Learning Approaches
 - Acquire knowledge automatically from real-world data
 - Most promising to date 😊
 - But wait! What kind of knowledge? Can we learn a mapping from input to output?
 - We need to cover a large **abstraction gap**!



The Path To Deep Learning

Closing the Abstraction Gap is Difficult!

- Images are the complex result of multiple interacting **factors of variation**
- Need to separate what is relevant from what is not!



illumination



deformation



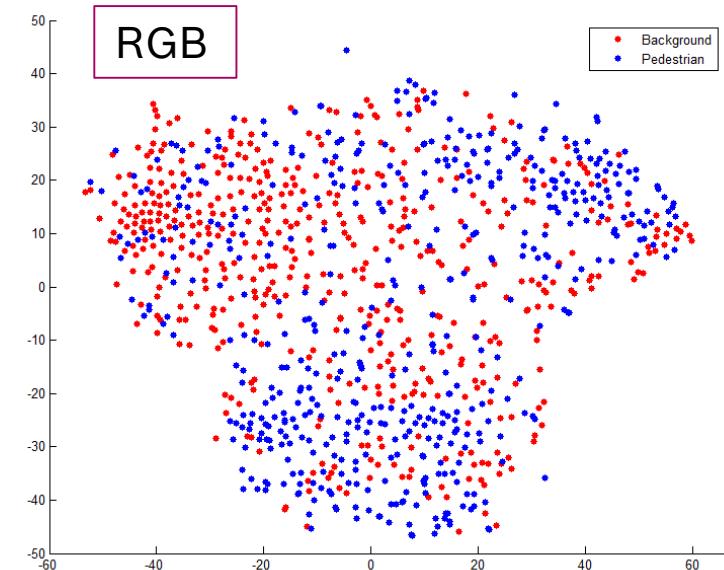
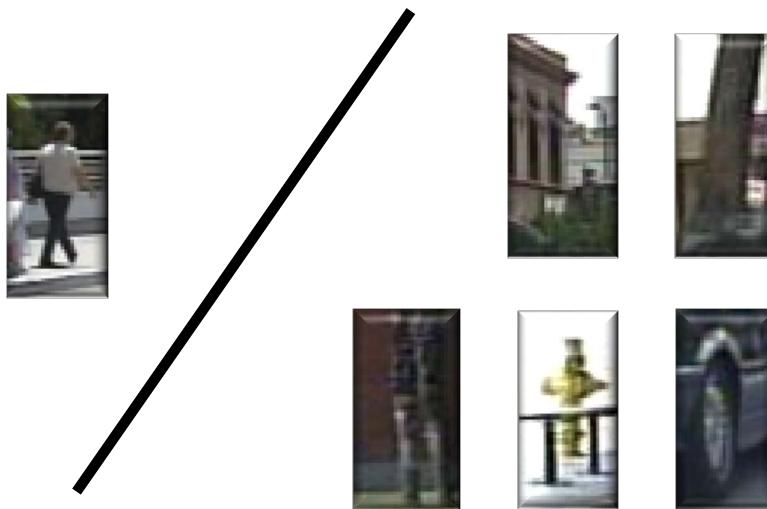
occlusion

Images taken from Fei-Fei, Karpathy & Johnson, Lecture Notes, 2016

The Path to Deep Learning

The Representation Problem

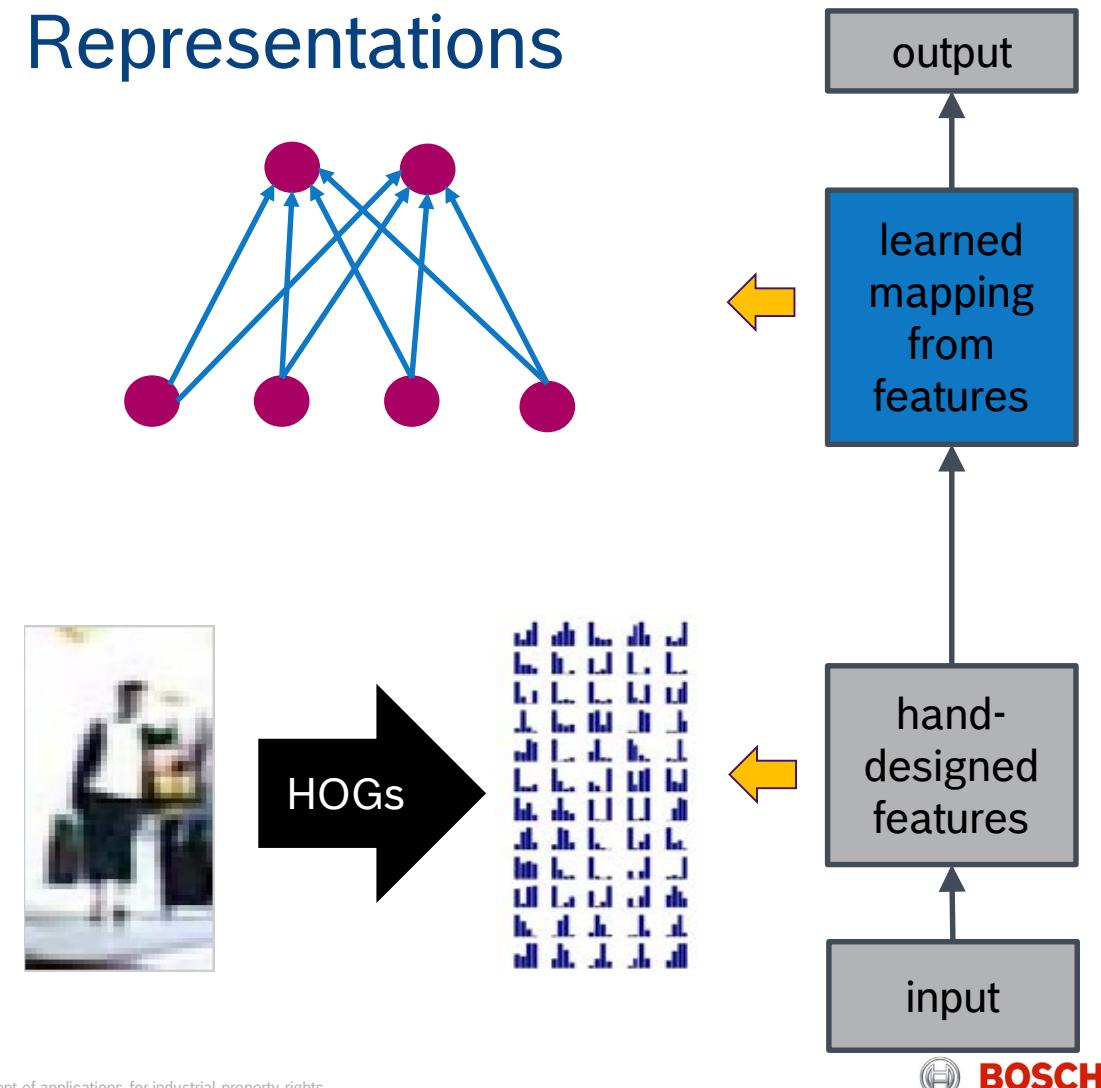
- We need **features**: relevant information extracted from the input
- How do we come up with good features?



The Path to Deep Learning

First Generation: Hand-Designed Representations

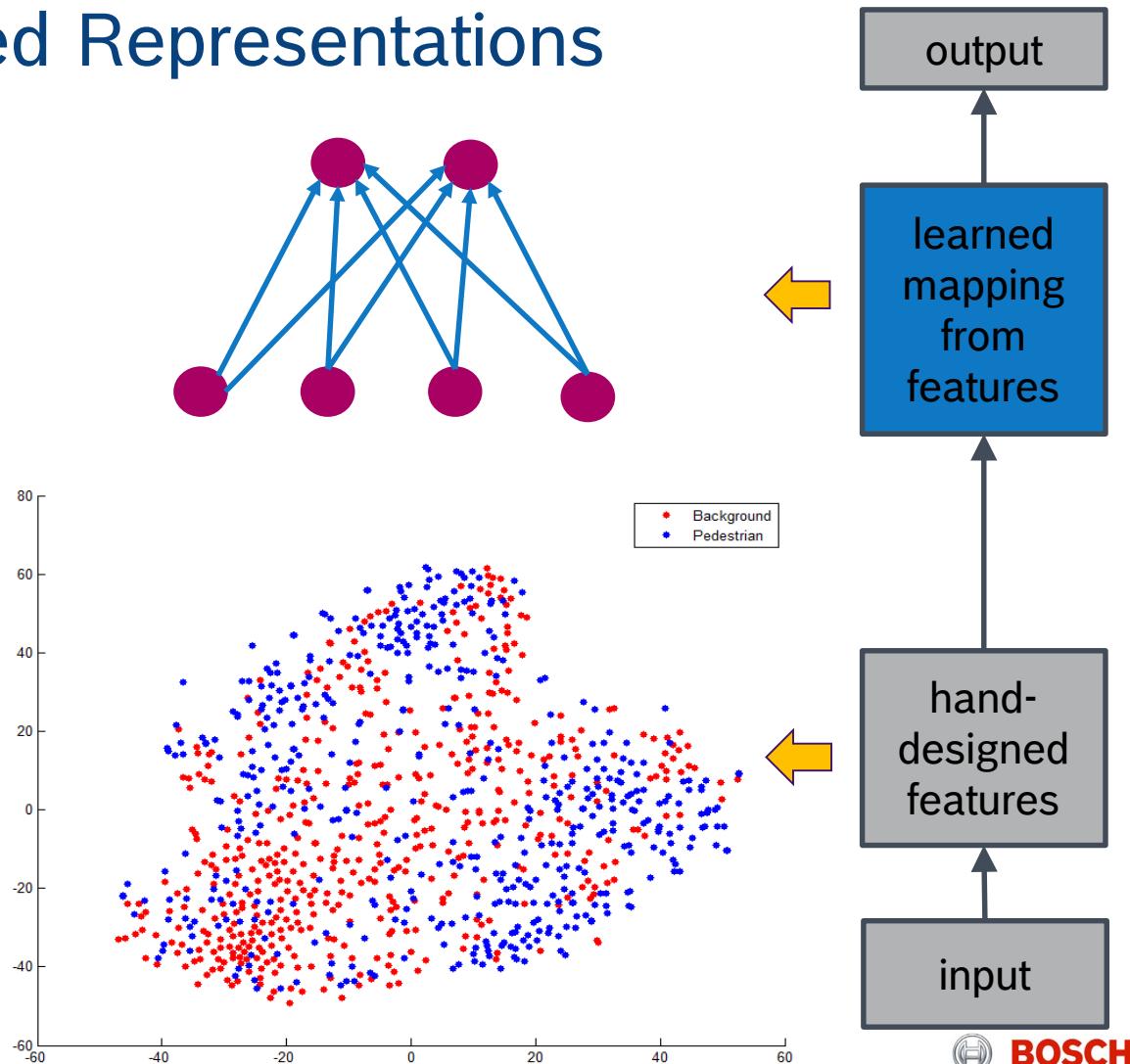
- Write a program to extract features!
- Examples:
 - Edges (Canny)
 - Histograms of Gradients (HoGs)
 - Haar-like features
 - Aggregated Channel Features (ACF)
- Pros:
 - better than rule-based systems
 - can inject human bias
- Cons:
 - still worse than human performance
 - large feature-output abstraction gap
 - hard to design good features
 - does not scale



The Path to Deep Learning

First Generation: Hand-Designed Representations

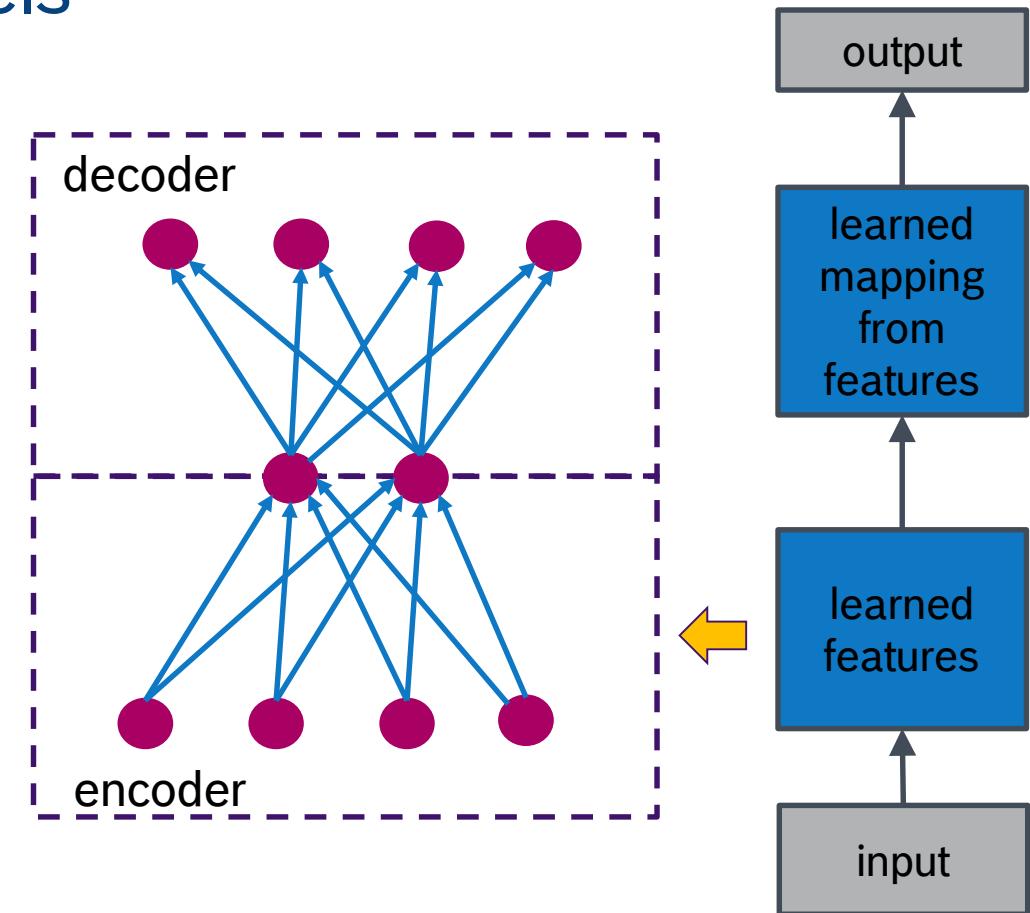
- Write a program to extract features!
- Examples:
 - Edges (Canny)
 - Histograms of Gradients (HoGs)
 - Haar-like features
 - Aggregated Channel Features (ACF)
- Pros:
 - better than rule-based systems
 - can inject human bias
- Cons:
 - still worse than human performance
 - large feature-output abstraction gap
 - hard to design good features
 - does not scale



The Path to Deep Learning

Second Generation: Shallow Models

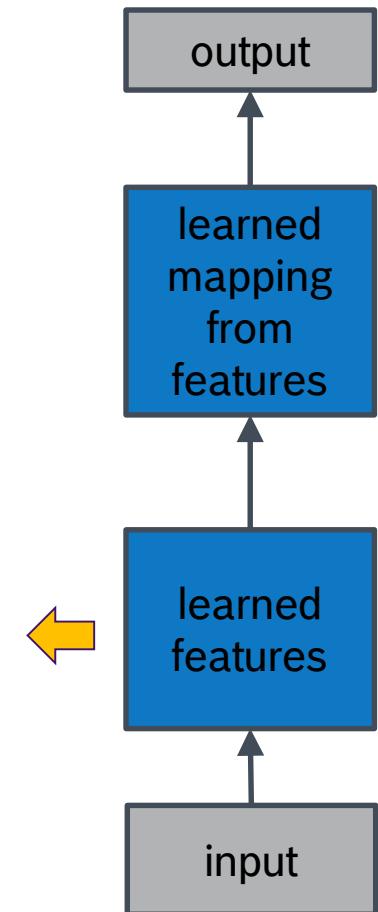
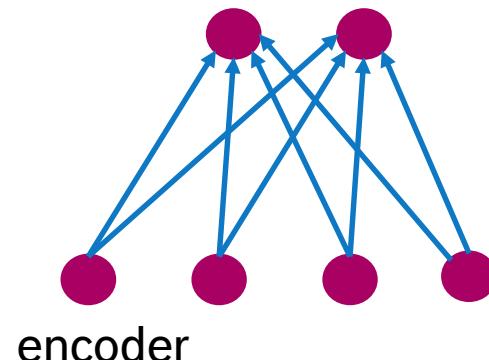
- Learn the representation too!
 - Separately: e.g. **auto-encoders**
 - Jointly with the mapping: **end-to-end learning**
- Better, but we still need to cover a large **abstraction gap**



The Path to Deep Learning

Second Generation: Shallow Models

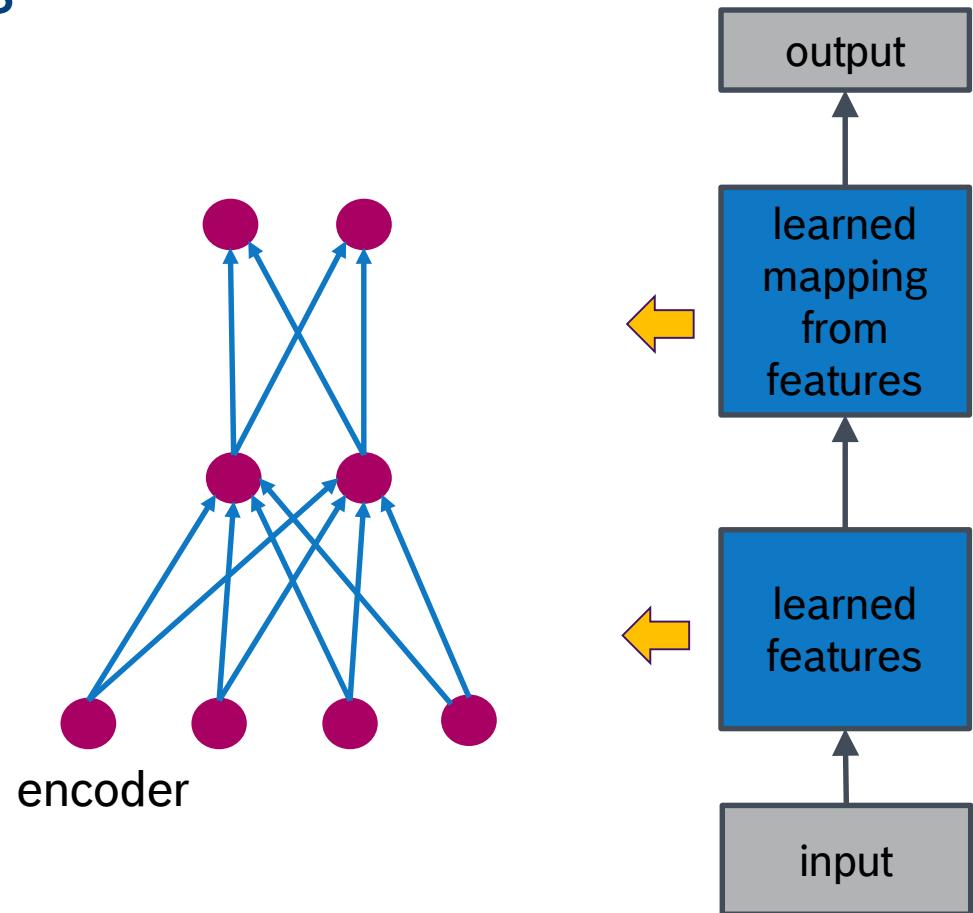
- Learn the representation too!
 - Separately: e.g. **auto-encoders**
 - Jointly with the mapping: **end-to-end learning**
- Better, but we still need to cover a large **abstraction gap**



The Path to Deep Learning

Second Generation: Shallow Models

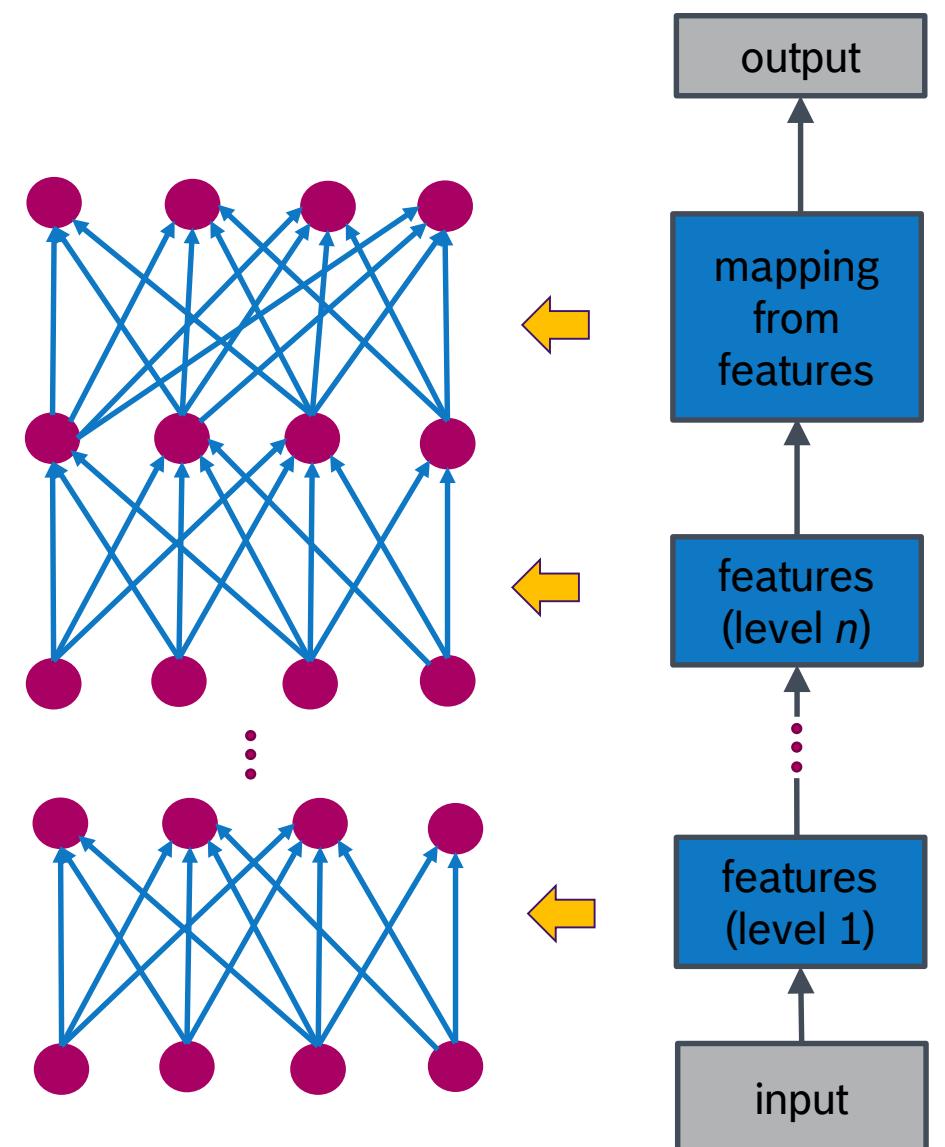
- Learn the representation too!
 - Separately: e.g. **auto-encoders**
 - Jointly with the mapping: **end-to-end learning**
- Better, but we still need to cover a large **abstraction gap**



The Path to Deep Learning

Third Generation: Deep Learning

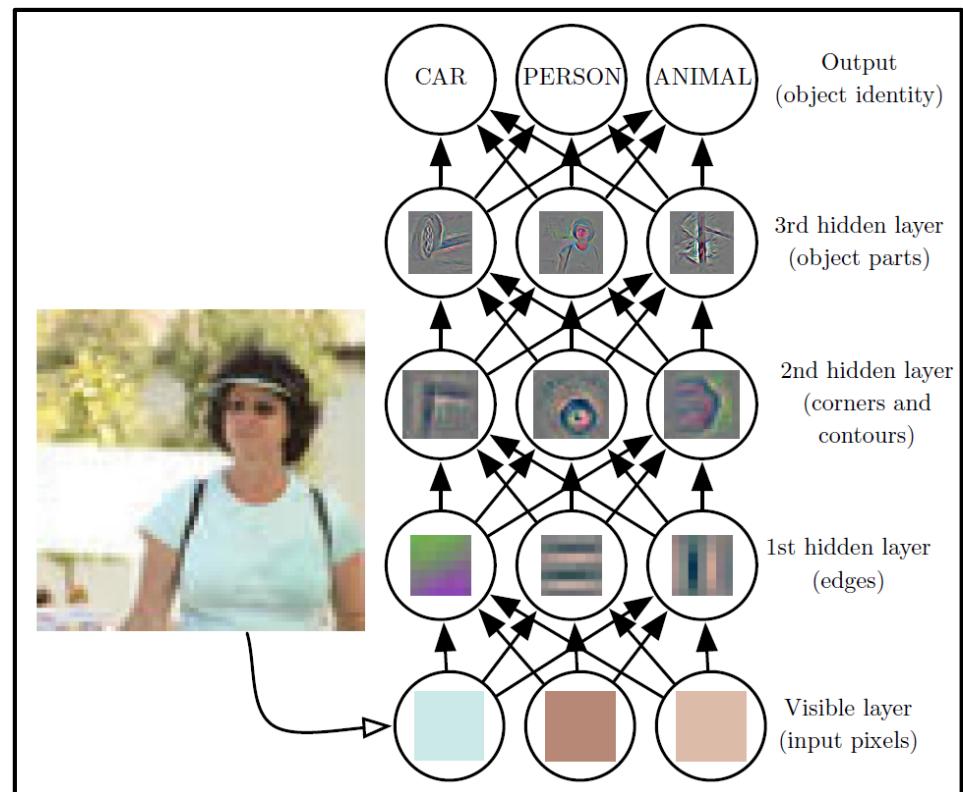
- Why not learn a hierarchy of features?
- This is **deep learning**
- Examples:
 - **deep auto-encoders**
 - **multi-layer-perceptron (MLP)**
 - **convolutional neural networks (CNNs)**



The Path to Deep Learning

Third Generation: Deep Learning

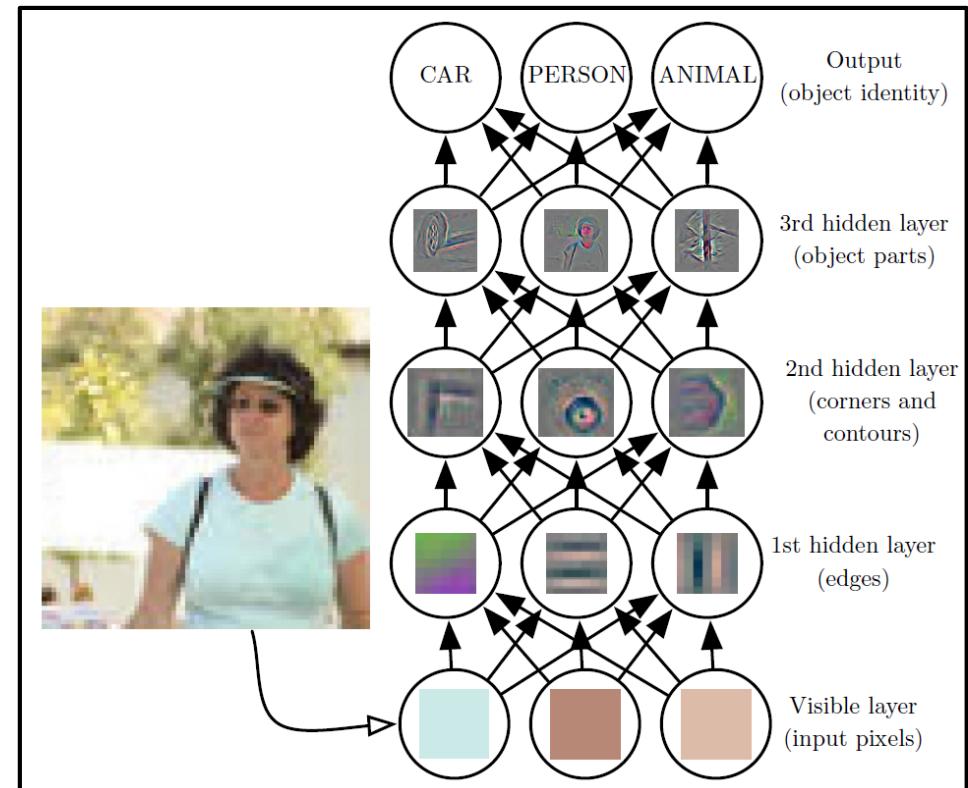
- Pros:
 - Near human performance on many hard problems
 - Fully scalable to any problem domain
- Cons:
 - Need large amounts of data
 - Expensive to train and run



The Path To Deep Learning

A Dual View of Deep Learning

- **Representational:** function composition
 - layer: a simple function (multi-dimensional)
 - layer output: a new representation of the input
- **Computational:** computer program
 - layer: set of parallel instructions
 - layer output: state of computer memory (may not be entirely input-related!)

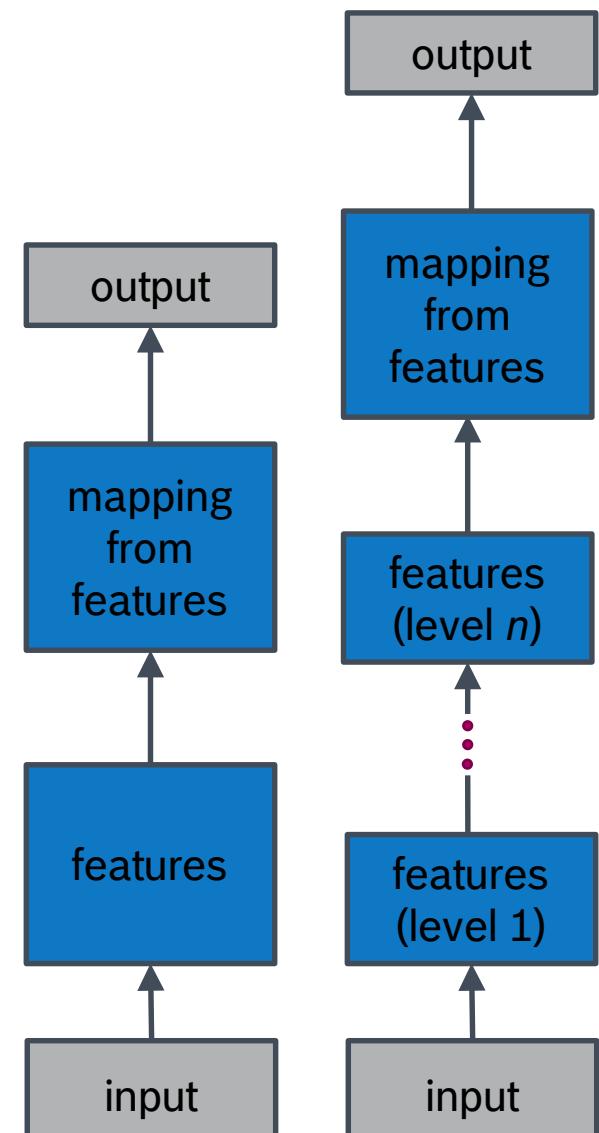


The Path To Deep Learning

Deep or not deep?

- No single definition of depth:
 - Maximum number of instructions to evaluate the entire output?
 - Maximum number of concepts to arrive at the output representation?
- No clear definition of deep:
 - Having at least 4 layers? (some authors)
 - In any case:

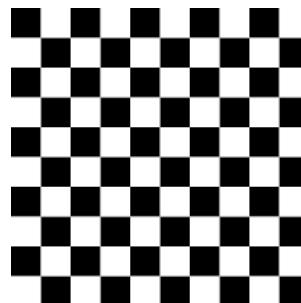
A **deep model** is a model with more learned instructions/concepts than a traditional machine learning model.



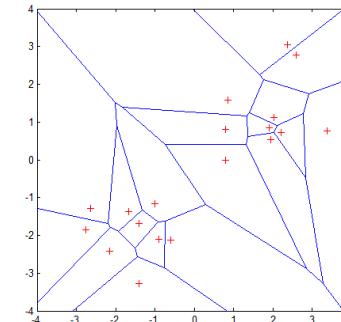
The Path to Deep Learning

Distributed Representations

- We need to **dis-entangle** factors of variation
- Example: recognize cars/trucks/birds of three colors (red, green, blue)
- Encodings:
 - Option 1: all combinations => 9 features
 - Option 2: **distributed representation** => 6 features (3 for object type, 3 for color)
- Deep models can learn distributed representations!



Entangled features



kNN in image space needs one example in each cell!

The Path To Deep Learning

Recap

- We need to address the **abstraction gap**
- Extremely Difficult:
 - Representations are not conspicuous!
 - Hard to dis-entangle factors of variation
- Hand designed features
 - Time consuming to design
 - Do not generalize across problems
 - Brittle to variations (illuminations, pose changes)
- Learned feature hierarchies (**end-to-end learning**)
 - Trade training data for design time
 - Generalize across problems
 - Increased variance
 - Distributed representations of the world
- **Deep learning** is just end-to-end learning with a lot of feature levels

REFERENCES

Deep Learning for Computer Vision

References

- [1] Goodfellow, I. and Bengio, Y. and Courville, A., “*Deep Learning*”, MIT Press, 2016, <http://www.deeplearningbook.org/>
- [2] Bishop, C. M. “Pattern Recognition and Machine Learning”, Springer, 2006
- [3] Murphy, K.P. “*Machine Learning: a Probabilistic Perspective*”, MIT Press, 2012
- [4] Fei-Fei, Karpathy and Johnson, *Lecture Notes*, 2016,
<http://cs231n.stanford.edu/slides/2017/>
- [5] Zhang, A. and Lipton, Z.C. and Li, M. and Smola, J.A., “*Dive into Deep Learning*”, 2017, <https://d2l.ai/>

CONVOLUTIONAL NEURAL NETWORKS

Szabolcs Pável
(Ștefan Máthé)

Outline

- ▶ Math Refresher: Calculus
- ▶ Optimization for Deep Learning
- ▶ Convolutional Neural Networks
- ▶ Perspectives

MATH REFRESHER: CALCULUS

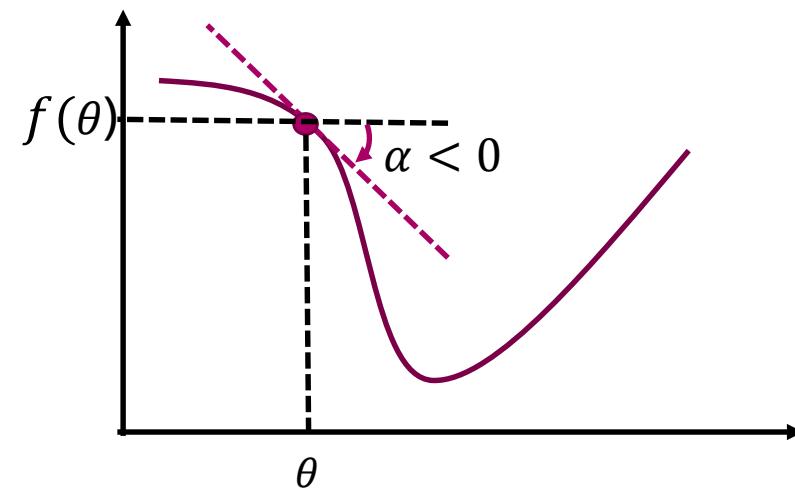
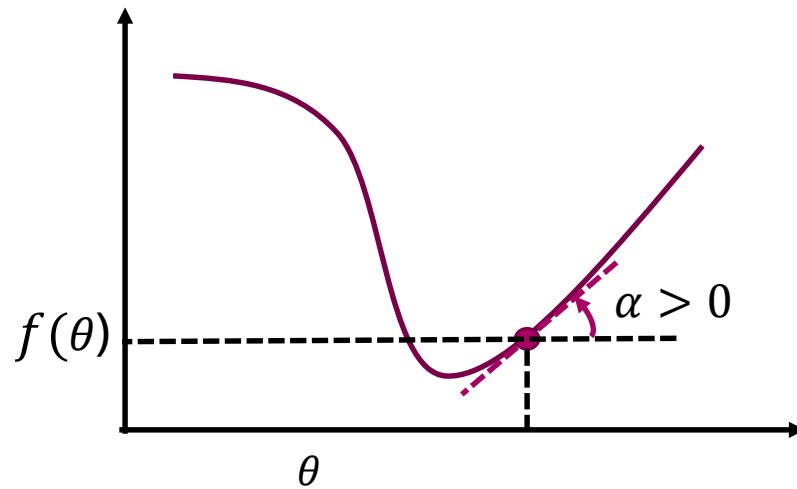
Quick Math Refresher

The Derivative

- Describes the sensitivity of a 1 dimensional scalar function to an infinitesimal change in input

$$f: \mathbb{R} \rightarrow \mathbb{R} \quad f'(\theta) = \lim_{\varepsilon \rightarrow 0} \frac{f(\theta + \varepsilon) - f(\theta)}{\varepsilon} = \tan \alpha$$

- The derivative is a scalar
- Interpretation: the tangent of the slope of the function



Quick Math Refresher

The Partial Derivative

- ▶ Assume we have a d -dimensional scalar function:

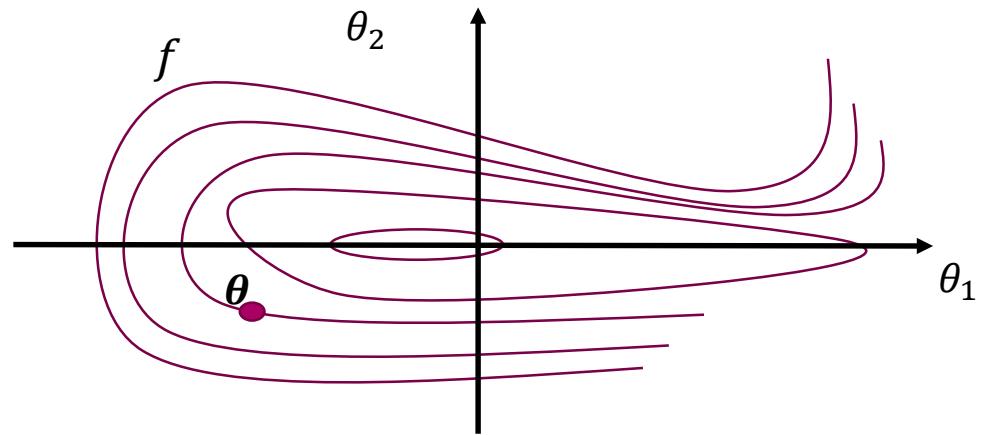
$$f: \mathbb{R}^d \rightarrow \mathbb{R}$$

- ▶ The partial derivative for dimension i describes how sensitive f is to infinitesimal changes along dimension i

$$\frac{\partial f}{\partial \theta_i}(\theta) = \lim_{\varepsilon \rightarrow 0} \frac{f(\theta + \varepsilon \Delta_i) - f(\theta)}{\varepsilon}$$

where: $\Delta_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

i-th dimension



Quick Math Refresher

The Partial Derivative

- ▶ Assume we have a d -dimensional scalar function:

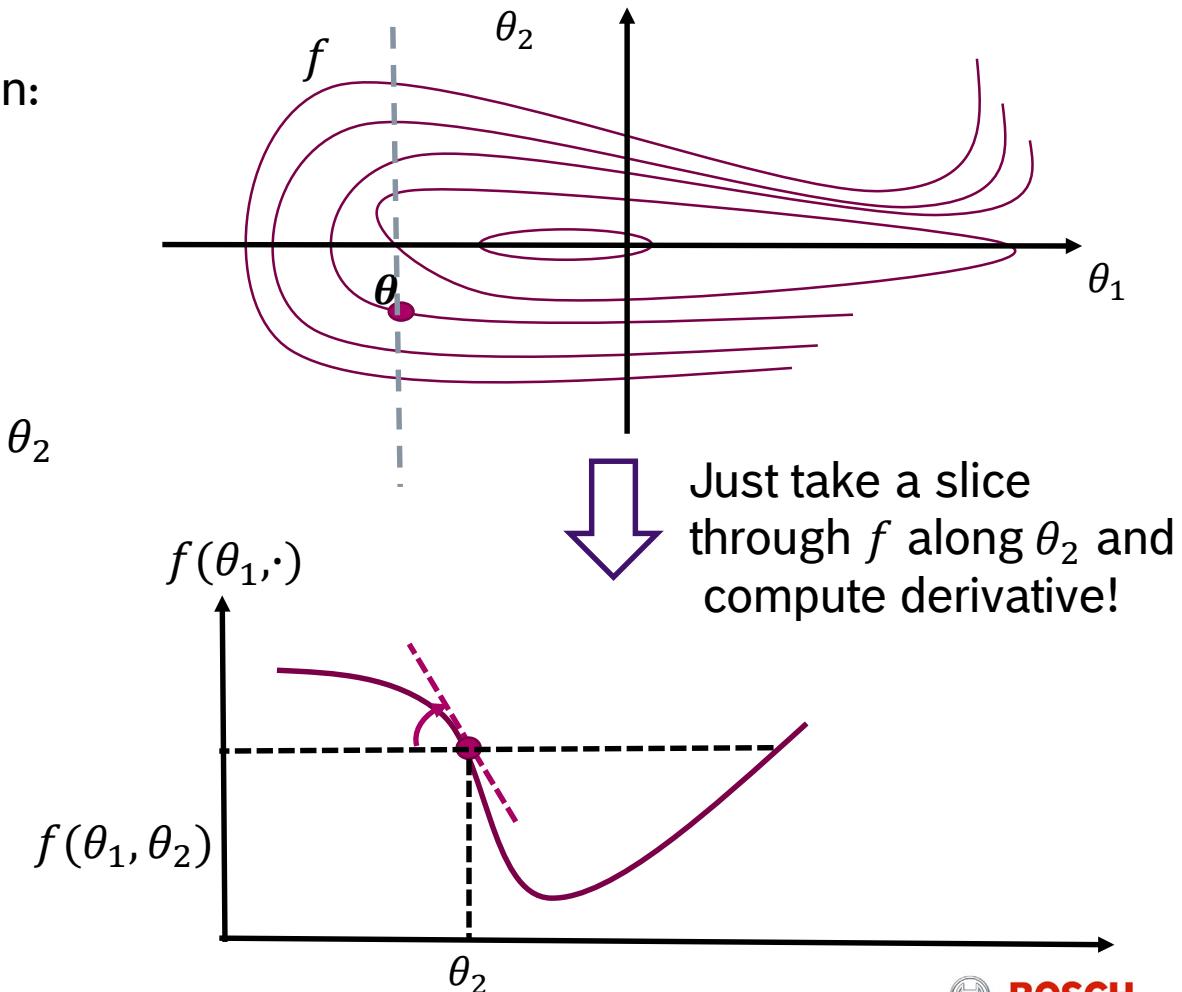
$$f: \mathbb{R}^d \rightarrow \mathbb{R}$$

- ▶ The partial derivative for dimension i describes how sensitive f is to infinitesimal changes along dimension i

$$\frac{\partial f}{\partial \theta_i}(\theta) = \lim_{\varepsilon \rightarrow 0} \frac{f(\theta + \varepsilon \Delta_i) - f(\theta)}{\varepsilon}$$

where: $\Delta_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

i-th dimension



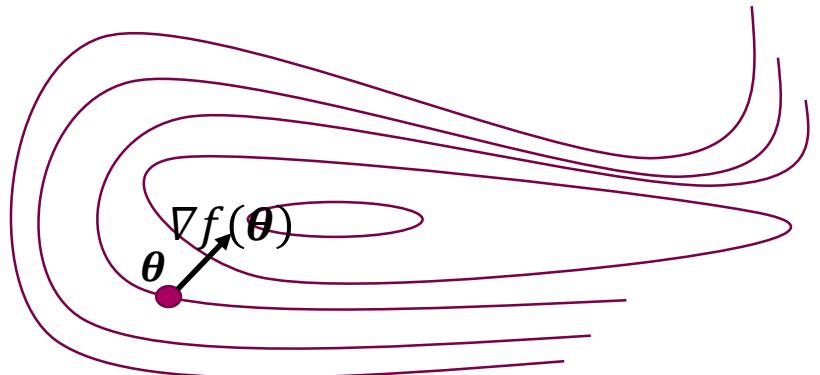
Quick Math Refresher

The Gradient

- ▶ Generalizes the gradient to a d-dimensional scalar function
- ▶ Just stack together the partial derivatives along each axis
- ▶ The gradient is a vector
- ▶ Interpretation: the direction of steepest ascent

$$f: \mathbb{R}^d \rightarrow \mathbb{R}$$

$$\nabla f(\theta) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \vdots \\ \frac{\partial f}{\partial \theta_d} \end{bmatrix}$$



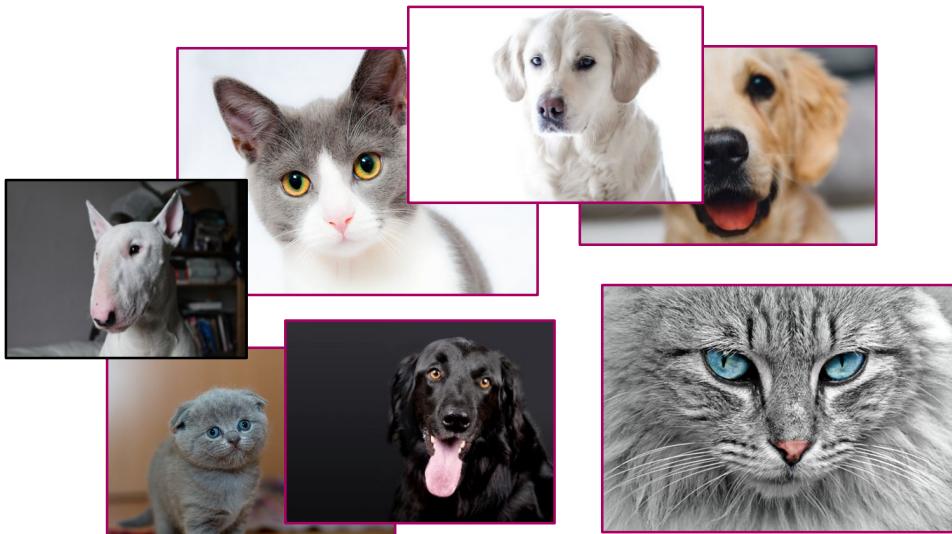
OPTIMIZATION

Optimization

Problem Formulation

- Almost all machine learning problems are a form of (constrained / approximate) minimization of an **objective function** (e.g. training negative log likelihood)

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

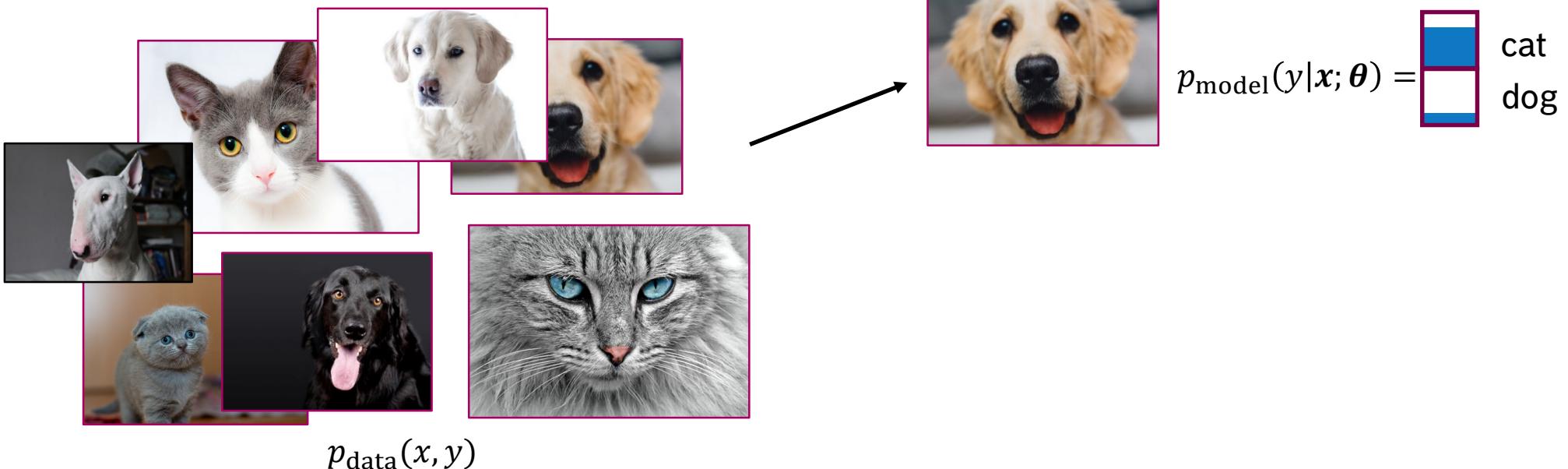


Optimization

Problem Formulation

- Almost all machine learning problems are a form of (constrained / approximate) minimization of an **objective function** (e.g. training negative log likelihood)

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

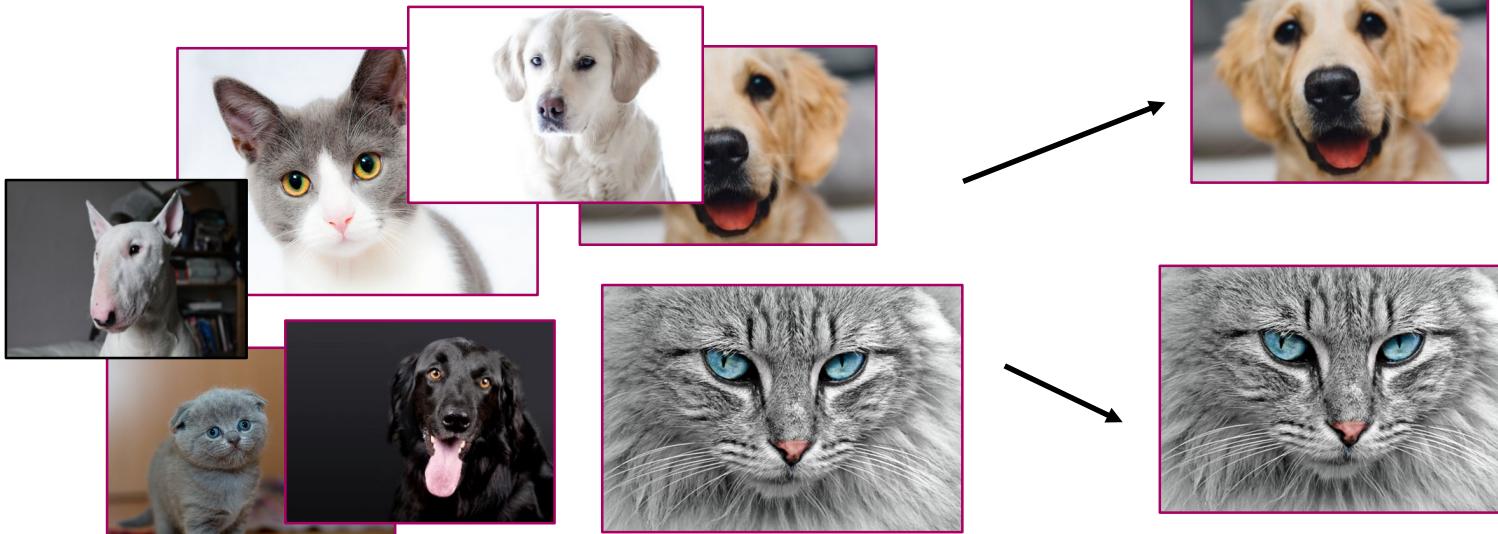


Optimization

Problem Formulation

- Almost all machine learning problems are a form of (constrained / approximate) minimization of an **objective function** (e.g. training negative log likelihood)

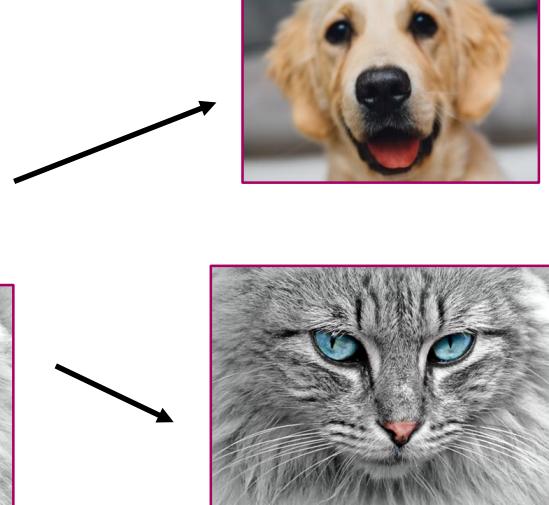
$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



$$p_{\text{data}}(x, y)$$

$$p_{\text{model}}(y|x; \boldsymbol{\theta}) = \begin{bmatrix} \text{blue} \\ \text{white} \end{bmatrix}$$

cat
dog



$$p_{\text{model}}(y|x; \boldsymbol{\theta}) = \begin{bmatrix} \text{white} \\ \text{blue} \end{bmatrix}$$

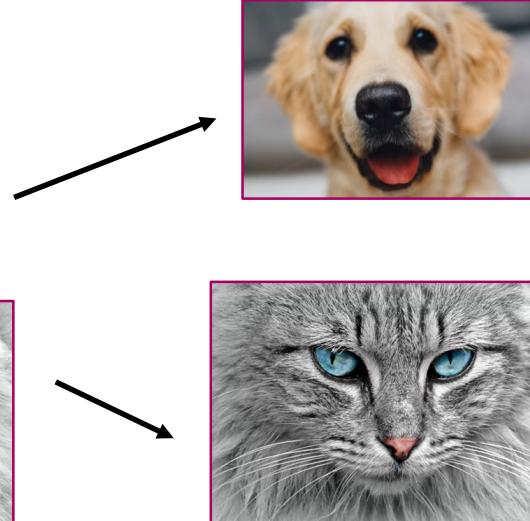
cat
dog

Optimization

Problem Formulation

- Almost all machine learning problems are a form of (constrained / approximate) minimization of an **objective function** (e.g. training negative log likelihood)

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



$$p_{\text{model}}(y|x; \boldsymbol{\theta}) = \begin{bmatrix} \text{cat} \\ \text{dog} \end{bmatrix}$$

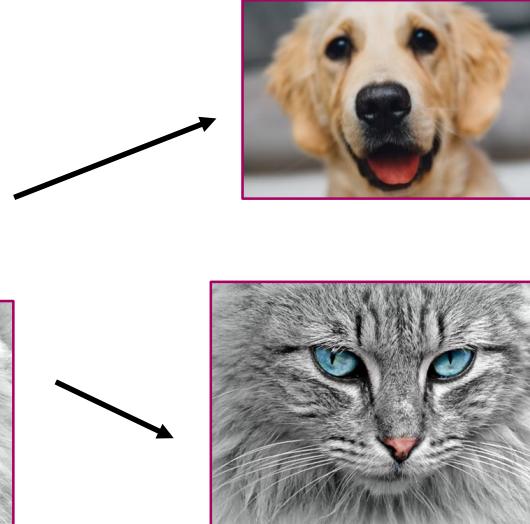
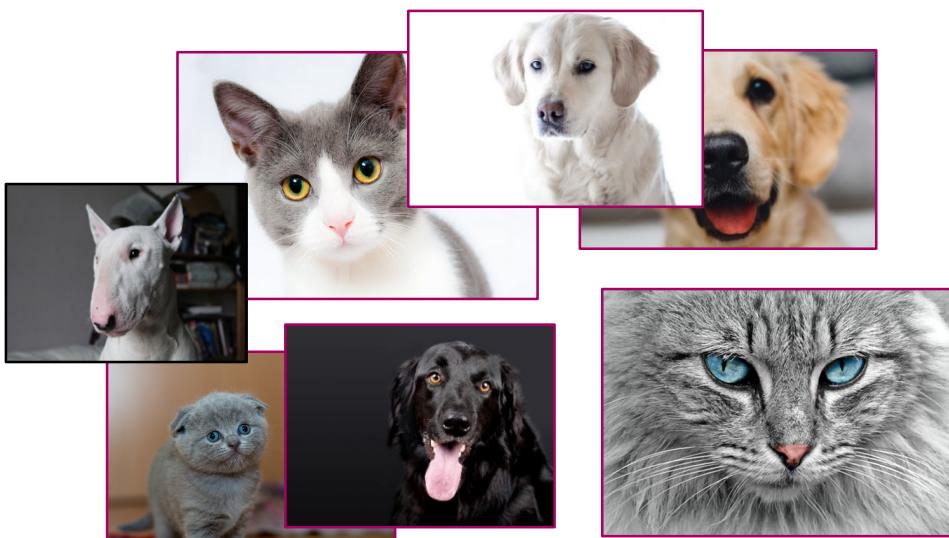
$$p_{\text{model}}(y|x; \boldsymbol{\theta}) = \begin{bmatrix} \text{cat} \\ \text{dog} \end{bmatrix}$$

Optimization

Problem Formulation

- Almost all machine learning problems are a form of (constrained / approximate) minimization of an **objective function** (e.g. training negative log likelihood)

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



$$p_{\text{model}}(y|x; \boldsymbol{\theta}) = \begin{bmatrix} \text{cat} \\ \text{dog} \end{bmatrix}$$

$$p_{\text{model}}(y|x; \boldsymbol{\theta}) = \begin{bmatrix} \text{cat} \\ \text{dog} \end{bmatrix}$$

Optimization

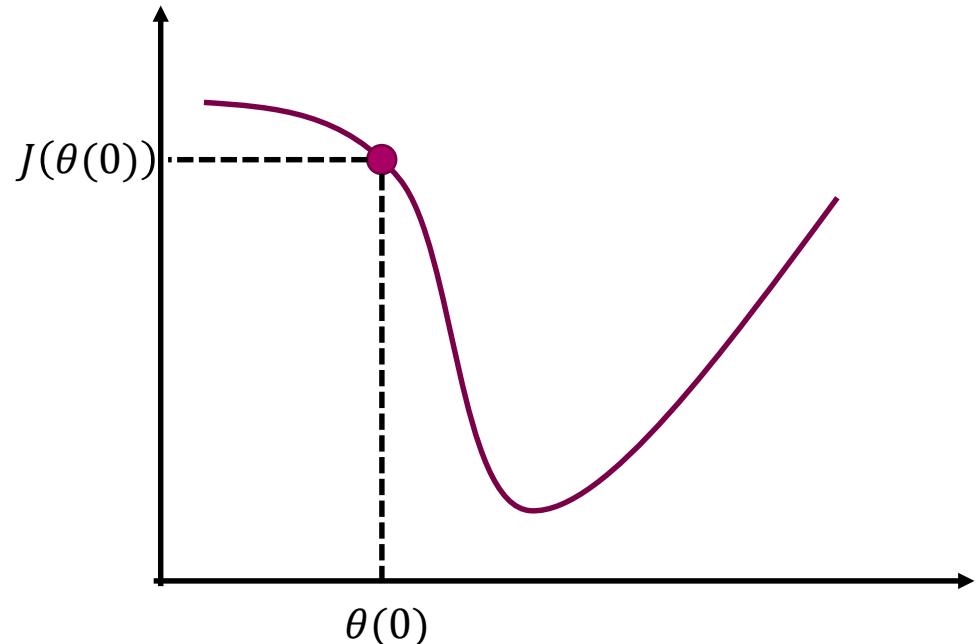
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

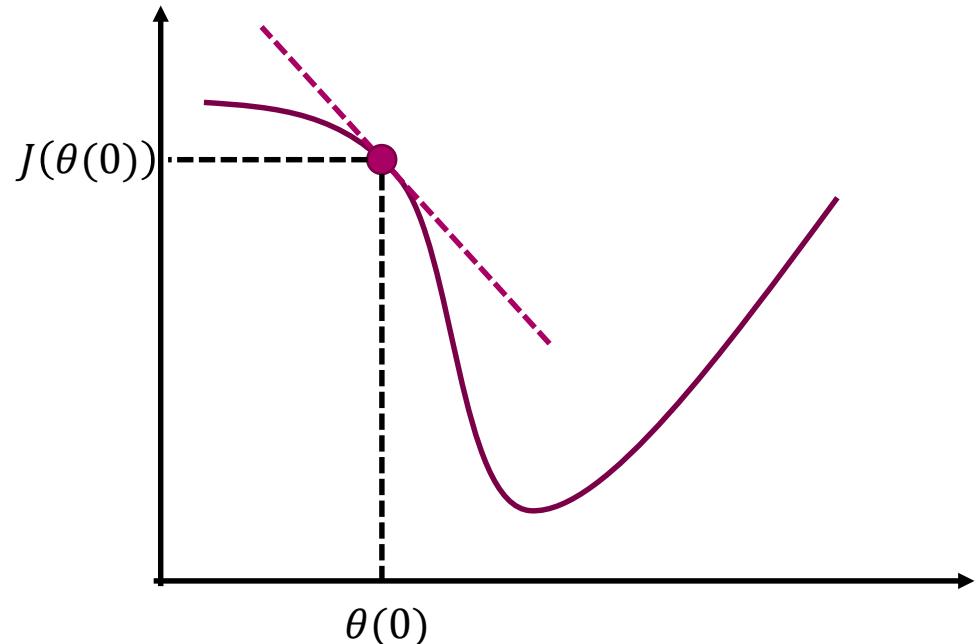
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

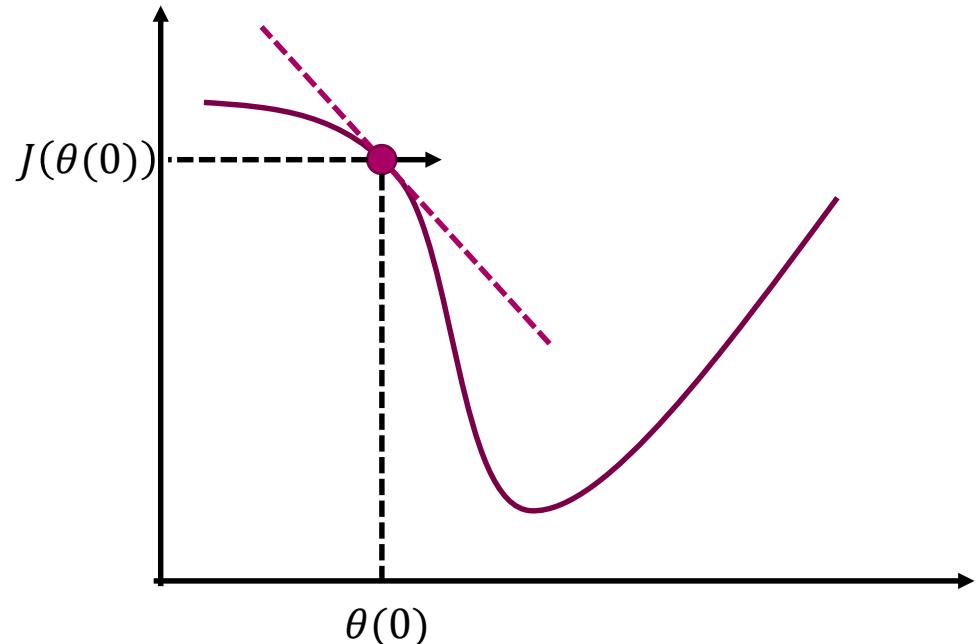
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

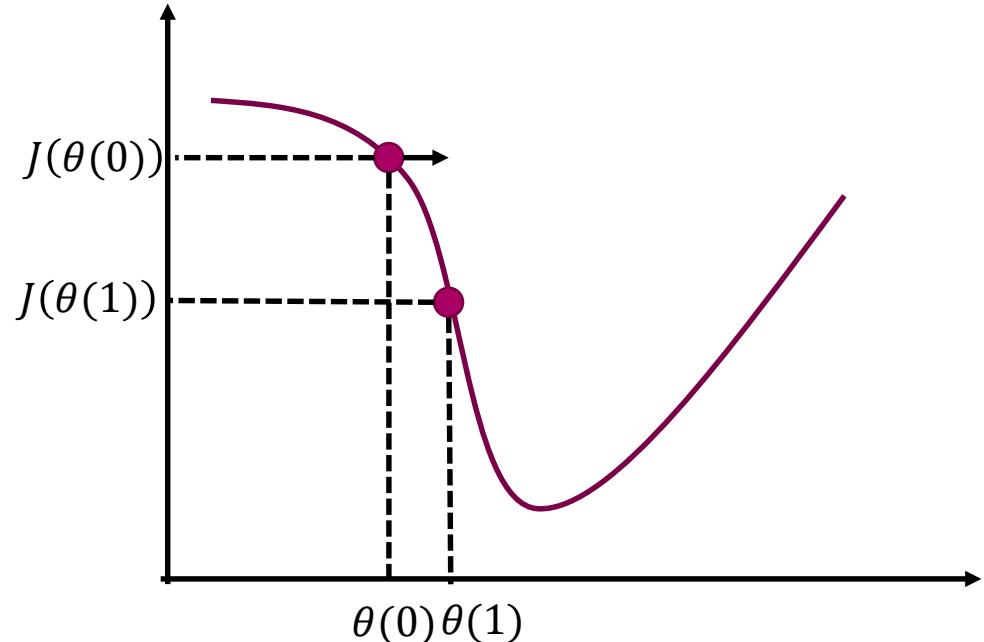
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

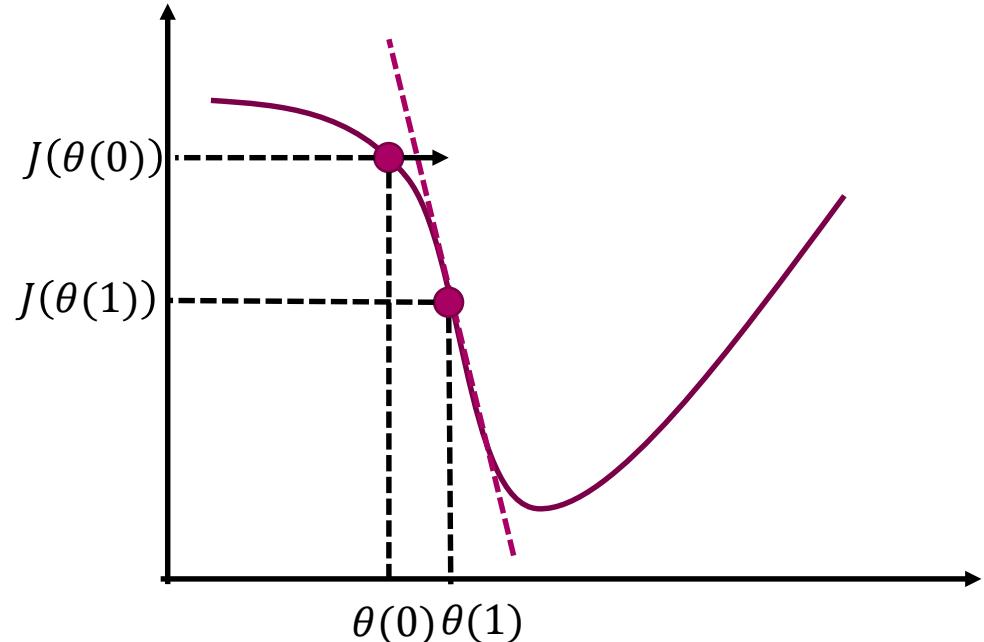
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

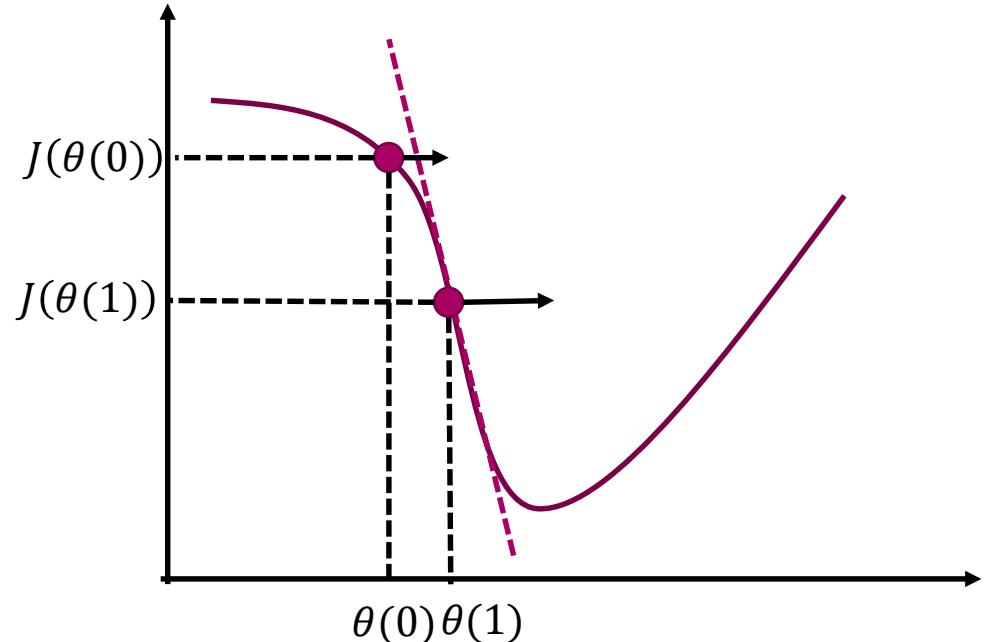
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

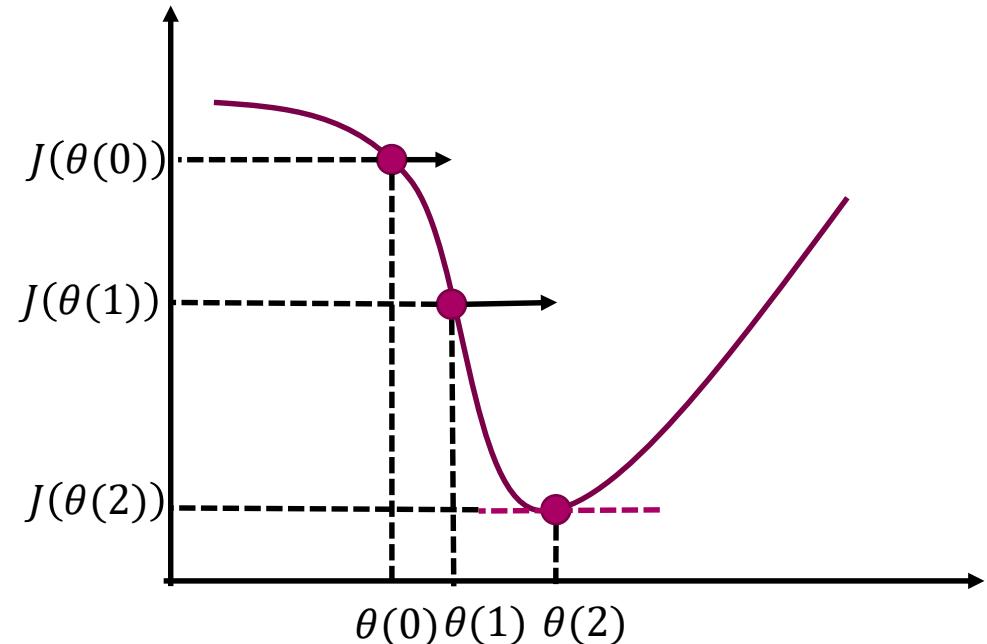
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

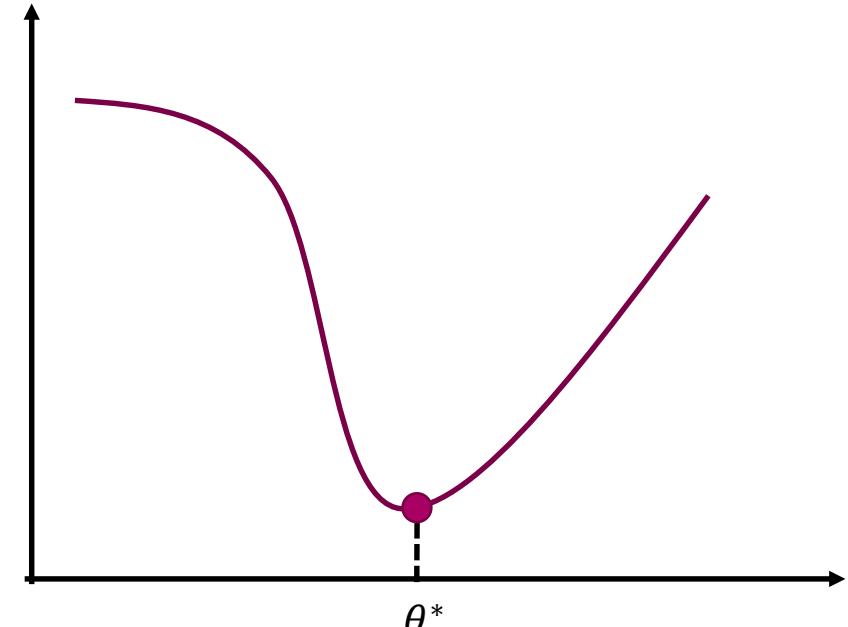
Gradient Descent Algorithm (1D Case)

```
function gradient-descent-1d
    input: a function  $J$ 
    output: a local optimum  $\theta^*$  of  $J$ 

    // Start from a random initial guess.
     $\theta(0) \leftarrow \text{rand}(\mathbb{R});$ 

    // Take steps in the direction of negative slope until
    // convergence.
     $t \leftarrow 0;$ 
    do {
         $\theta(t + 1) \leftarrow \theta(t) - \eta J'(\theta(t));$ 
         $t \leftarrow t + 1;$ 
    } while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

    return  $\theta(t);$ 
```



Optimization

Gradient Descent Algorithm (n-D case)

- ▶ Use a 1st order Taylor expansion of J :

$$J(\theta) \cong J(\theta_0) + (\theta - \theta_0)^T \nabla J(\theta_0)$$

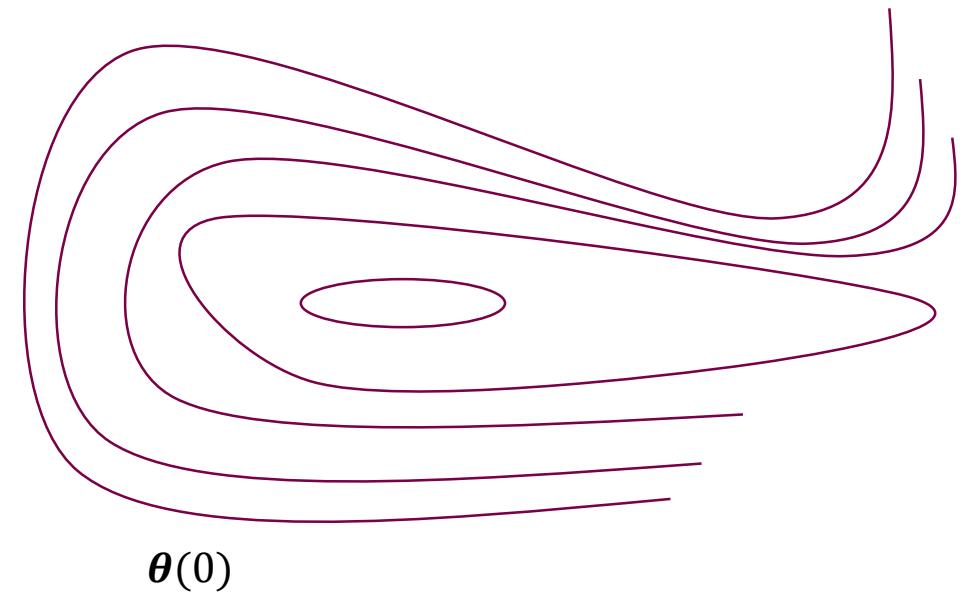
- ▶ Move in the direction with negative slope

```
function gradient-descent
input: a function  $J$ 
output: a local optimum  $\theta^*$  of  $J$ 

// Start from a random initial guess.
 $\theta(0) \leftarrow \text{rand}(\mathbb{R}^p);$ 

// Take steps in the direction of negative slope until
// convergence.
 $t \leftarrow 0;$ 
do {
     $\theta(t+1) \leftarrow \theta(t) - \eta \nabla J(\theta(t));$ 
     $t \leftarrow t + 1;$ 
} while ( $|\theta(t) - \theta(t-1)| > \varepsilon;$ 

return  $\theta(t);$ 
```



Optimization

Gradient Descent Algorithm (n-D case)

- ▶ Use a 1st order Taylor expansion of J :

$$J(\theta) \cong J(\theta_0) + (\theta - \theta_0)^T \nabla J(\theta_0)$$

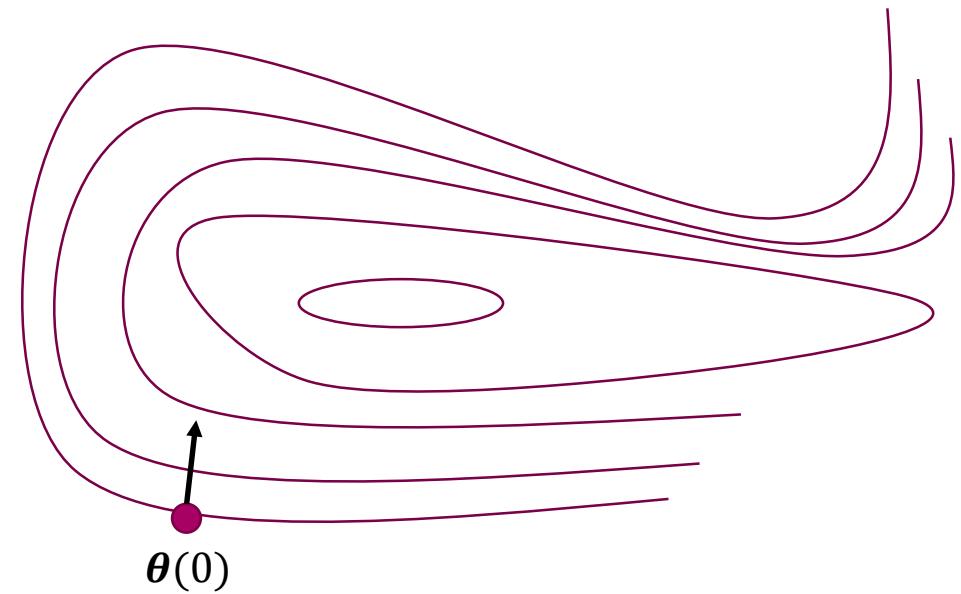
- ▶ Move in the direction with negative slope

```
function gradient-descent
input: a function  $J$ 
output: a local optimum  $\theta^*$  of  $J$ 

// Start from a random initial guess.
 $\theta(0) \leftarrow \text{rand}(\mathbb{R}^p);$ 

// Take steps in the direction of negative slope until
// convergence.
 $t \leftarrow 0;$ 
do {
     $\theta(t+1) \leftarrow \theta(t) - \eta \nabla J(\theta(t));$ 
     $t \leftarrow t + 1;$ 
} while ( $|\theta(t) - \theta(t-1)| > \varepsilon;$ 

return  $\theta(t);$ 
```



Optimization

Gradient Descent Algorithm (n-D case)

- ▶ Use a 1st order Taylor expansion of J :

$$J(\theta) \cong J(\theta_0) + (\theta - \theta_0)^T \nabla J(\theta_0)$$

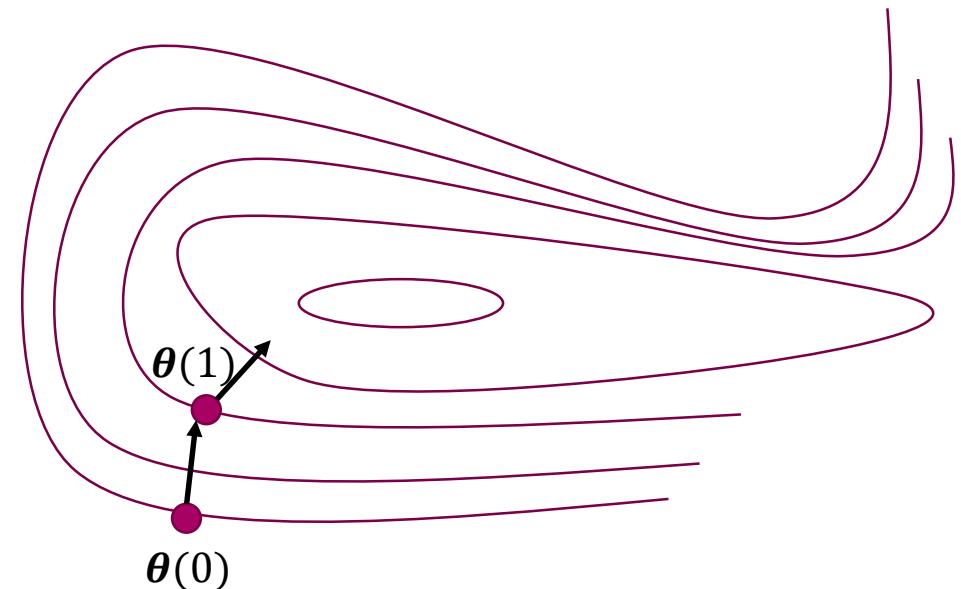
- ▶ Move in the direction with negative slope

```
function gradient-descent
input: a function  $J$ 
output: a local optimum  $\theta^*$  of  $J$ 

// Start from a random initial guess.
 $\theta(0) \leftarrow \text{rand}(\mathbb{R}^p);$ 

// Take steps in the direction of negative slope until
// convergence.
 $t \leftarrow 0;$ 
do {
     $\theta(t + 1) \leftarrow \theta(t) - \eta \nabla J(\theta(t));$ 
     $t \leftarrow t + 1;$ 
} while ( $|\theta(t) - \theta(t - 1)| > \varepsilon;$ 

return  $\theta(t);$ 
```



Optimization

Gradient Descent Algorithm (n-D case)

- ▶ Use a 1st order Taylor expansion of J :

$$J(\theta) \cong J(\theta_0) + (\theta - \theta_0)^T \nabla J(\theta_0)$$

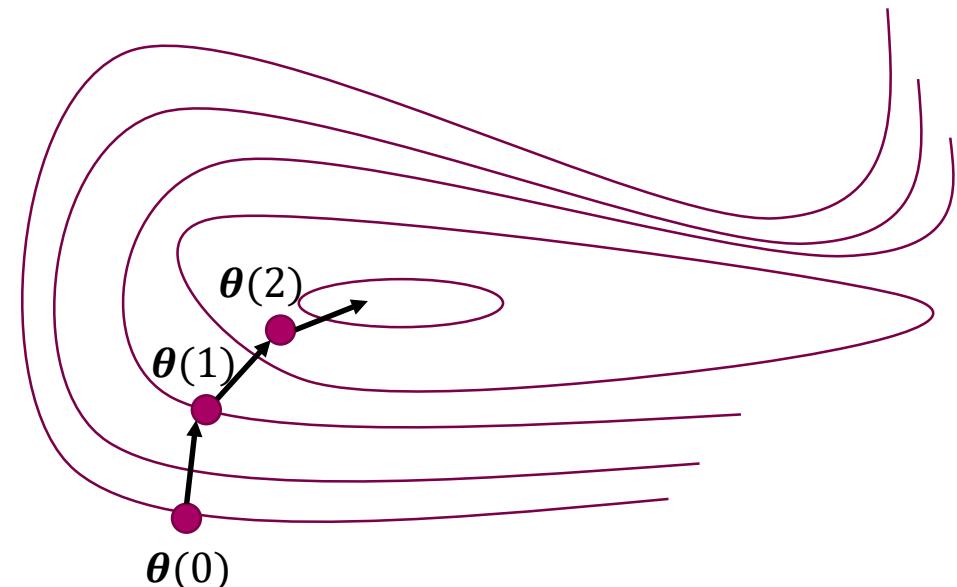
- ▶ Move in the direction with negative slope

```
function gradient-descent
input: a function  $J$ 
output: a local optimum  $\theta^*$  of  $J$ 

// Start from a random initial guess.
 $\theta(0) \leftarrow \text{rand}(\mathbb{R}^p);$ 

// Take steps in the direction of negative slope until
// convergence.
 $t \leftarrow 0;$ 
do {
     $\theta(t+1) \leftarrow \theta(t) - \eta \nabla J(\theta(t));$ 
     $t \leftarrow t + 1;$ 
} while ( $|\theta(t) - \theta(t-1)| > \varepsilon;$ 

return  $\theta(t);$ 
```



Optimization

Gradient Descent Algorithm (n-D case)

- ▶ Use a 1st order Taylor expansion of J :

$$J(\theta) \cong J(\theta_0) + (\theta - \theta_0)^T \nabla J(\theta_0)$$

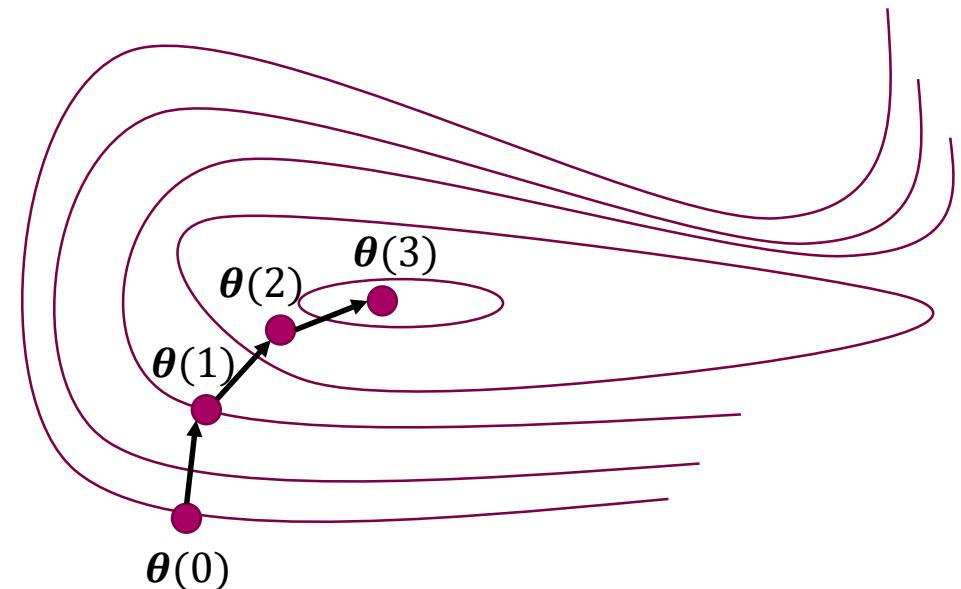
- ▶ Move in the direction with negative slope

```
function gradient-descent
input: a function  $J$ 
output: a local optimum  $\theta^*$  of  $J$ 

// Start from a random initial guess.
 $\theta(0) \leftarrow \text{rand}(\mathbb{R}^p);$ 

// Take steps in the direction of negative slope until
// convergence.
 $t \leftarrow 0;$ 
do {
     $\theta(t+1) \leftarrow \theta(t) - \eta \nabla J(\theta(t));$ 
     $t \leftarrow t + 1;$ 
} while ( $|\theta(t) - \theta(t-1)| > \varepsilon;$ 

return  $\theta(t);$ 
```



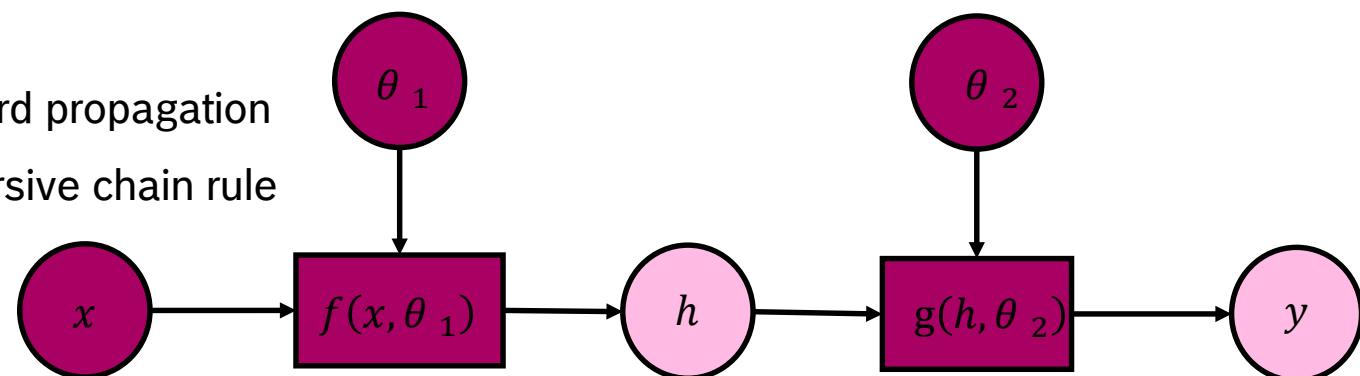
Optimization

Computing the Gradient: The Backpropagation Algorithm

- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis

- ▶ **Steps:**

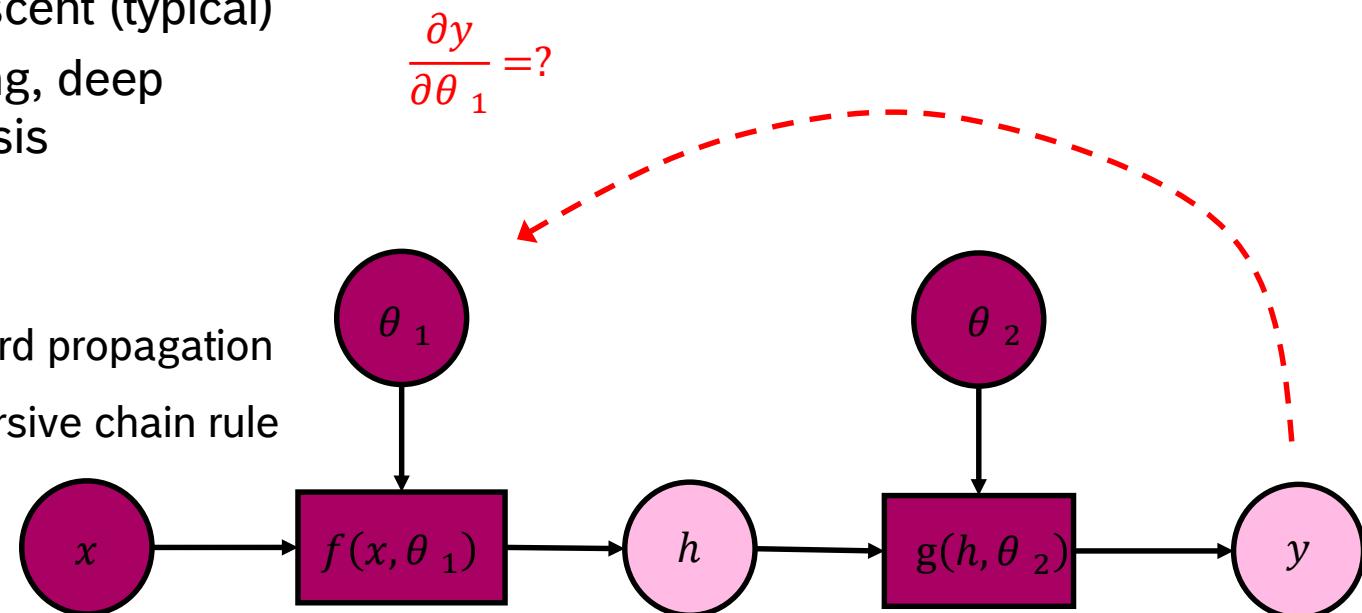
- ▶ Step 1: Compute y by forward propagation
- ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

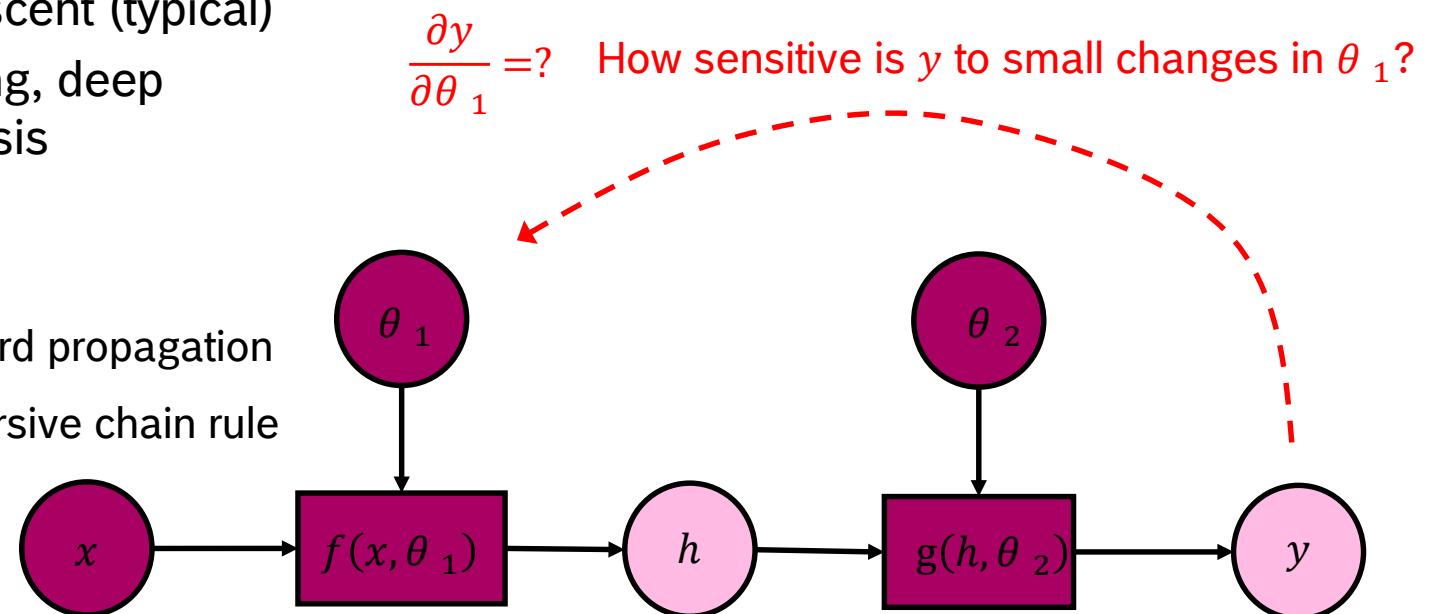
- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis
- ▶ **Steps:**
 - ▶ Step 1: Compute y by forward propagation
 - ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

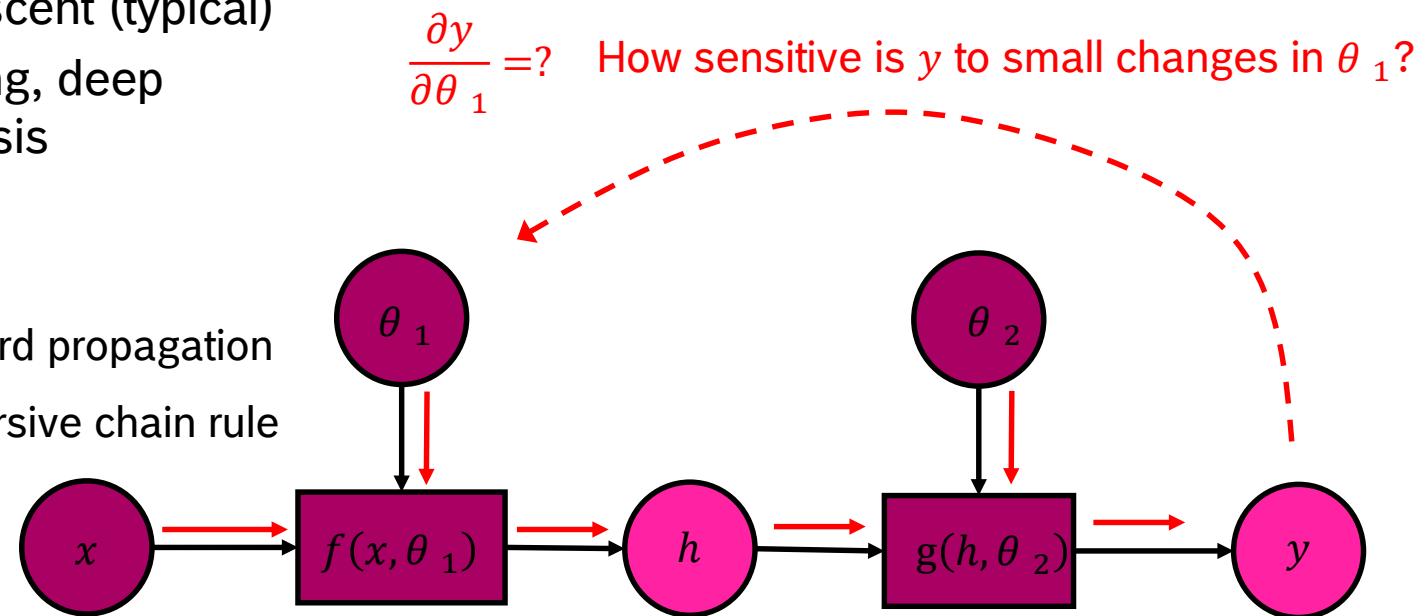
- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis
- ▶ **Steps:**
 - ▶ Step 1: Compute y by forward propagation
 - ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

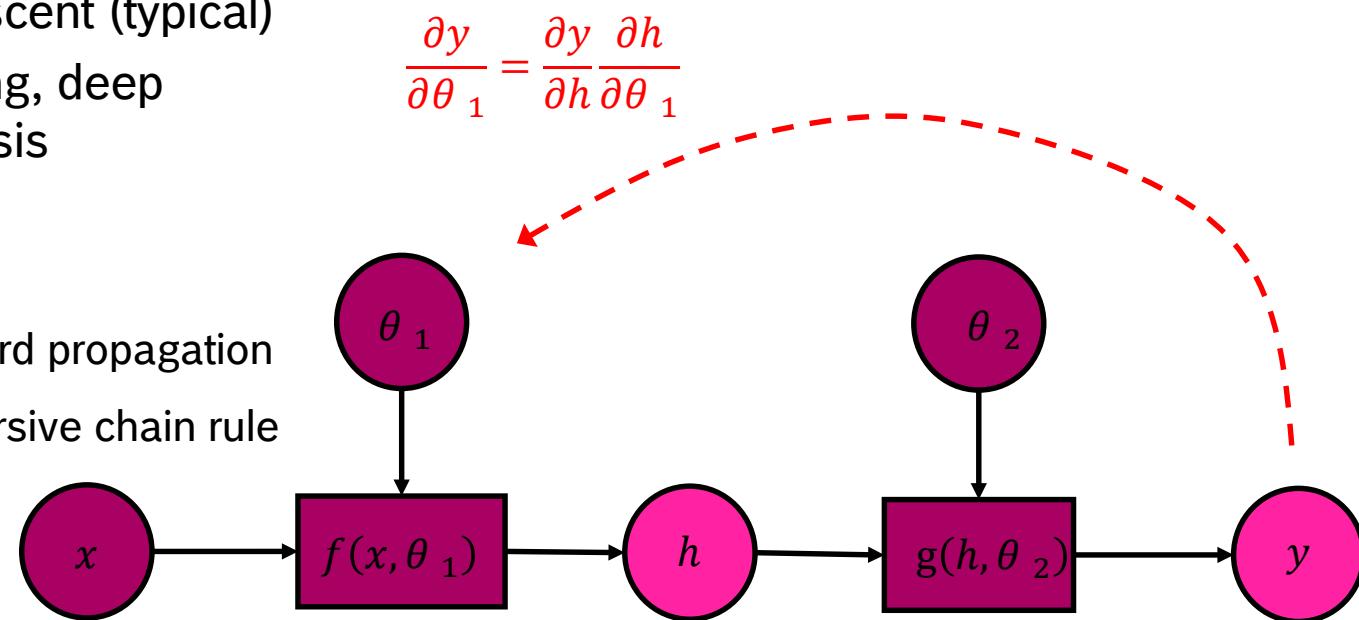
- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis
- ▶ **Steps:**
 - ▶ Step 1: Compute y by forward propagation
 - ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis
- ▶ **Steps:**
 - ▶ Step 1: Compute y by forward propagation
 - ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

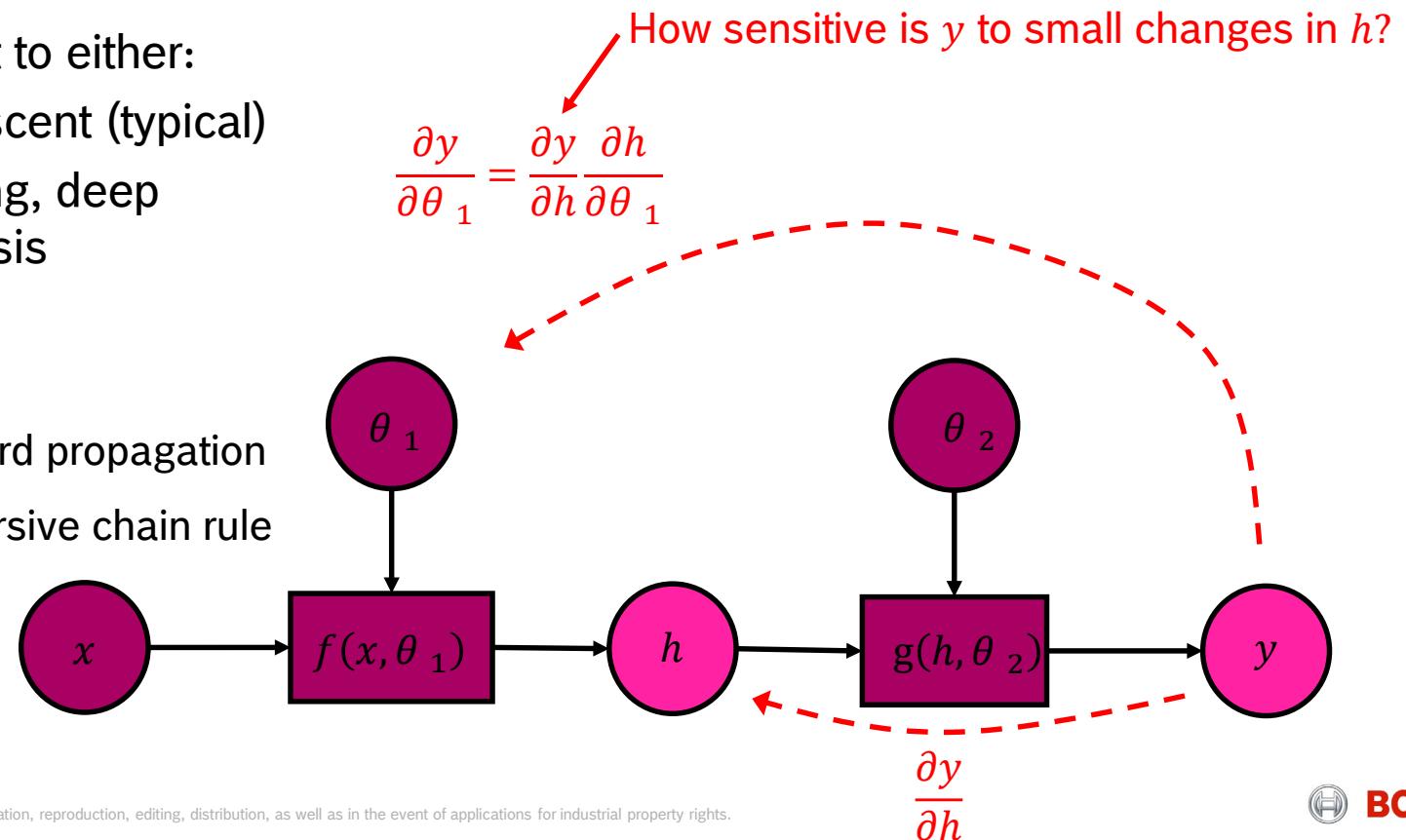
- ▶ **Input:** function composition

- ▶ **Output:** gradient with respect to either:

- ▶ parameter – gradient descent (typical)
- ▶ input – adversarial training, deep dreaming, network analysis

- ▶ **Steps:**

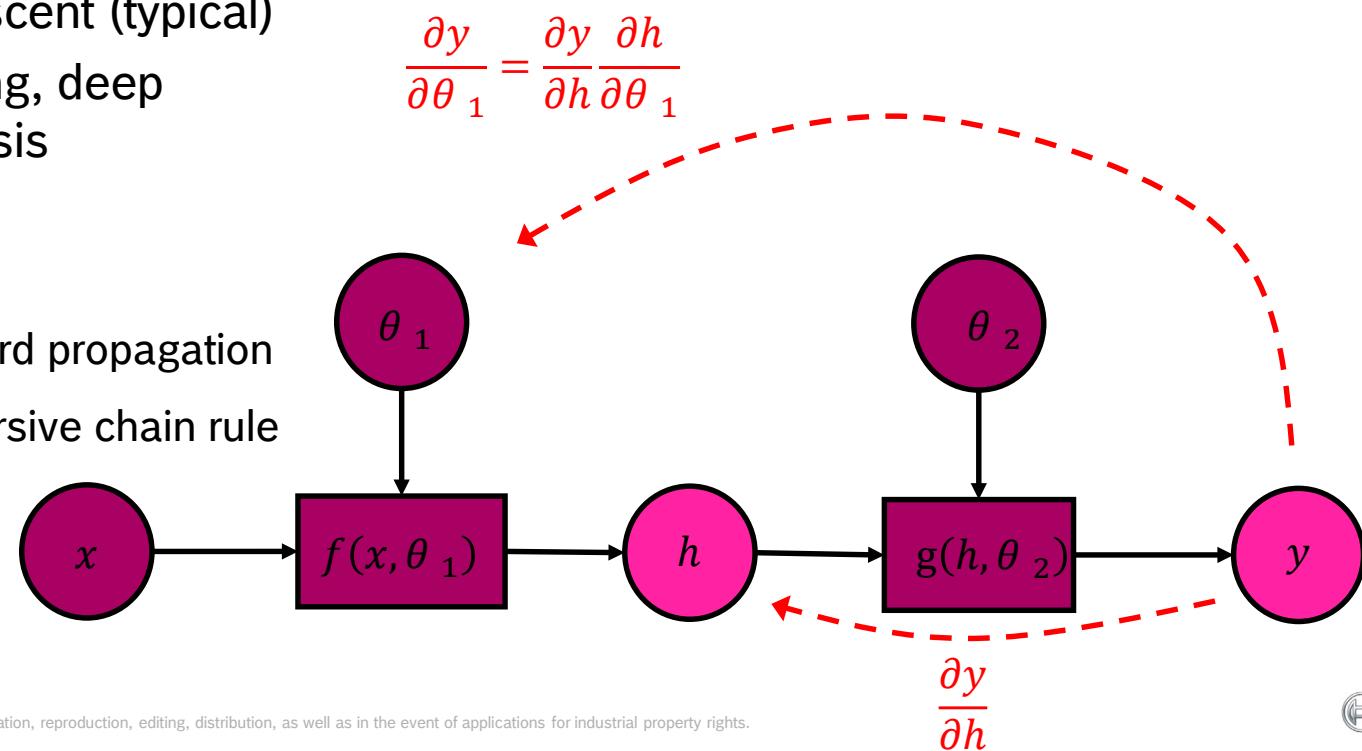
- ▶ Step 1: Compute y by forward propagation
- ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

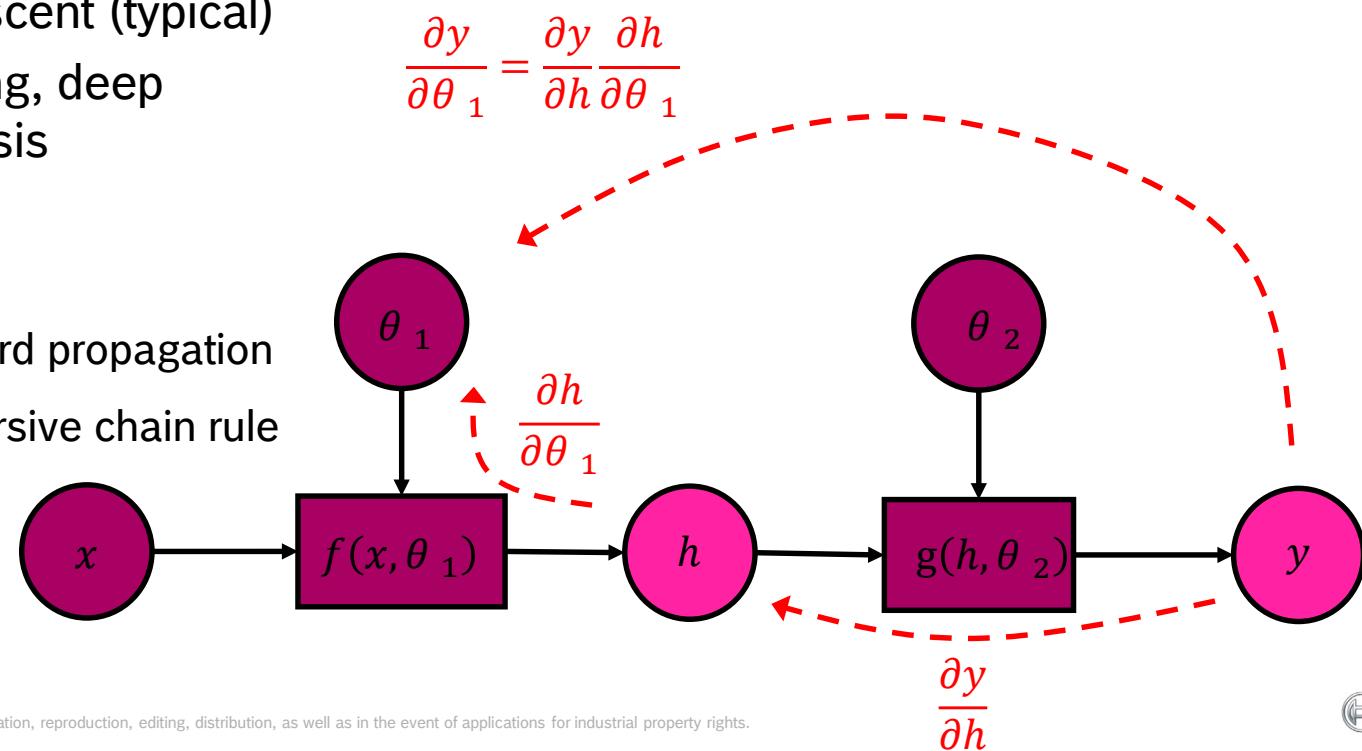
- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis
- ▶ **Steps:**
 - ▶ Step 1: Compute y by forward propagation
 - ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Computing the Gradient: The Backpropagation Algorithm

- ▶ **Input:** function composition
- ▶ **Output:** gradient with respect to either:
 - ▶ parameter – gradient descent (typical)
 - ▶ input – adversarial training, deep dreaming, network analysis
- ▶ **Steps:**
 - ▶ Step 1: Compute y by forward propagation
 - ▶ Step 2: Compute $\frac{\partial y}{\partial \theta}$ by recursive chain rule applications



Optimization

Backpropagation: Extensions

► Variants:

- ▶ **symbol-to-number**: nodes compute gradient numerically (Torch, Caffe, Pytorch)
- ▶ **symbol-to-symbol**: augment the graph with nodes that compute the gradient (Theano, Tensorflow) – can simplify symbolic expressions!

► Efficiency:

- ▶ Not optimal (special case of the **reverse mode differentiation algorithm**)
- ▶ Finding the optimal formula is NP-complete [Naumann, 2008]

► Implementation:

- ▶ **automatic differentiation** (transparent gradient computation)
- ▶ new way of writing programs:

“Deep Learning est mort. Vive Differentiable Programming!” [Yann LeCun]

Optimization

Stochastic Gradient Descent

- ▶ **Problem:**

- ▶ Empirical risk is too expensive (sums over all training samples):

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^n \log p_{\text{model}}(y_i | \mathbf{x}_i; \boldsymbol{\theta})$$

Optimization

Stochastic Gradient Descent

- ▶ **Problem:**

- ▶ Empirical risk is too expensive (sums over all training samples):

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^n \log p_{\text{model}}(y_i | \mathbf{x}_i; \boldsymbol{\theta})$$

- ▶ **Solution:**

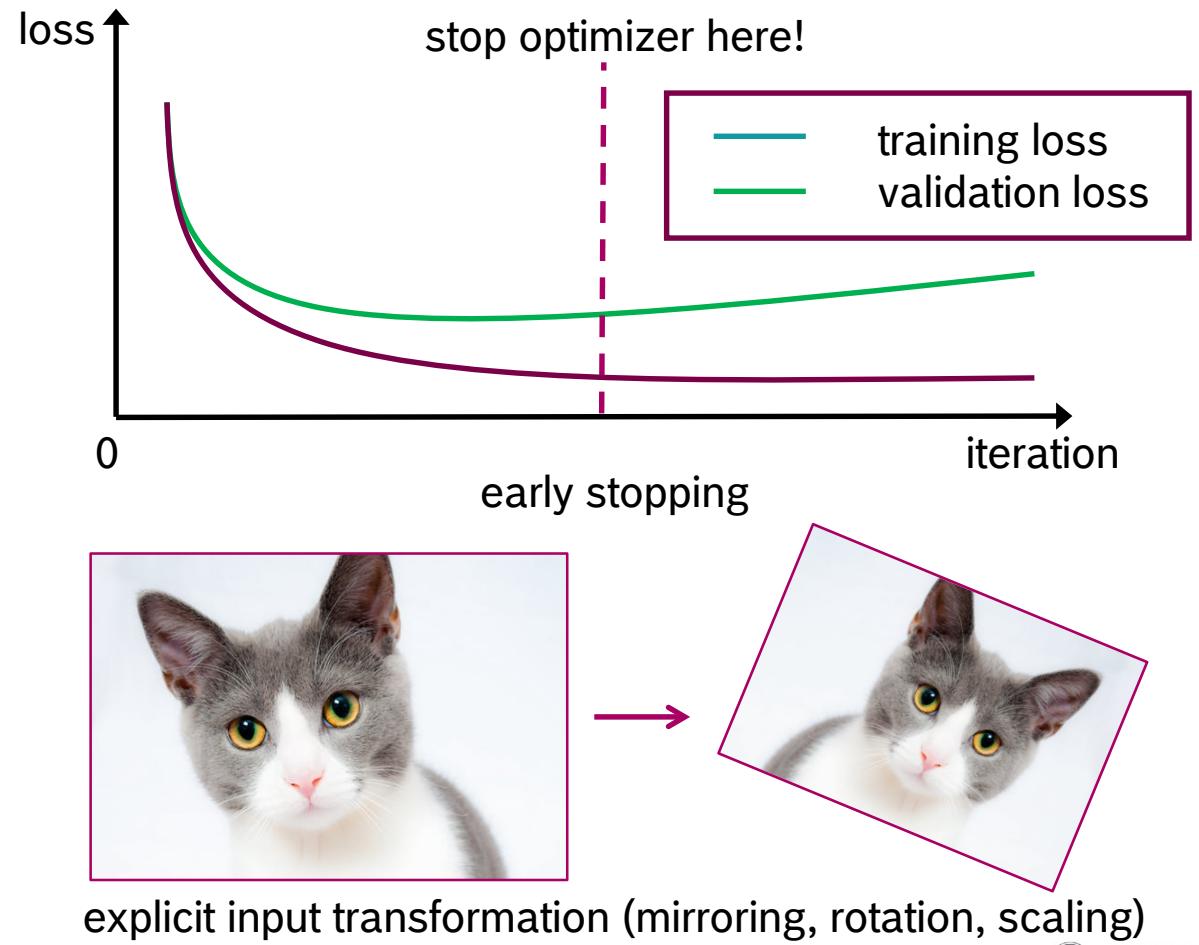
- ▶ Compute gradient on a random set of samples, i.e. a **minibatch**
 - ▶ Use this as the approximation of the gradient for the full batch

- ▶ **Minibatch size:**

- ▶ Increases gradient accuracy (but less than linearly!)
 - ▶ Increases parallelism utilization
 - ▶ Increases memory utilization (assuming full parallelism)
 - ▶ Decreases regularization effect

Optimization Regularization In Deep Learning (Revisited)

- ▶ **Optimizer:**
 - ▶ early stopping:
 - ▶ minibatch size
 - ▶ constrained optimization
 - ▶ **Data augmentation:**
 - ▶ explicit input transformations
 - ▶ adversarial training

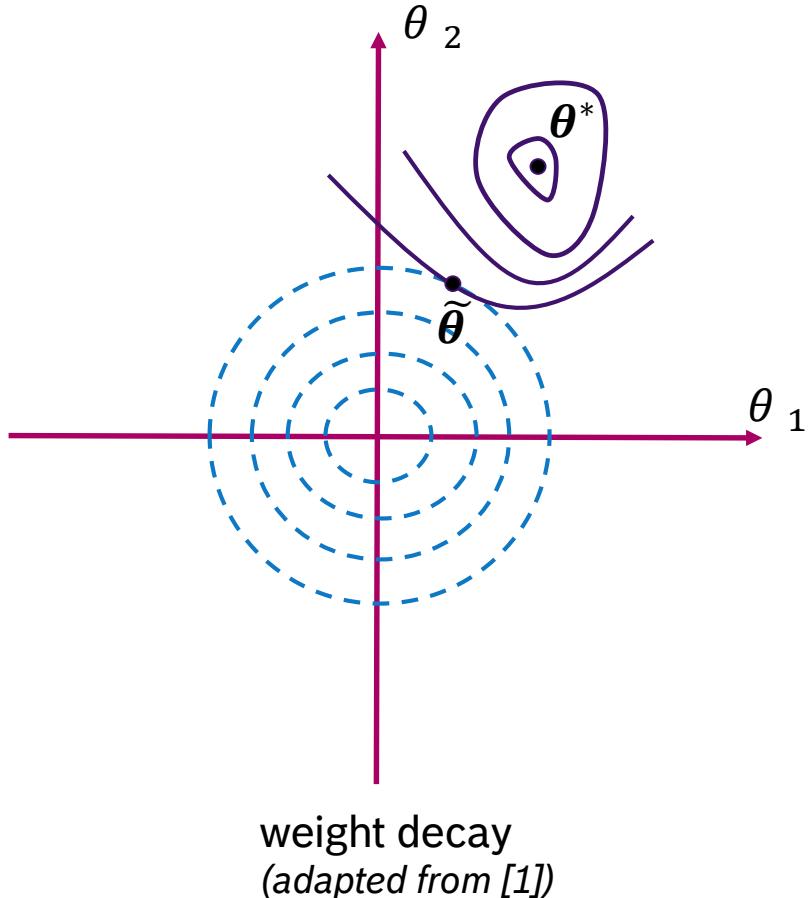


Optimization

Regularization In Deep Learning (Revisited)

- ▶ **Model structure:**
 - ▶ sharing parameters
 - ▶ trimming parameters
 - ▶ multi-task learning
 - ▶ dropout

- ▶ **Objective function:**
 - ▶ weight decay
 - ▶ label smoothing



Optimization

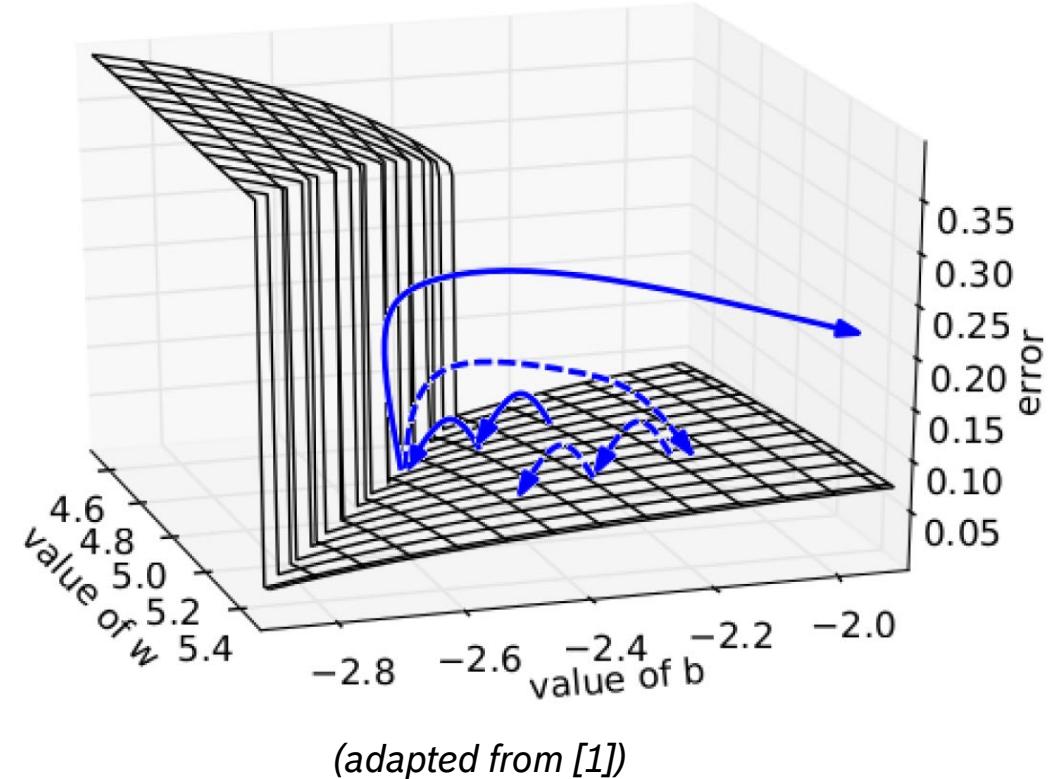
Meta algorithms: Batch Normalization

► Problem:

- ▶ sensitivity to one parameters often depends on others parameters
- ▶ highly nonlinear behavior!
- ▶ dangerous to update parameters at once

► Solution:

- ▶ re-parameterize the network
- ▶ add batch normalization layer:
 - ▶ computes mean and variance of each feature over the mini-batch
 - ▶ for each feature: subtracts mean and divides by variance
- ▶ layer disappears at runtime (collapsed with the conv/fully connected layer)



CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks

Deep Learning Recap

► Machine Learning:

- ▶ Search for a program that maps features to correct answers
- ▶ Feature design is hard!

► End-to-end Learning:

- ▶ Features are real-world observations
- ▶ Learn several intermediate representation layers

► Deep Learning:

- ▶ End-to-end learning with a lot of layers
- ▶ How do we avoid the parameter space explosion? (overfitting)

Convolutional Neural Networks

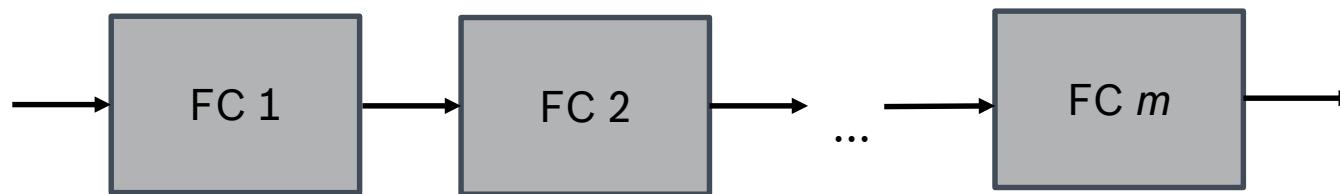
A First Try: The Multi-Layer Perceptron

► Idea:

- Level 1: match the input image to a set of patterns
- Level 2: match the level 1 features to a set of patterns
- ...

► Implementation:

- Fully Connected (FC) Layers

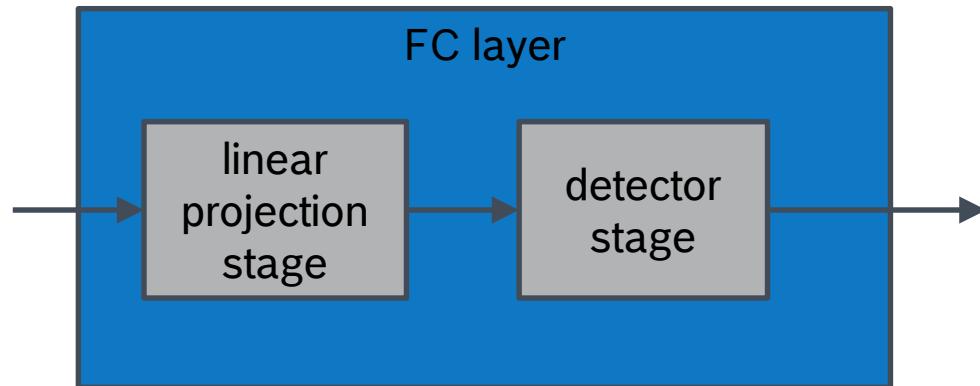


Convolutional Neural Networks

The Stages of an FC Layer

► Stages:

- **Linear projection:** project feature vector x along a learned direction w
- **Detector:** is the projection y “strong enough” for a pattern match? (match strength z)

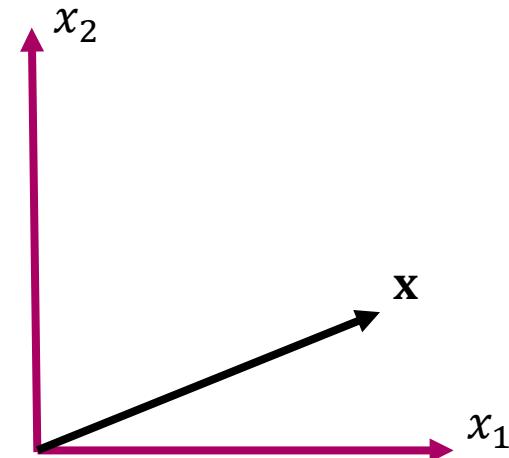
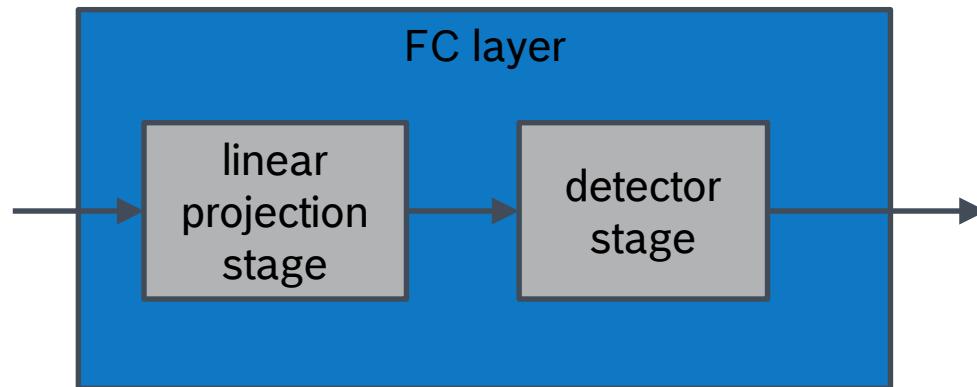


Convolutional Neural Networks

The Stages of an FC Layer

► Stages:

- **Linear projection:** project feature vector x along a learned direction w
- **Detector:** is the projection y “strong enough” for a pattern match? (match strength z)

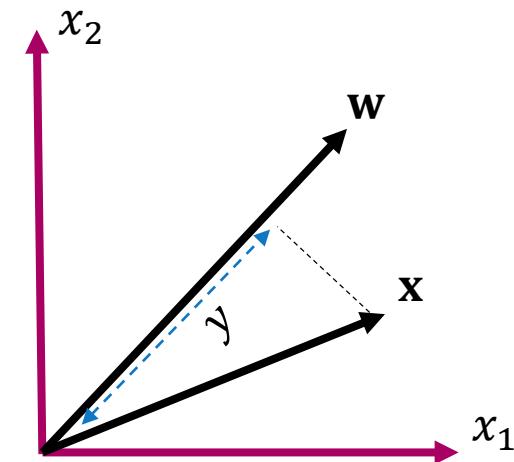
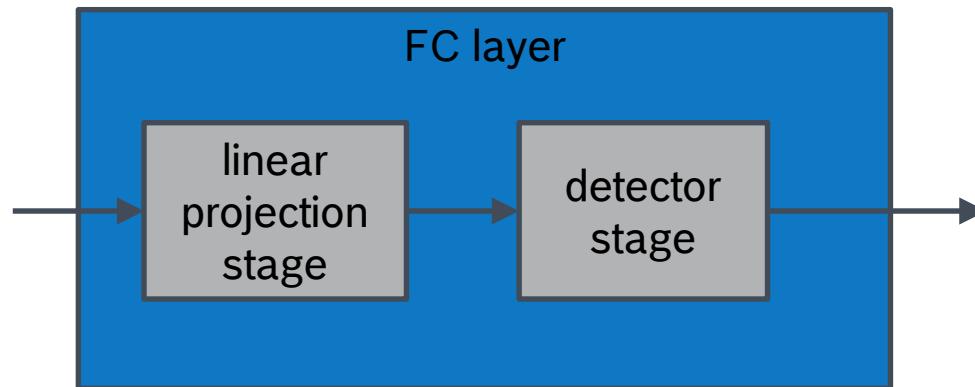


Convolutional Neural Networks

The Stages of an FC Layer

► Stages:

- **Linear projection:** project feature vector x along a learned direction w
- **Detector:** is the projection y “strong enough” for a pattern match? (match strength z)

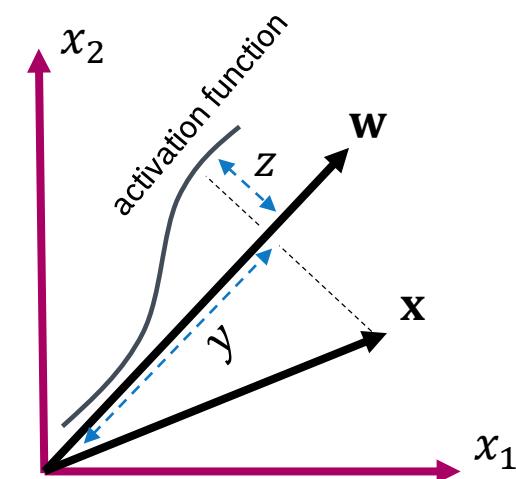
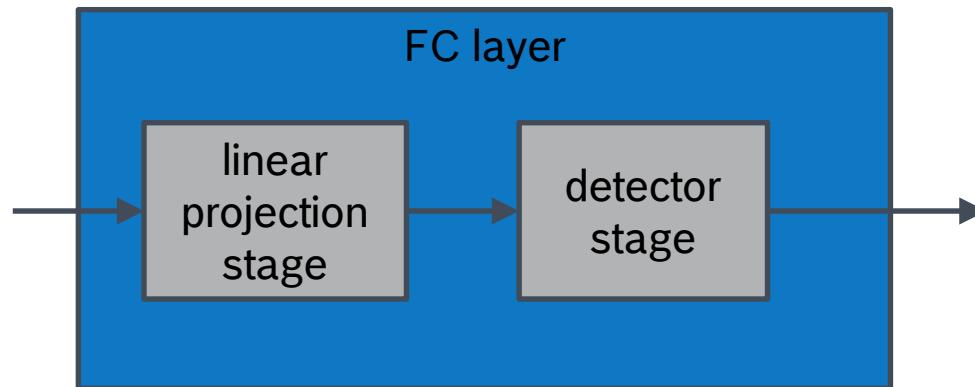


Convolutional Neural Networks

The Stages of an FC Layer

► Stages:

- **Linear projection:** project feature vector x along a learned direction w
- **Detector:** is the projection y “strong enough” for a pattern match? (match strength z)



Convolutional Neural Networks

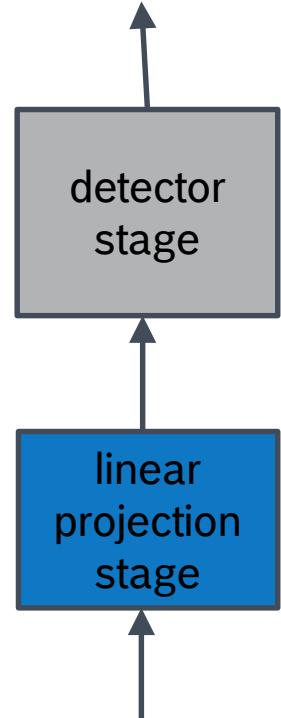
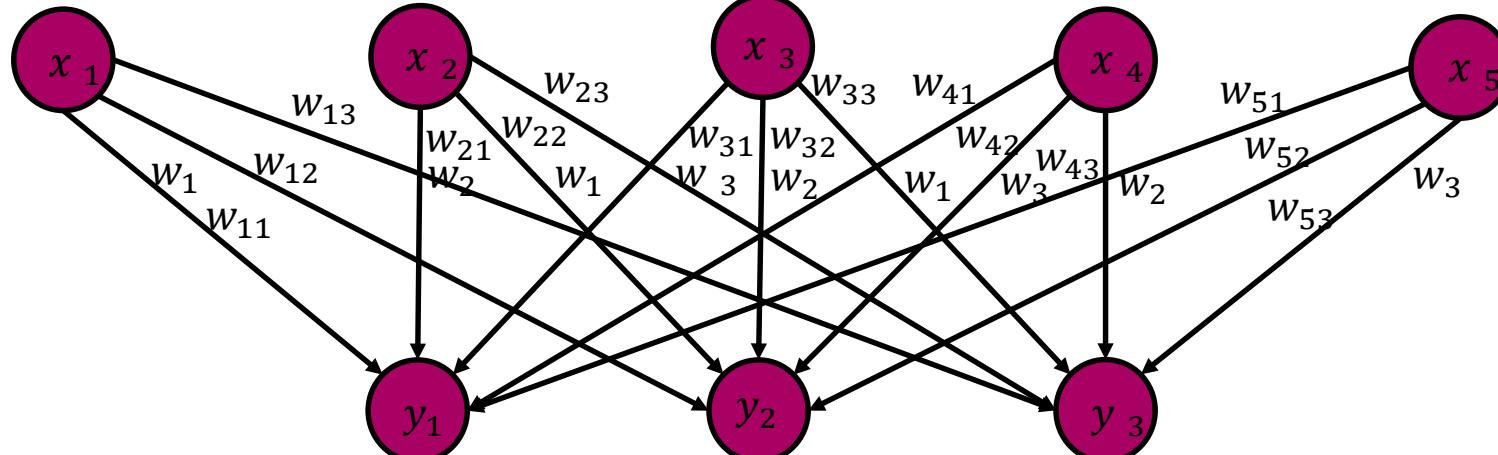
FC Layer: The Linear Projection Stage

► Input:

- plain feature vector
- do not exploit spatial/temporal structure

► Output:

- each element a different linear combination of the input elements: $y_j = \sum_{i=1}^{N_{\text{in}}} w_{ij} \cdot x_i$



Convolutional Neural Networks

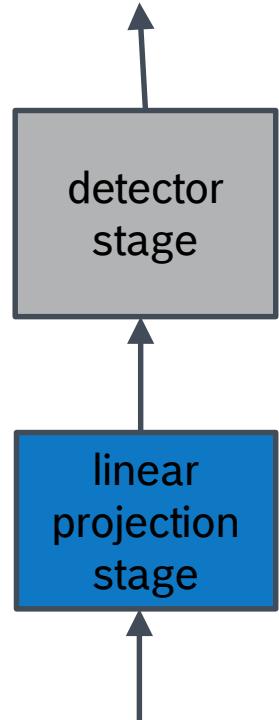
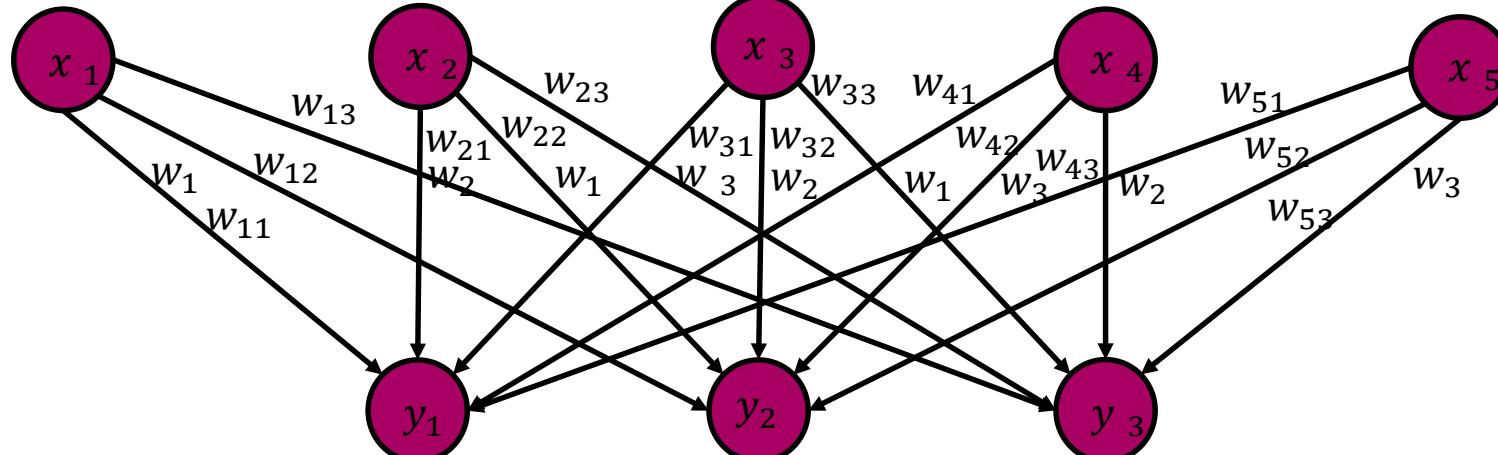
FC Layer: Complexity

► Parameters:

- $N_{\text{in}} \cdot N_{\text{out}}$
- Example: 1 million parameters for a 256x256 input image!

► Time:

- Multiply-and-accumulates (MACs): $N_{\text{in}} \cdot N_{\text{out}}$



Convolutional Neural Networks

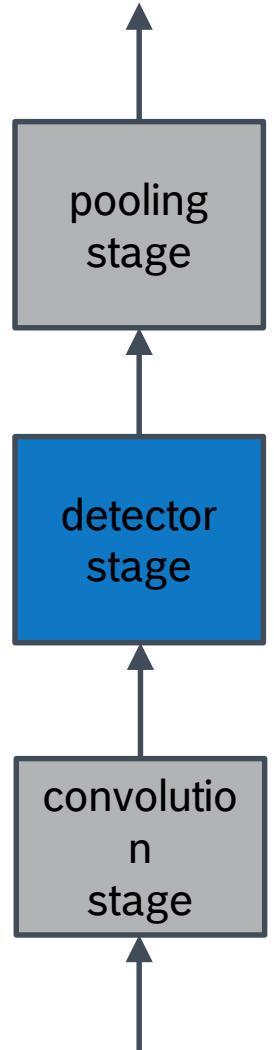
The Detector Stage

► Requirements:

- Highly nonlinear behavior (soft thresholding)
- Balance modeling power, efficiency, ease of optimization

► State-of-the-art:

- Most obvious choices do not work well! (e.g. softplus)
- Open research field



Convolutional Neural Networks

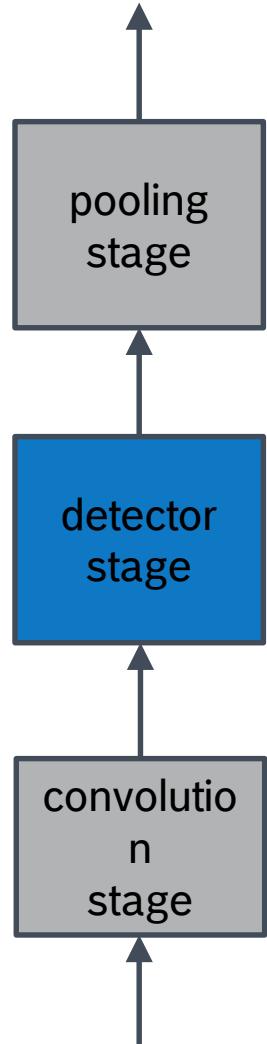
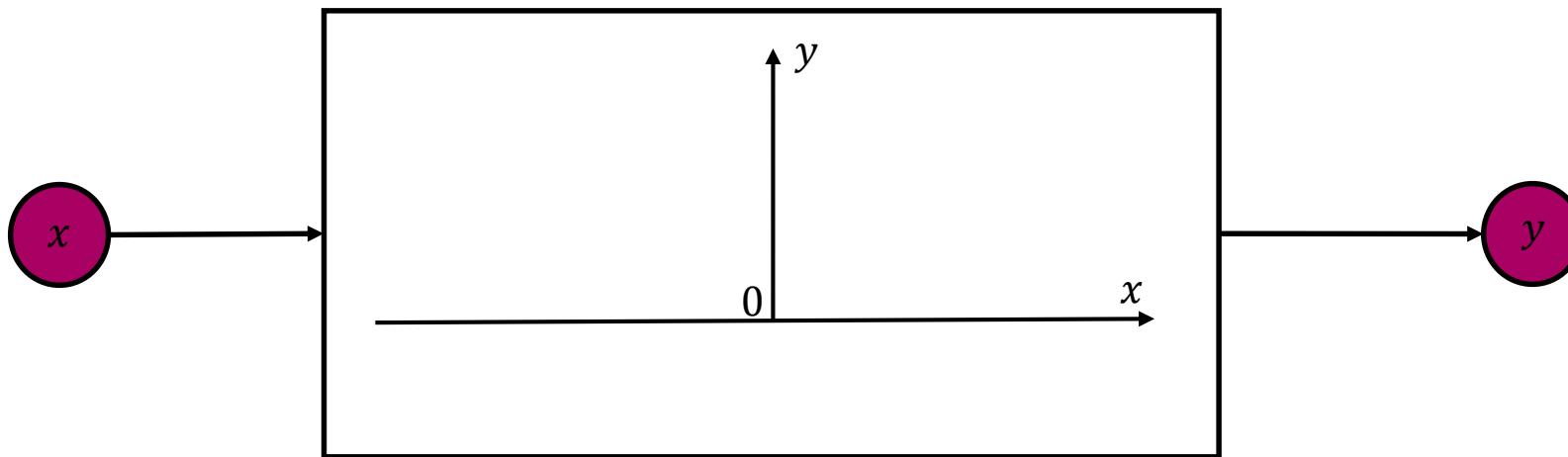
The Detector Stage

► Requirements:

- Highly nonlinear behavior (soft thresholding)
- Balance modeling power, efficiency, ease of optimization

► State-of-the-art:

- Most obvious choices do not work well! (e.g. softplus)
- Open research field



Convolutional Neural Networks

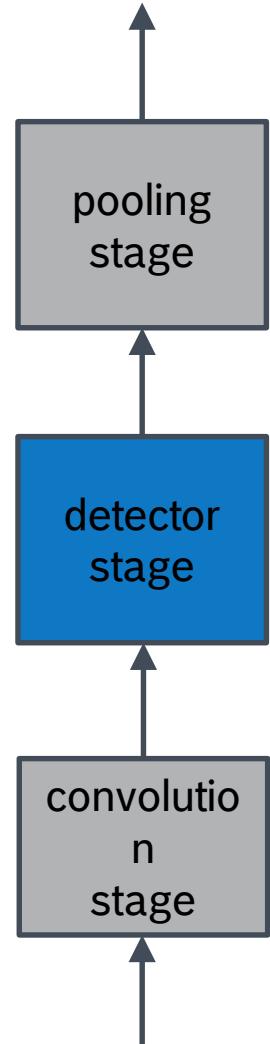
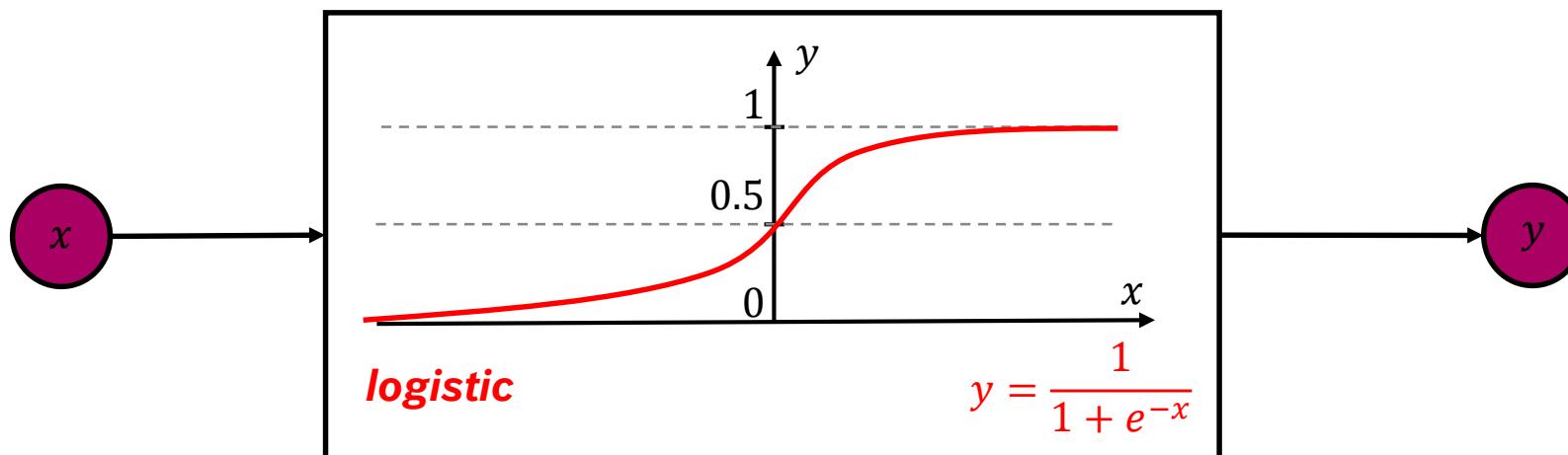
The Detector Stage

► Requirements:

- Highly nonlinear behavior (soft thresholding)
- Balance modeling power, efficiency, ease of optimization

► State-of-the-art:

- Most obvious choices do not work well! (e.g. softplus)
- Open research field



Convolutional Neural Networks

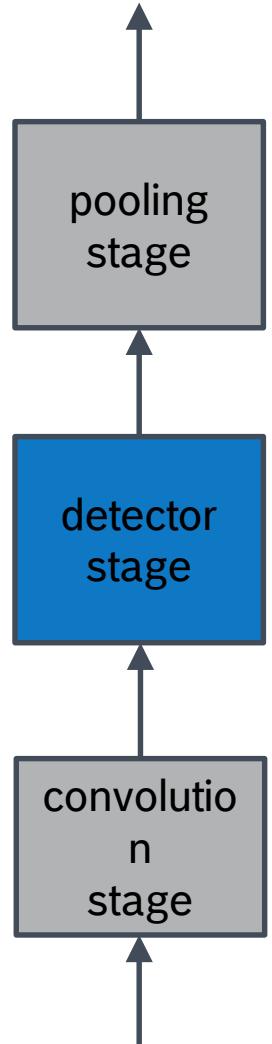
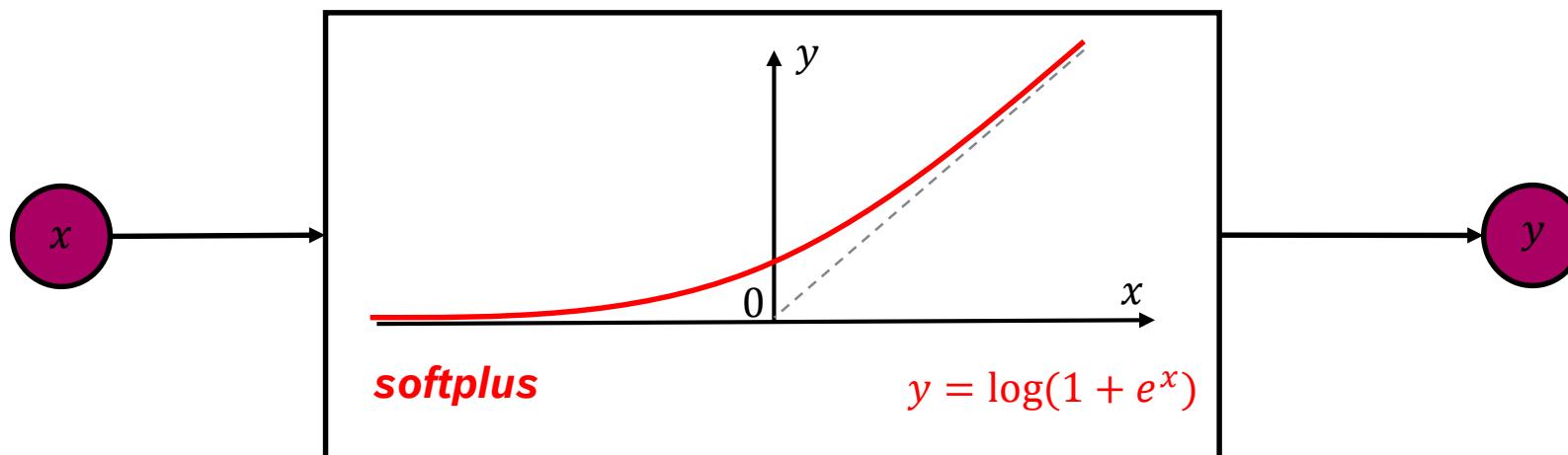
The Detector Stage

► Requirements:

- Highly nonlinear behavior (soft thresholding)
- Balance modeling power, efficiency, ease of optimization

► State-of-the-art:

- Most obvious choices do not work well! (e.g. softplus)
- Open research field



Convolutional Neural Networks

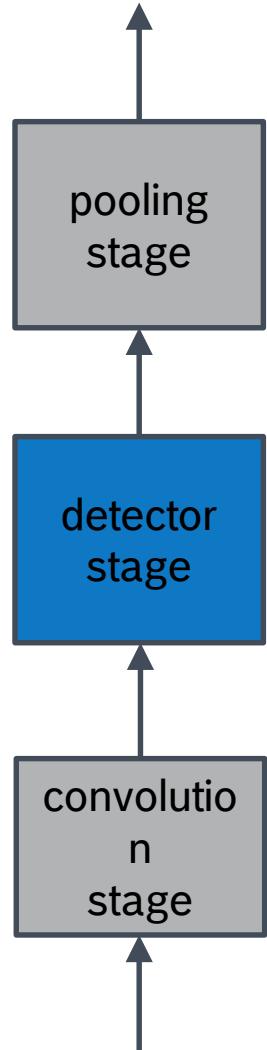
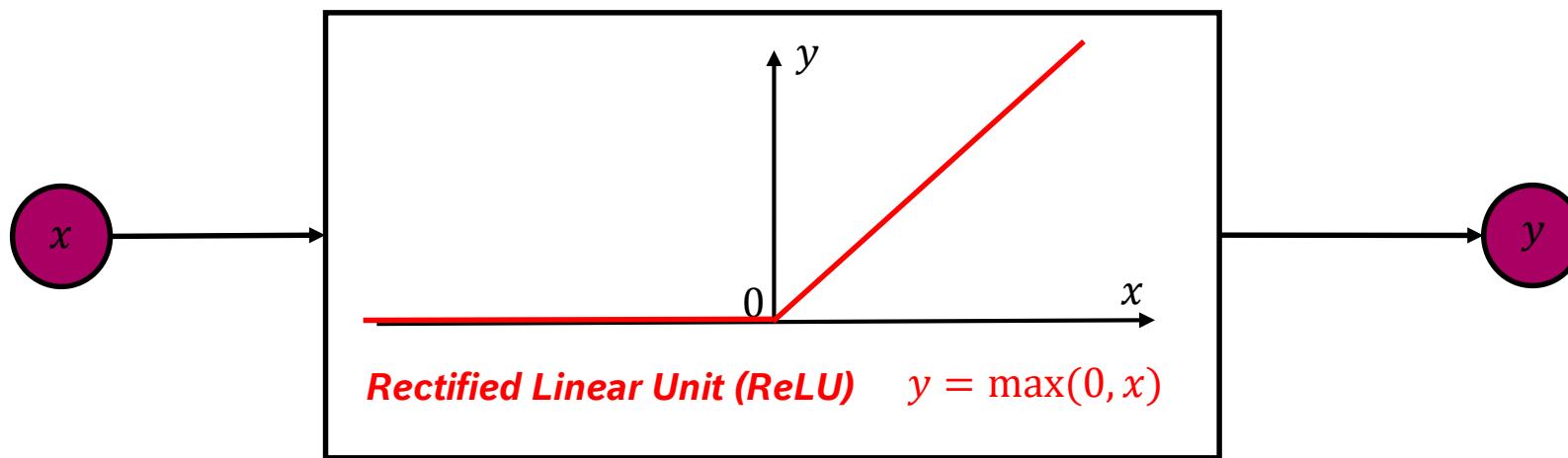
The Detector Stage

► Requirements:

- Highly nonlinear behavior (soft thresholding)
- Balance modeling power, efficiency, ease of optimization

► State-of-the-art:

- Most obvious choices do not work well! (e.g. softplus)
- Open research field



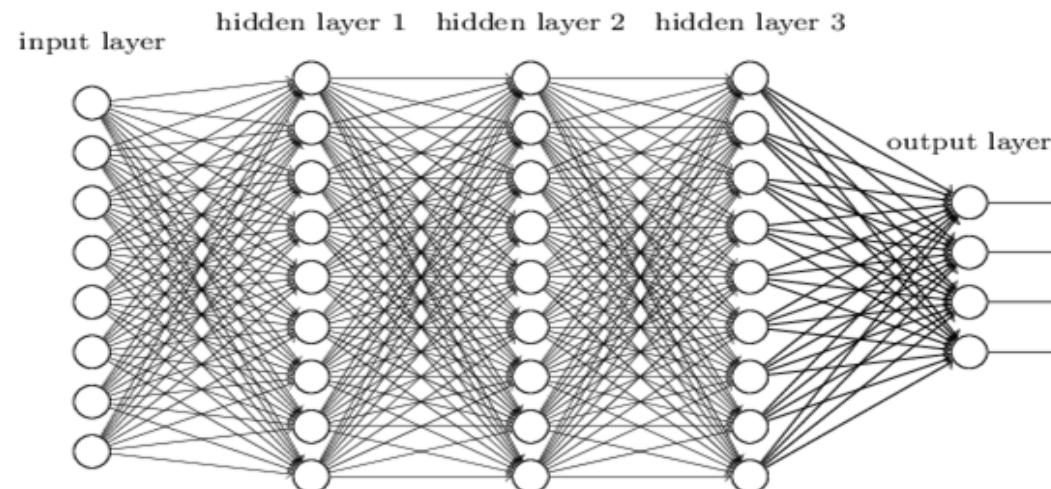
Convolutional Neural Networks

Does the Multi Layer Perceptron Work?

- Rarely! 😞

- Problems:

- Huge number of parameters
- Low spatial invariance



Convolutional Neural Networks

Intuition

► Solution:

- Scan for smaller local patterns and successively group them into larger ones
- Introduce Convolutional (CONV) Layers

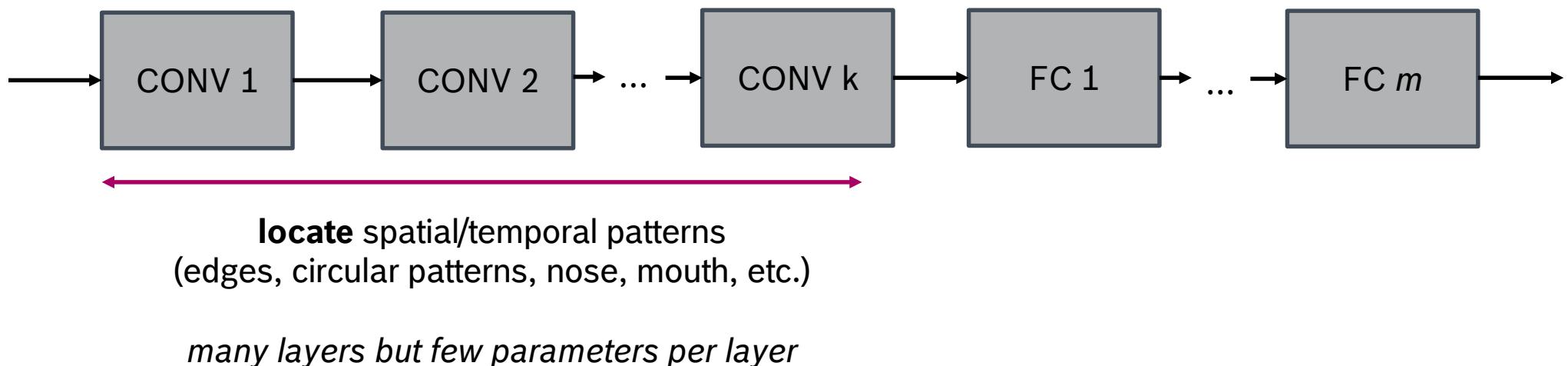


Convolutional Neural Networks

Intuition

► Solution:

- Scan for smaller local patterns and successively group them into larger ones
- Introduce Convolutional (CONV) Layers

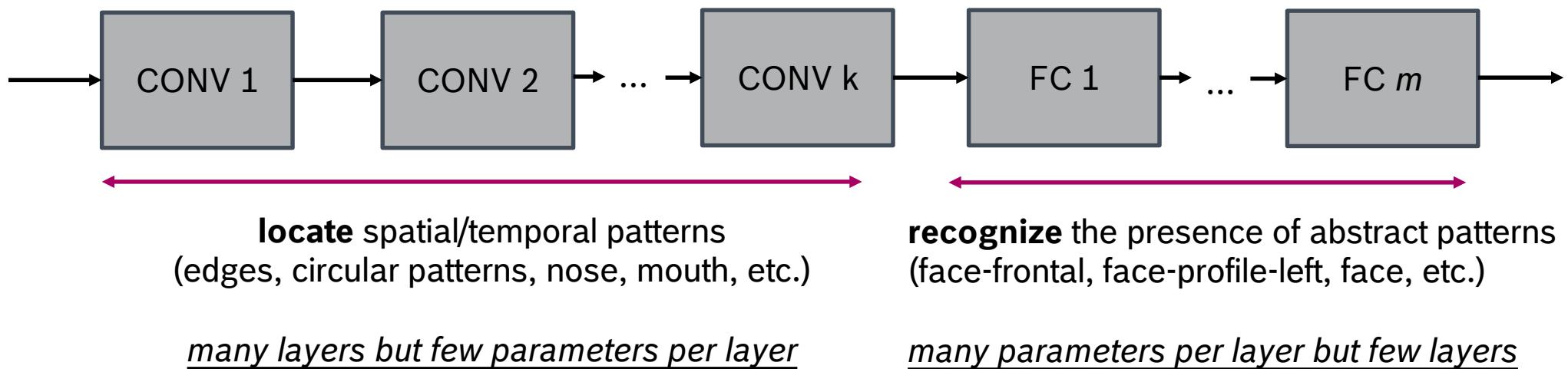


Convolutional Neural Networks

Intuition

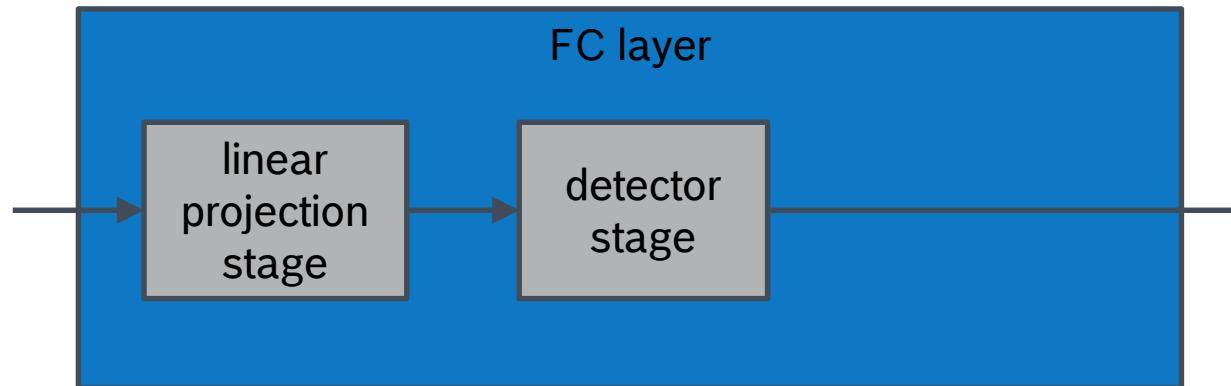
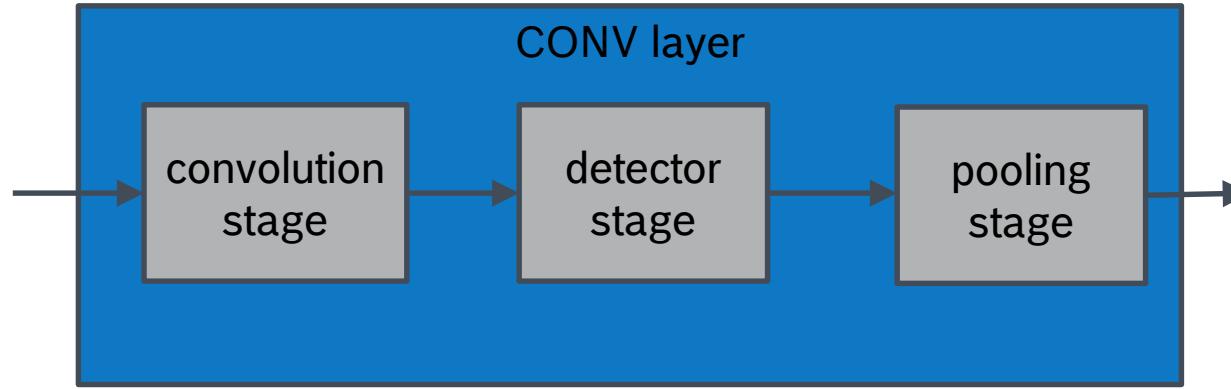
► Solution:

- Scan for smaller local patterns and successively group them into larger ones
- Introduce Convolutional (CONV) Layers



Convolutional Neural Networks

Layer Stages



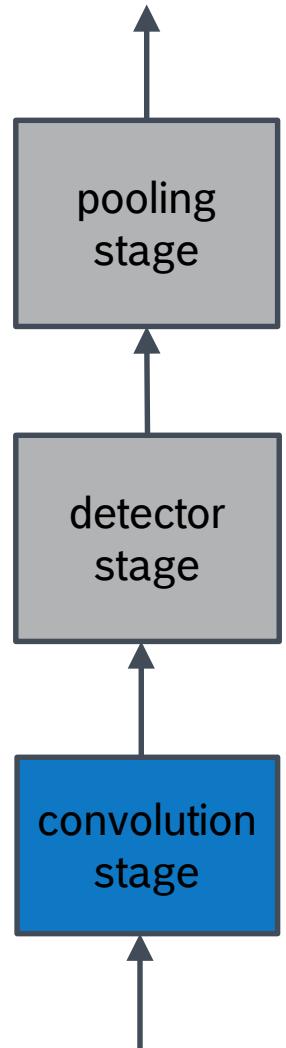
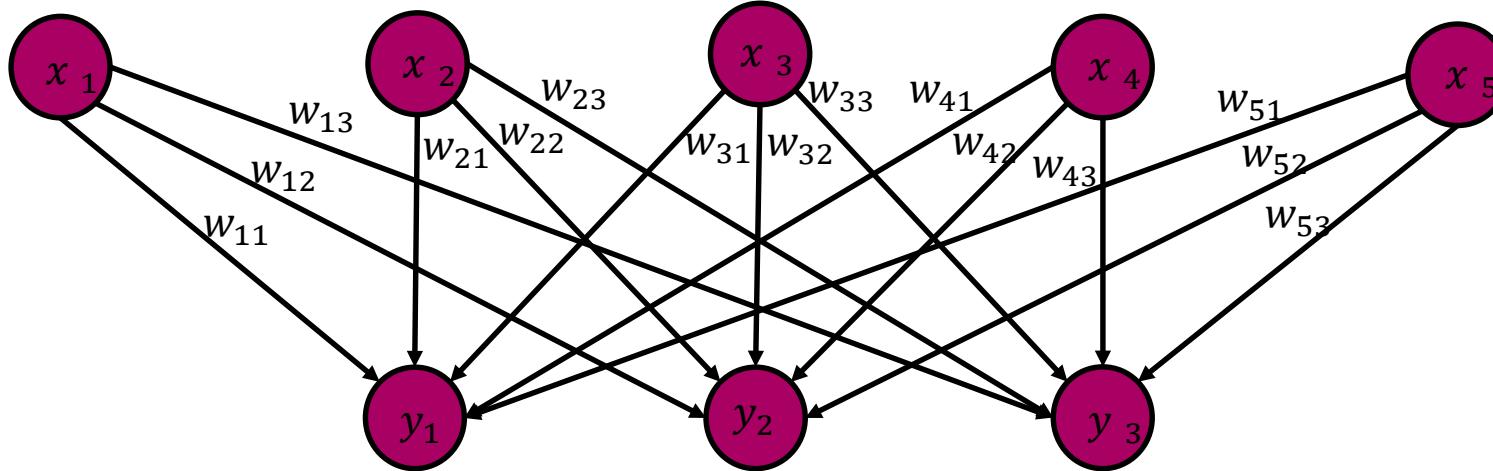
Convolutional Architectures

CONV Layer: The Convolution Stage (1D case)

► Idea:

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**):

$$y_j = \sum_{i=1}^{N_w} w_i \cdot x_{i+j-1}$$



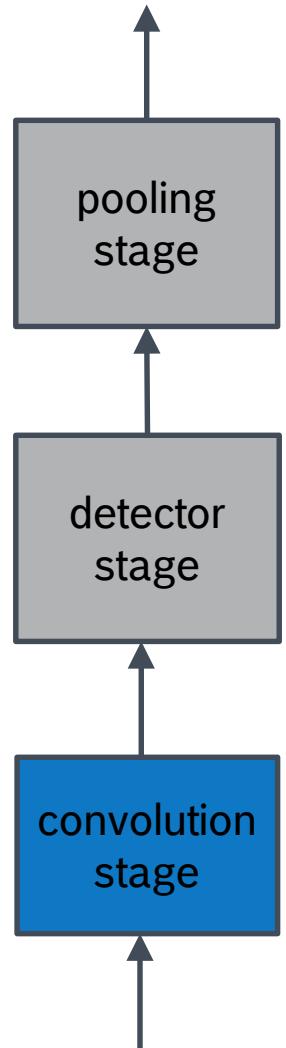
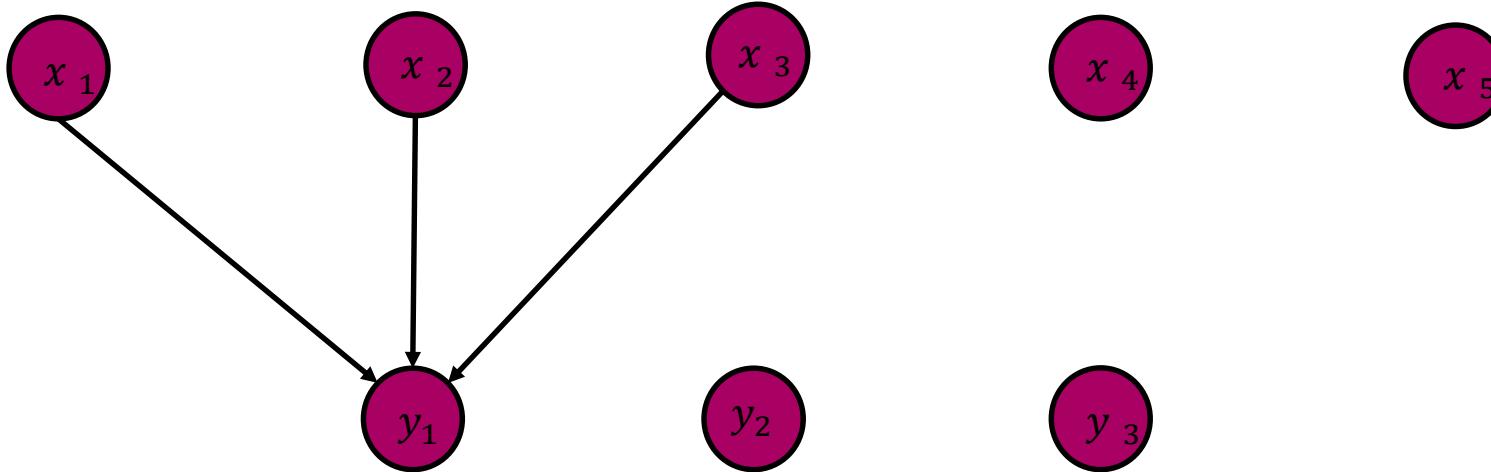
Convolutional Architectures

CONV Layer: The Convolution Stage (1D case)

► Idea:

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**):

$$y_j = \sum_{i=1}^{N_w} w_i \cdot x_{i+j-1}$$



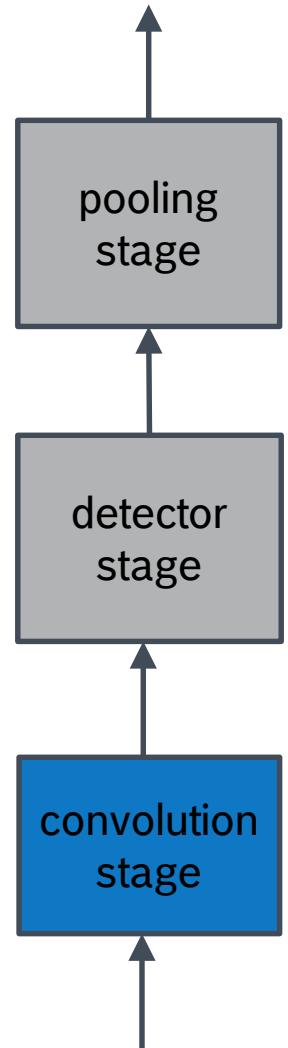
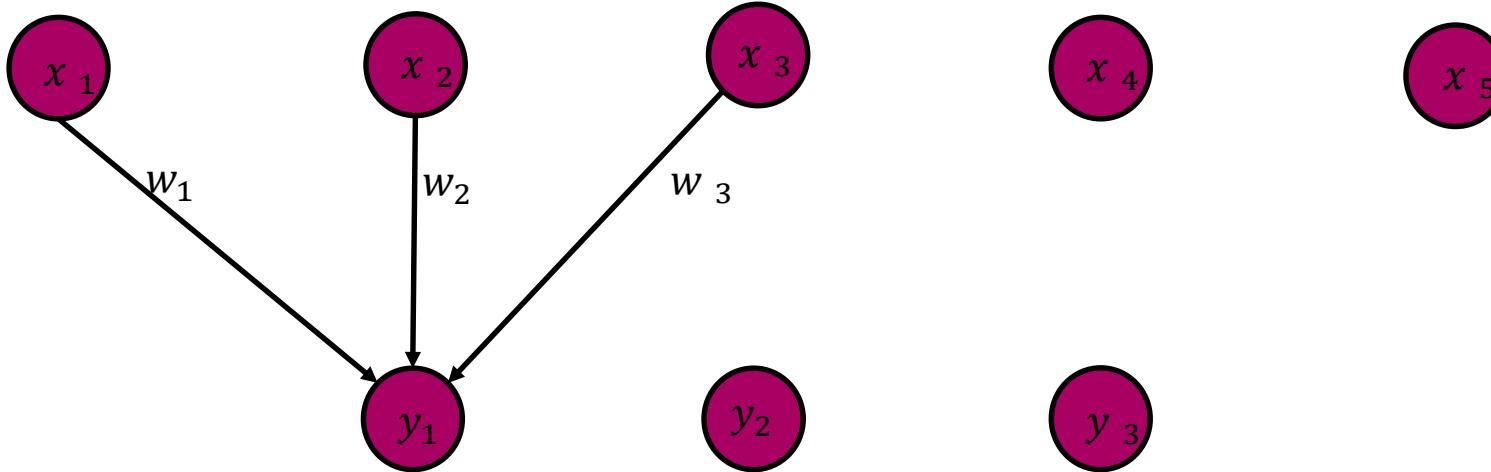
Convolutional Architectures

CONV Layer: The Convolution Stage (1D case)

► Idea:

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**):

$$y_j = \sum_{i=1}^{N_w} w_i \cdot x_{i+j-1}$$



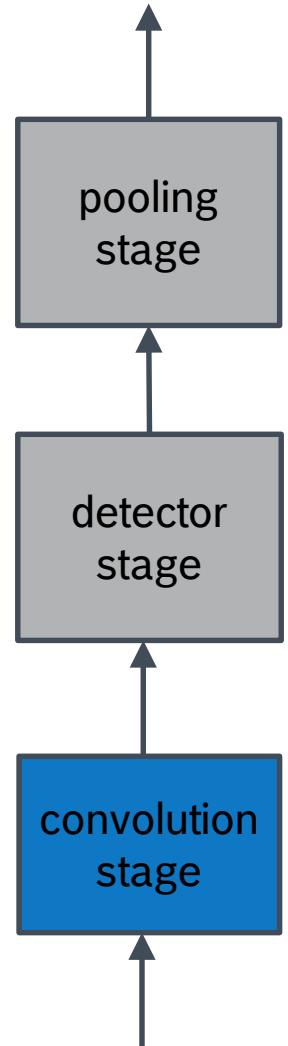
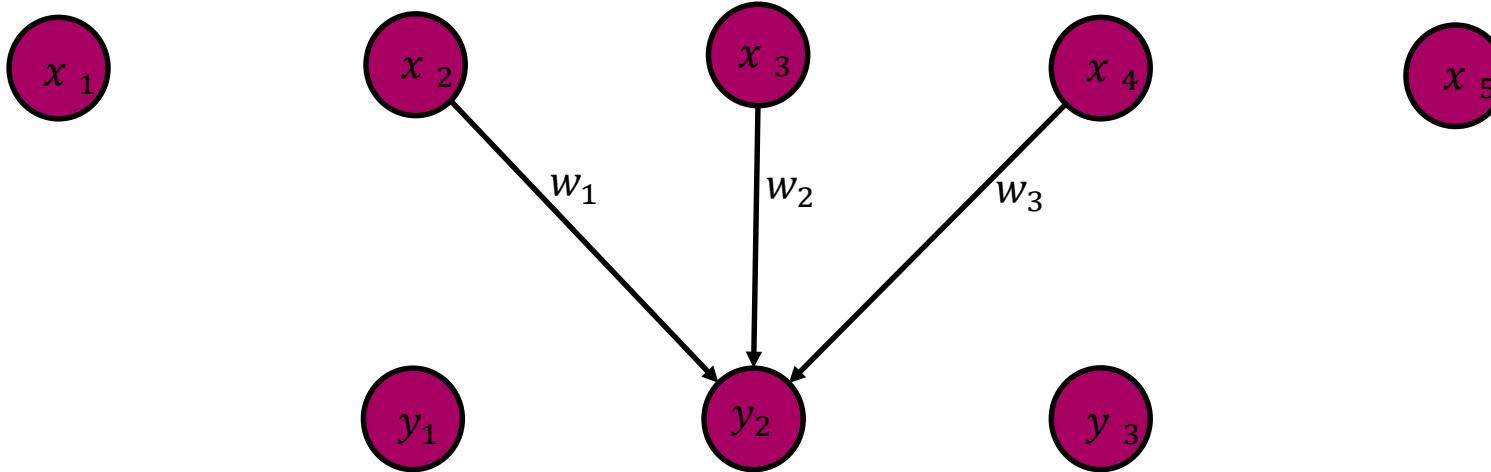
Convolutional Architectures

CONV Layer: The Convolution Stage (1D case)

► Idea:

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**):

$$y_j = \sum_{i=1}^{N_w} w_i \cdot x_{i+j-1}$$



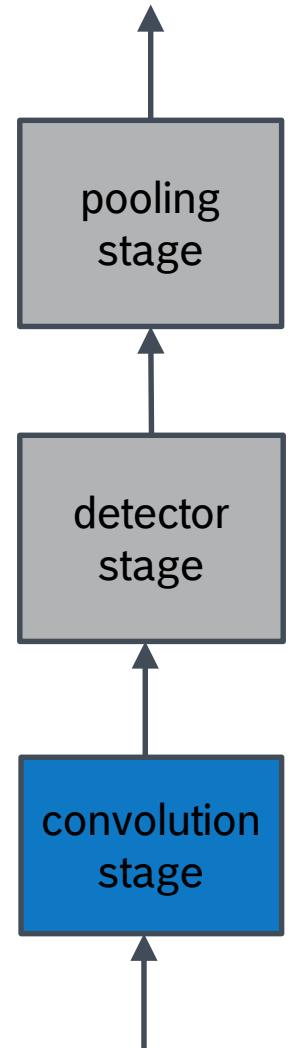
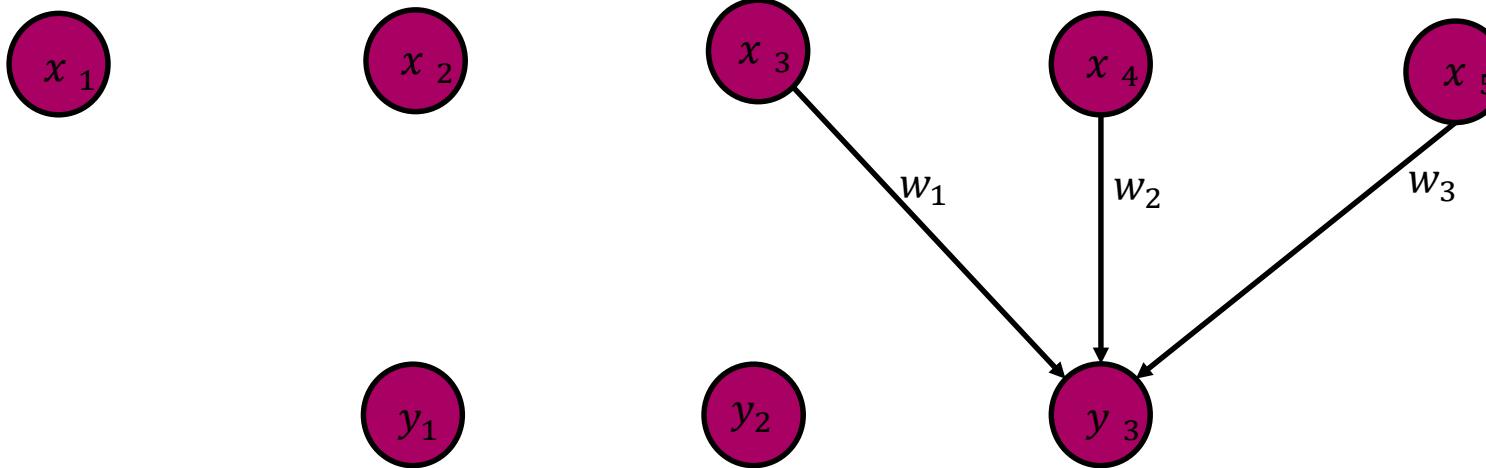
Convolutional Architectures

CONV Layer: The Convolution Stage (1D case)

► Idea:

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**):

$$y_j = \sum_{i=1}^{N_w} w_i \cdot x_{i+j-1}$$



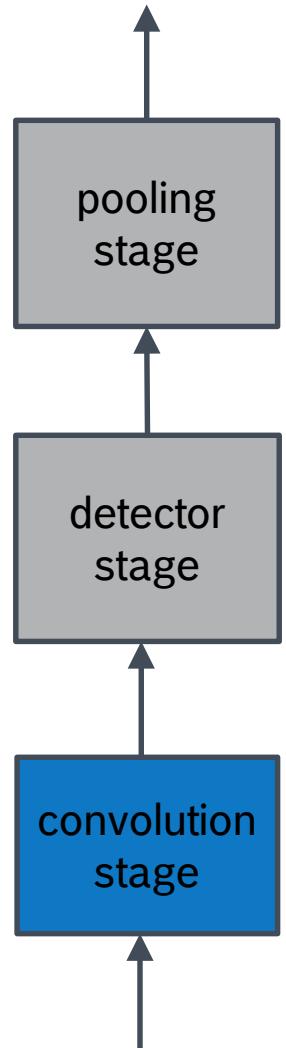
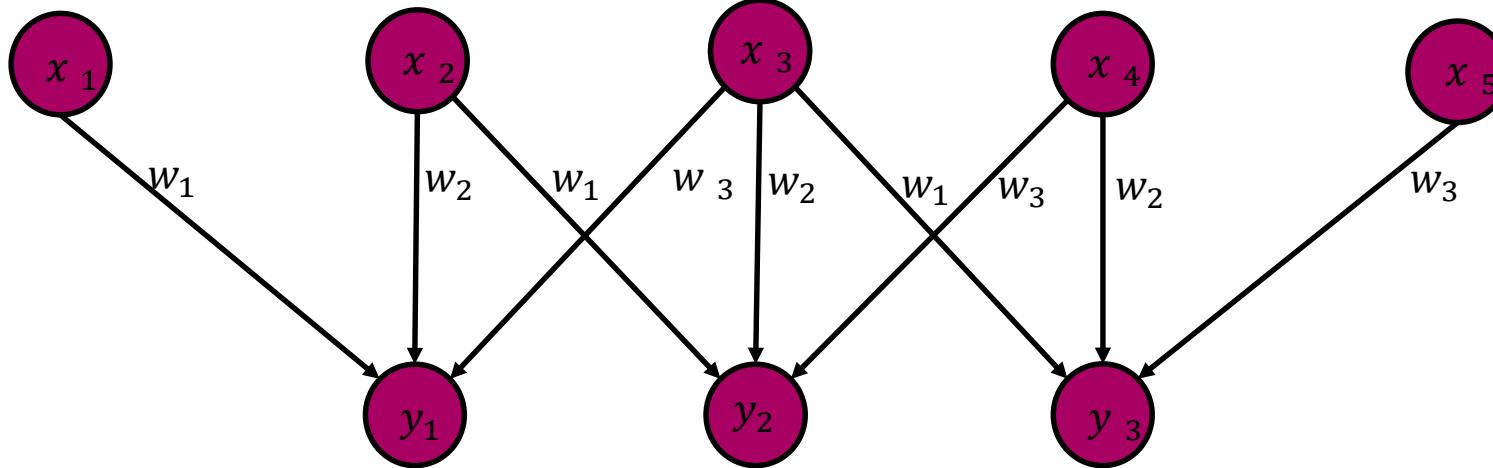
Convolutional Architectures

CONV Layer: The Convolution Stage (1D case)

► Idea:

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**):

$$y_j = \sum_{i=1}^{N_w} w_i \cdot x_{i+j-1}$$

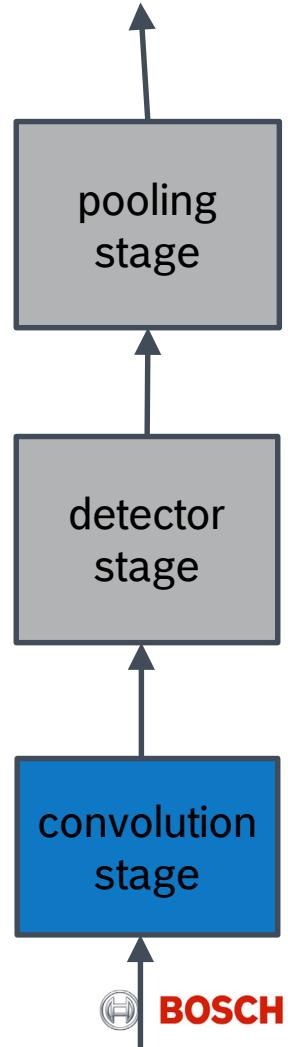
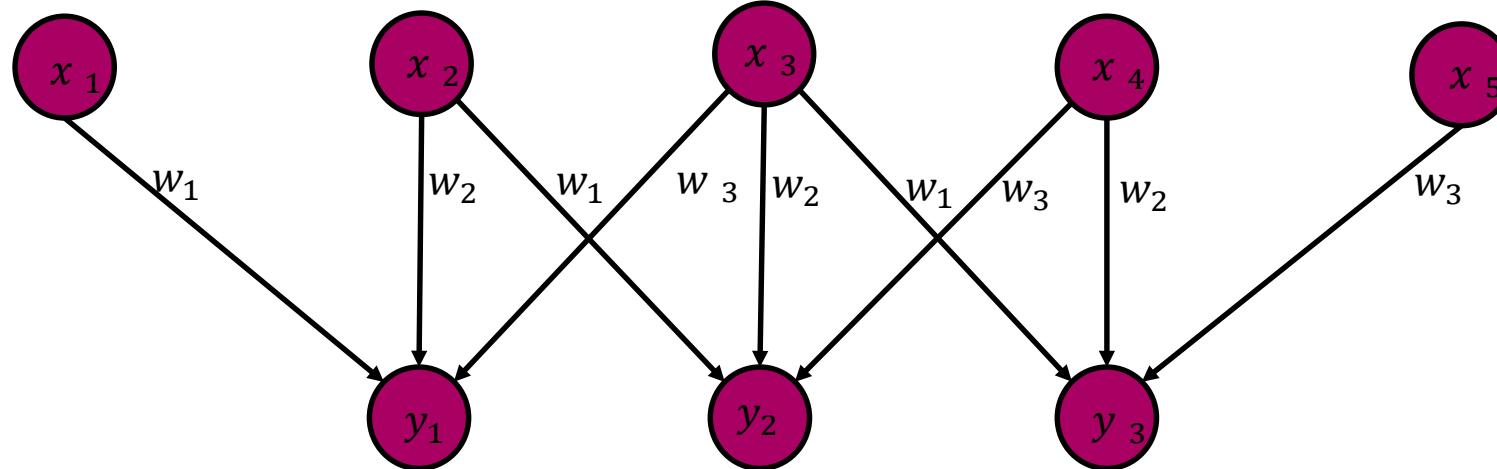


Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

► **Number of Parameters:** K_w (does not depend on the input size!)

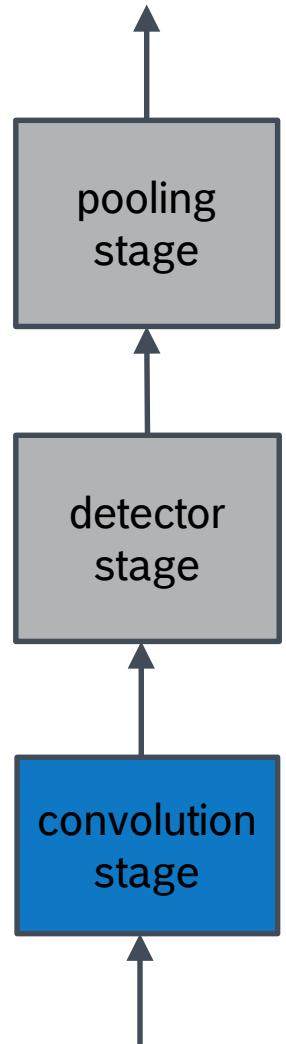
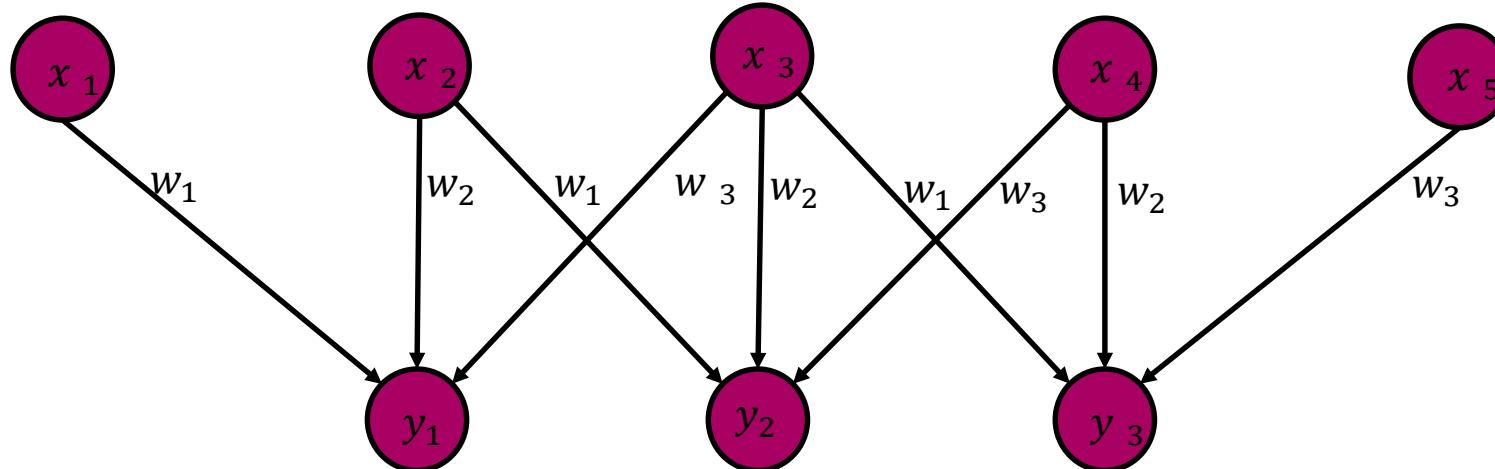
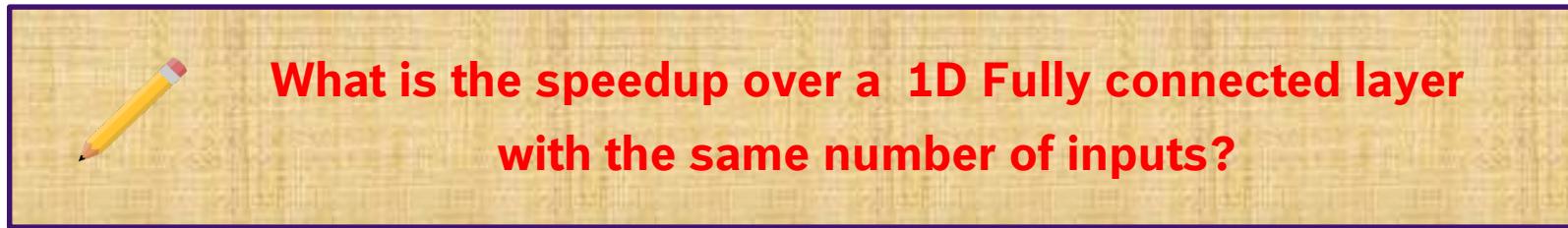
- We have actually regularized the FC layer!
- Parameter trimming: $w_{ij} = 0$, for $0 \leq i - j < N_w$
- Parameter tying: $w_{ij} = w_{kl}$, for $(k - i) = (l - j)$



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

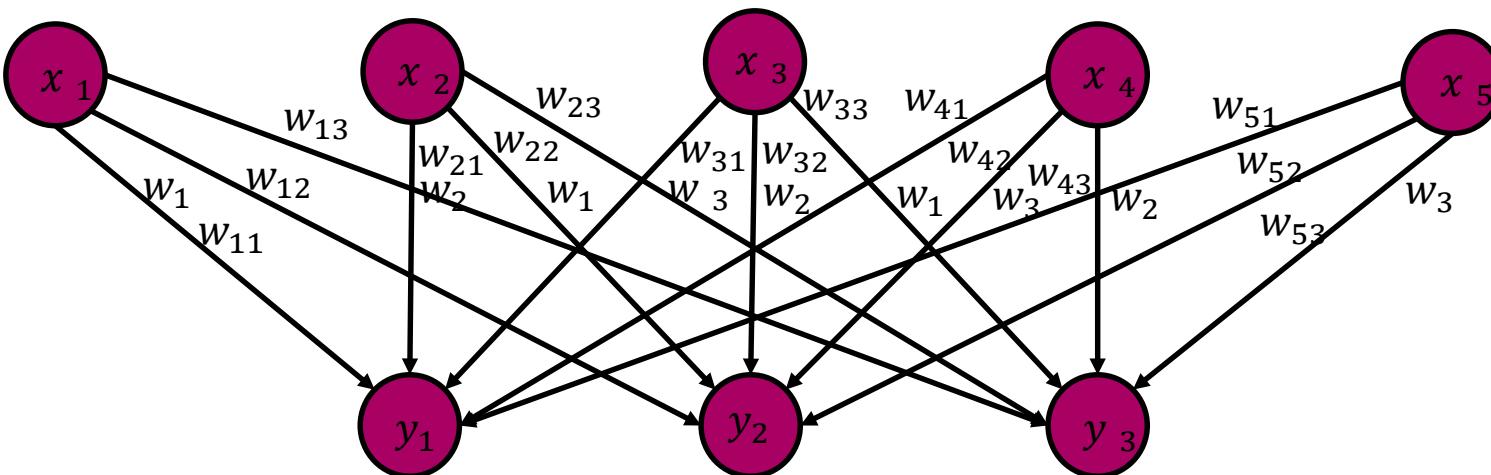
- **Time Complexity:** $N_w \cdot N_{\text{out}}$ MACs



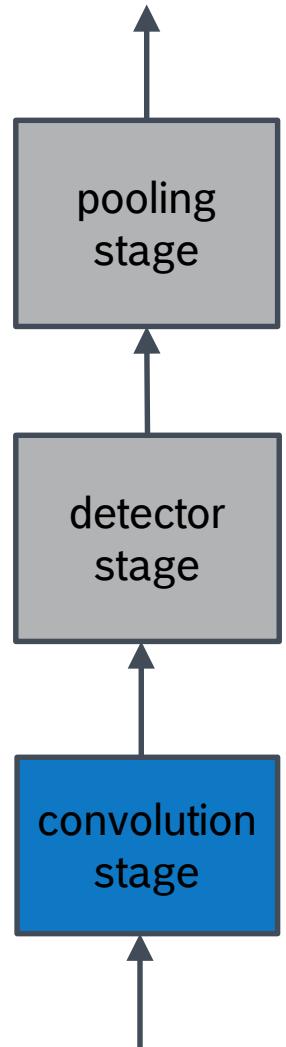
Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



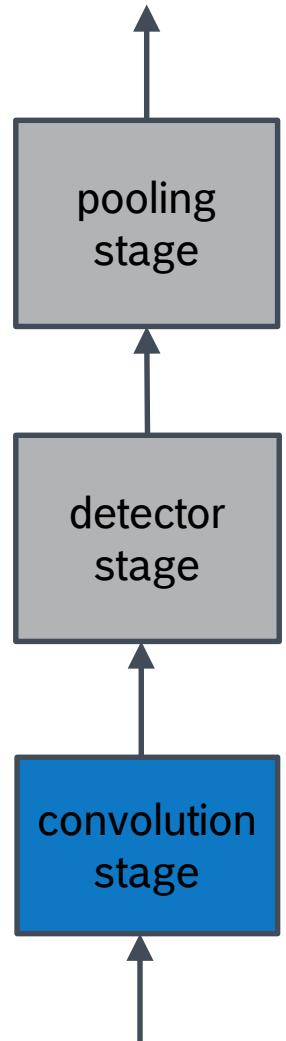
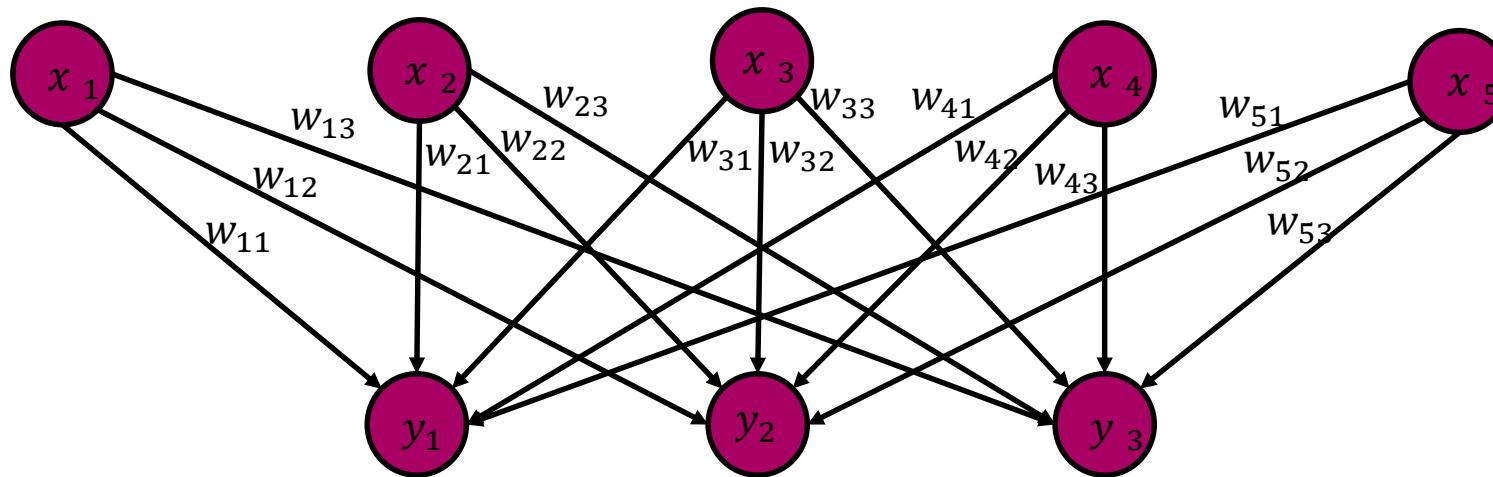
- **Equivariant** to translation, but not to scale and rotation
- Regularization: parameter trimming and tying (infinite prior)
- Biologically inspired (one of the few stories of success)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

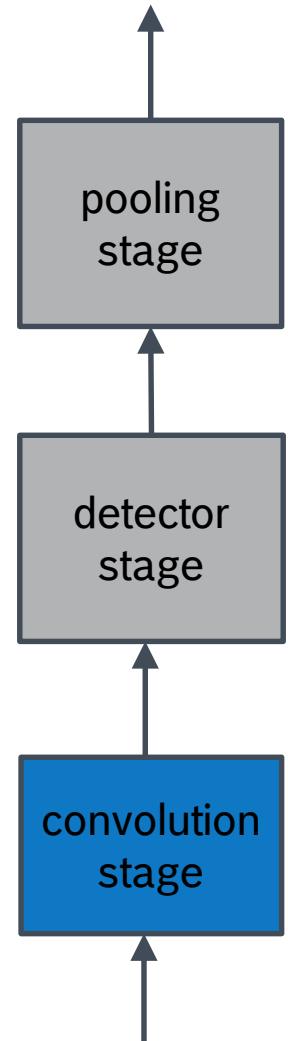
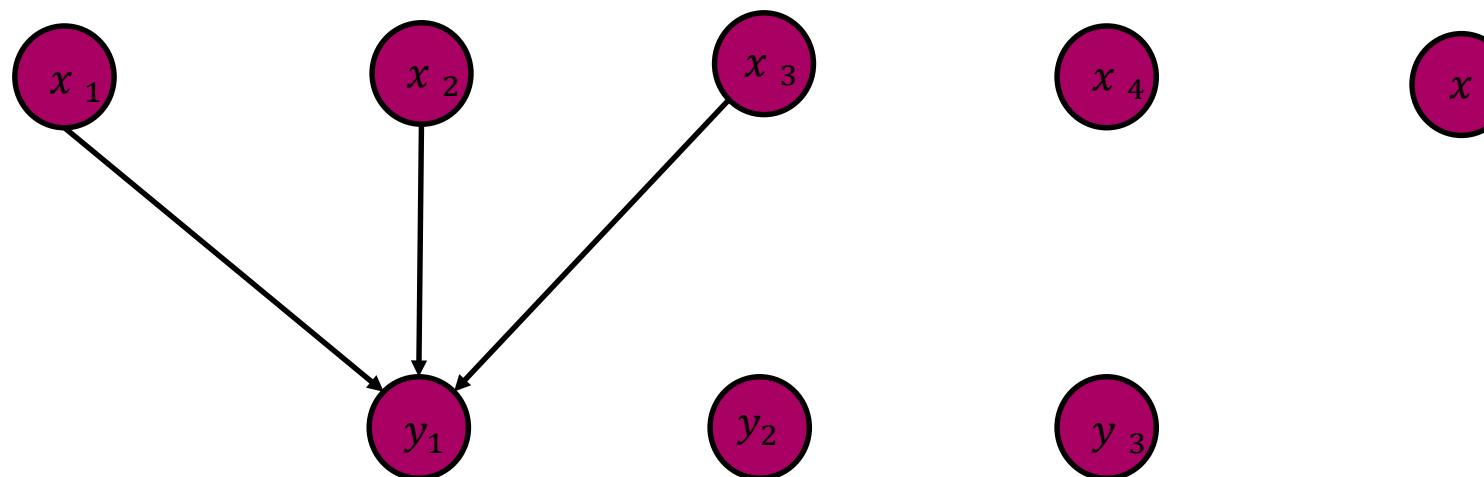
- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

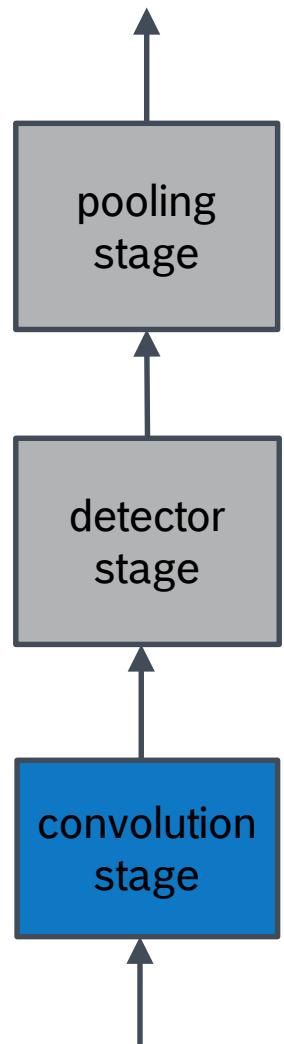
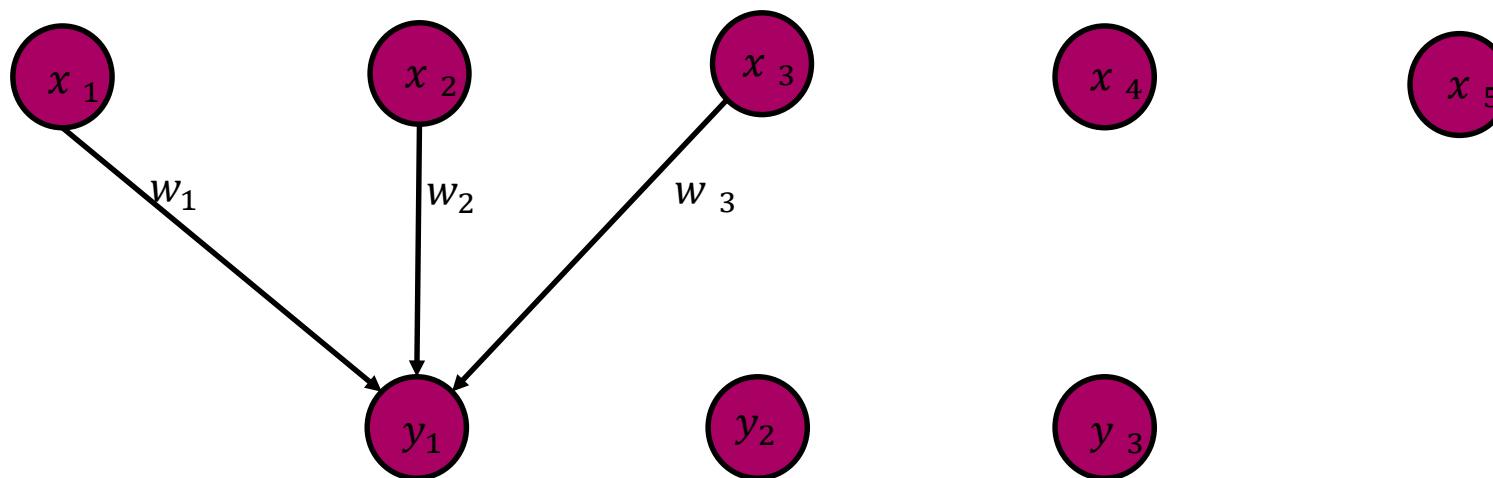
- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

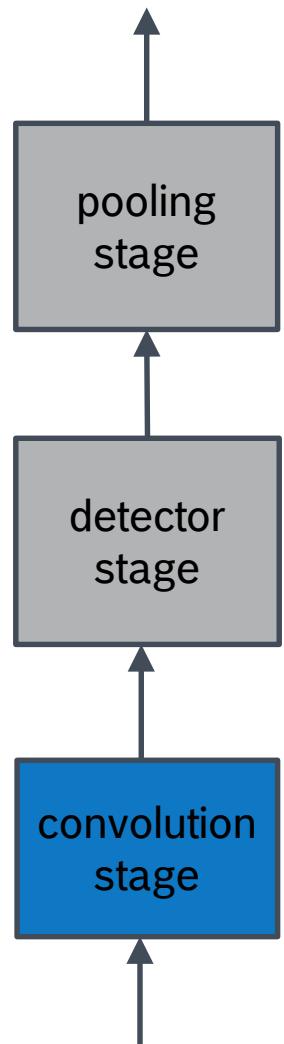
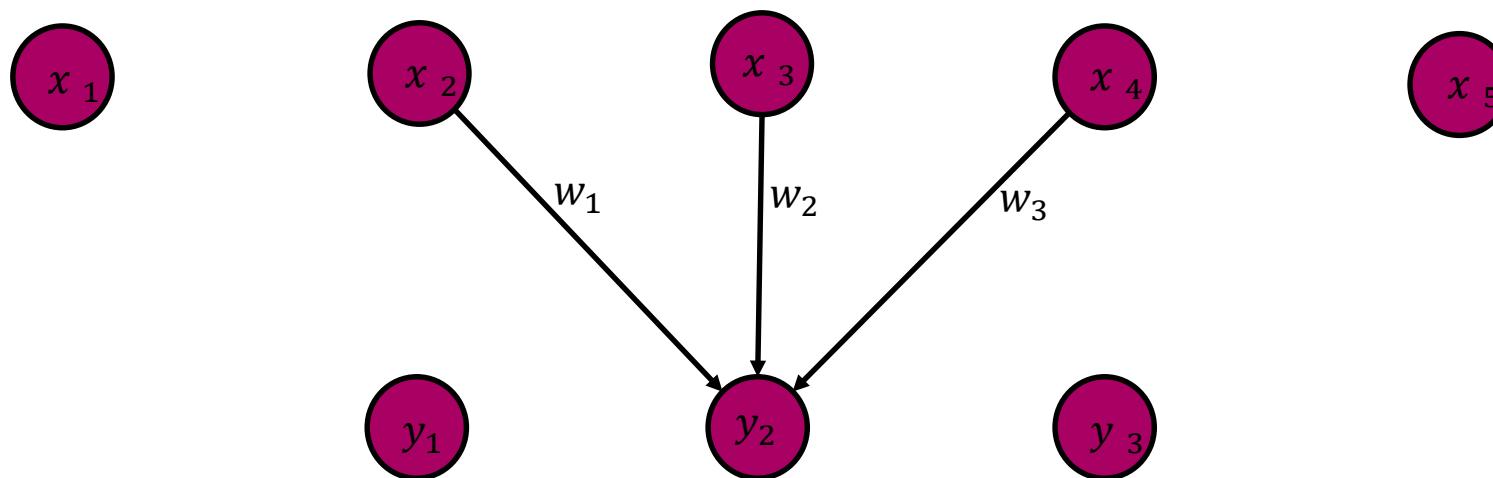
- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

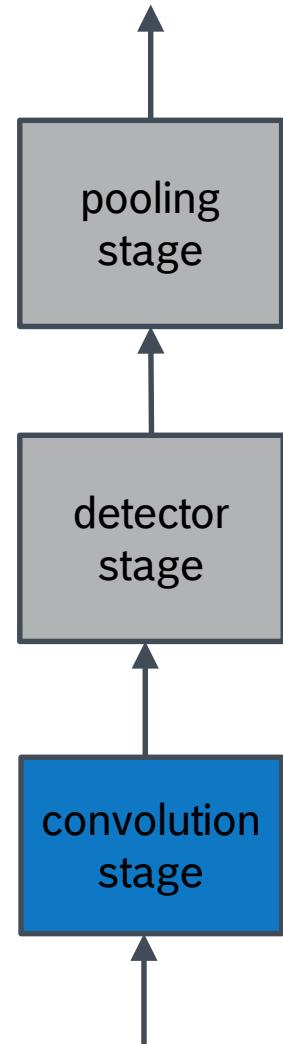
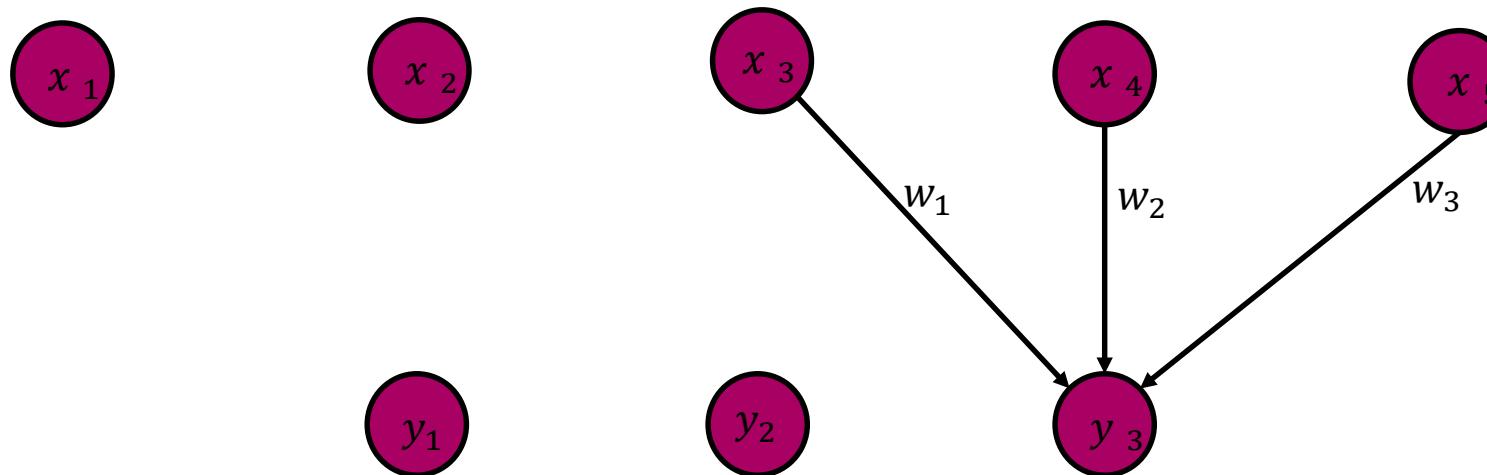
- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

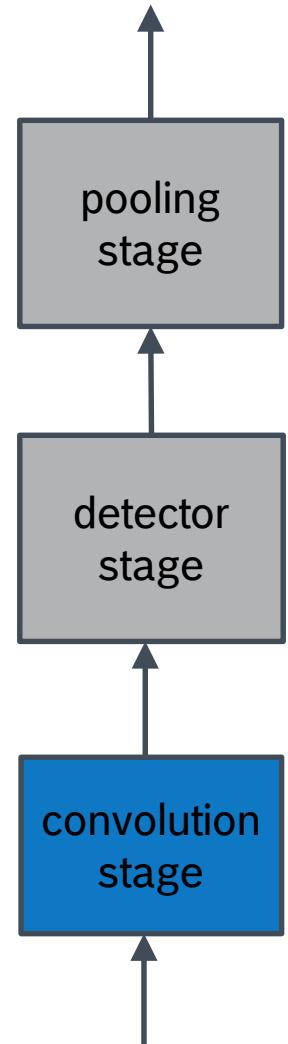
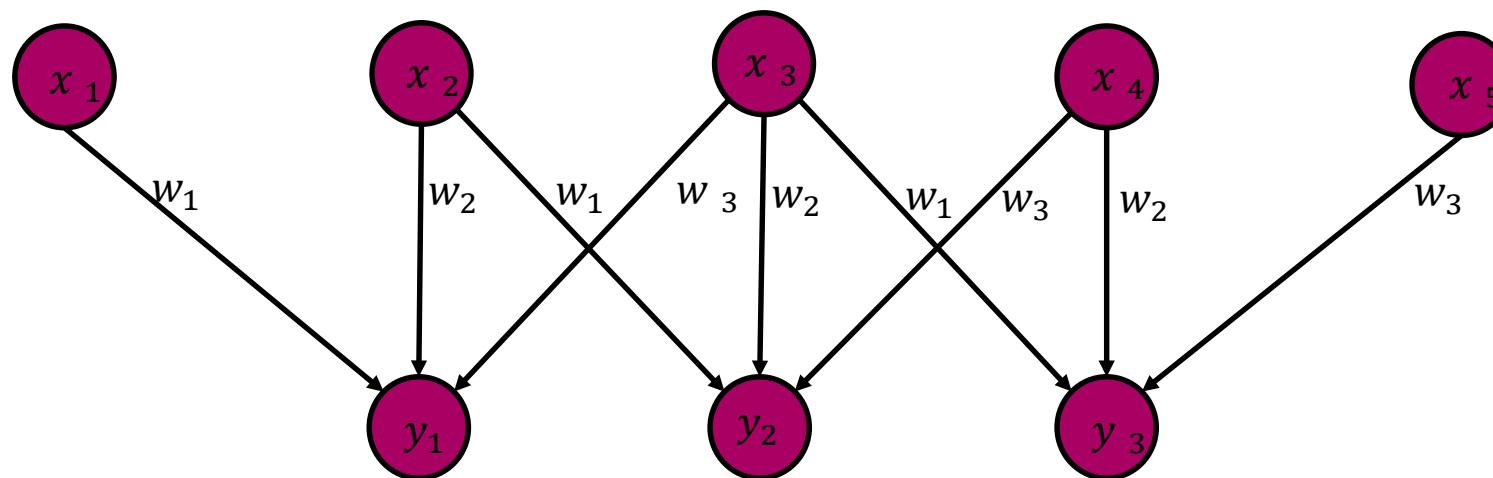
- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

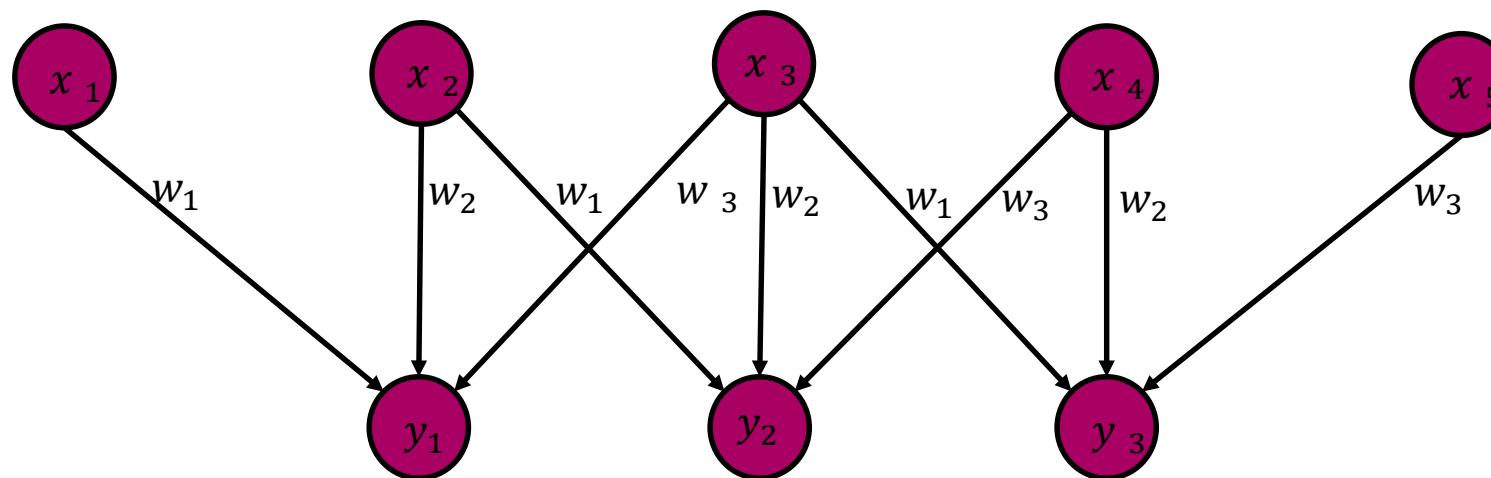
- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



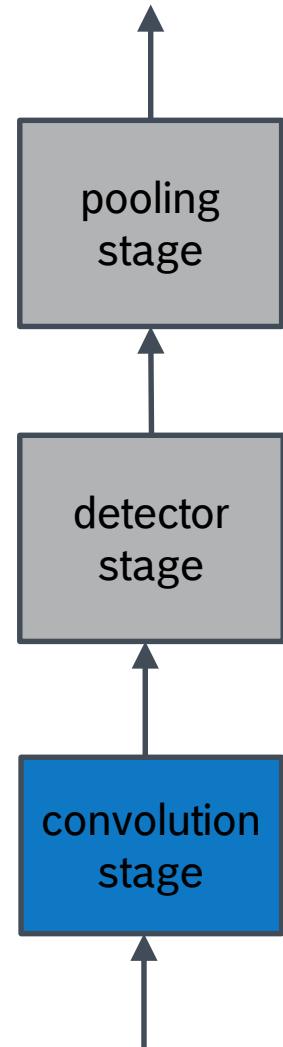
Convolutional Neural Networks

CONV Layer: The Convolution Stage (1D case)

- Exploit grid-like topology in real data (time series, images)
- Replace **fully connected** topology with **convolution** (or **correlation**)



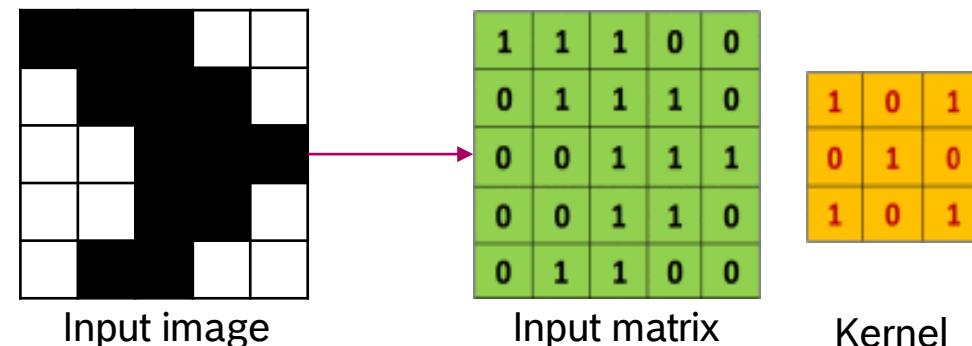
- **Equivariant** to translation, but not to scale and rotation
- Regularization: parameter trimming and tying (infinite prior)
- Biologically inspired (one of the few stories of success)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (2D case)

- **Input:** $W_{\text{in}} \times H_{\text{in}}$ matrix (e.g. binary image)



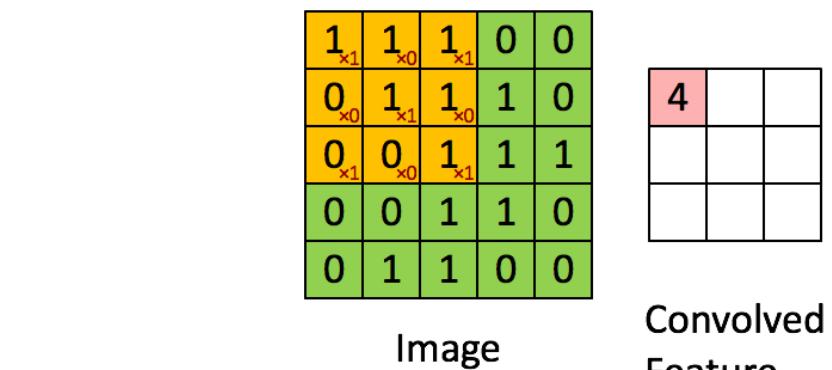
- **Parameters:** $W_k \times H_k$ matrix (filter)

► Operation:

- Slide the kernel over the input matrix
- At each position calculate element wise multiplication
- Calculate the sum of products

► Output:

- $W_{\text{out}} \times H_{\text{out}}$ matrix (activation map)



Source: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Convolutional Neural Networks

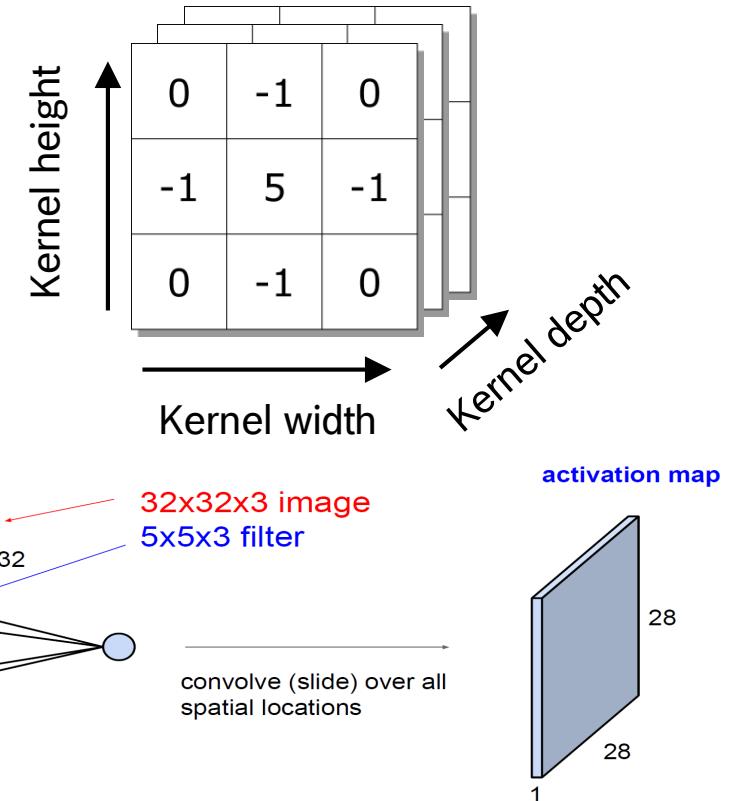
CONV Layer: The Convolution Stage (3D input)

- ▶ **Input:** $D_{\text{in}} \times W_{\text{in}} \times H_{\text{in}}$ tensor (D_{in} stacked matrices)
 - ▶ D_{in} is the depth of the input (e.g. 3 for an RGB image)

- ▶ **Parameters:** $D_{\text{in}} \times W_k \times H_k$ matrix (kernel)
 - ▶ Must have the same depth as the input

- ▶ **Operation:**
 - ▶ Slide the kernel over the width and height of the input
 - ▶ At each position calculate element wise multiplication
 - ▶ Calculate the sum of products
 - ▶ This is still 2D convolution (we do **not** slide along depth!)

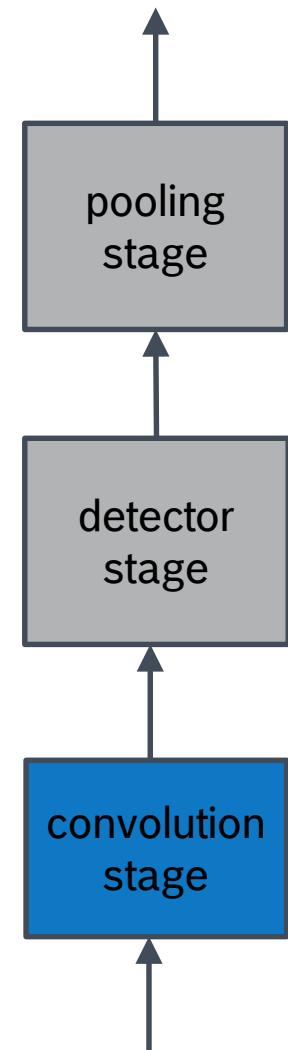
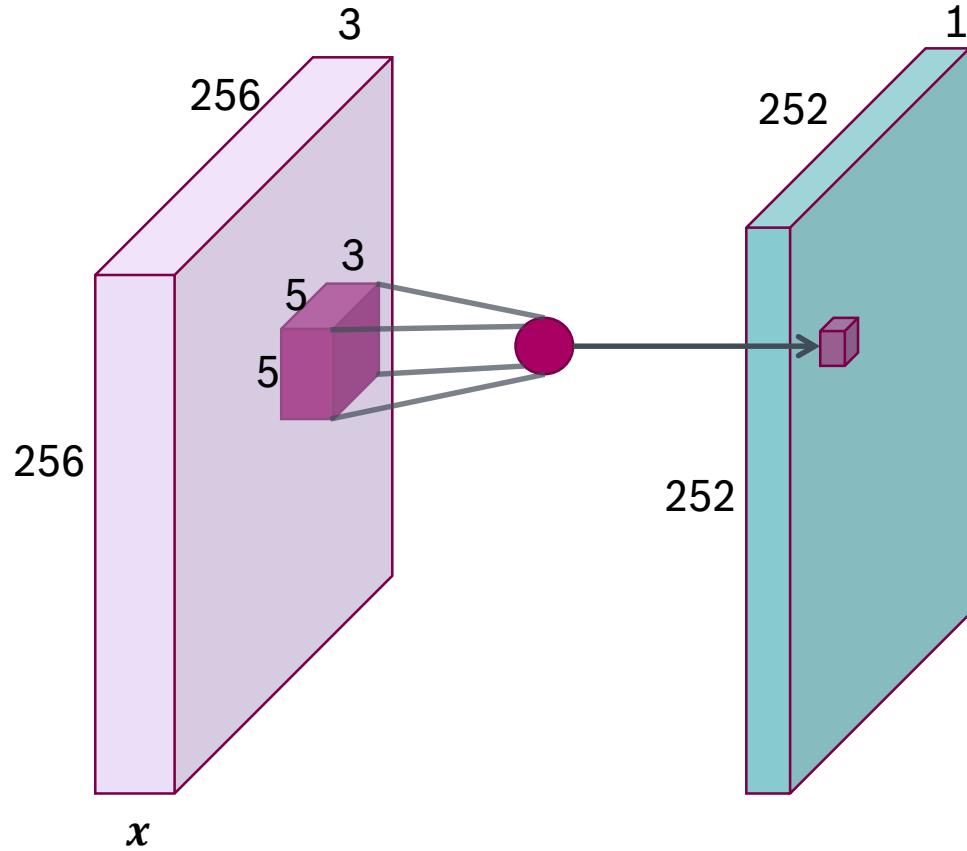
- ▶ **Output:** $W_{\text{out}} \times H_{\text{out}}$ matrix (depth 1)



Source: https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/convolutional_neural_networks.html

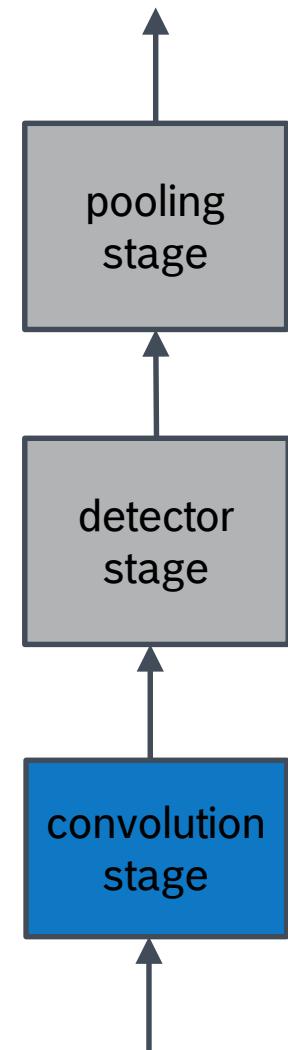
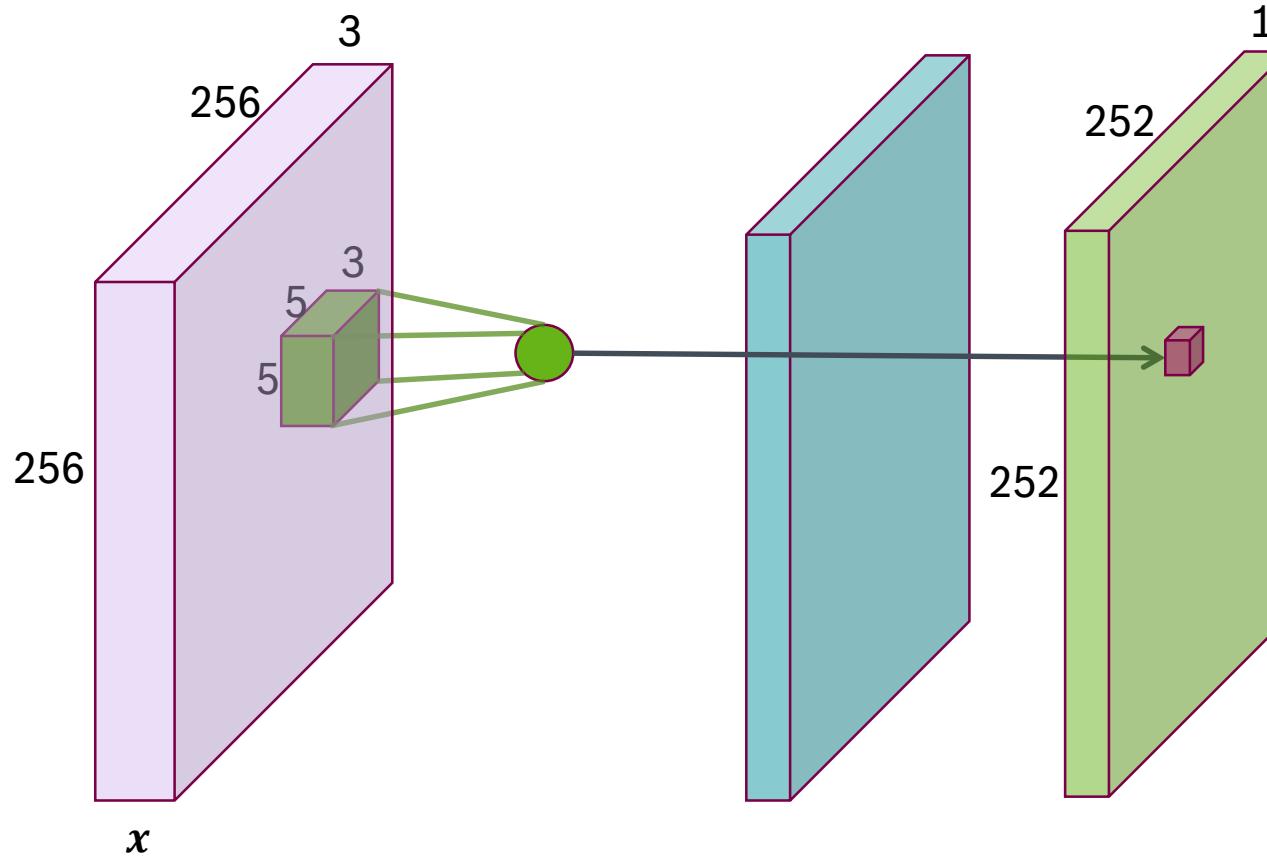
Convolutional Neural Networks

CONV Layer: The Convolution Stage (several kernels)



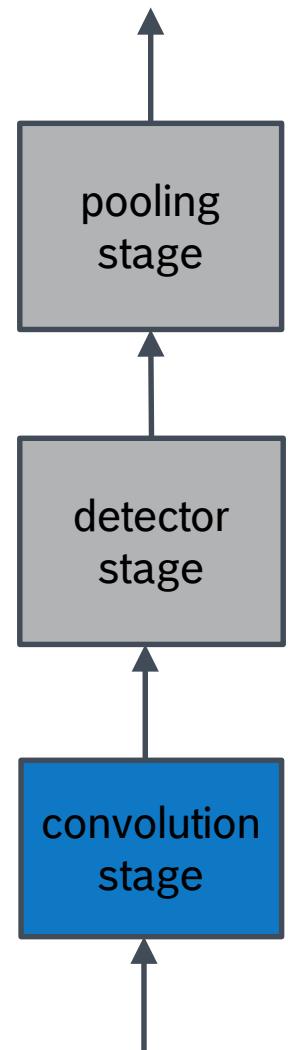
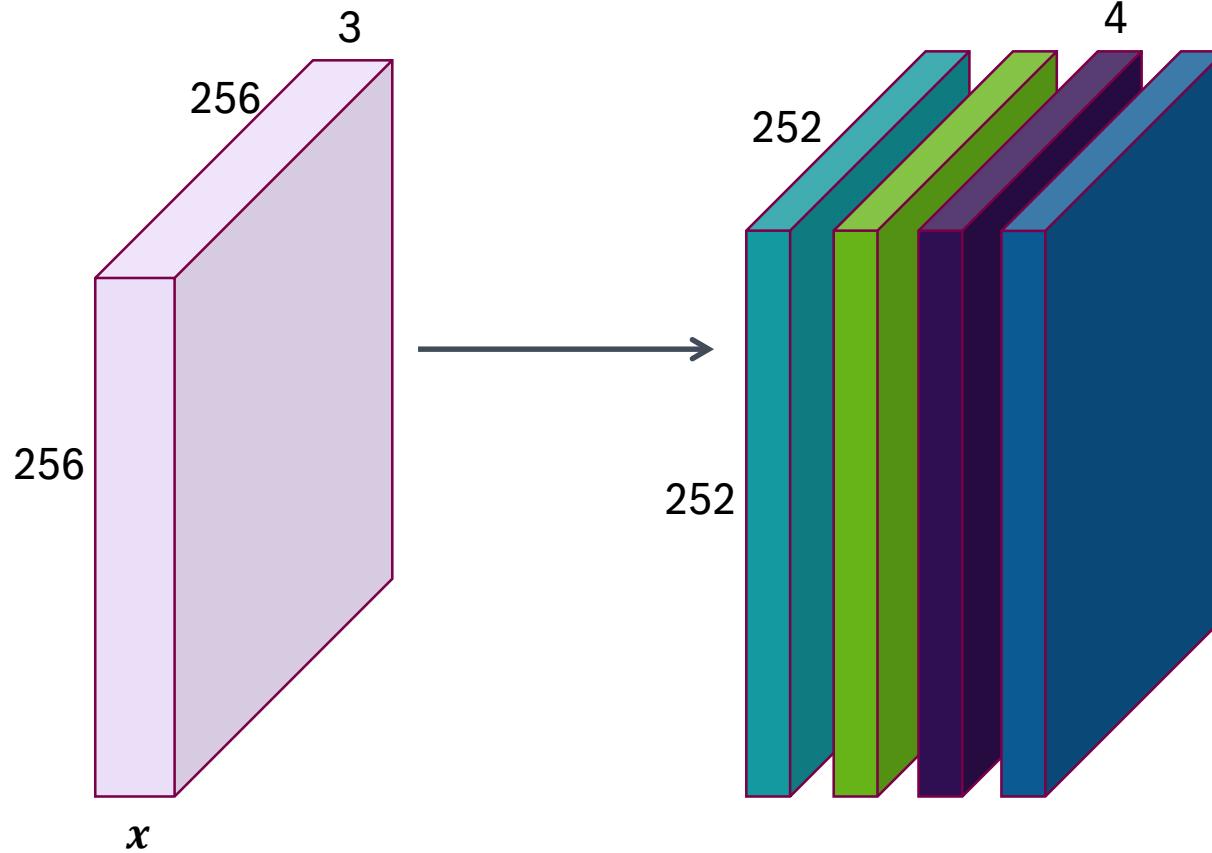
Convolutional Neural Networks

CONV Layer: The Convolution Stage (several kernels)



Convolutional Neural Networks

CONV Layer: The Convolution Stage (several kernels)

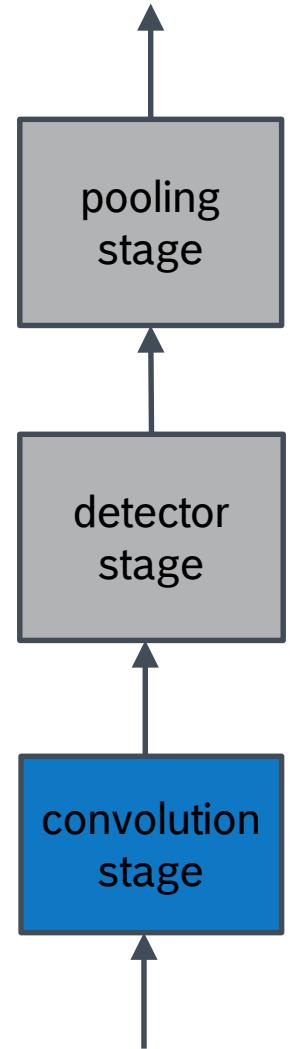


Convolutional Neural Networks

CONV Layer: The Convolution Stage (general case)

- ▶ **Input:** 3D tensor X of size $D_{\text{in}} \times W_{\text{in}} \times H_{\text{in}}$ (D_{in} stacked matrices)
- ▶ **Parameters:** 4D tensor W of size $D_{\text{out}} \times D_{\text{in}} \times W_k \times H_k$ - **kernel bank**
 - ▶ D_{out} 3D tensors (**kernels**)
 - ▶ Each kernel consist of D_{in} 2D matrices (filters)
- ▶ **Output:** 3D tensor Y of size $D_{\text{out}} \times (W_{\text{in}} - W_k) \times (H_{\text{in}} - H_k)$ (D_{out} stacked matrices)
- ▶ **Operation:**
 - ▶ Slide each 3D kernel over the input tensor
 - ▶ At each position calculate element wise multiplication and sum up the products

$$y_{j,x,y} = \sum_{i=1}^{D_{\text{in}}} \sum_{\Delta y=1}^{H_K} \sum_{\Delta x=1}^{W_K} w_{j,i,\Delta x,\Delta y} \cdot x_{i,x+\Delta x-1,y+\Delta y-1}$$



Convolutional Neural Networks

Conv Layer: The Stride Meta-Parameter

► Motivation:

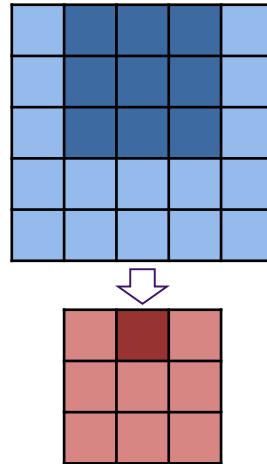
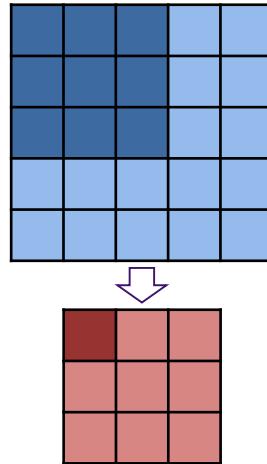
- Reduce the output size
- Reduce computational complexity

► Approach:

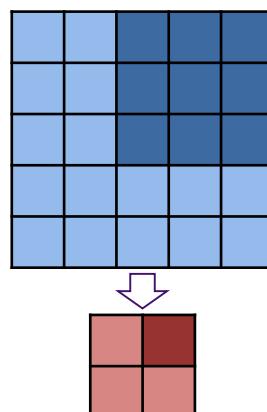
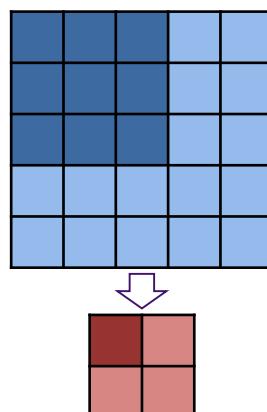
- control the step along each axis when sliding the kernel

► Meta-parameters:

- **horizontal stride**: number of entries we move along the X axis at each step
- **vertical stride**: number of entries we move along the Y axis at each step



Horizontal stride = 1



Horizontal stride = 2

Convolutional Neural Networks

CONV Layer: The Padding Meta-Parameter

► Motivation:

- Avoid losing a few entries at the border
- Keep input/output size ratio independent of the input size (1/stride)

► Approach:

- Add zeros along the border of the input

► Meta-parameters:

- **padding**: number of zeros to add along the border in each slice
- Common values (assume square kernels of odd size):
 - $\frac{w_k - 1}{2}$: **same padding** scheme (preserve input size when stride=1)
 - 0: **valid padding** scheme (only “valid” kernel positions that fit inside the input generate output)

Convolutional Neural Networks

CONV Layer: The Padding Meta-Parameter

► Motivation:

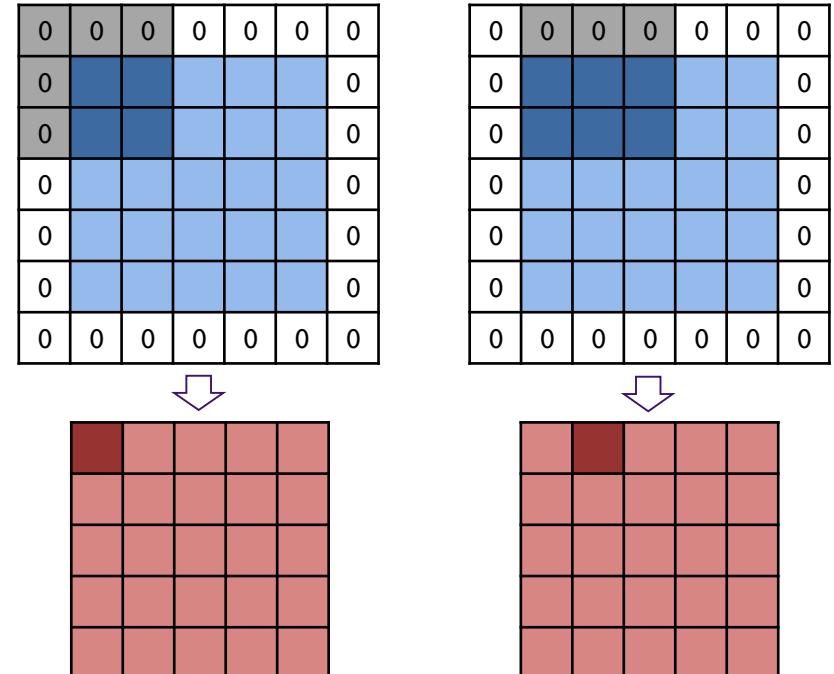
- Avoid losing a few entries at the border
- Keep input/output size ratio independent of the input size (1/stride)

► Approach:

- Add zeros along the border of the input

► Meta-parameters:

- **padding**: number of zeros to add along the border in each slice
- Common values (assume square kernels of odd size):
 - $\frac{w_k - 1}{2}$: **same padding** scheme (preserve input size when stride=1)
 - 0: **valid padding** scheme (only “valid” kernel positions that fit inside the input generate output)



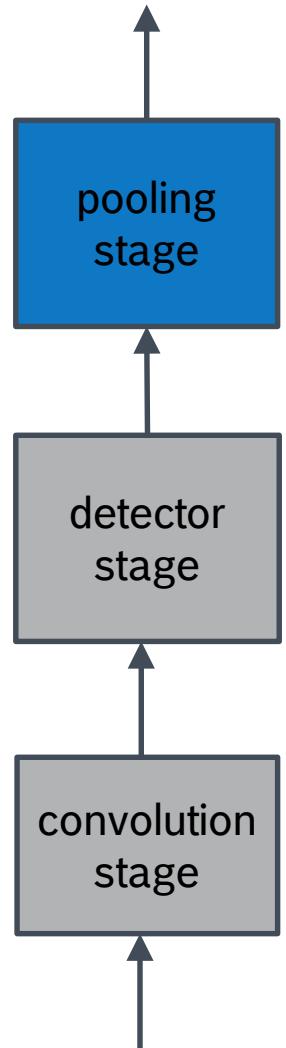
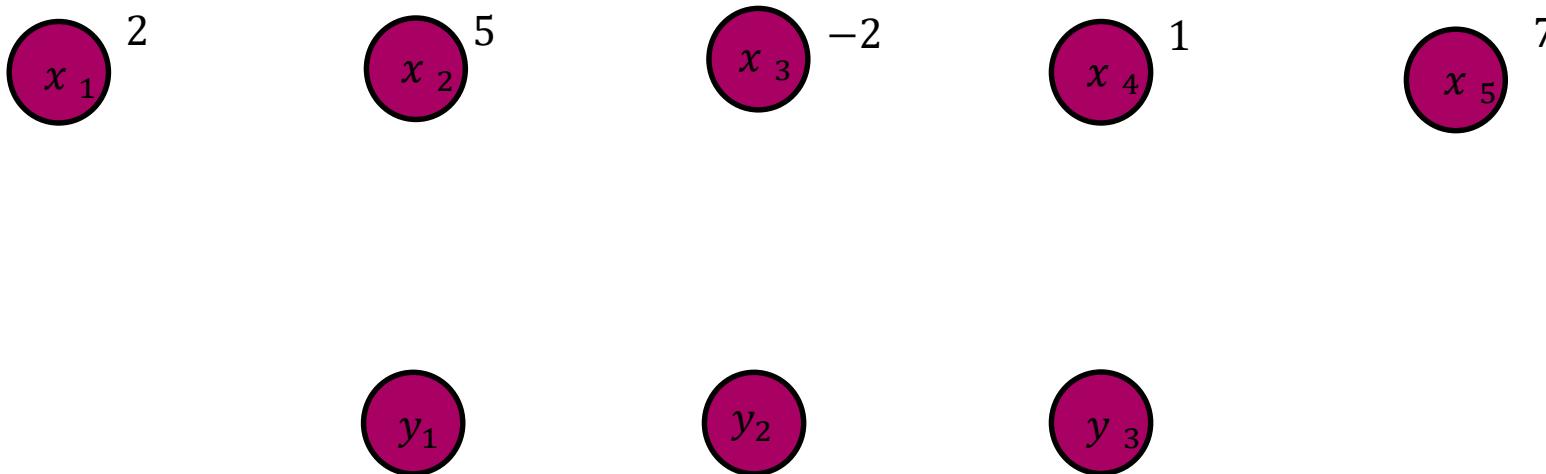
padding = 1

Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

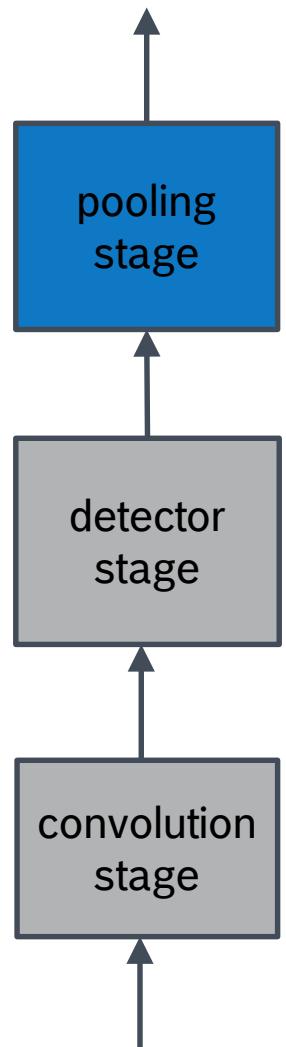
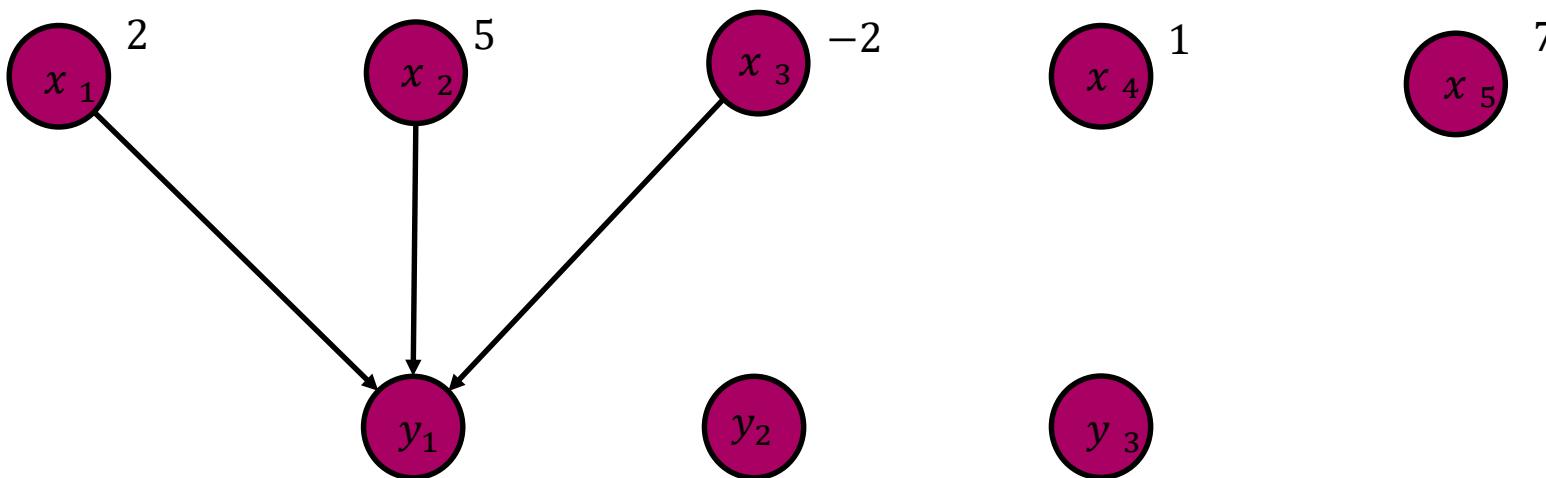


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

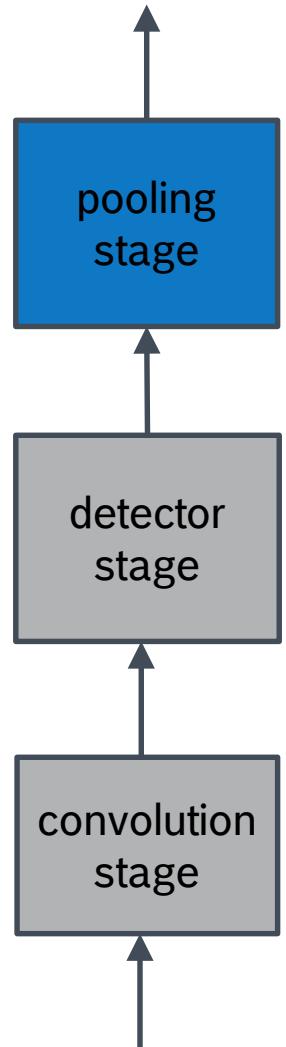
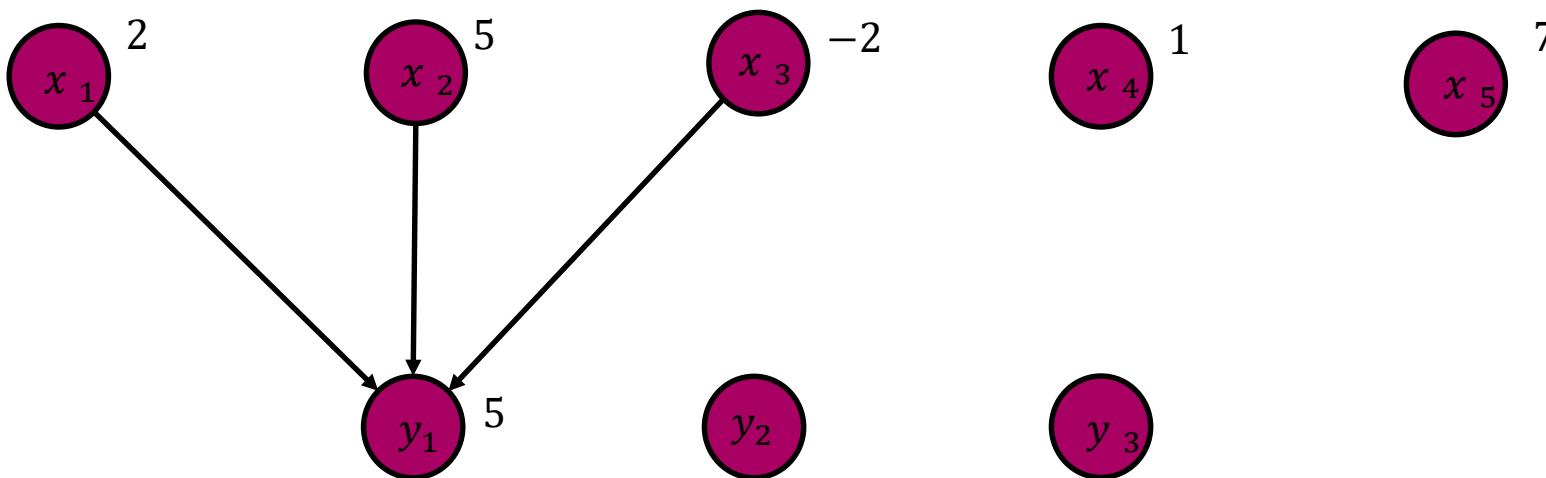


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

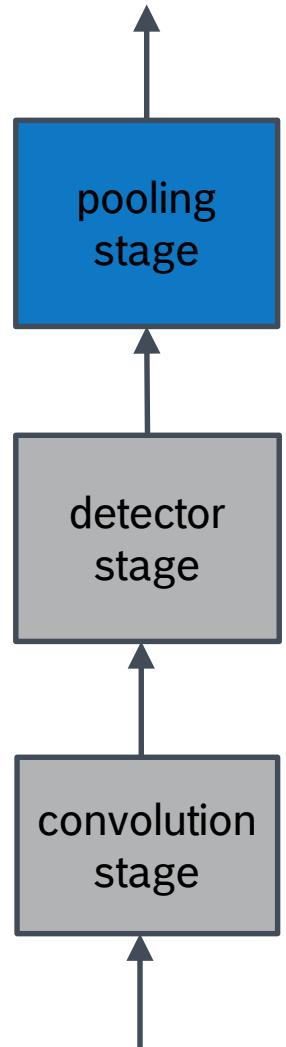
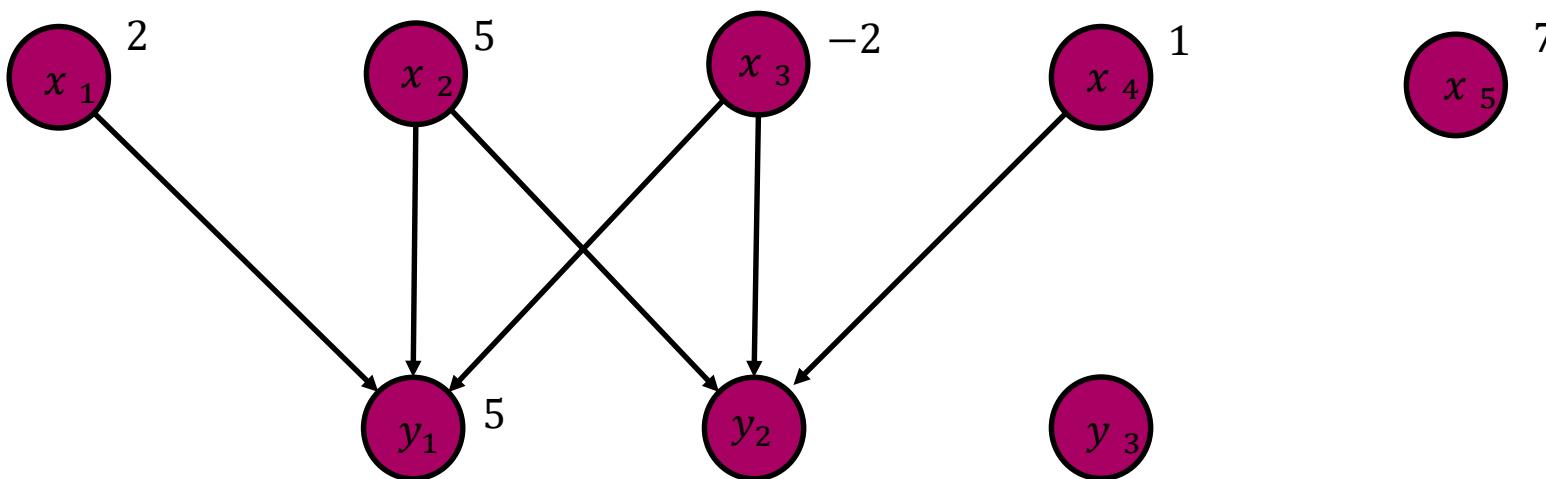


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

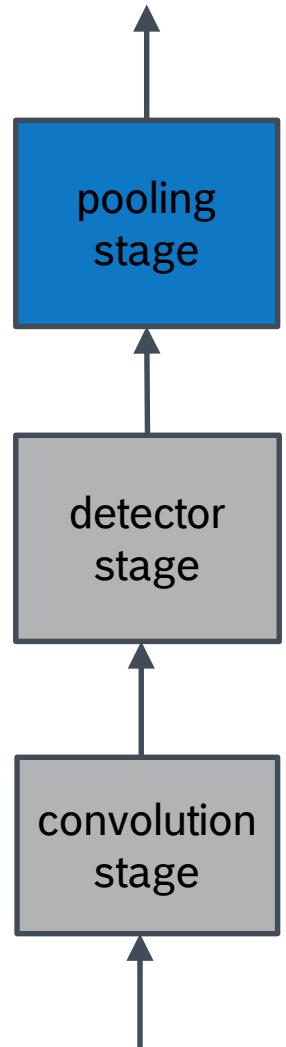
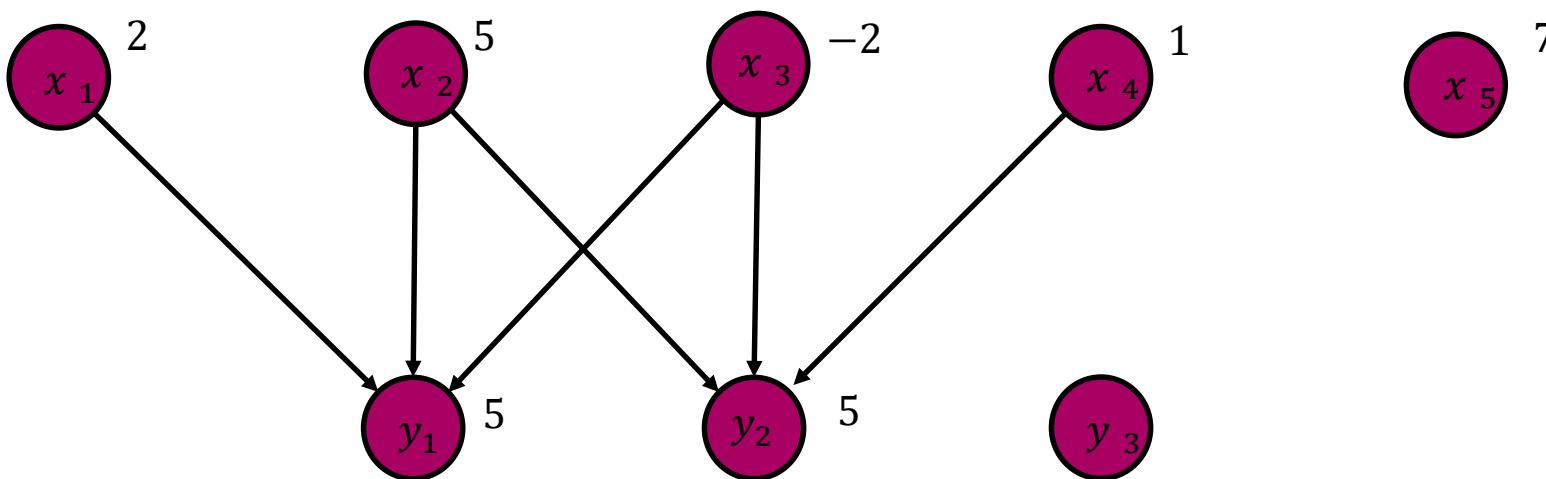


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

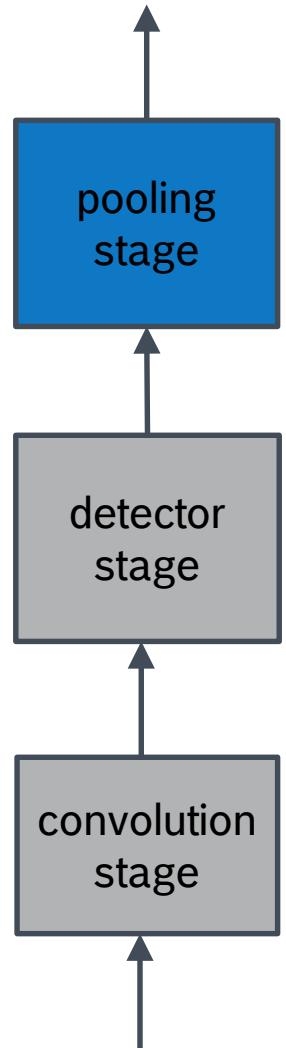
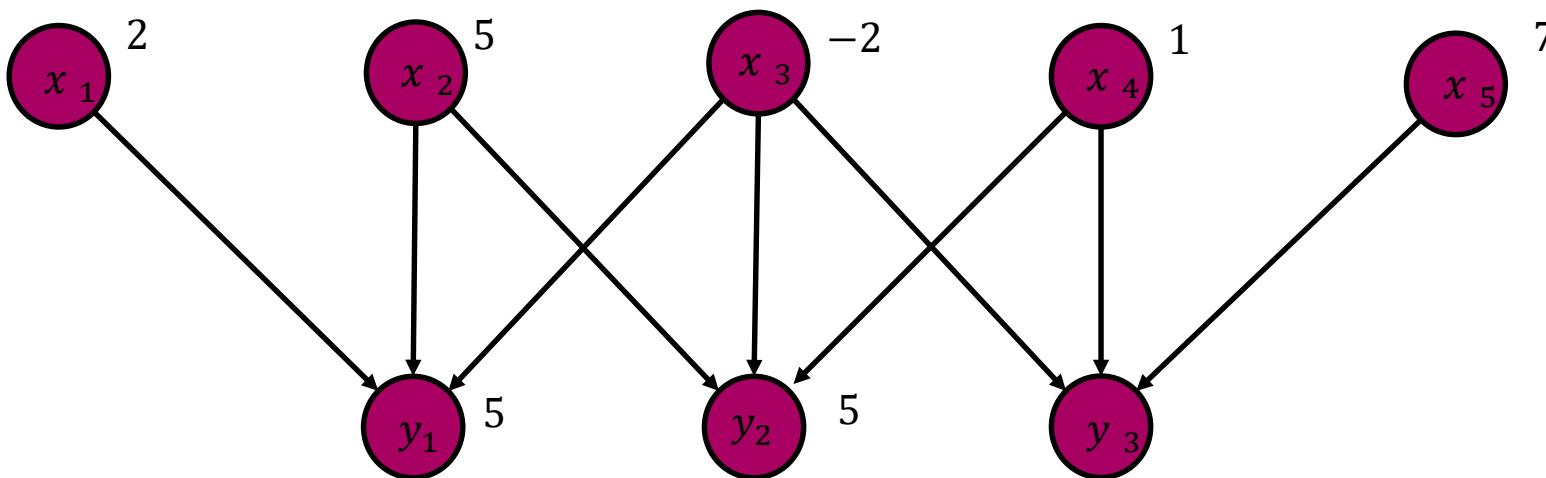


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

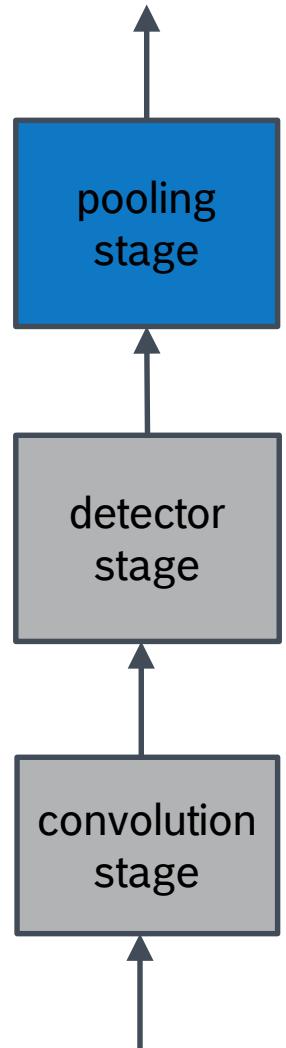
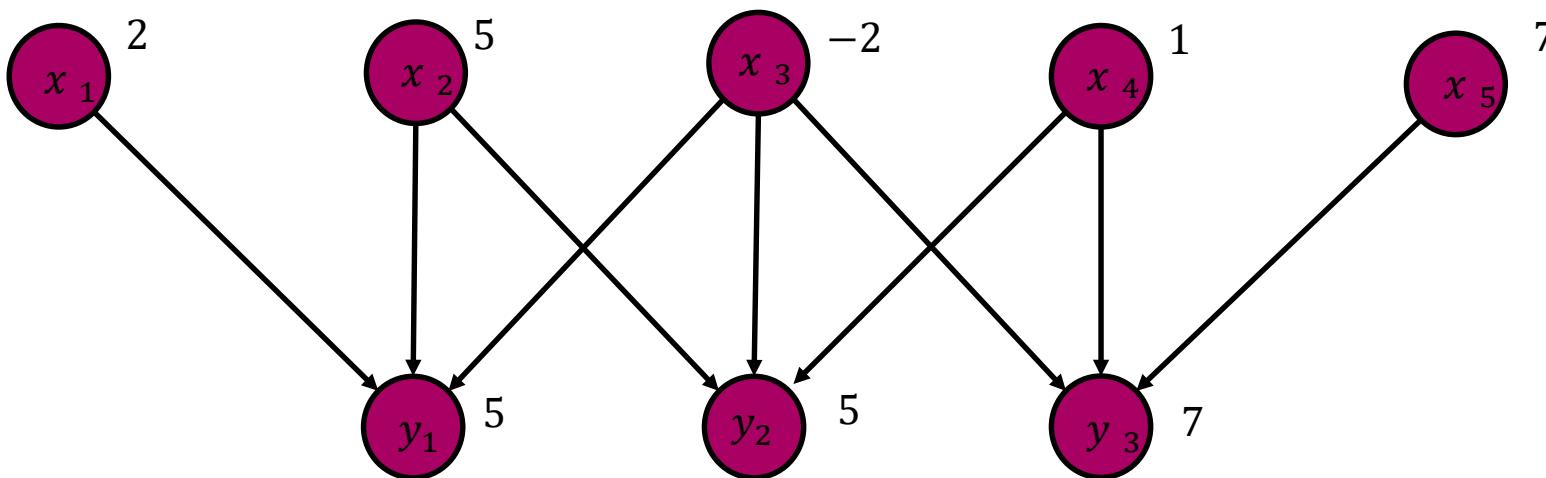


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)

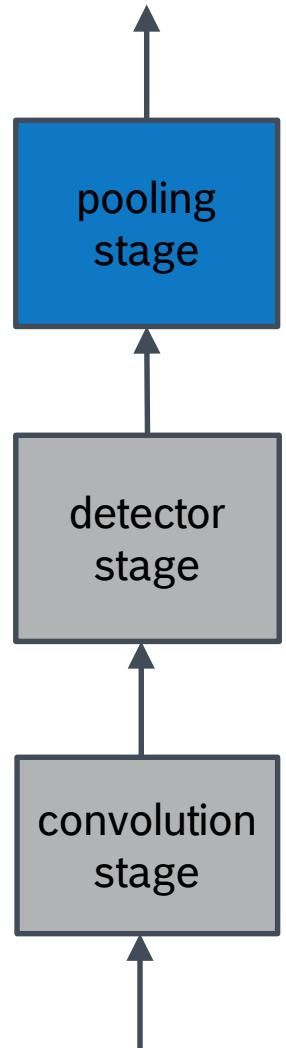
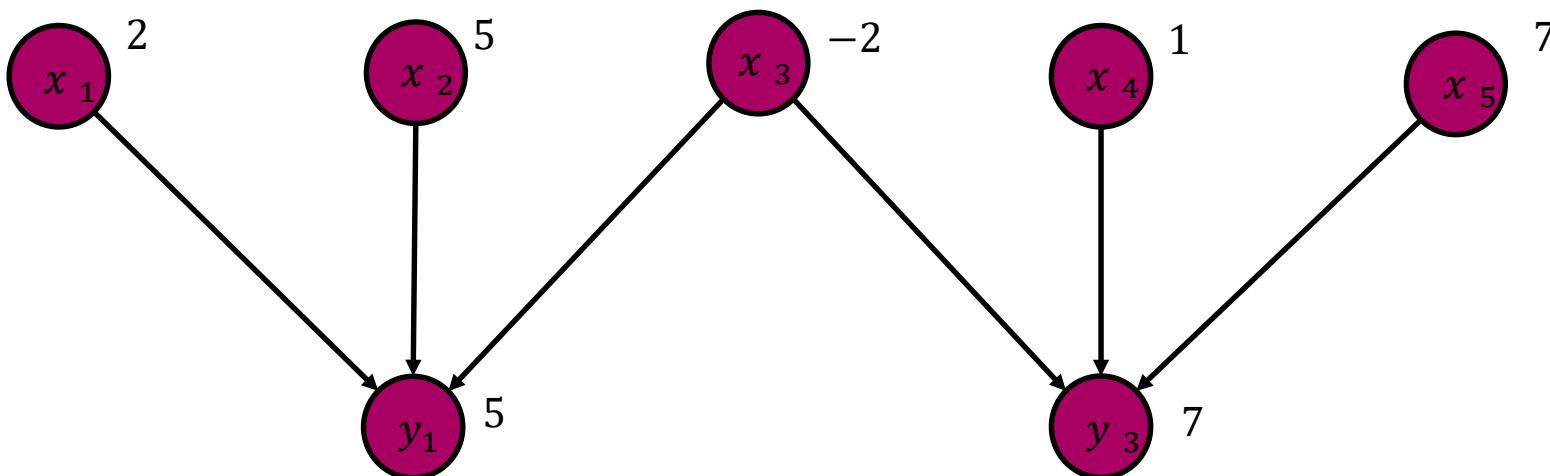


Convolutional Neural Networks

The Pooling Stage (1D case)

► Idea:

- Add **invariance** to small translations
- Collect statistics (max, average, etc.) over neighboring features
- Downsample signal (optionally)
- Handling inputs of variable size (dynamic pooling regions)



Convolutional Neural Networks

The Pooling Stage (2D case)

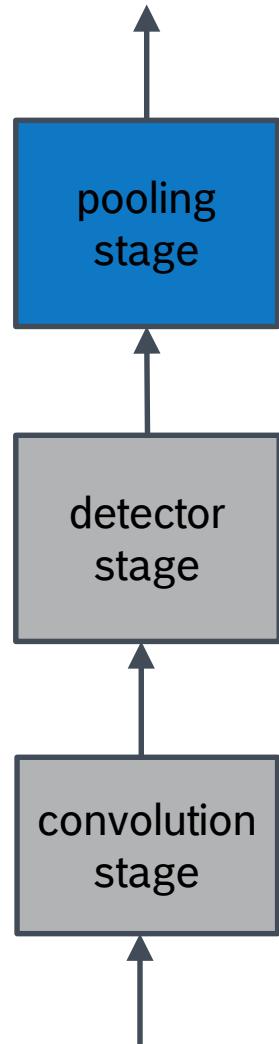
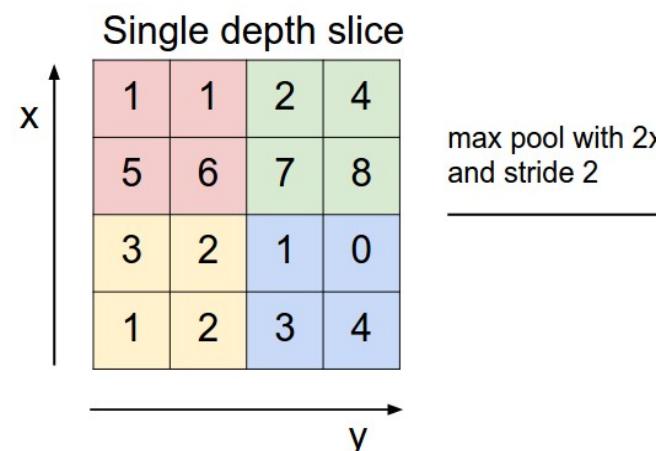
► Operation:

- ▶ Slide window along width and height (with a given stride)
- ▶ Compute statistics inside the window

► Meta-Parameters:

▶ Statistics:

- ▶ **max**: the winner detection takes all (most common in practice)
 - ▶ **average**: smooth out detection noise
- ▶ **window size**: controls spatial invariance
- ▶ **stride**: controls downsampling strength



Source: <http://cs231n.github.io/convolutional-networks/>

Convolutional Neural Networks

The Pooling Stage (general case)

► **Input:** 3D tensor X of size $D_{\text{in}} \times W_{\text{in}} \times H_{\text{in}}$

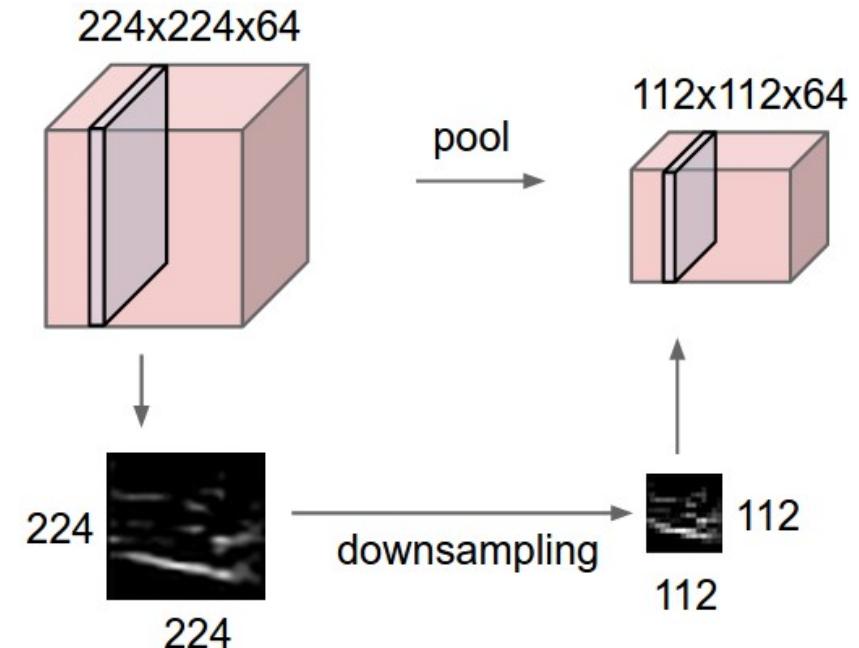
► **Parameters:** none

► **Meta-parameters:**

- statistic (max, avg)
- kernel width/height (W_K, H_K)
- horizontal/vertical stride (s_x, s_y)

► **Output:** 3D tensor Y of size $D_{\text{in}} \times W_{\text{in}} \times H_{\text{in}}$

► **Operation:** perform 2D pooling in each input slice independently

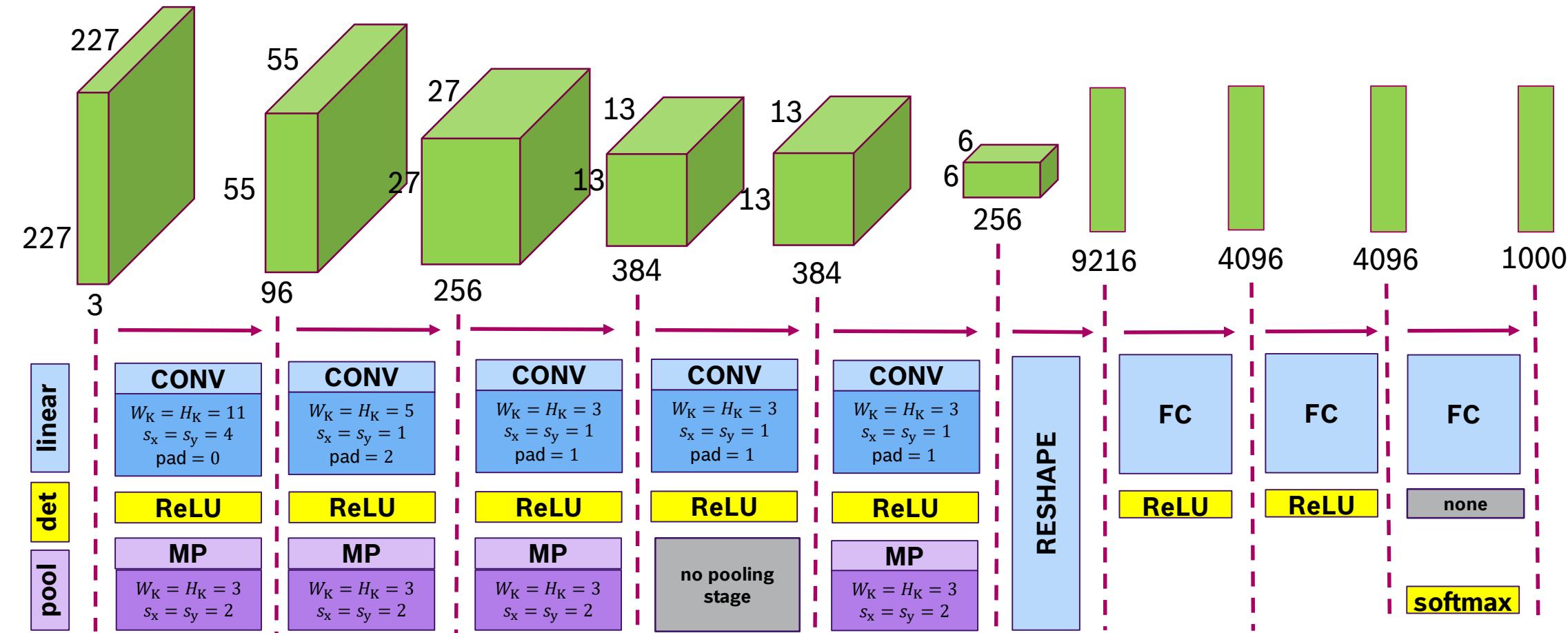


$$y_{j,x,y} = \max_{\substack{0 \leq \Delta x < W_K \\ 0 \leq \Delta y < H_K}} y_{j,s_x \cdot x + \Delta x, s_y \cdot y + \Delta y}$$

Source: <http://cs231n.github.io/convolutional-networks/>

Convolutional Neural Networks

Case Study: Image Classification (AlexNet)



Convolutional Neural Networks

The Softmax Layer: Outputting Probability Distributions

► Issue:

- ▶ How do we convert the output into a Multinoulli distribution?
- ▶ Condition 1: Entries must be between 0 and 1
- ▶ Condition 2: Must sum up to 1

► Solutions:

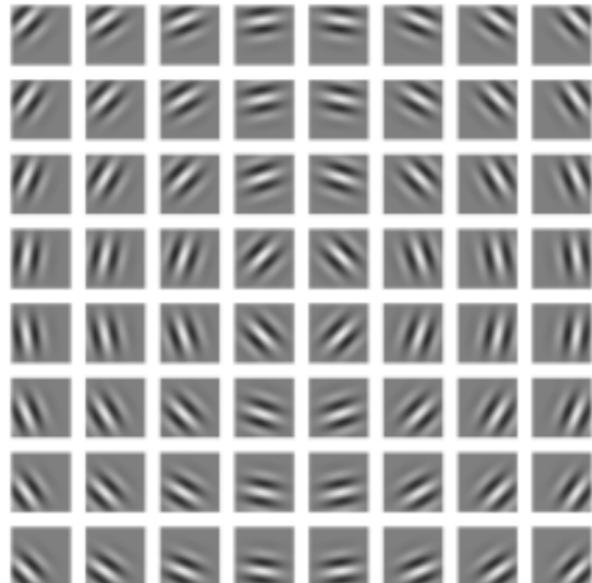
- ▶ For binary classification: sigmoid for one class (condition 1), implicitly represent the other
- ▶ General case: enforce competition using a **softmax layer**:

$$y_j = \frac{e^{x_j}}{\sum_{i=1}^{N_{\text{in}}} e^{x_i}}$$

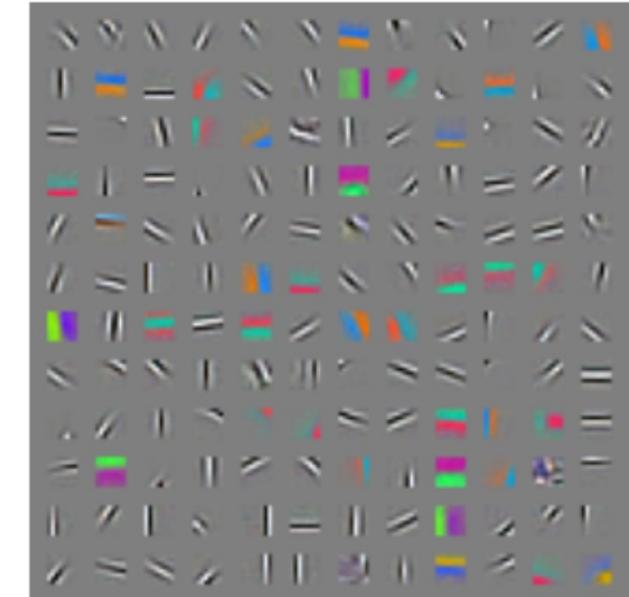
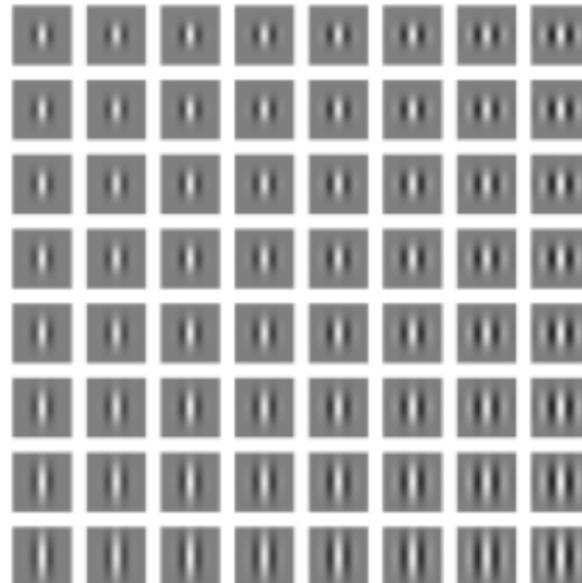
The Path to Deep Learning

Learned Representations

- ▶ Kernels look similar to stimuli for early activation patterns for layers in the brain (V1 area)



Gabor Filters (similar to V1 activators)



Learned kernels in the first convolutional layer

Convolutional Neural Network Recap

► Convolution:

- ▶ Hierarchical local pattern composition (biologically inspired)
- ▶ Uniform equivariant processing across space (and/or time)
- ▶ Can be seen as a strongly regularized fully connected layer
- ▶ Huge reduction in the number of parameters!

► Spatial pooling:

- ▶ Add spatial invariance by inserting pooling layers
- ▶ Reduce feature map sizes (computational advantage)

► Few additional layers (Connectivist Approach):

- ▶ FC: may be missing altogether, see Fully Convolutional Neural Networks (FCNs)
- ▶ Softmax: convert outputs into probability distributions

PERSPECTIVES

Perspectives

Active Research Areas

- ▶ The **statistical** challenge: labeled data is expensive
 - ▶ Supervised transfer learning: fueled the breakthrough (ImageNet)
 - ▶ Unsupervised pre-training: not that successful! Why?
 - ▶ Unsupervised/weakly supervised learning
 - ▶ Active learning
- ▶ The **computational** challenge: inference is expensive
 - ▶ Model compression/speedup (weight/response quantization)
 - ▶ Implicit generative models (e.g. GANs)
- ▶ The **optimization** challenge: slow/lack of convergence needs tuning
 - ▶ More or less the same algorithms as 3 decades ago! (SGD family)
 - ▶ Second order optimization (e.g. Hessian-Free optimization)

REFERENCES

Deep Learning for Computer Vision

References

- [1] Goodfellow, I. and Bengio, Y. and Courville, A., “*Deep Learning*”, MIT Press, 2016, <http://www.deeplearningbook.org/>
- [2] Bishop, C. M. “Pattern Recognition and Machine Learning”, Springer, 2006
- [3] Murphy, K.P. “*Machine Learning: a Probabilistic Perspective*”, MIT Press, 2012
- [4] Fei-Fei, Karpathy & Johnson, *Lecture Notes*, 2016, <http://cs231n.stanford.edu/slides/2017/>

CNN ARCHITECTURES

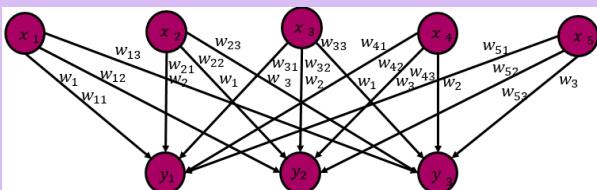
Szabolcs Pável
(Ștefan Máthé)

RECAP

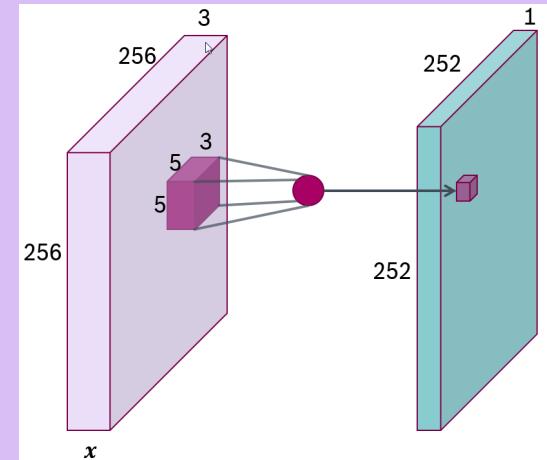
Recap

Basic layer types

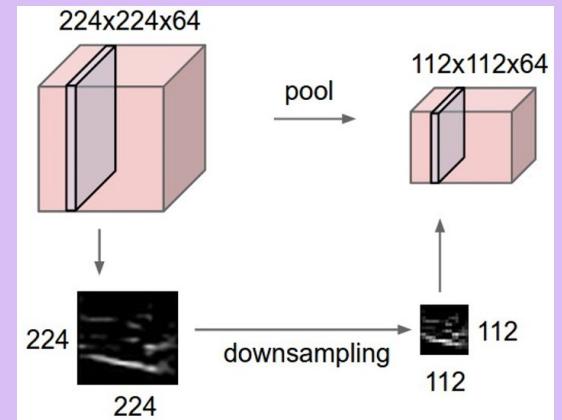
► Fully connected layer



► Convolution layer

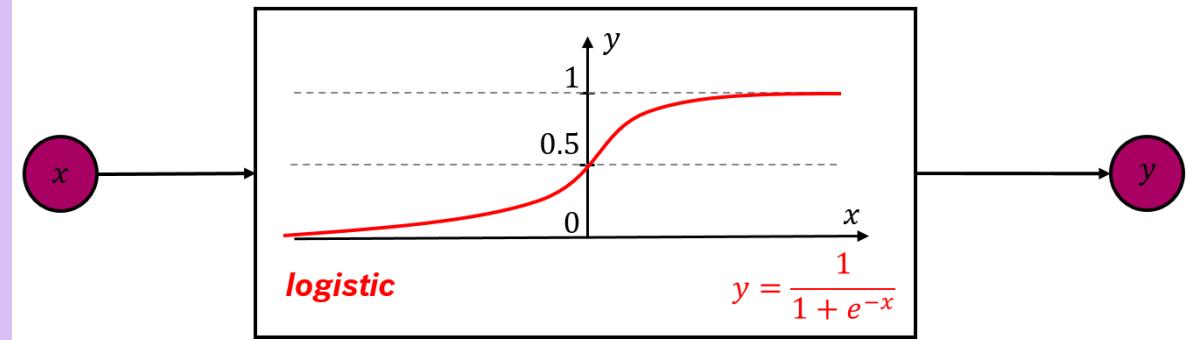


► Max-pooling layer

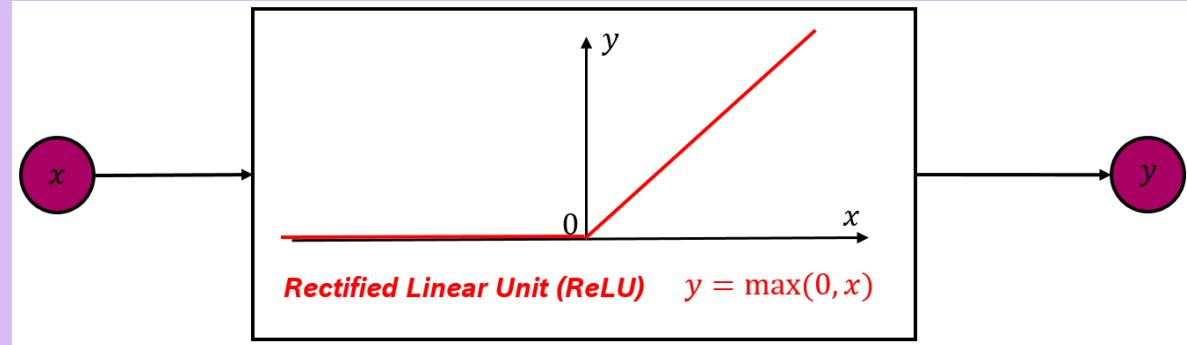


Recap Non-linearities

► Logistic (Sigmoid)

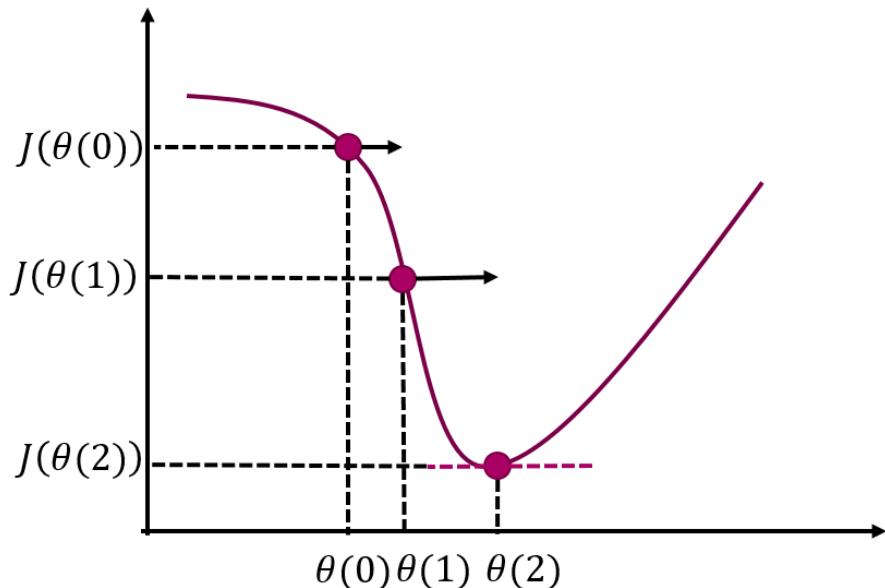


► ReLU (Rectified Linear Unit)

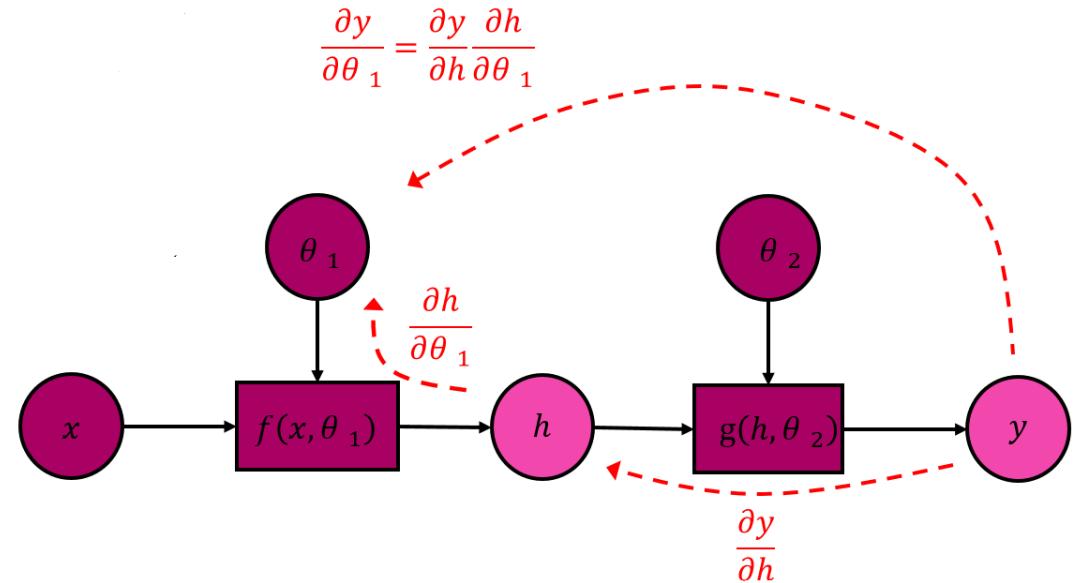


Recap

Optimization, gradient descent, and backpropagation



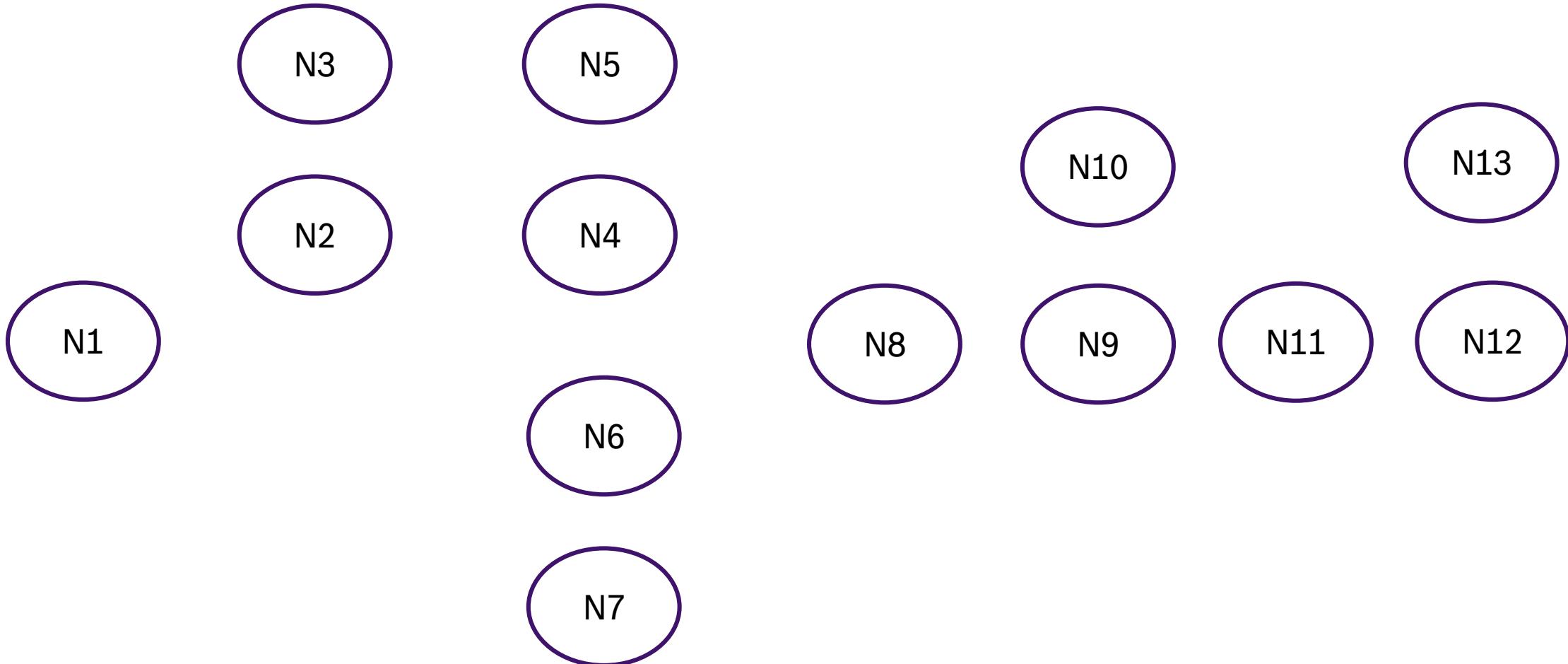
$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



$$J(\boldsymbol{\theta}) \cong J(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \nabla J(\boldsymbol{\theta}_0)$$

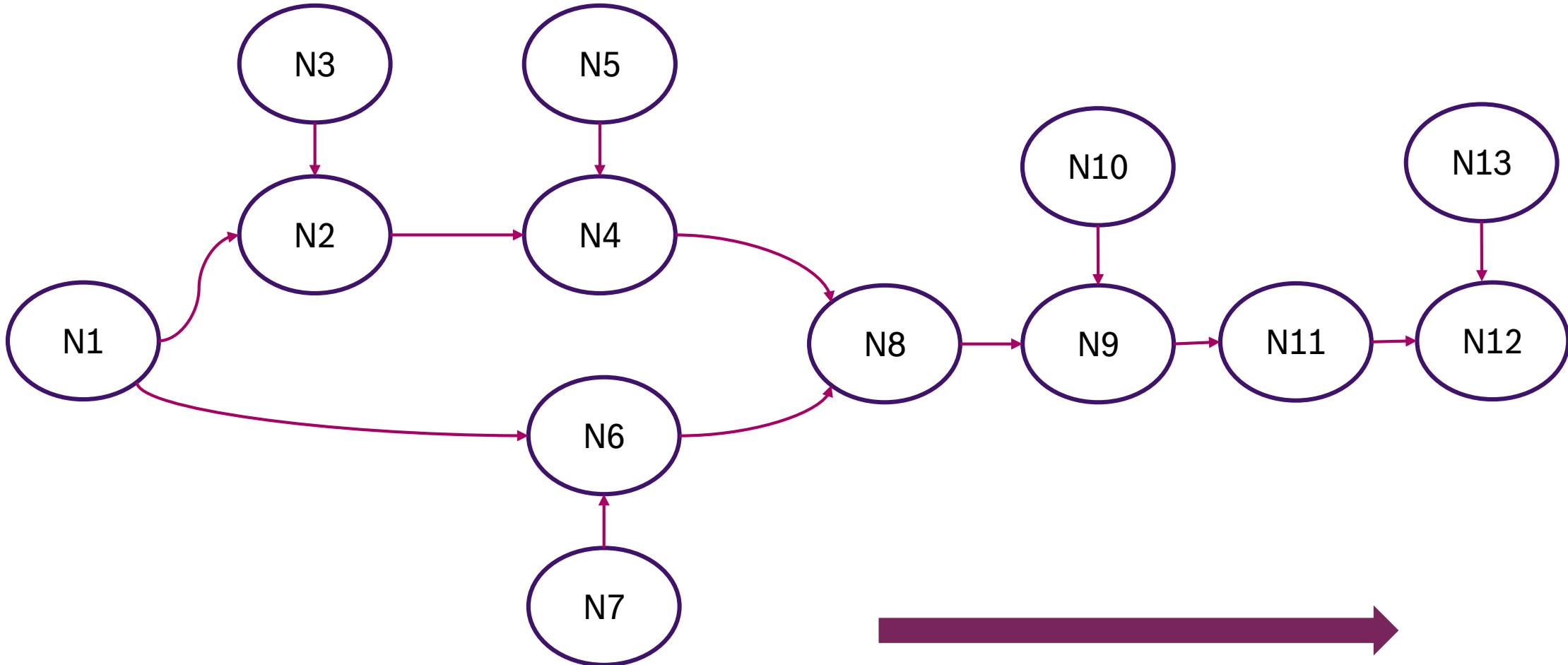
Recap

Deep learning as differentiable computational graphs



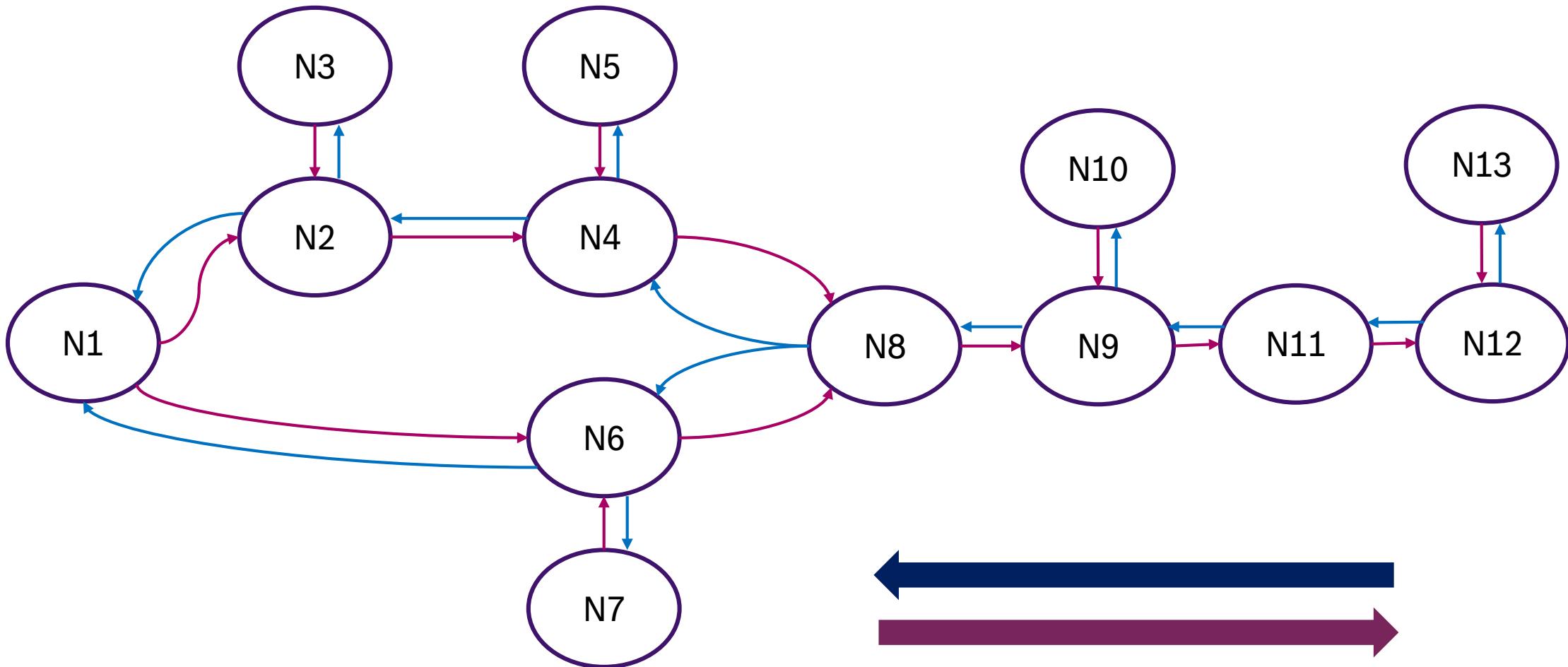
Recap

Deep learning as differentiable computational graphs



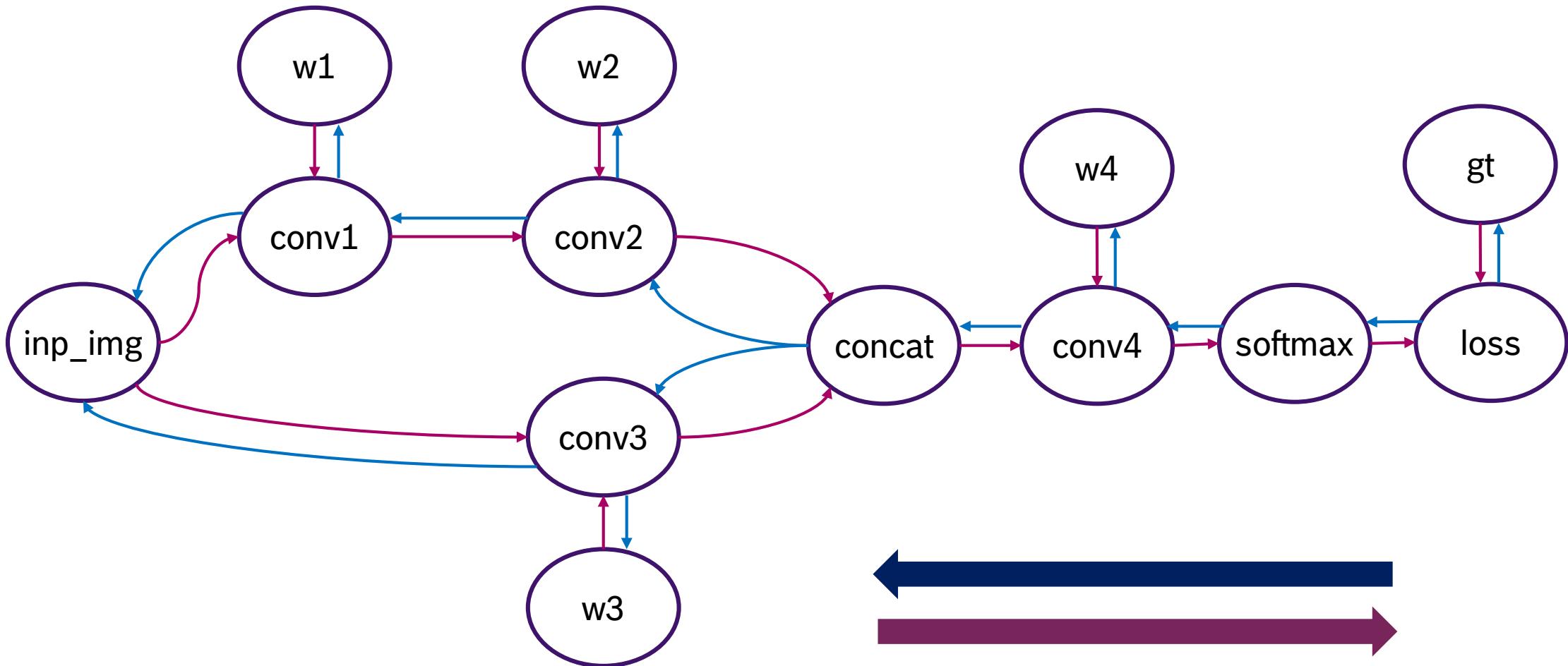
Recap

Deep learning as differentiable computational graphs



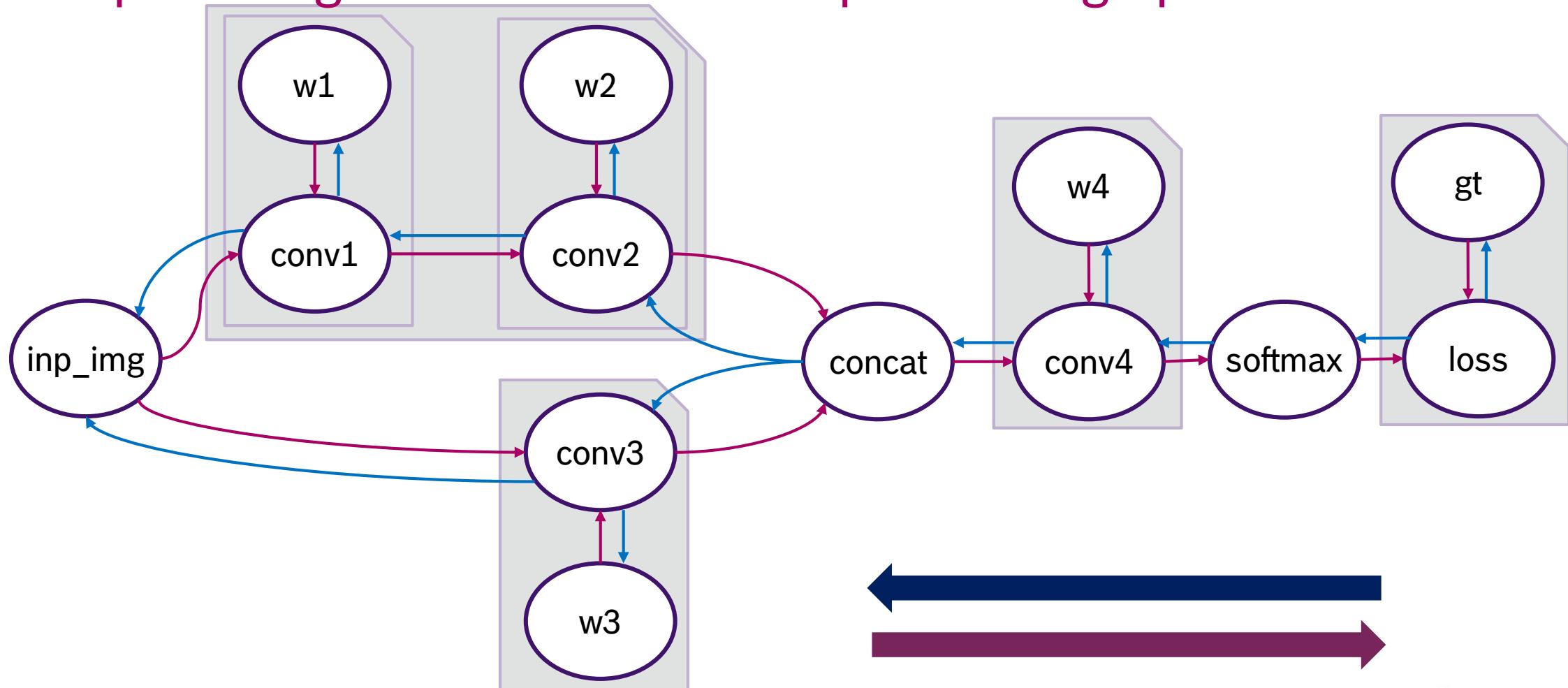
Recap

Deep learning as differentiable computational graphs



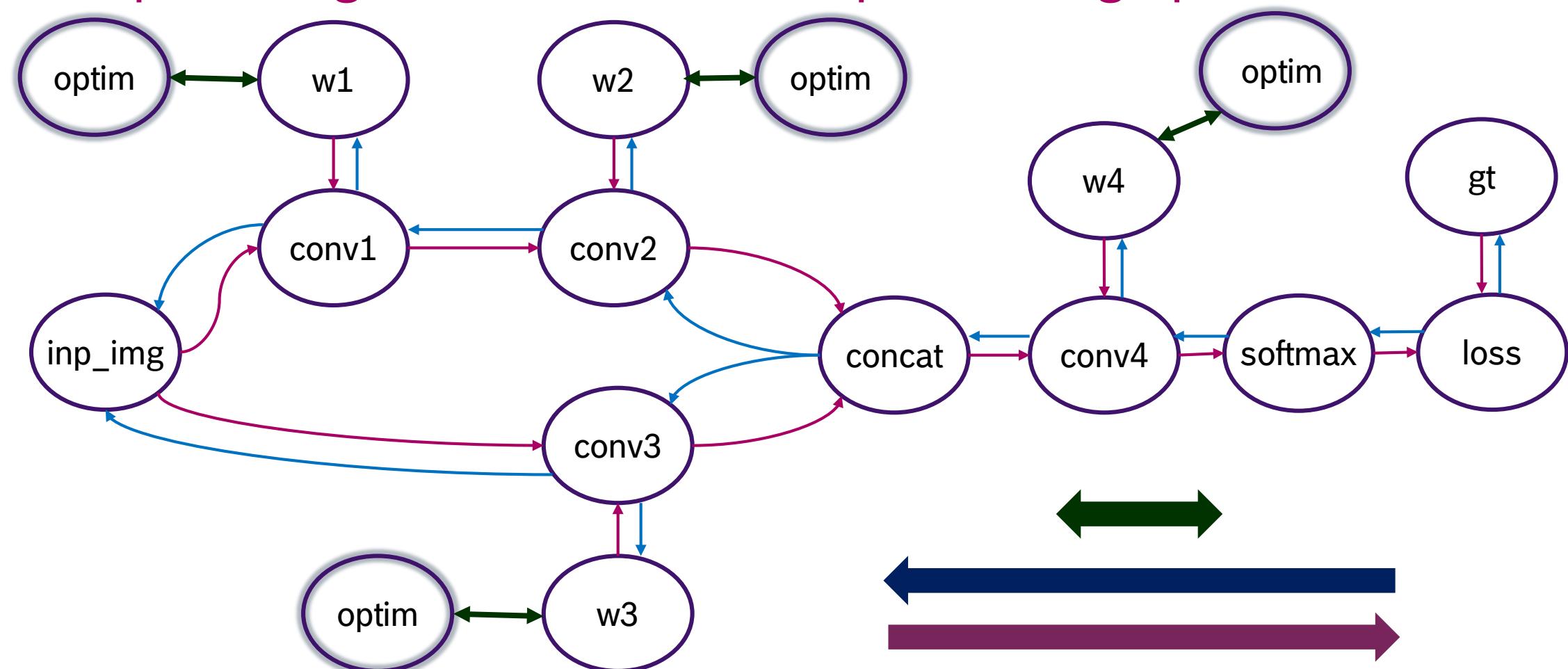
Recap

Deep learning as differentiable computational graphs



Recap

Deep learning as differentiable computational graphs

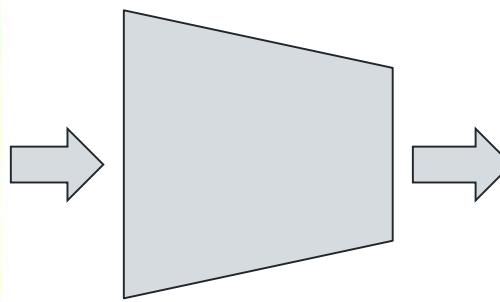


TASKS AND ARCHITECTURES

IMAGE CLASSIFICATION

Image classification

Task definition



CNN

c	\hat{y}_c
cat	0.8
dog	0.1
house	0.01
car	0.05
plane	0.04

→
Most
probable
class (cat)

We can minimize:

- NLL:

$$L(\theta) = -\log(p_{model}(y|x, \theta))$$

- Cross-entropy (Multinoulli):

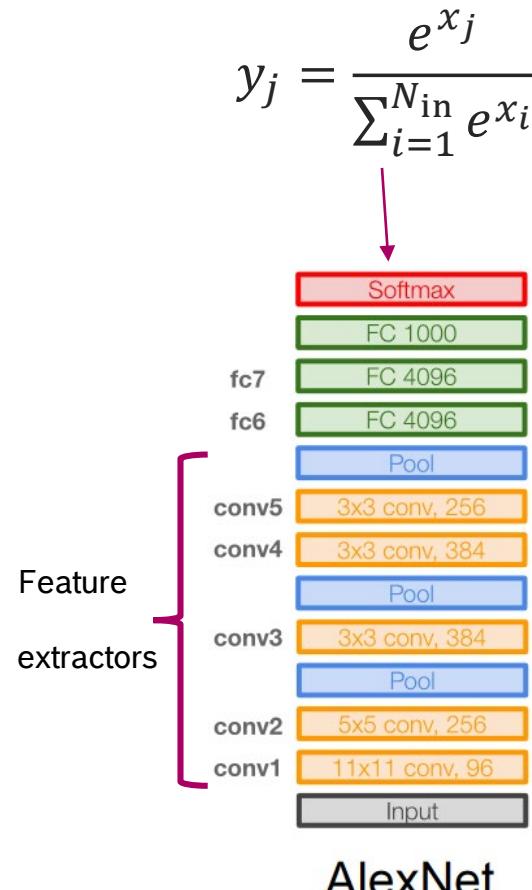
$$L(\theta) = - \sum_c y_c \log(\hat{y}_c)$$

Image classification

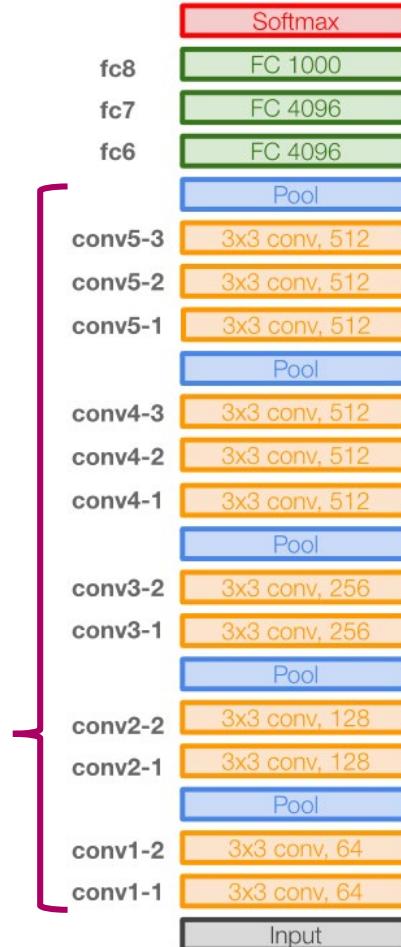
Typical architectures – AlexNet & VGG

About

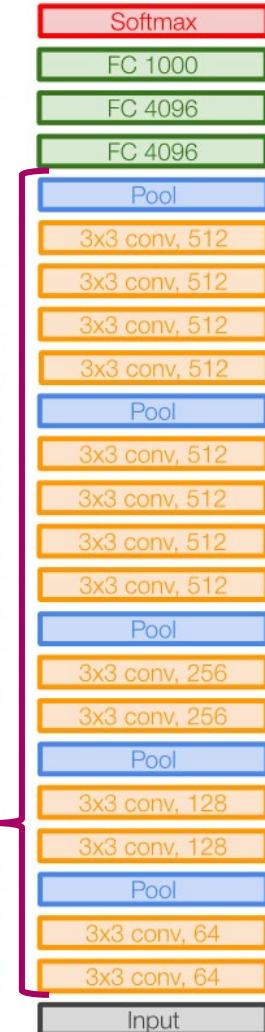
- AlexNet and VGG represent the *prototypical* image classification architectures
- They consist from a **feature extractor** and some **decision layers** on top of that
- The feature extractor gradually reduces the initial height / width dimensions, while increasing the level of abstraction



AlexNet



VGG16



VGG19

Image classification

Case study – AlexNet

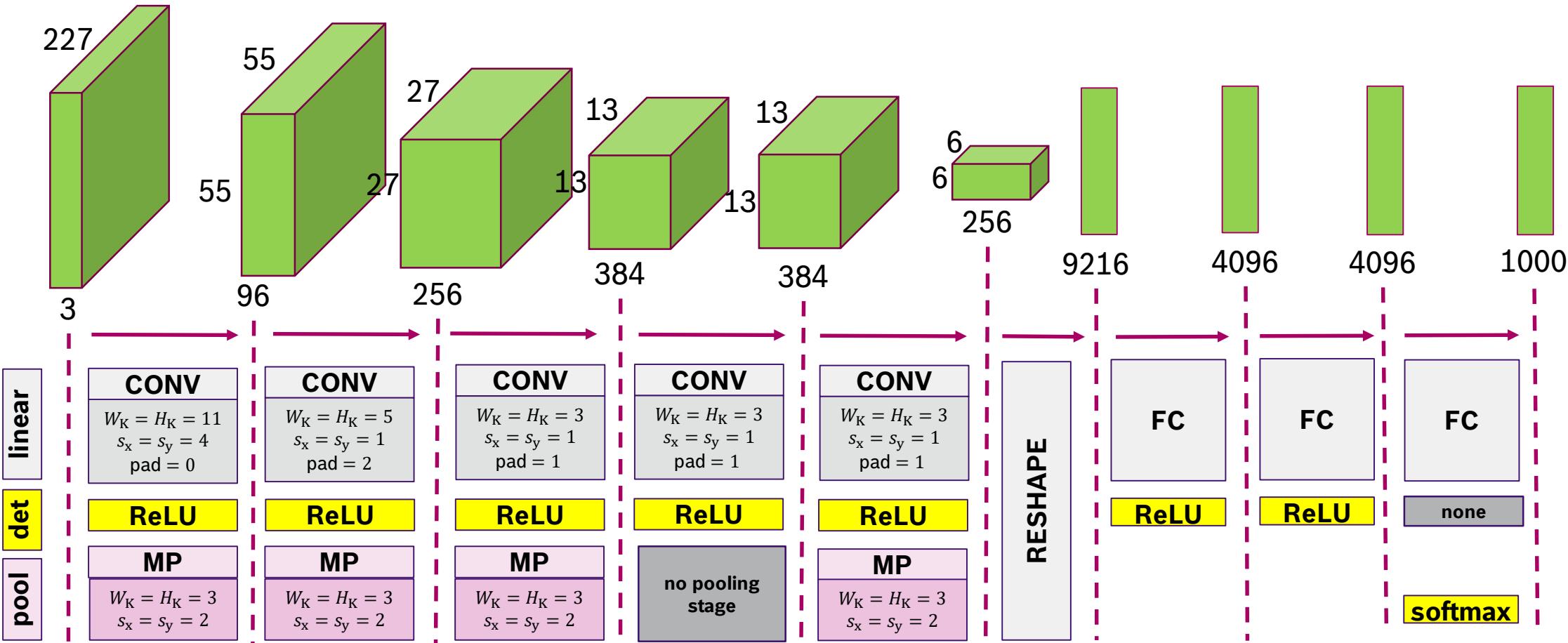


Image classification

Typical architectures – AlexNet & VGG

Issues

- Vanishing gradients -> lower layers are difficult to train
- Unfeasible thus to have too many levels

Possible solution

- Step-by-step training -> add a new level only after you trained the previous

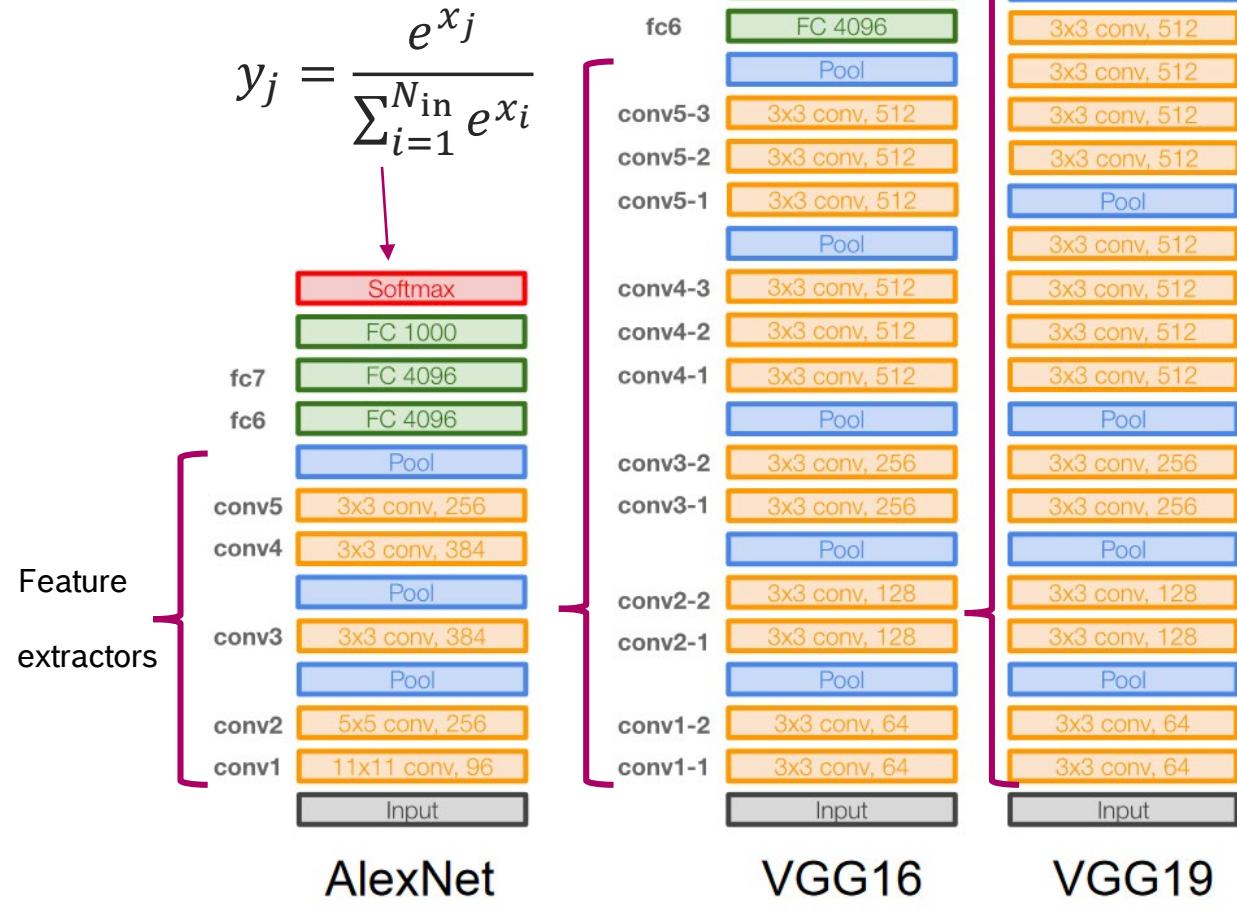


Image classification Block based architectures – GoogLeNet

About

- GoogLeNet is a good example of a modularized CNN architecture
- Each module computes multiple convolution types
- The loss is computed at **multiple levels** for stronger gradients
- With careful output dimension management for each module, we can get quite deep models

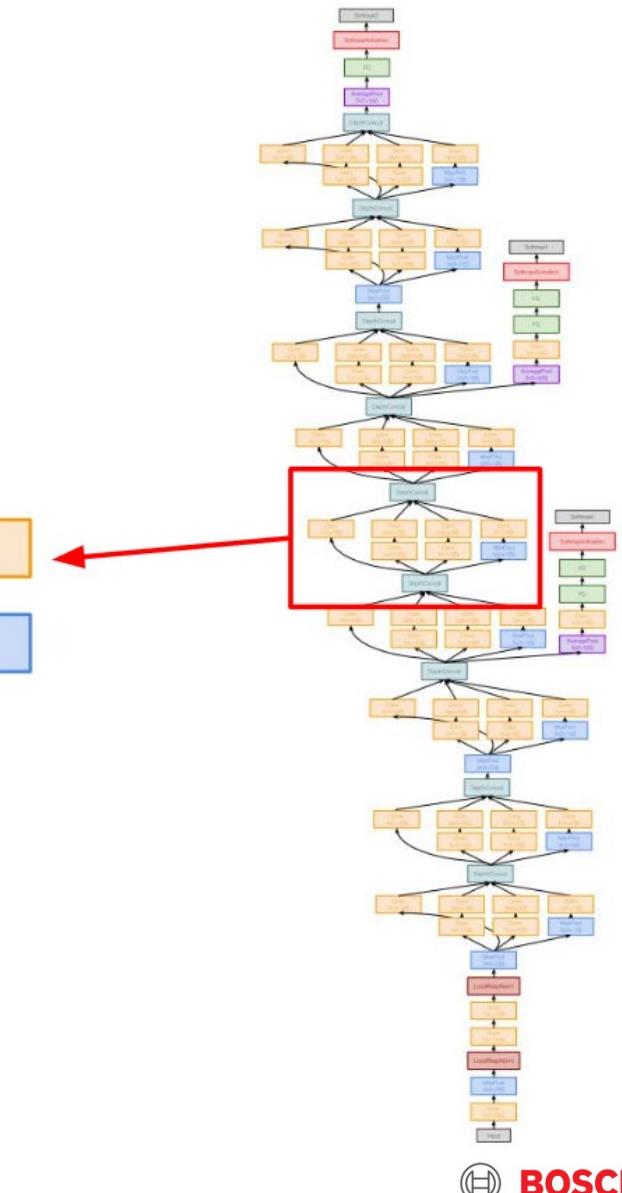
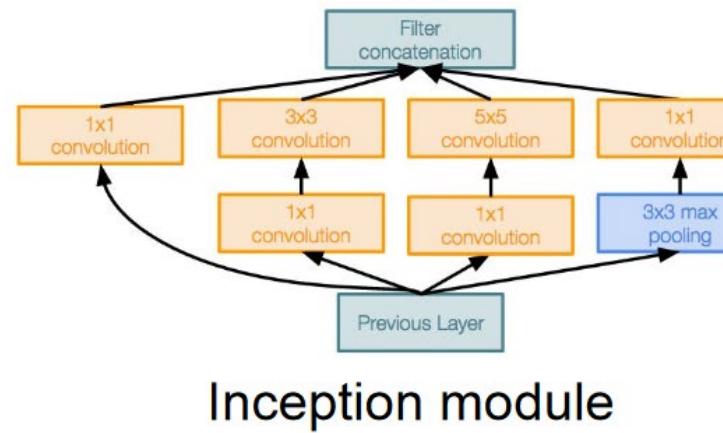


Image classification

Block based architectures – ResNet

About

- [ResNet](#) tackles vanishing gradients by adding **skip connections** between modules
- The resulting **residual blocks** are chained together, the output of each block is an **additive transformation** of the input

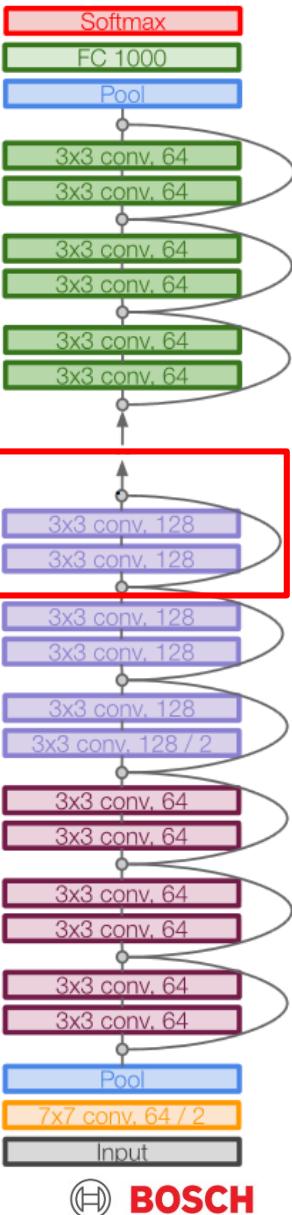
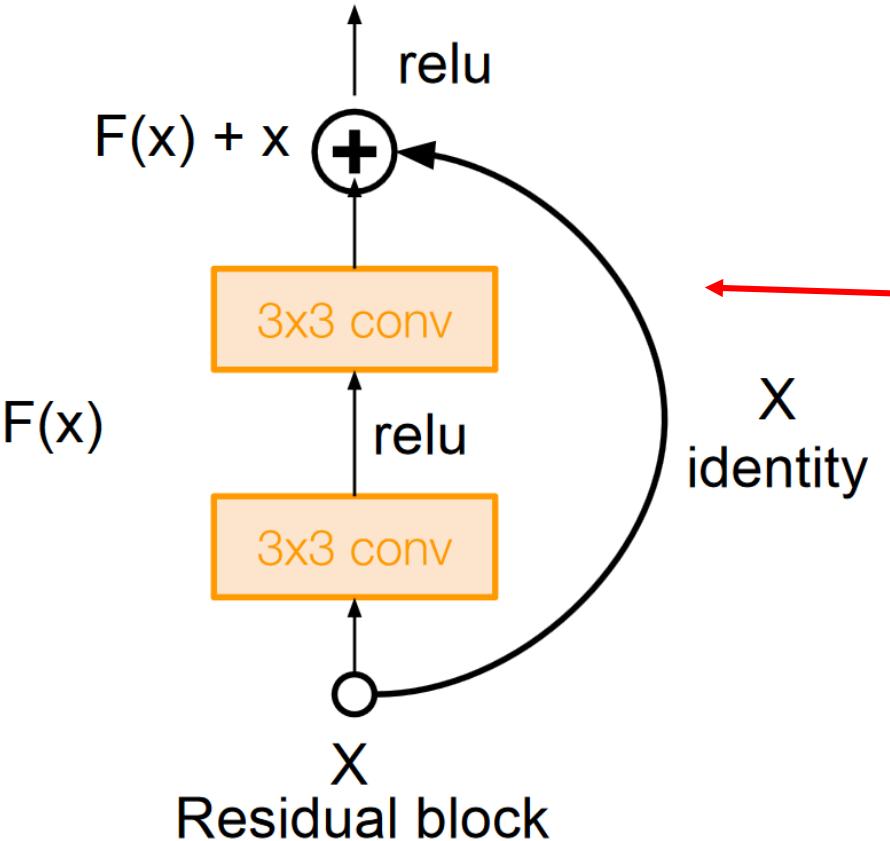
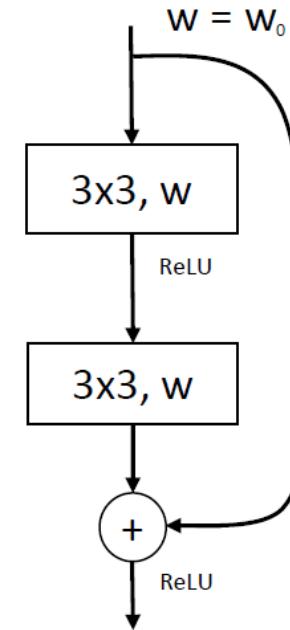


Image classification

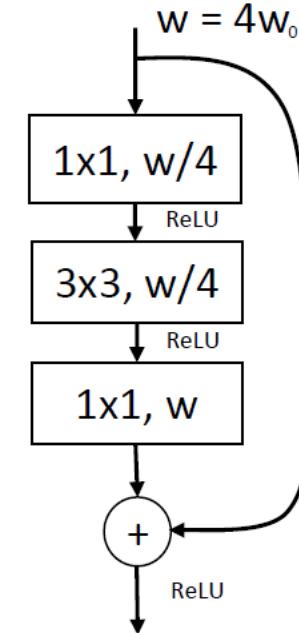
Block based architectures – ResNet bottleneck

About

- The residual block with **bottleneck** reduces the number parameters -> we can get even deeper models
- Based around the **information compression** assumption / theory



- Normal residual block
- better behaved gradients
- faster training



- Residual block with **bottleneck**
- reduces computational burden

Image classification

Training problem

Issue

- No matter how good our gradients are, training huge convolutional neural networks with a **simple first order optimization method** (gradient descent) still **requires large amounts of training data**
- For most tasks, there not much labeled data available
- **Manual labeling is an expensive process!**



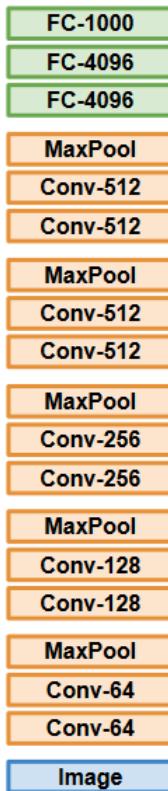
Possible solution:

- Feature extractor **pre-training**, a.k.a. **transfer learning**

Image classification

Transfer learning

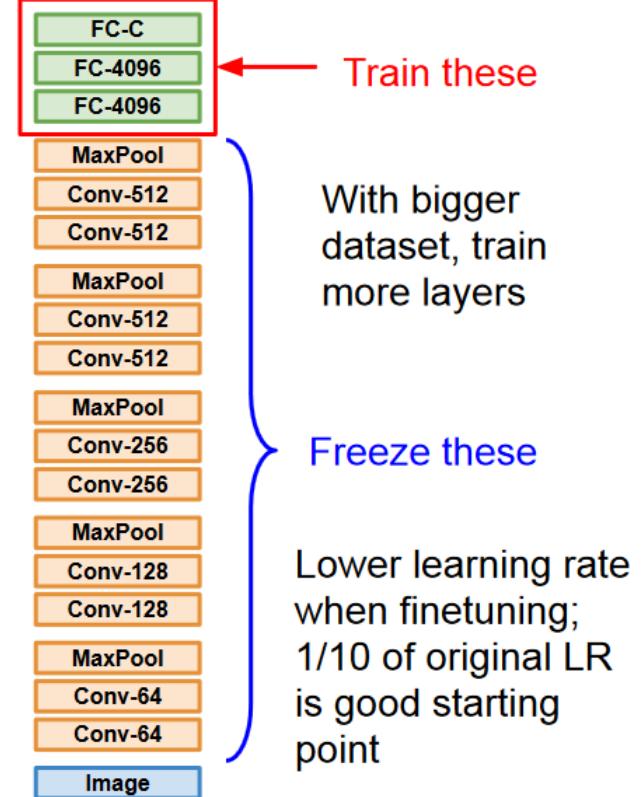
1. Train on Imagenet



2. Small Dataset (C classes)



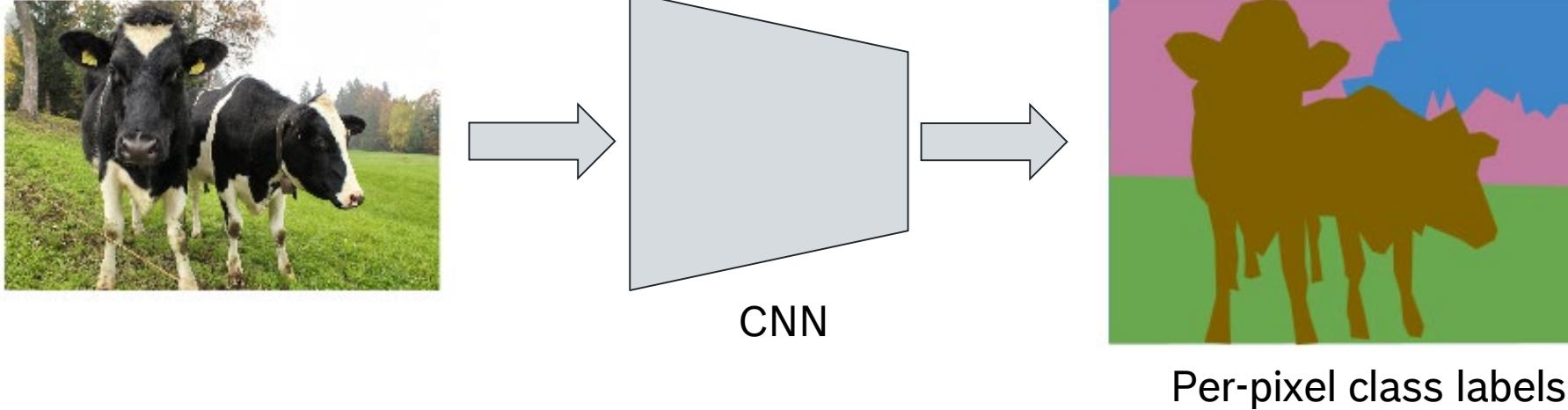
3. Bigger dataset



SEMANTIC SEGMENTATION

Semantic segmentation

Task definition



Semantic segmentation -> Per-pixel classification -> **we can use similar loss functions**

$$L(\theta) = -\log(p_{model}(y|x, \theta)) = -\sum_c y_c \log(\hat{y}_c)$$

Semantic segmentation

Example

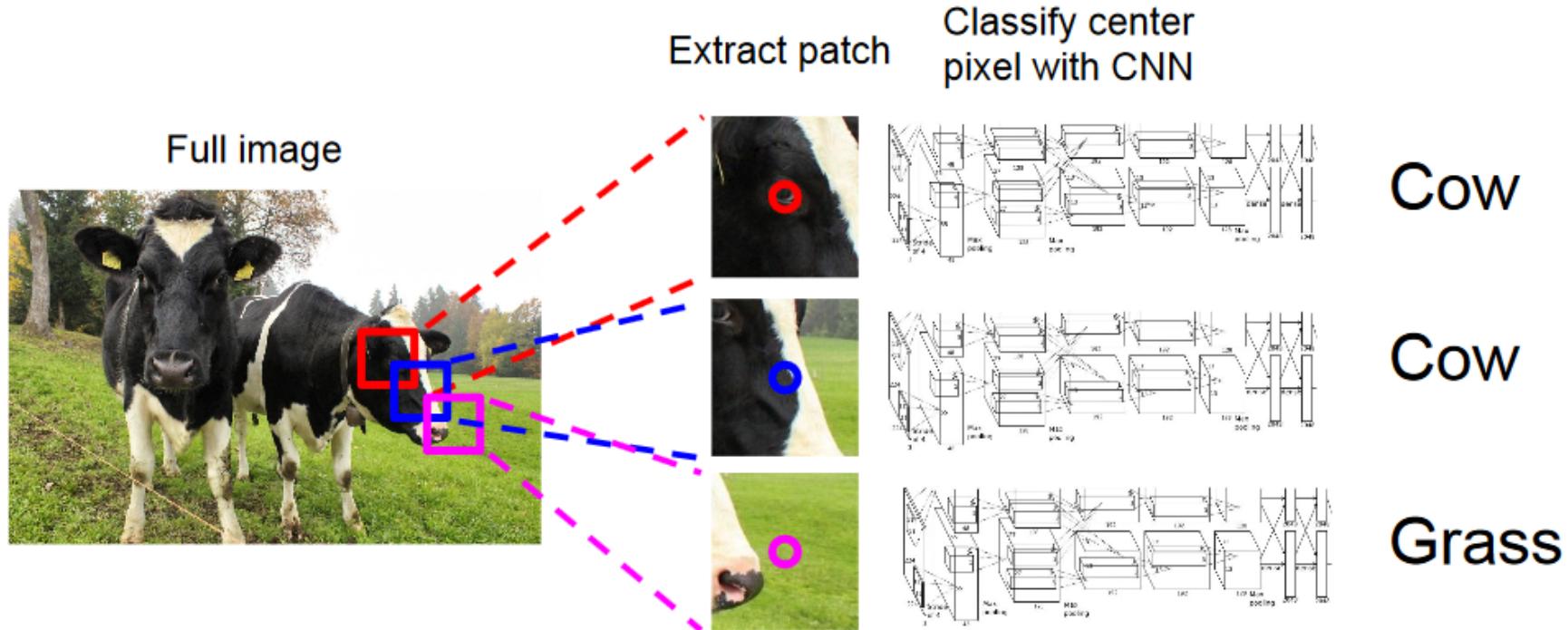


self, etc.	dynamic	ground	road	sidewalk
parking	rail track	building	wall	fence
guard rail	bridge	tunnel	pole	polegroup
traffic light	traffic sign	vegetation	terrain	sky
person	rider	car	truck	bus
caravan	trailer	train	motorcycle	bicycle



Semantic segmentation

Naive approach



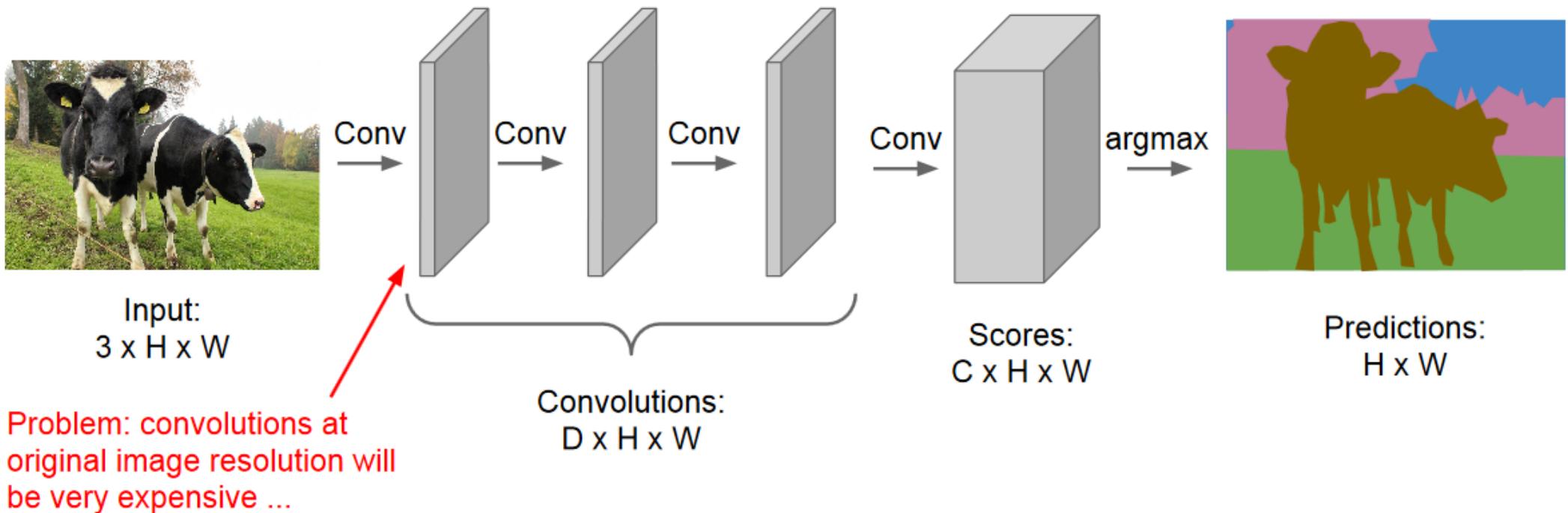
Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic segmentation

Better approach – but impractical

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Semantic segmentation

Better approach

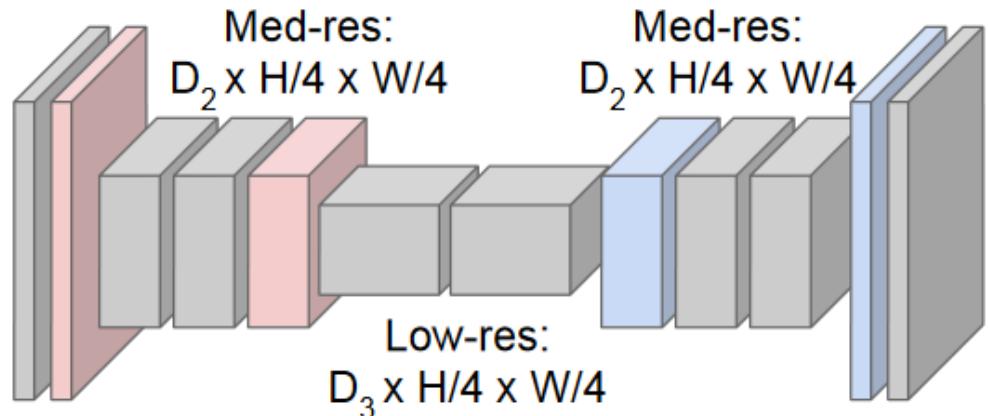
Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Transposed convolution
(a.k.a. “deconvolution”) for
upsampling



Input:
 $3 \times H \times W$

High-res:
 $D_1 \times H/2 \times W/2$



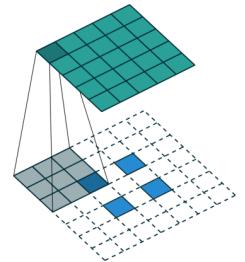
High-res:
 $D_1 \times H/2 \times W/2$



Predictions:
 $H \times W$

Semantic segmentation

Better approach



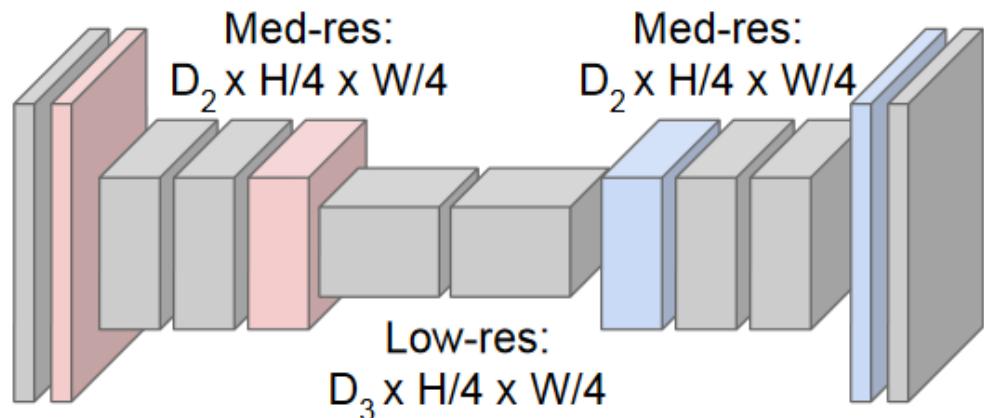
Transposed convolution
(a.k.a. “deconvolution”) for
upsampling

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



Input:
 $3 \times H \times W$

High-res:
 $D_1 \times H/2 \times W/2$



High-res:
 $D_1 \times H/2 \times W/2$



Predictions:
 $H \times W$

Semantic segmentation

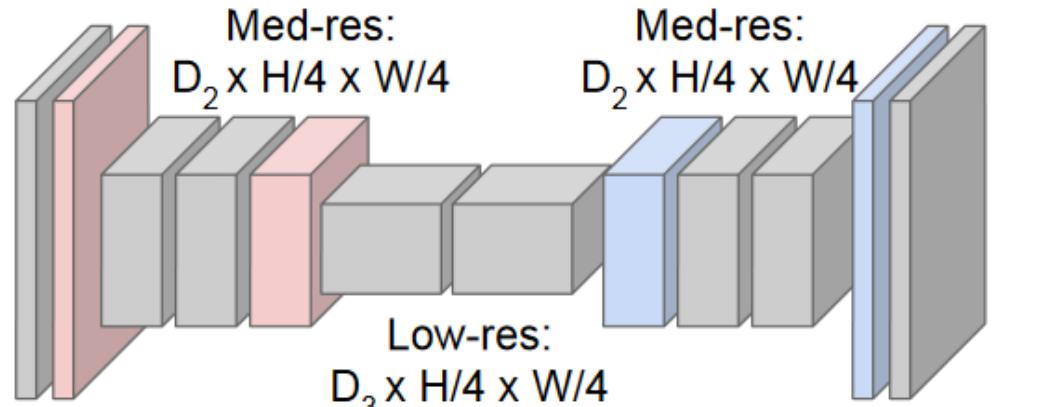
Better approach

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Input:
 $3 \times H \times W$

High-res:
 $D_1 \times H/2 \times W/2$



Predictions:
 $H \times W$

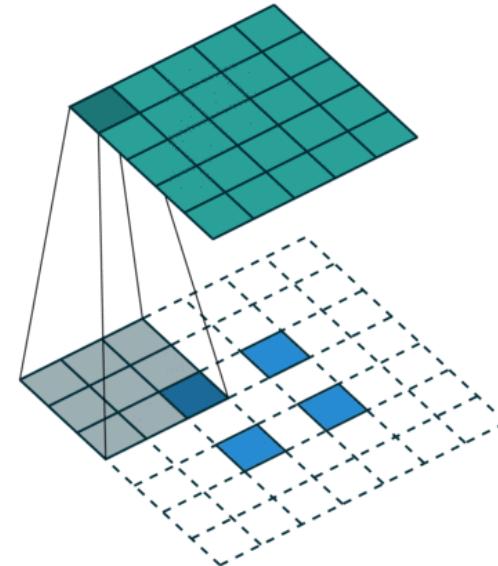
Feature extractor -> can be pre-trained on image classification

Semantic segmentation

Transposed convolution

About

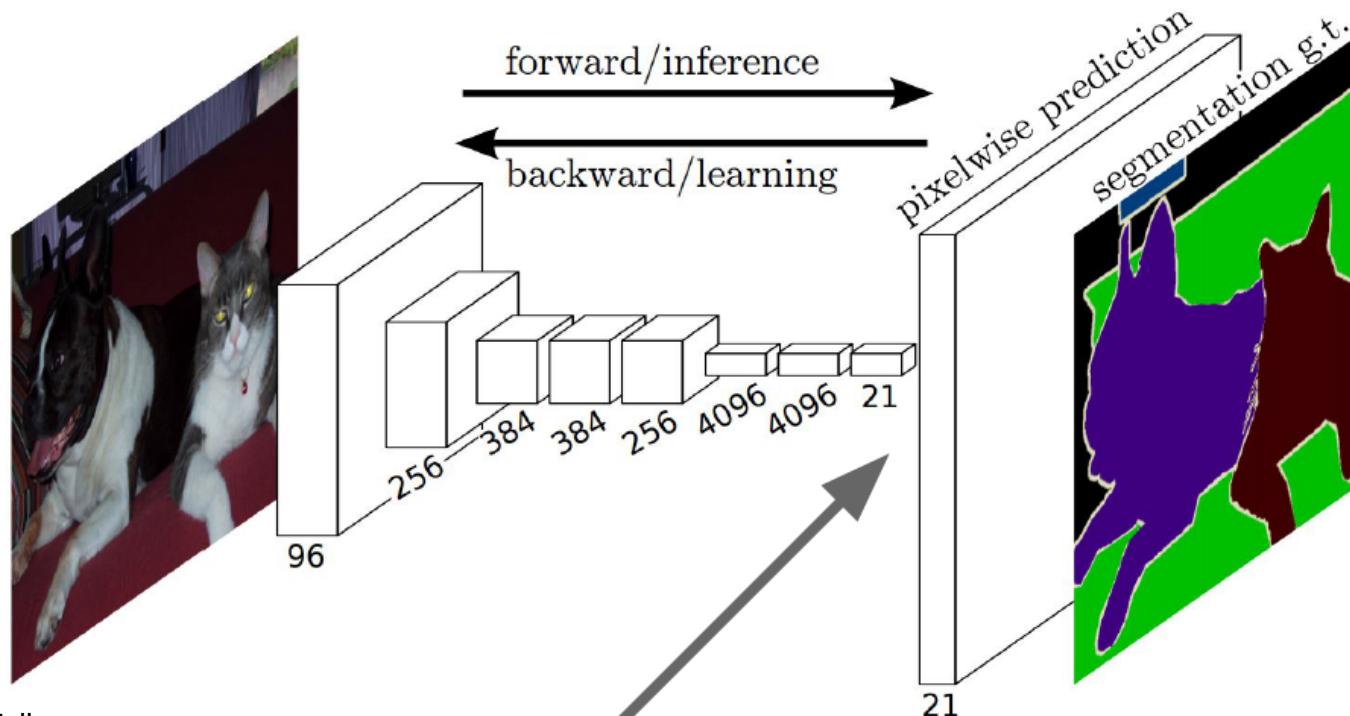
- Upsamples its input
- It is parameterized, therefore it can be trained
- Should obtain better results than standard interpolation methods



Transposed convolution
(a.k.a. “deconvolution”)

Semantic segmentation

Simplest architecture

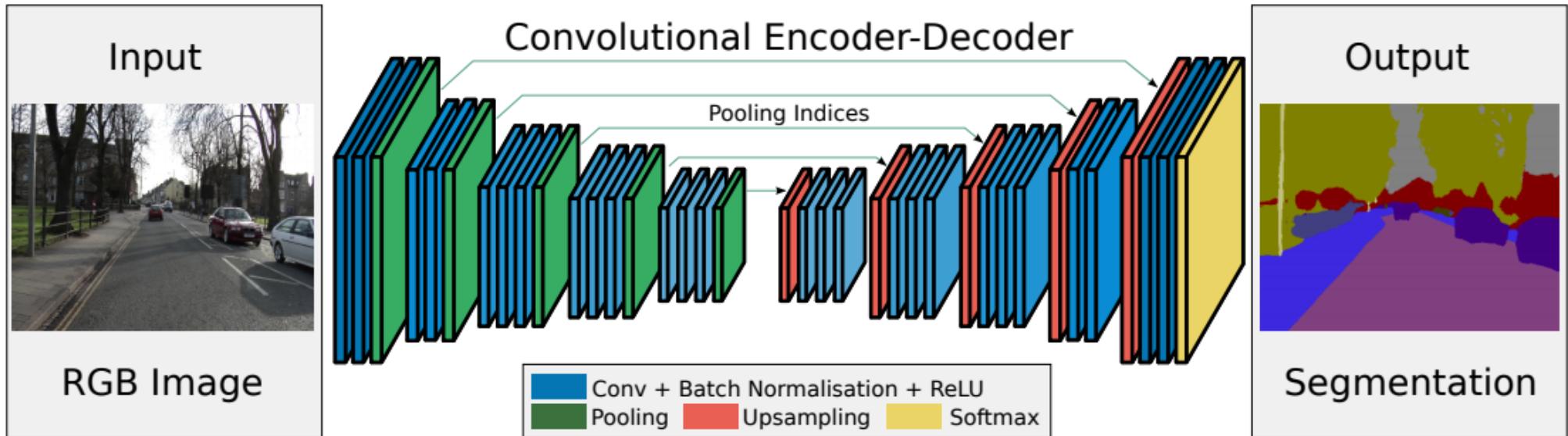


Long, Shelhamer, and Darrell, “Fully
Convolutional Networks for Semantic
Segmentation”, CVPR 2015

Learnable upsampling!

Semantic segmentation

Better architectures



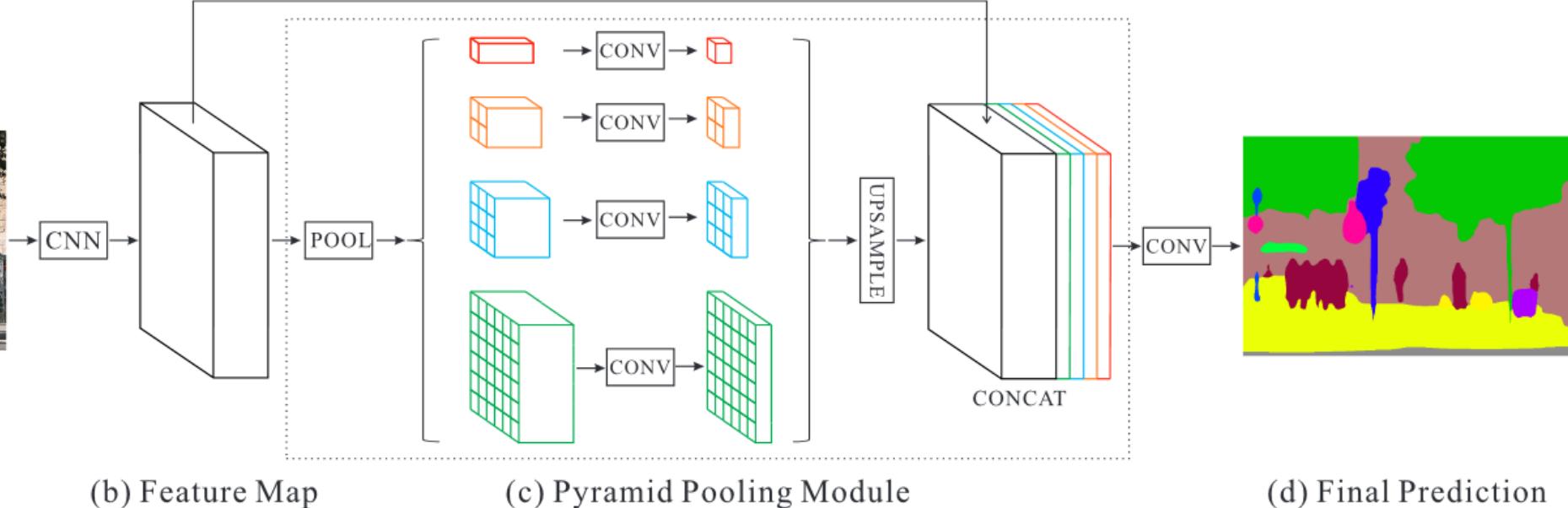
Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "**Segnet**: A deep convolutional encoder-decoder architecture for image segmentation." *arXiv preprint arXiv:1511.00561*(2015).

Semantic segmentation

Better architectures



(a) Input Image

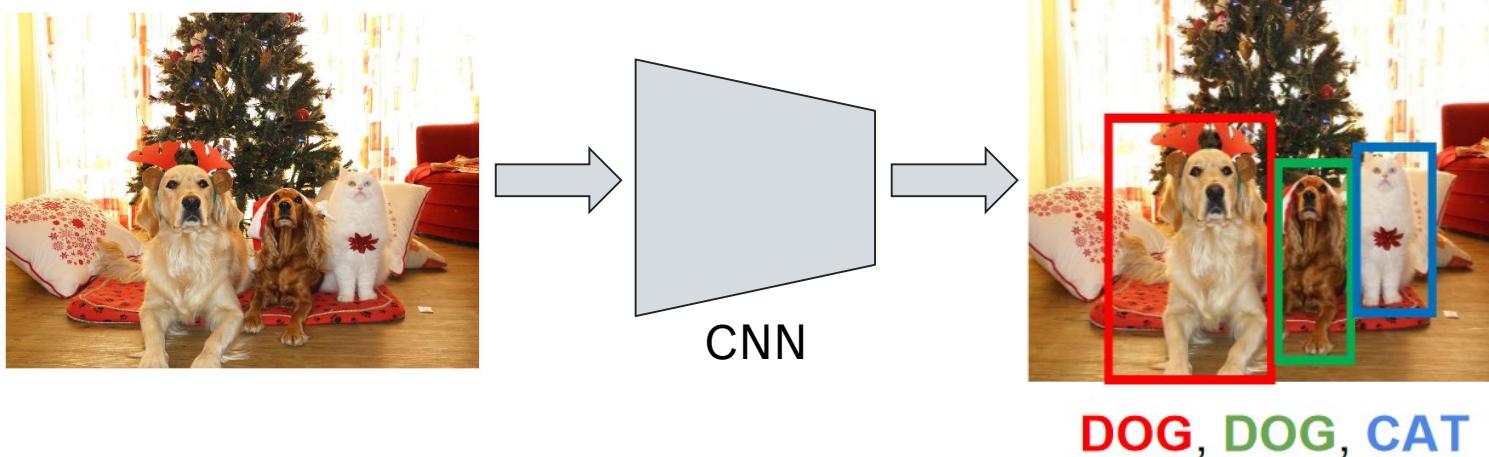


Zhao, Hengshuang, et al. "Pyramid scene parsing network." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

OBJECT DETECTION

Object detection

Task definition



Object detection -> predict **bounding boxes** and the **associated classes**

-> Compound loss function

$$L(\theta) = \alpha L_{class}(\theta) + \beta L_{coord}(\theta)$$

$$\left\{ \begin{array}{l} L_{class}(\theta) = CE(\theta) = - \sum_c y_{class,c} \log(\hat{y}_{class,c}) \\ \\ L_{coord}(\theta) = MSE(\theta) = (y_{coord} - \hat{y}_{coord})^2 \end{array} \right.$$

Object detection Example

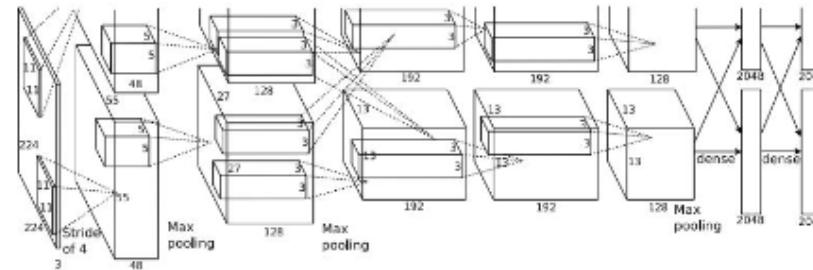


Redmon, Joseph, and Ali Farhadi.
"Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).

Object detection Naive approach



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



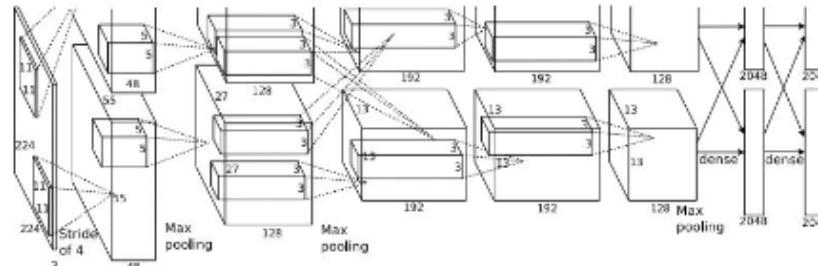
Dog? NO
Cat? NO
Background? YES

Object detection

Naive approach



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

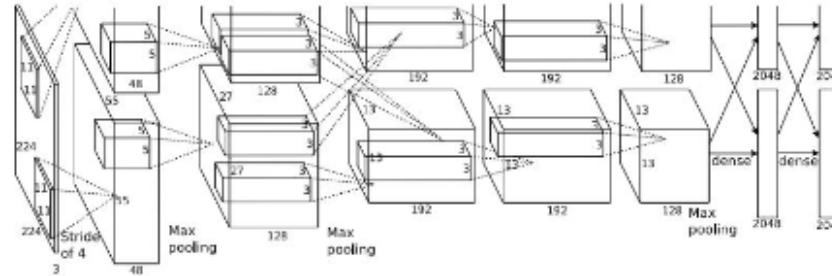


Dog? YES
Cat? NO
Background? NO

Object detection Naive approach



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



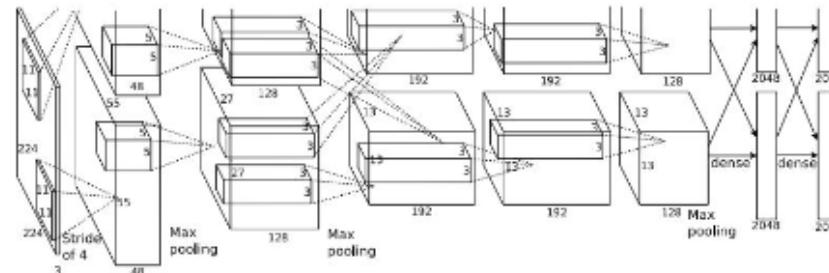
Dog? YES
Cat? NO
Background? NO

Object detection

Naive approach

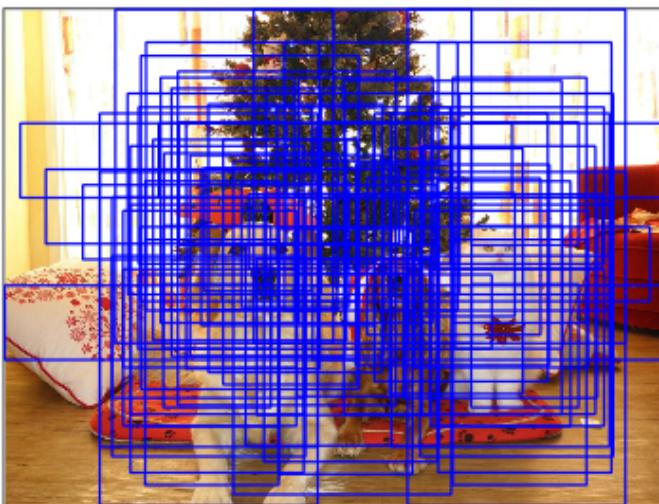


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

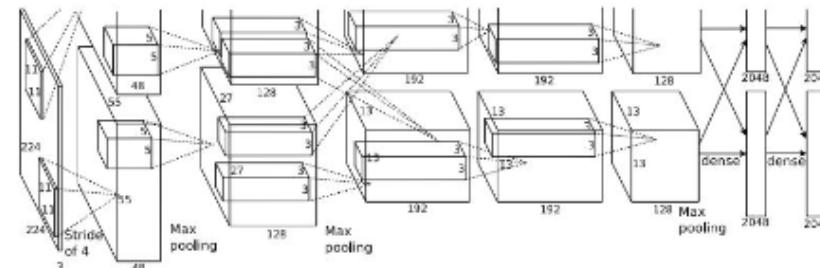


Dog? NO
Cat? YES
Background? NO

Object detection Naive approach



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

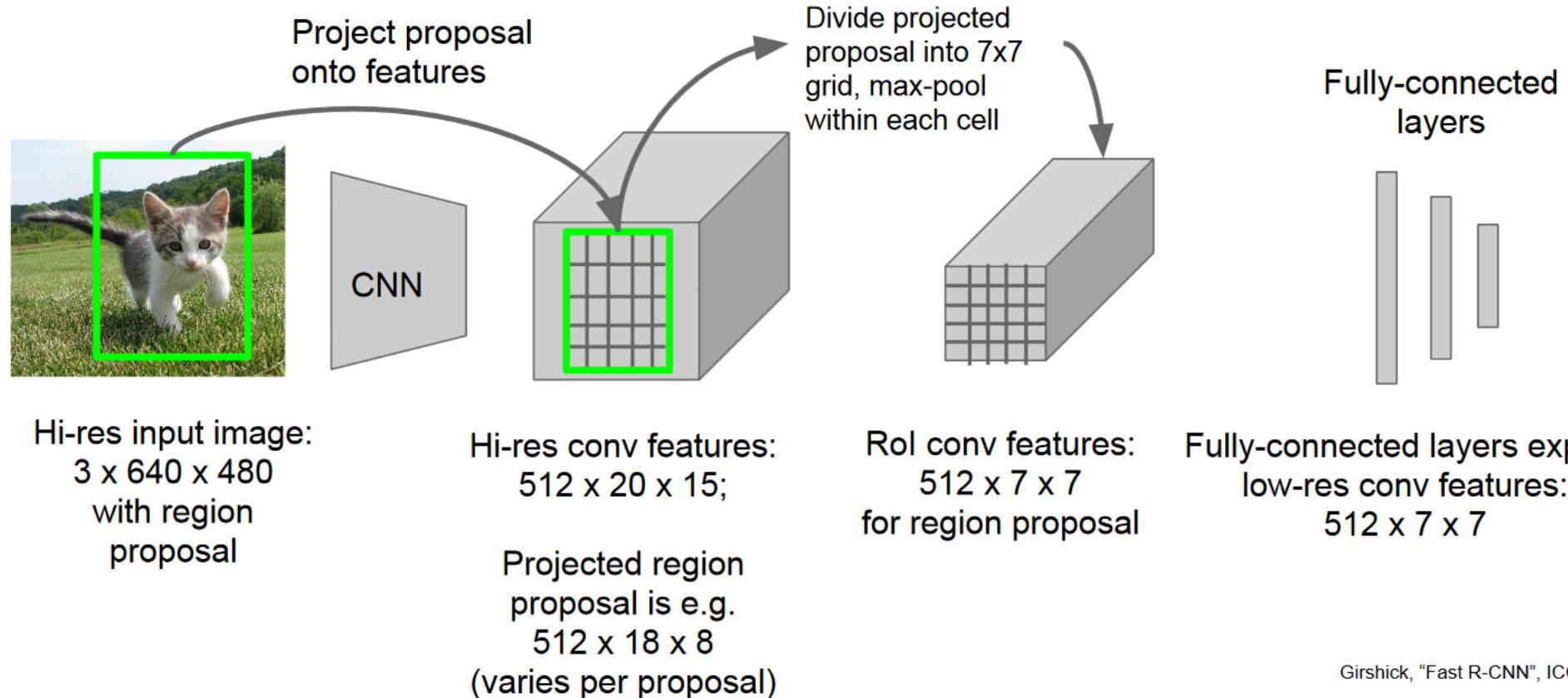


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Object detection

Better approach – only classify some proposals



Object detection

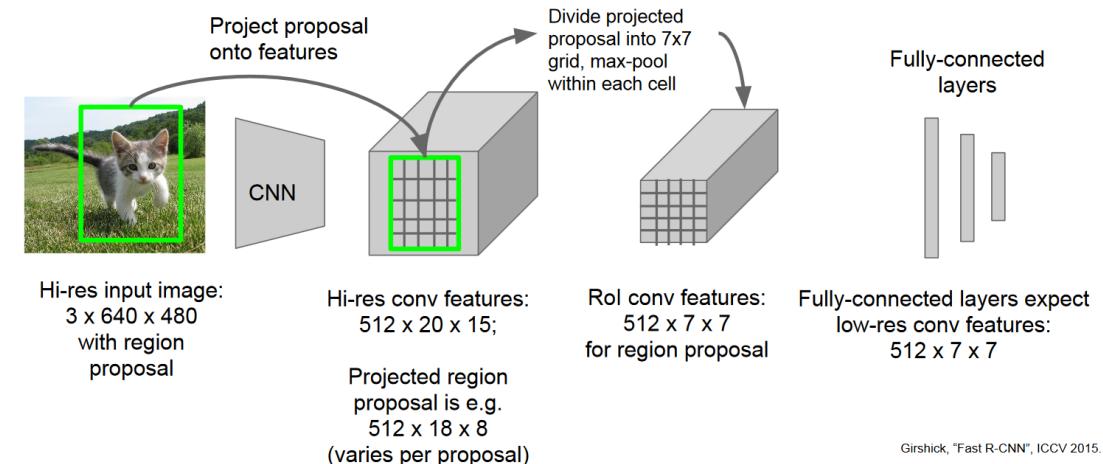
Better approach – only classify some proposals

Issues

- Independently processing each cropped feature block is very costly

Solution

- Ideally, we would like to obtain both the bounding boxes and the classes in a single pass



Object detection

One pass detectors – YOLO / SSD



Input image
 $3 \times H \times W$

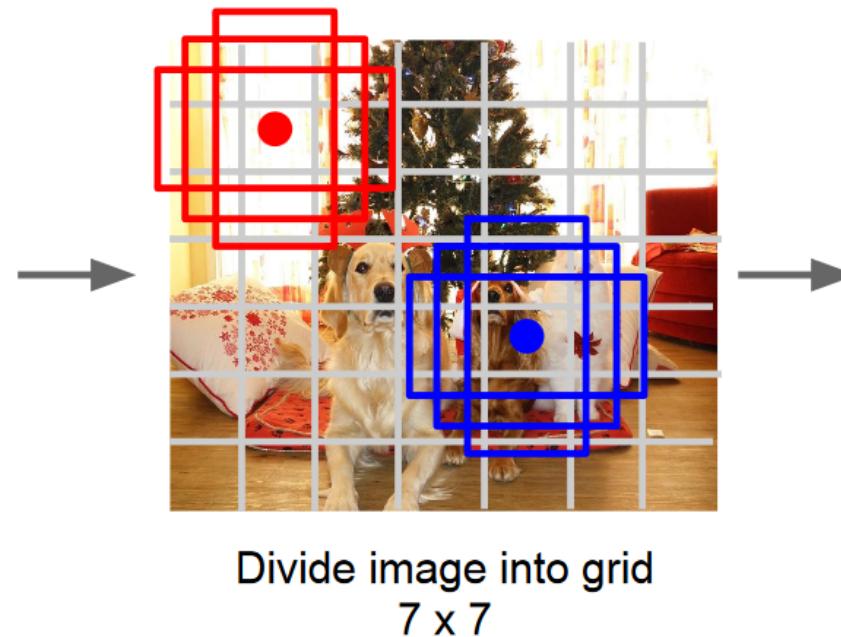


Image a set of **base boxes**
centered at each grid cell
Here $B = 3$

- Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers: $(dx, dy, dh, dw, confidence)$
 - Predict scores for each of C classes (including background as a class)

Output:
 $7 \times 7 \times (5 * B + C)$

Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Object detection

Bounding box parameterization in YOLO

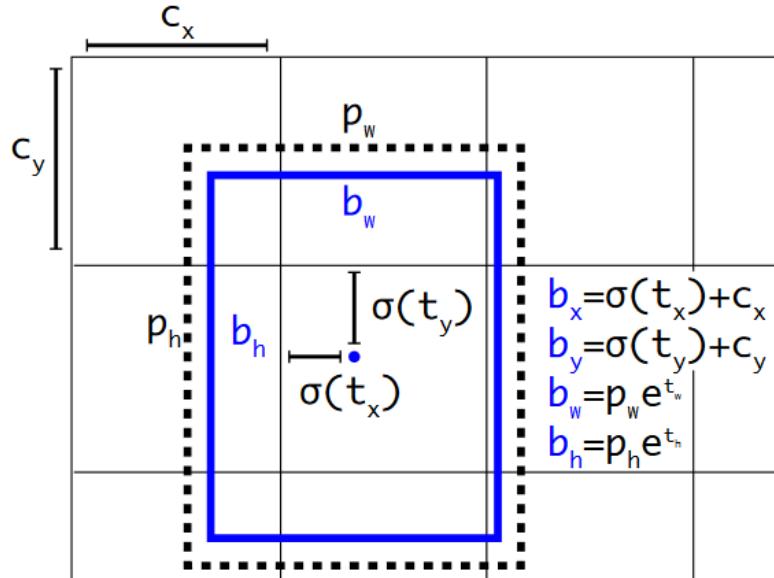


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

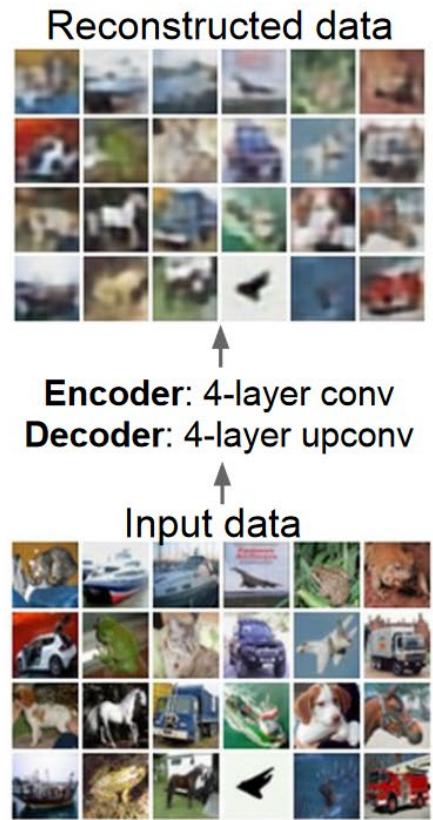
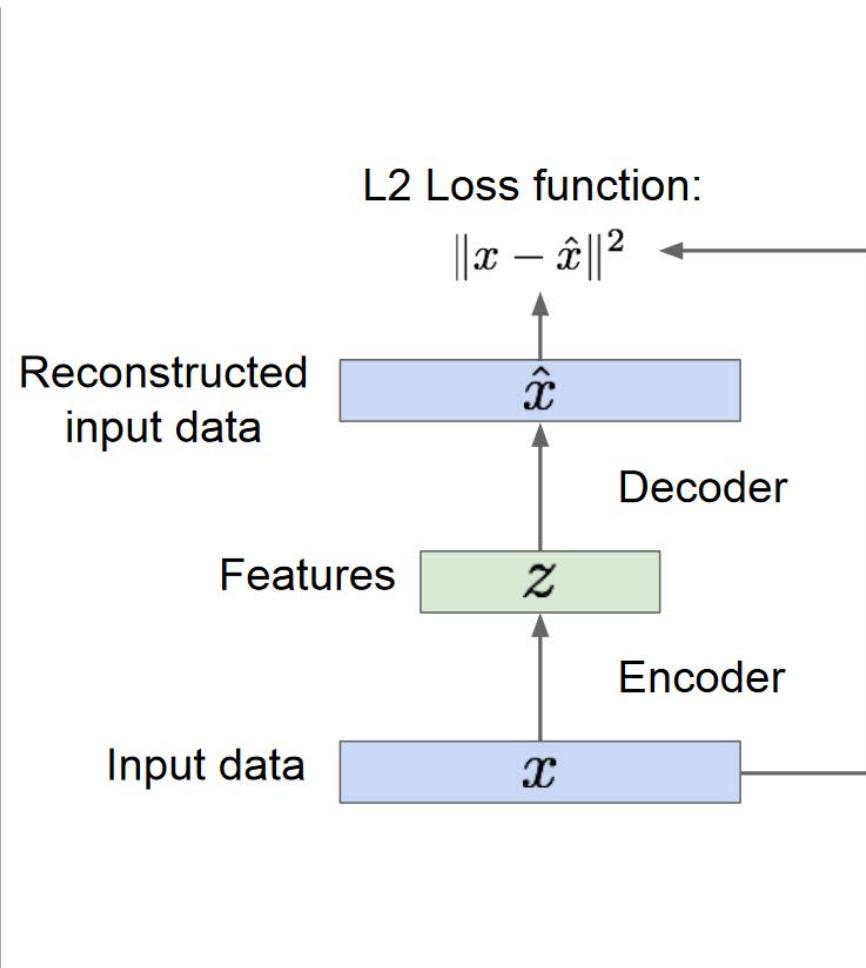
- $[c_x, c_y]$ • Coordinates of current cell (integers)
- $[\sigma(t_x), \sigma(t_y)]$ • Predicted bounding box center coordinates (relative to the current grid cell)
- $[b_w, b_h]$ • Predicted bounding box width and height (relative) to the anchor / base bounding box
- $[p_w, p_h]$ • Width and height of the anchor / base bounding box (pixels)

AUTOENCODERS

Autoencoders Overview

About

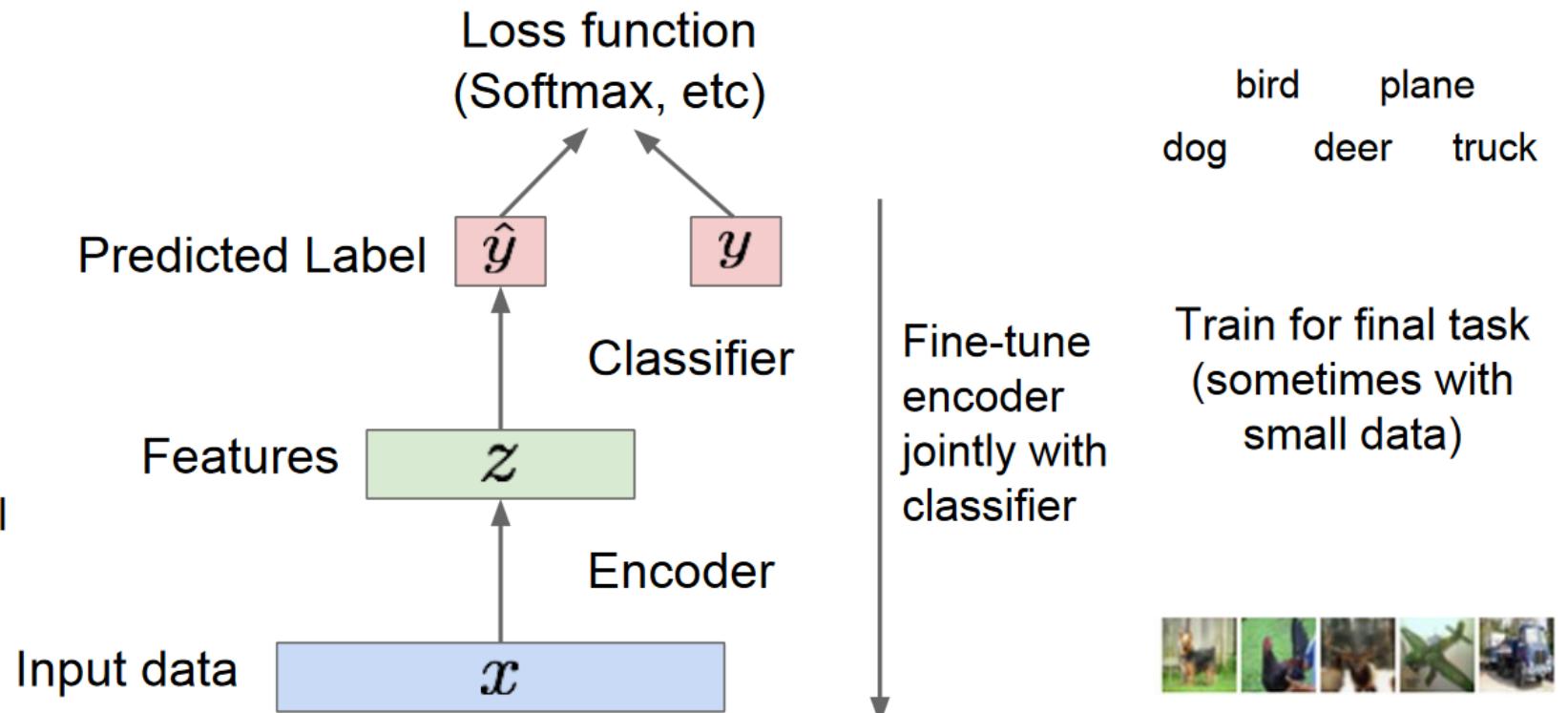
- They do not require labels (**unsupervised learning**)
- Based on information bottleneck (compression)
- Can be used for compression, denoising etc.
- Also great for **pre-training** segmentation, classification, or object detection networks



Autoencoders

Pre-training supervised models

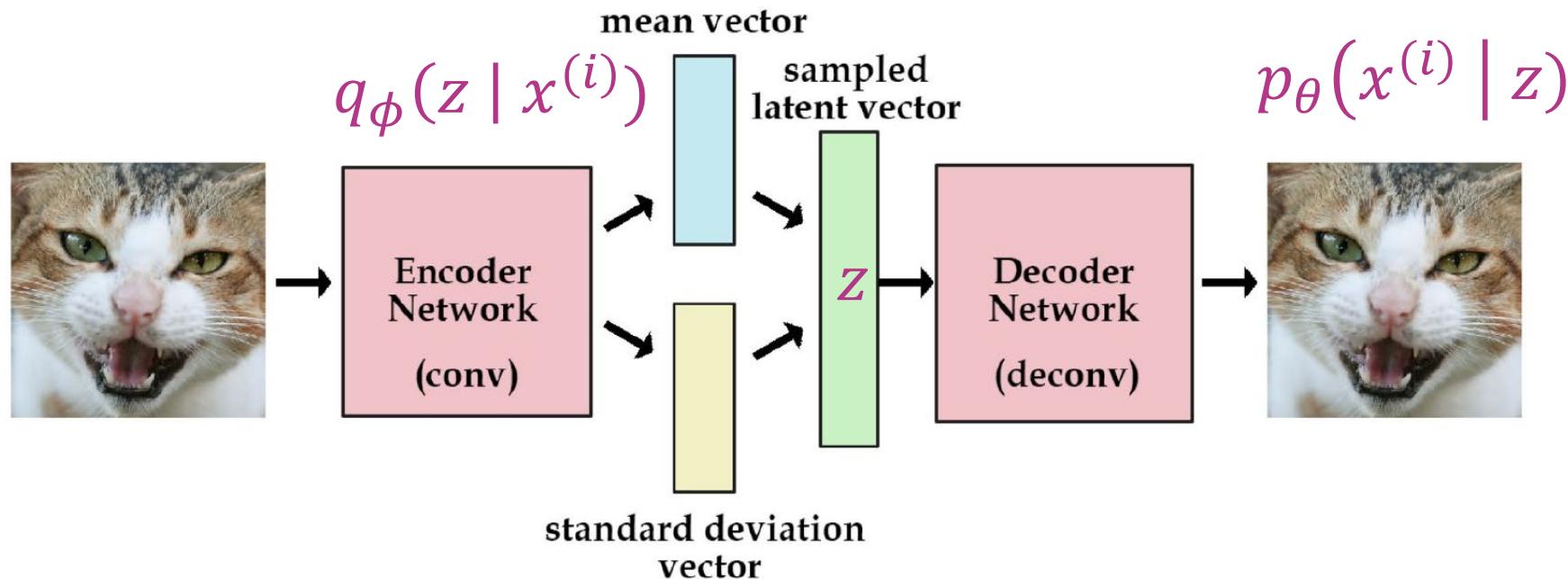
Encoder can be used to initialize a **supervised** model



GENERATIVE

Generative architectures

Variational Autoencoder (VAE)

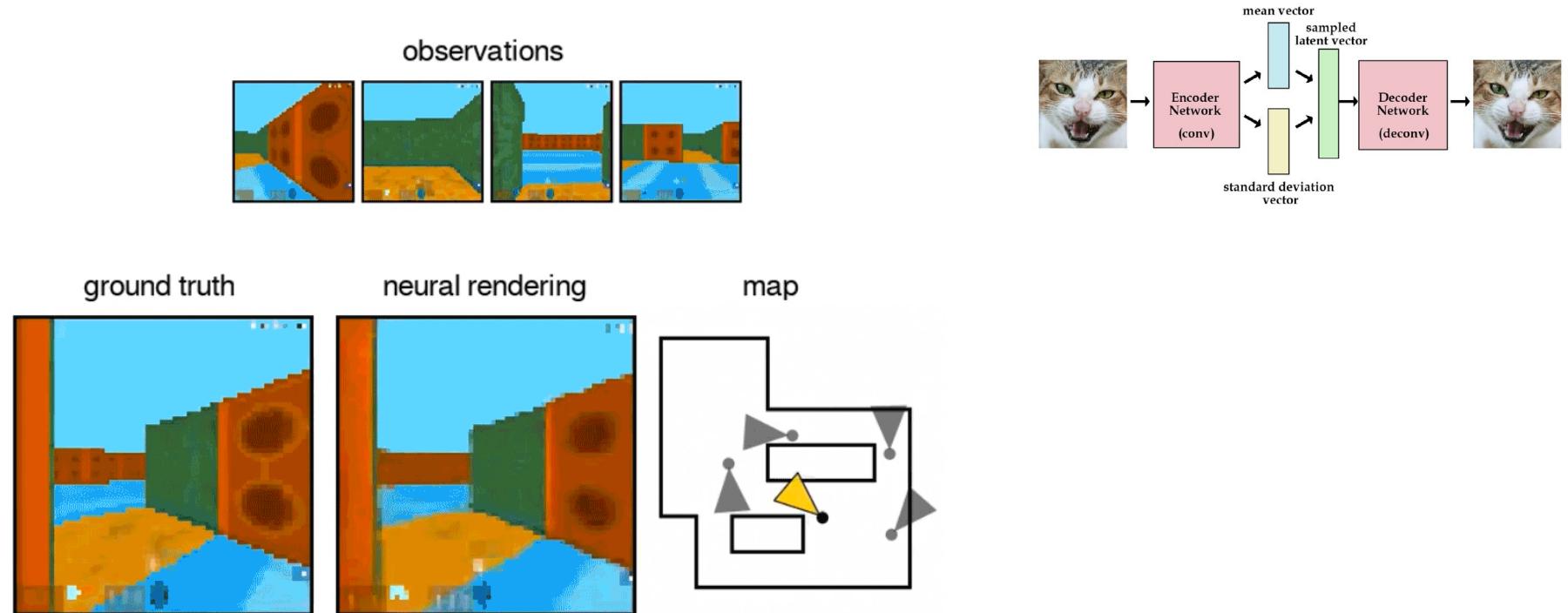


Maximize: $E_z \left[\log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) || p_{\theta}(z))$

<http://kvfrans.com/variational-autoencoders-explained/>

Generative architectures

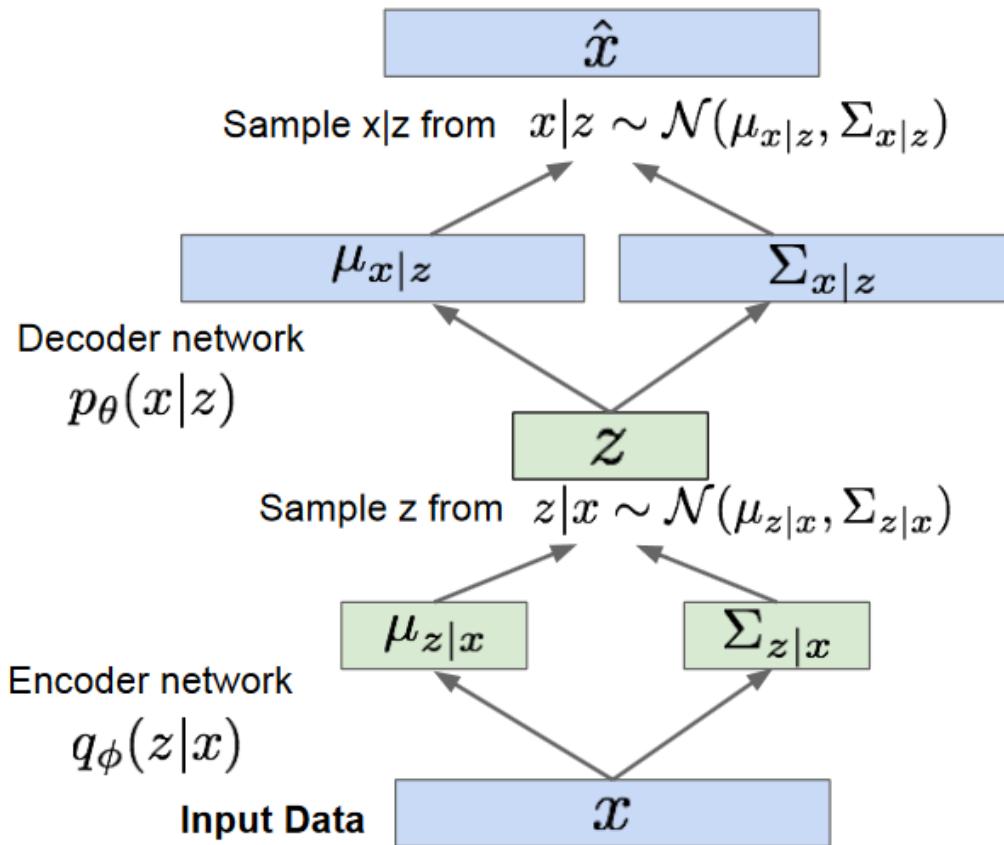
Variational Autoencoder (VAE)



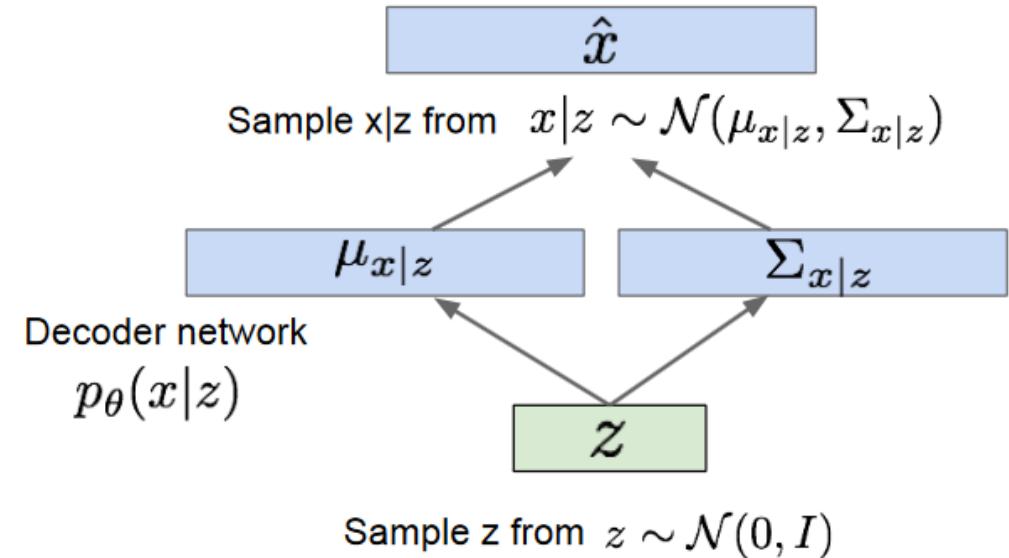
<https://deepmind.com/blog/neural-scene-representation-and-rendering/>

Generative architectures

Variational Autoencoder (VAE)



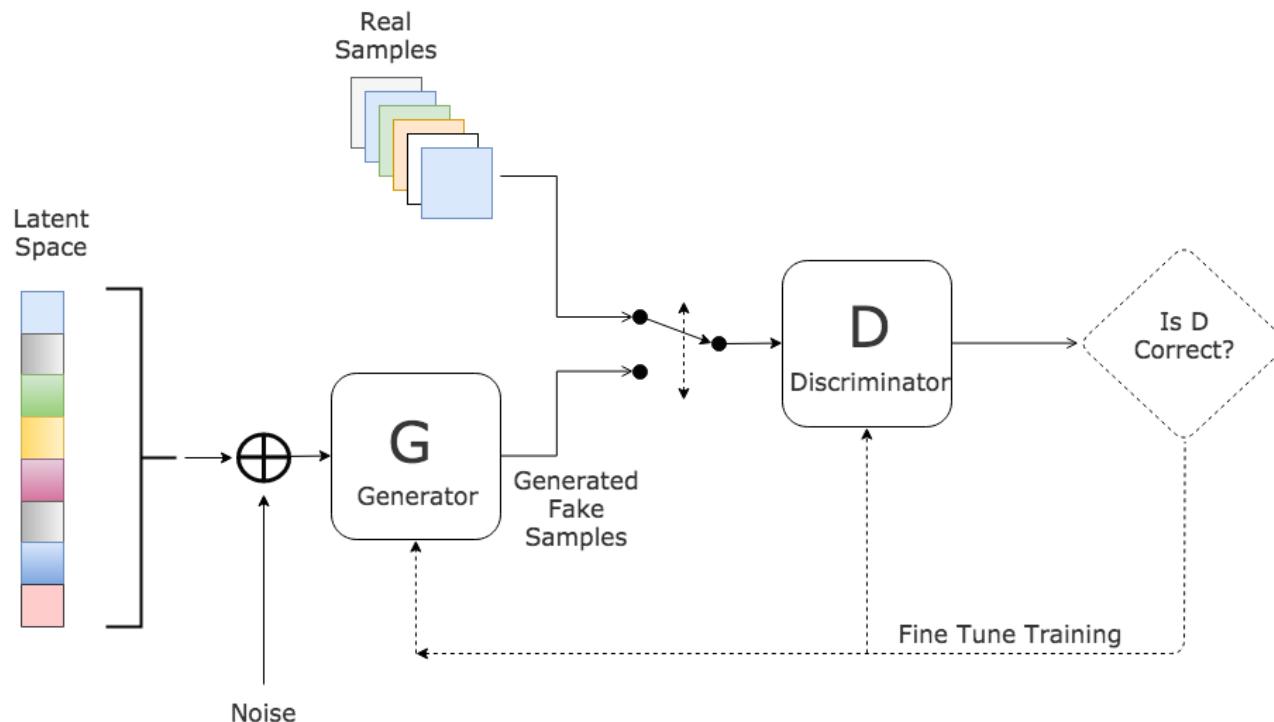
To generate new images use:



Generative architectures

Generative Adversarial Network (GAN)

Generative Adversarial Network

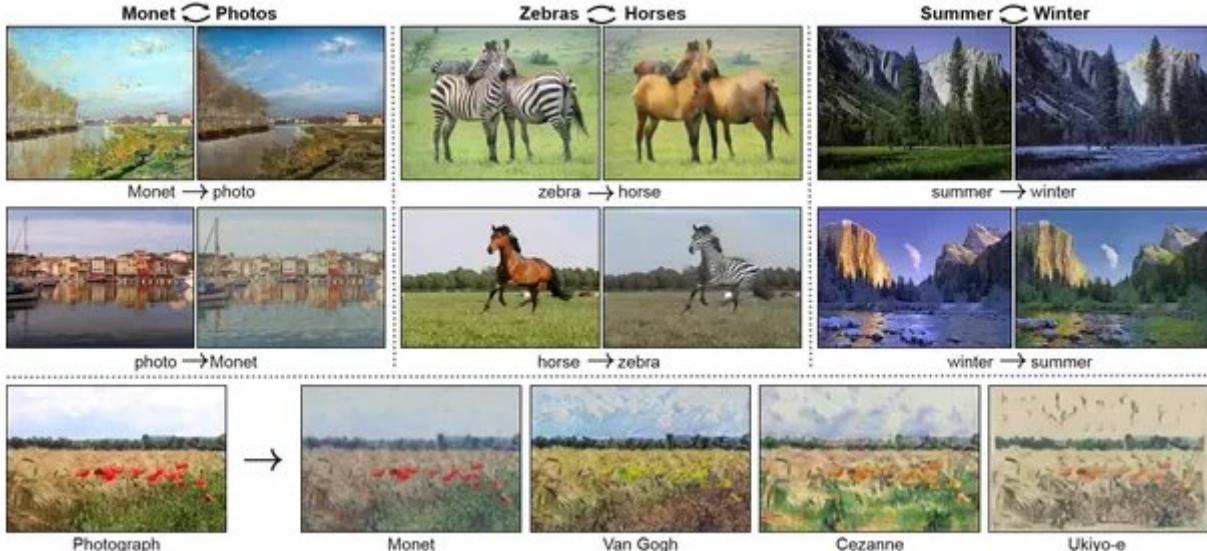


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Generative architectures

Generative Adversarial Network (GAN)



Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint*(2017).



Liu, Ming-Yu, Thomas Breuel, and Jan Kautz. "Unsupervised image-to-image translation networks." *Advances in Neural Information Processing Systems*. 2017.

<https://artix41.github.io/static/domain-adaptation-in-2017/index.html>

Generative architectures

Generative Adversarial Network (GAN)



<https://tcwang0509.github.io/pix2pixHD/>

OPTIMIZING FOR EMBEDDED

Optimizing for embedded

Some methods

- ▶ Pruning: remove “unimportant” weights or neurons
- ▶ Weight quantization: restrict weights to a discrete set of possible values; ternarization / binarization
- ▶ Filter design optimization: reduce the number of necessary operations (use 1x1 filters or depthwise convolution filters) (MobileNet, ShuffleNet)
- ▶ Knowledge distillation – Teacher/Student methods; train a smaller network using what the teacher has learned

https://medium.com/@nicolas_19145/state-of-the-art-in-compressing-deep-convolutional-neural-networks-cfd8c5404f22

THANK YOU FOR YOUR ATTENTION

SENSOR DATA FUSION

CONTENTS

CONTENTS

Slide structure

1. Introduction in data fusion
2. Preliminary notions
3. Problem to discuss
4. State Observer
5. Kalman Filter
6. Extended Kalman Filter

INTRODUCTION IN DATA FUSION

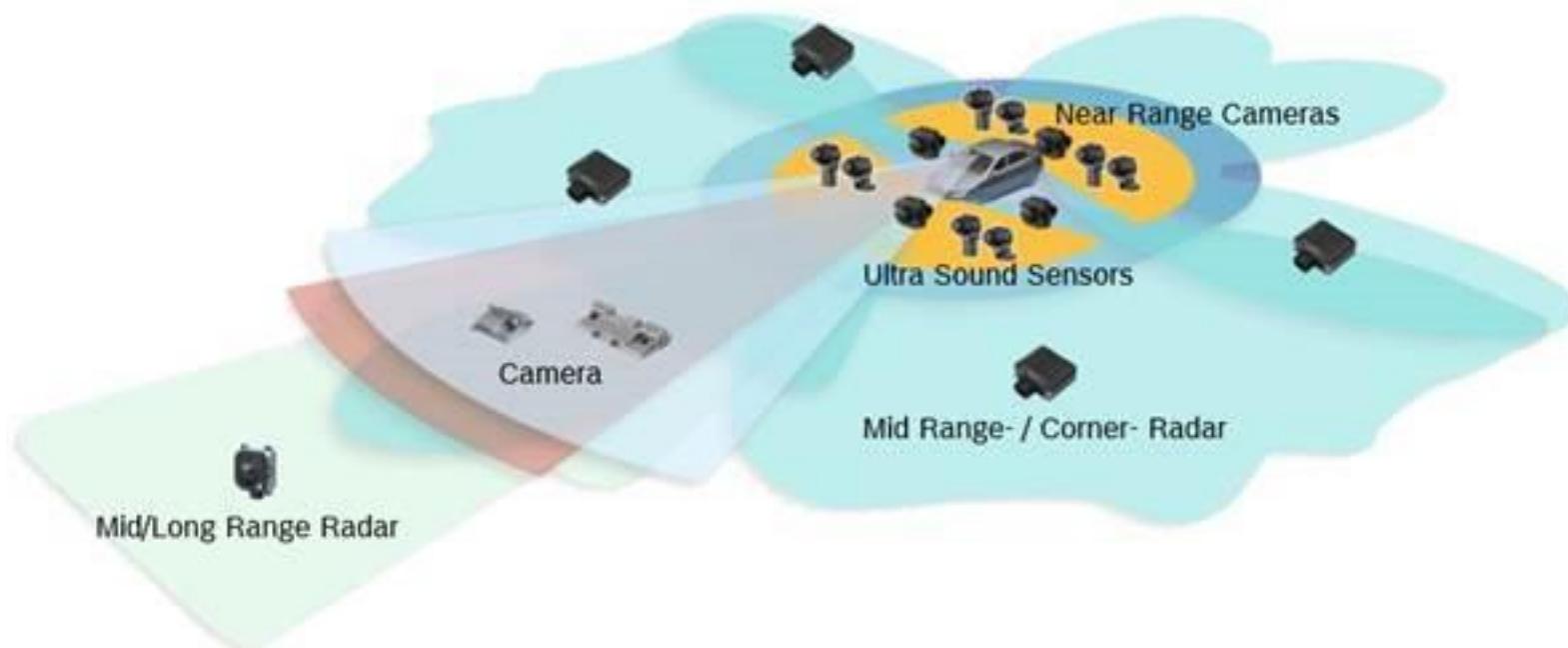
INTRODUCTION IN DATA FUSION

Why ?

- Each sensor has its uncertainty (error in measurement)
 - i.e. GPS – give position with an accuracy of 3 meter
- Each sensor has its drawbacks
 - i.e. GPS – does not work well in tunnels
 - IMU (inertial measurement unit) – accumulate error due to the integration process
- Each estimation method does not meet exactly the process
 - i.e. A theoretical model works perfect, but the process has variation due to mechanical issues

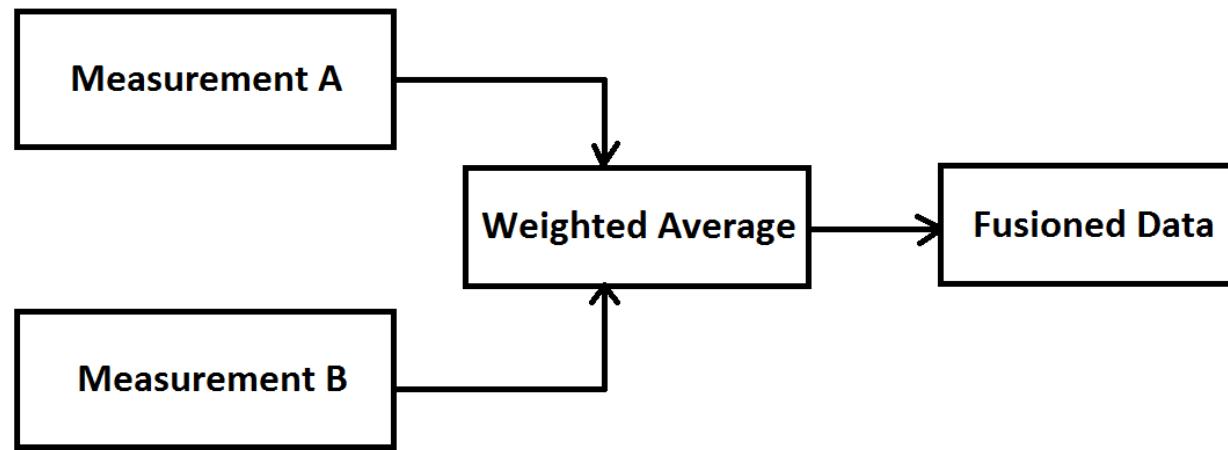
INTRODUCTION IN DATA FUSION

Why ?



INTRODUCTION IN DATA FUSION

How ?

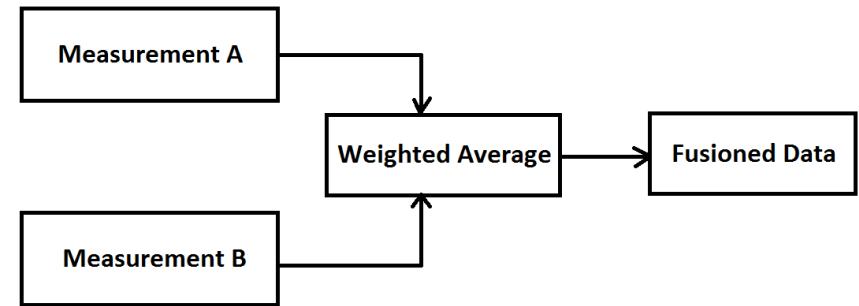


- Which are the weights ?
- Why we want this fusion ?

INTRODUCTION IN DATA FUSION

How ?

- Why we want this fusion ?



Measurement A

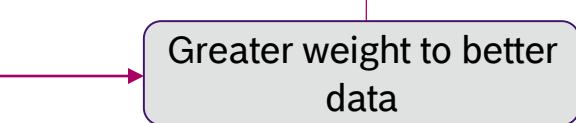
Weighted Average

Fusioned Data

Measurement B

Some “good” weights

- Which are the weights ?



INTRODUCTION IN DATA FUSION

Example

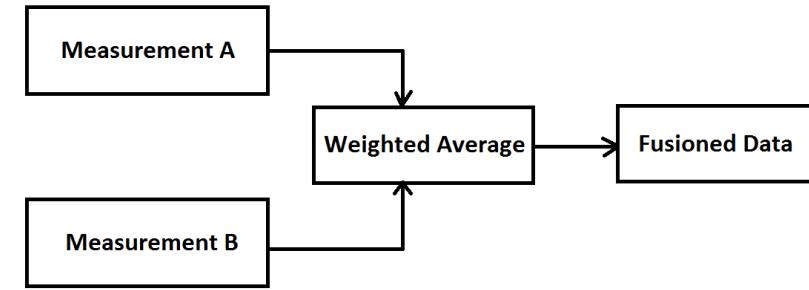


- Estimated relative position
- Accumulated error

A

- Absolute position
- Noisy

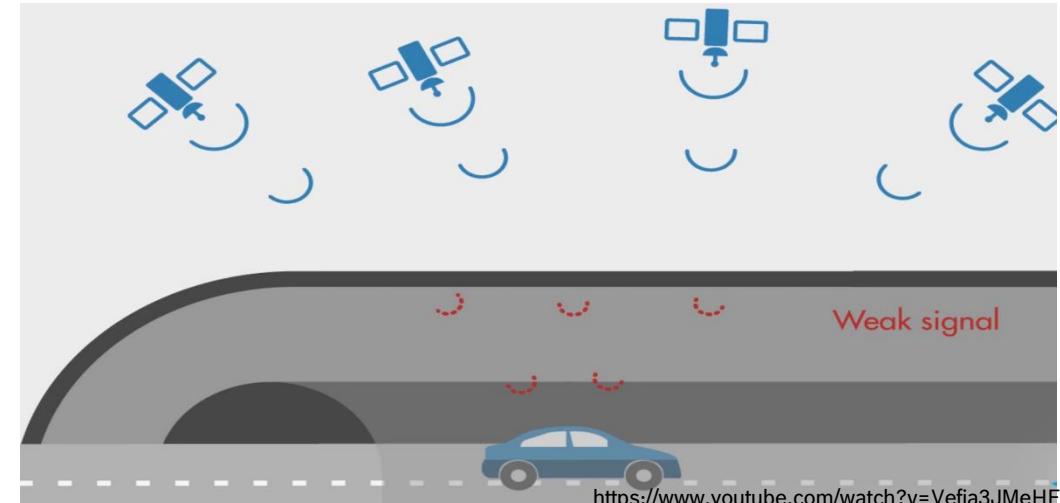
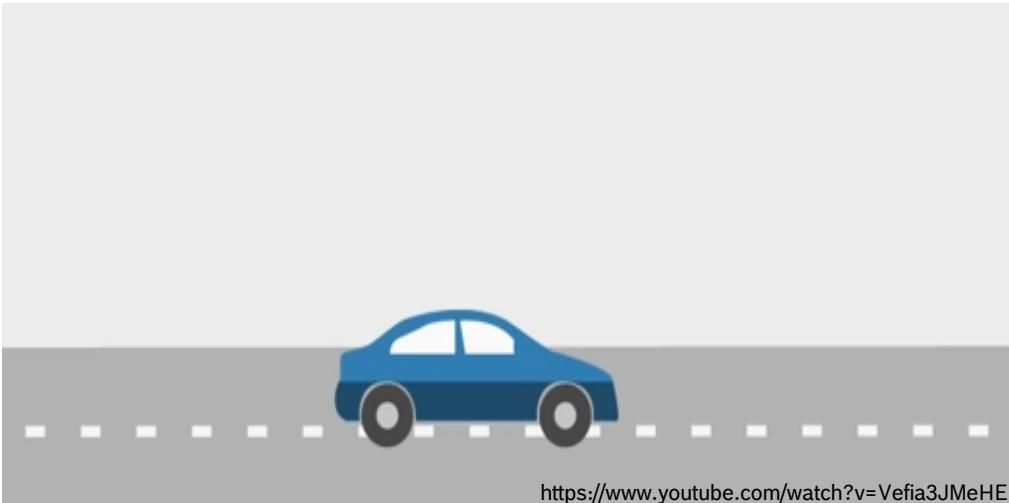
B



Which one you trust ?

INTRODUCTION IN DATA FUSION

Example



Good GPS – use A only for correction,
position is based on B

Bad GPS – use A exclusively to predict from
the other sensors

PRELIMINARY NOTIONS

Preliminary Notions

Random Variable

Def.: A function which link a random event with a number.



$$X : E \rightarrow \mathbb{R}$$

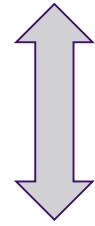
Example: Roll two dices. One roll represents an event. The random number associated with this event is the sum of the number of dices. This sum is a random variable.

Preliminary Notions

Expected Value

Take as random values the exam grades of all students from Control Engineering at math

$$E[X] = \frac{\sum_{i=1}^N (x_i)}{N}$$



Average of the students grades

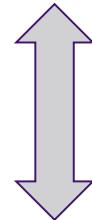
What we expect to be
the grade of a student?

Preliminary Notions

Variance

Take as random values the exam grades of all students from Control Engineering at math

$$Var(X) = E[(\bar{x} - X)^2] = \frac{\sum_{i=1}^N (\bar{x} - x_i)^2}{N}$$



$$[Var] = [X]^2$$

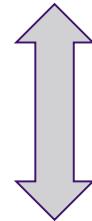
How far from an average student are the best and the worst student

Preliminary Notions

Standard Deviation

Take as random values the exam grades of all students from Control Engineering at math

$$\sigma(X) = \sqrt{Var(X)}$$



$$[\sigma] = [X]$$

More convenient

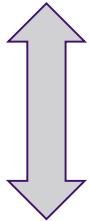
How far from an average student are the best and the
worst student

Preliminary Notions

Covariance

Take as random values the exam grades of all students from Control Engineering at math and English

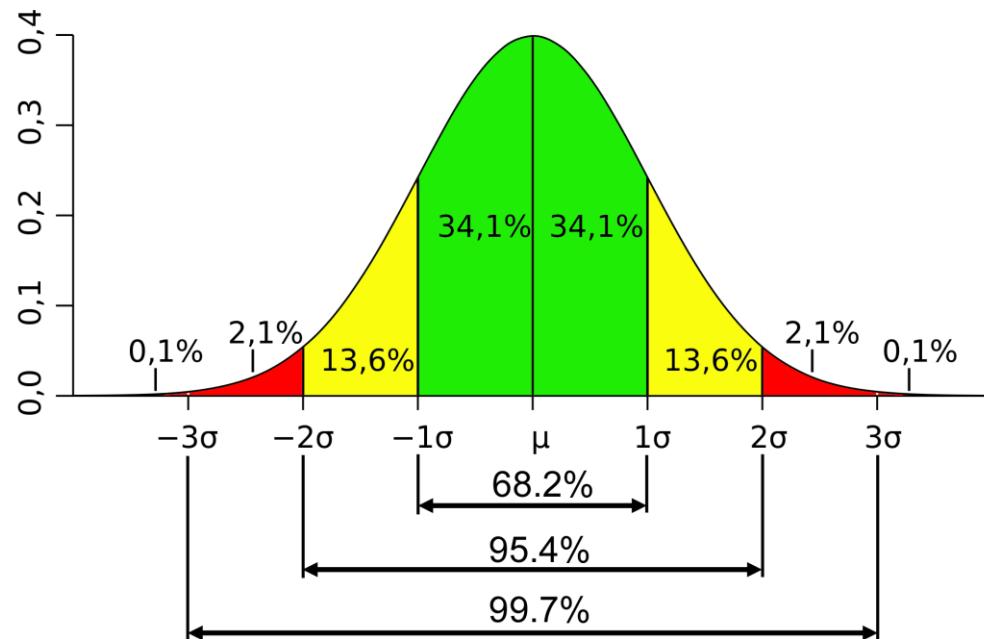
$$\text{Cov}(X, Y) = E[(\bar{x} - X)(\bar{y} - Y)] = \frac{\sum_{i=1}^N \sum_{j=1}^M (\bar{x} - x_i)(\bar{y} - y_j)}{N+M}$$



Express a linear dependence between two dimensions

Preliminary Notions

Gaussian Distribution



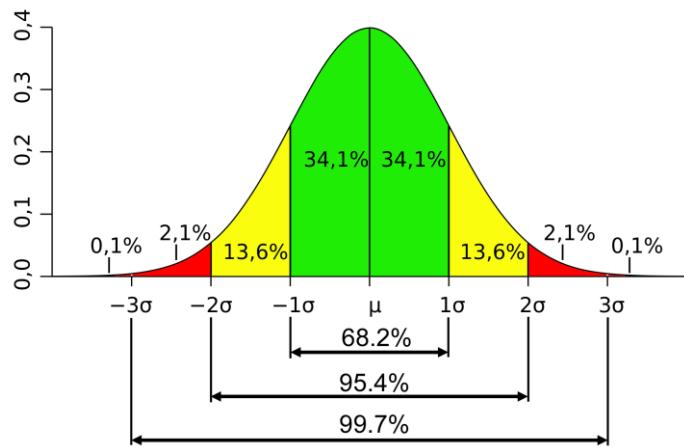
$$f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Probability density function

Take as random value the exam grade of all students from Control Engineering at math
How we interpret this graph ?

Preliminary Notions

Gaussian Distribution



$$f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$E[X_f] = \mu$$

Grade of an average student

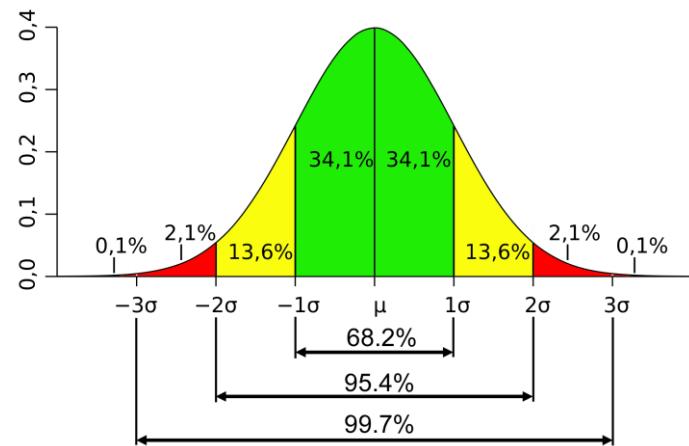
$$\text{Var}(X_f) = \sigma^2$$

How far away is the best and the worst student

Take as random value the exam grade of all students from Control Engineering at math
How we interpret this graph ?

Preliminary Notions

White Noise



$$f(x | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$E[X_f] = \mu = 0$$

We expect the noise to be zero

$$\text{Var}(X_f) = \sigma^2$$

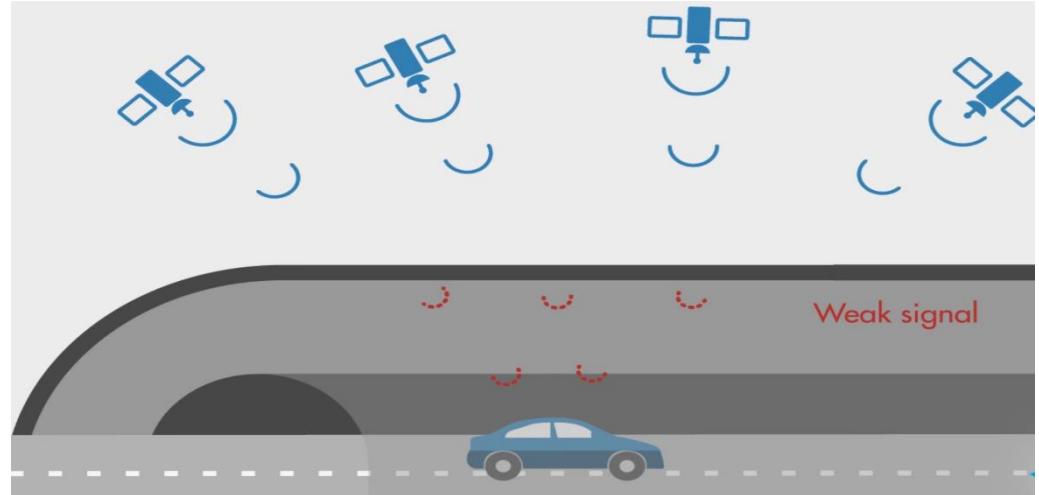
Depends on the signal source
↔ how distorted is the signal

A white noise is a deviation around the 0 which disturb the original signal
Example: Ground variation from an oscilloscope

PROBLEM TO DISCUSS

PROBLEM TO DISCUSS

Vehicle tracking

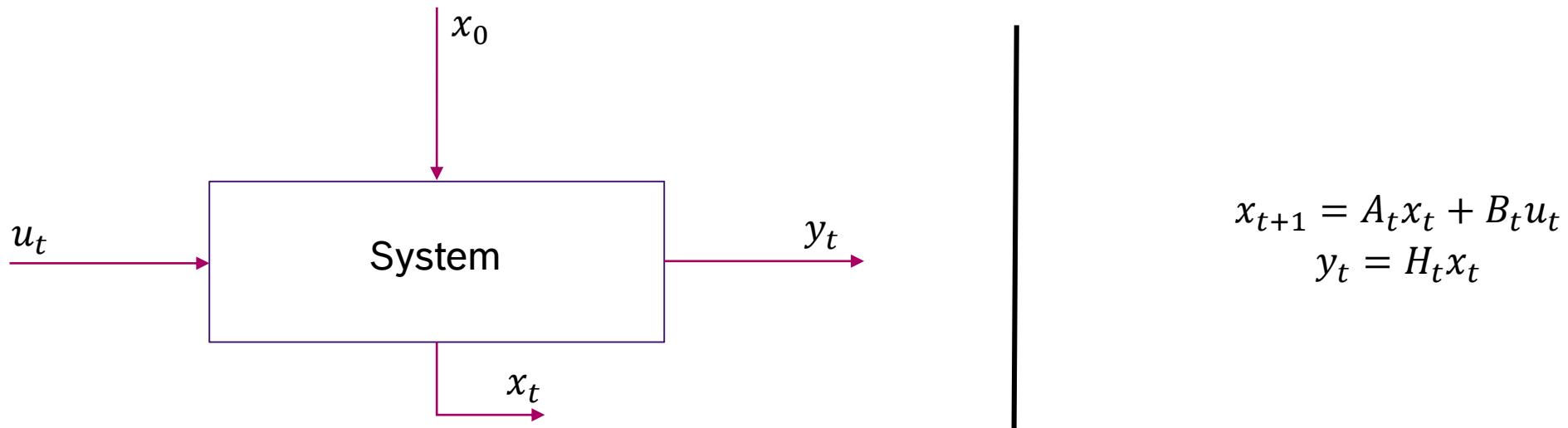


- Assume that the trajectory is a line
- You know the velocity of the car as input to the system
- You measure the position of the car via GPS system

STATE OBSERVER

STATE OBSERVER

Introduction

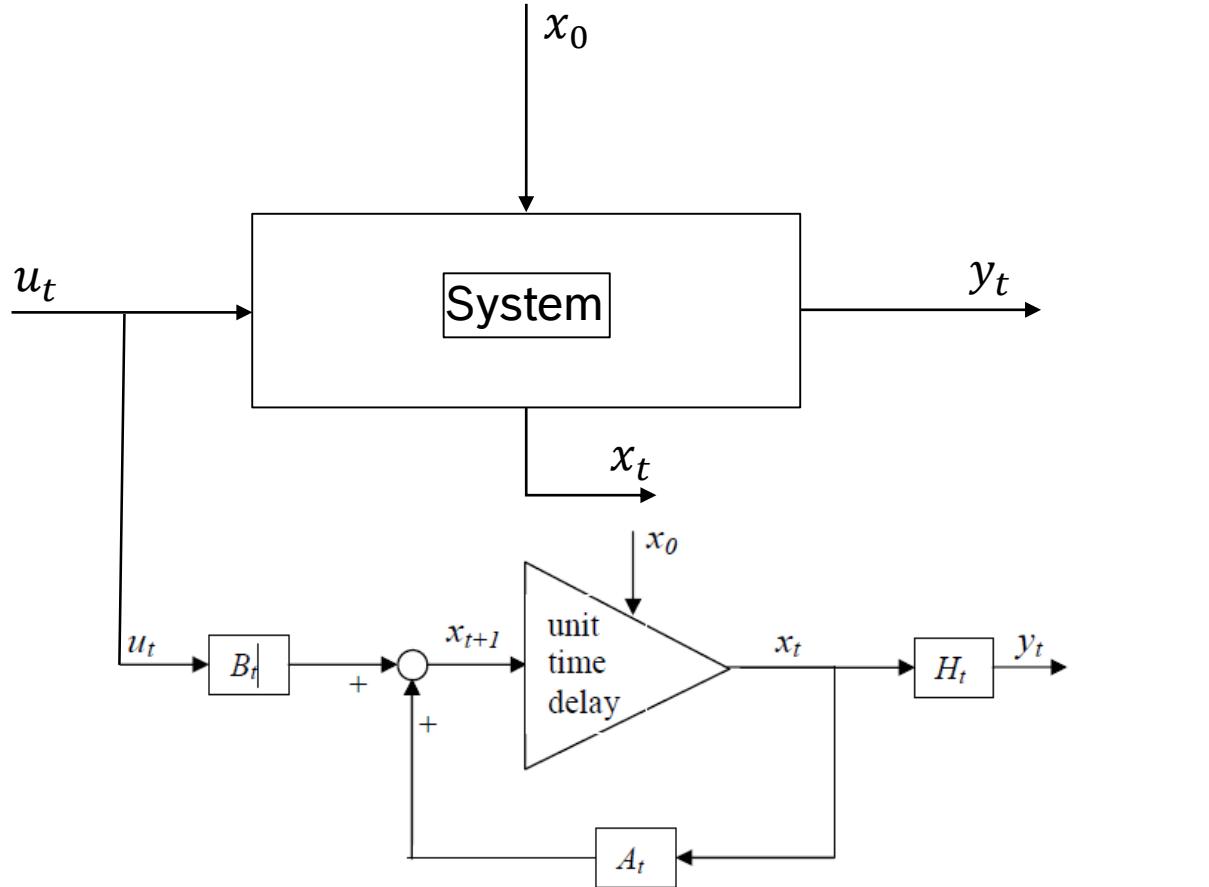


- We measure the output y_t
- What about the internal state x_t ? Can estimate the internal state ?

Image taken from www.quora.com

STATE OBSERVER

System estimator



$$x_{t+1} = A_t x_t + B_t u_t$$
$$y_t = H_t x_t$$

Is it the same ?

Image taken from www.quora.com

STATE OBSERVER

Optimal State Estimator

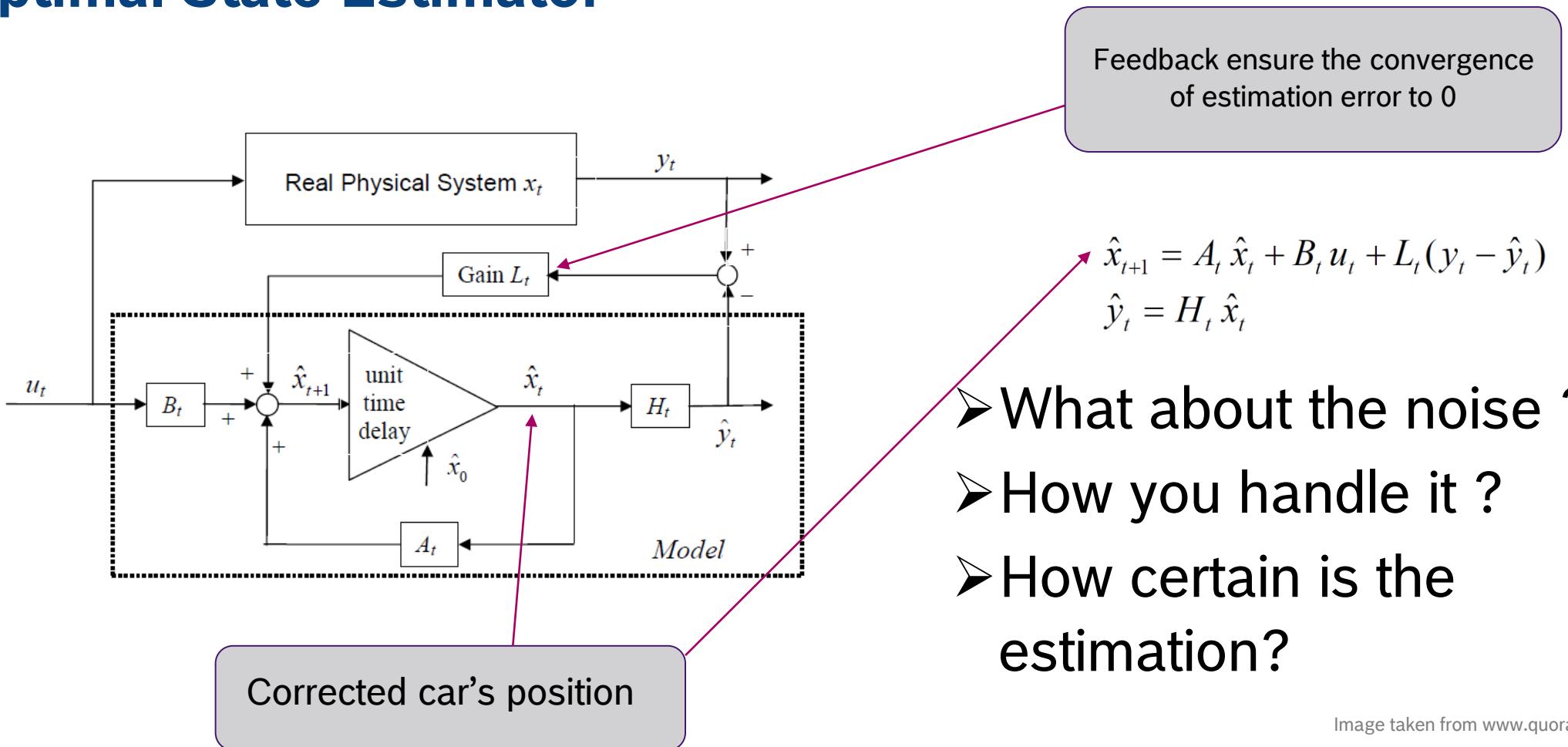


Image taken from www.quora.com

STATE OBSERVER

Optimal State Estimator

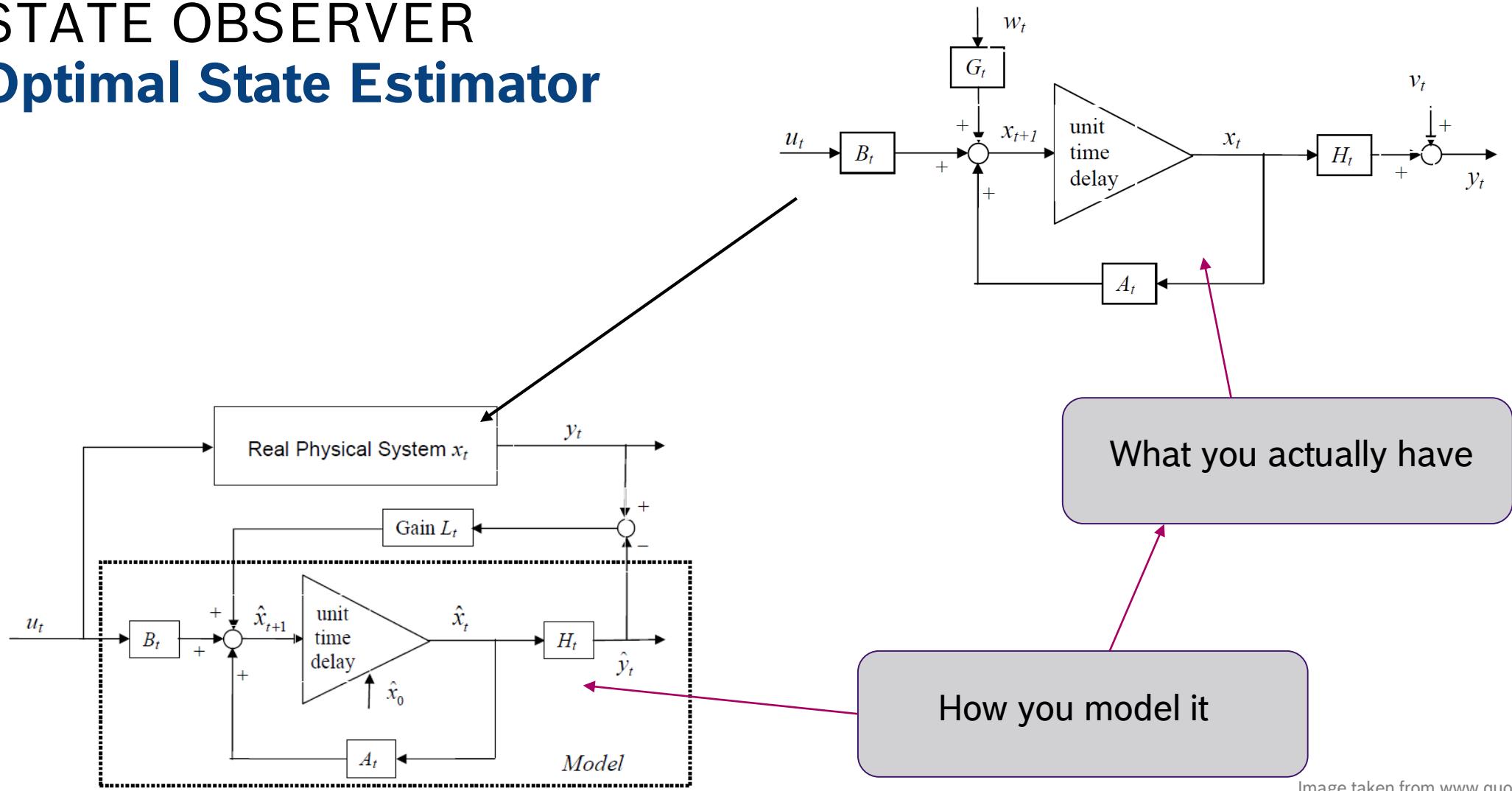
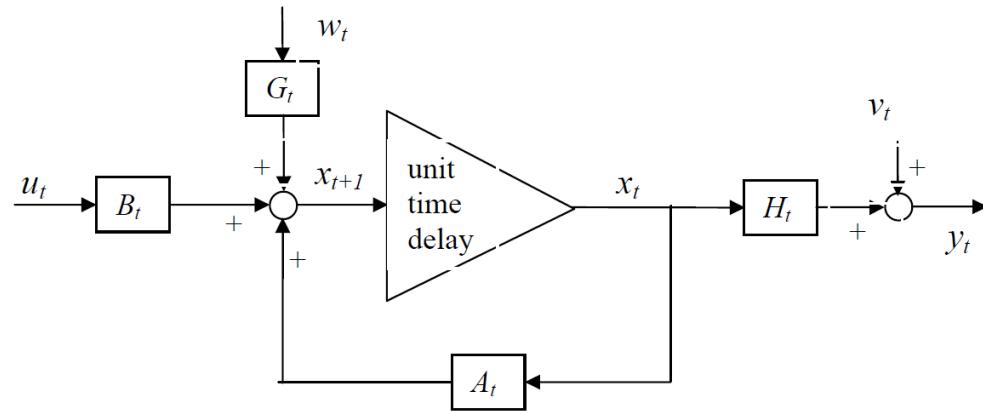


Image taken from www.quora.com

KALMAN FILTER

KALMAN FILTER

Optimal State Estimator For Stochastic Systems



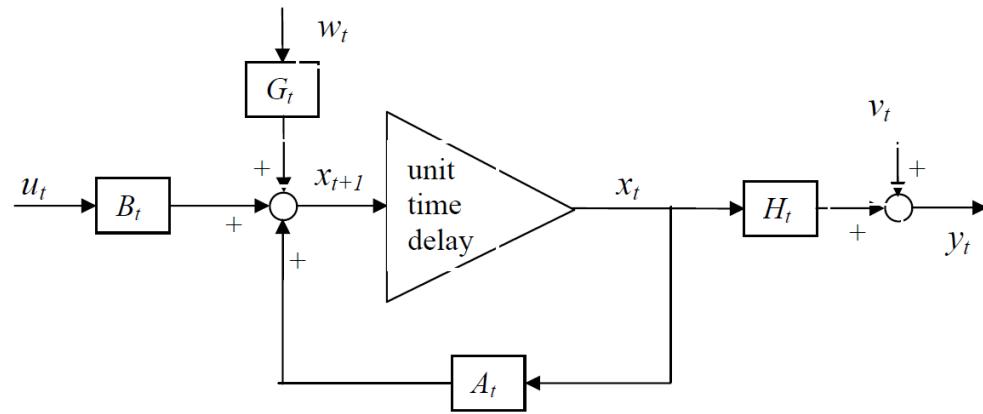
v_t = measurement noise
 w_t = process noise

- v_t, w_t – is assumed to be white noises
- $v_t = N(0, R_t)$, R_t is covariance of measurement noise
- $w_t = N(0, Q_t)$, Q_t is covariance of process noise

Obs.: Process and measurement noise are uncorrelated

KALMAN FILTER

Optimal State Estimator For Stochastic Systems



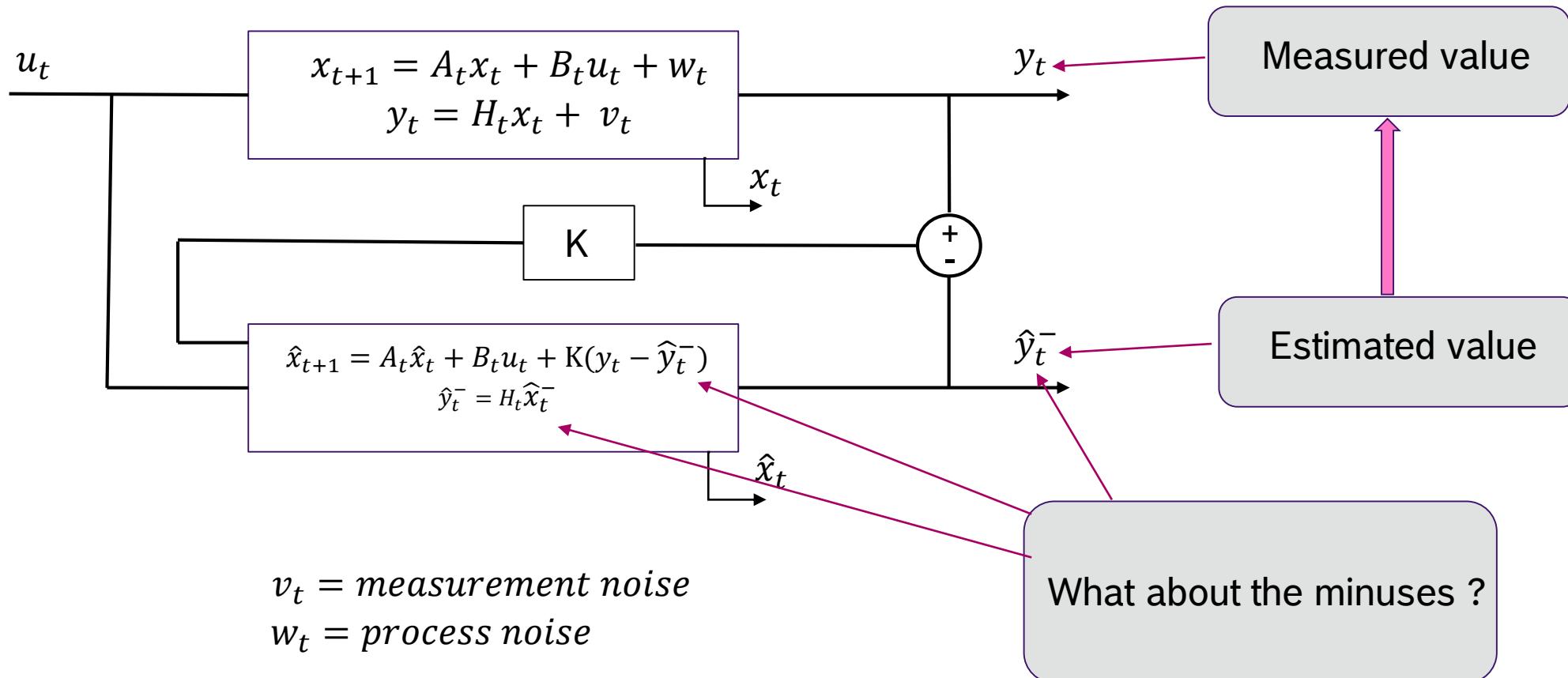
$$x_{t+1} = A_t x_t + B_t u_t + w_t$$
$$y_t = H_t x_t + v_t$$

v_t = measurement noise

w_t = process noise

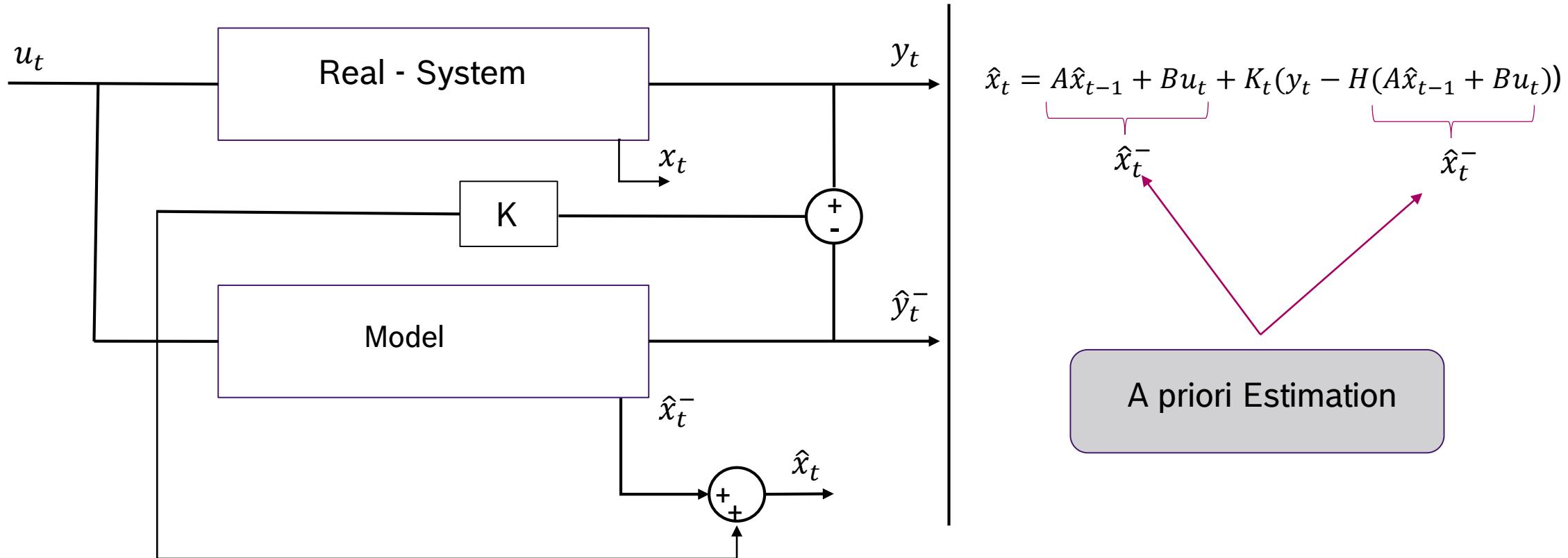
KALMAN FILTER

Optimal State Estimator For Stochastic Systems



KALMAN FILTER

Optimal State Estimator For Stochastic Systems



KALMAN FILTER

Optimal State Estimator For Stochastic Systems

$$\hat{x}_t = A\hat{x}_{t-1} + Bu_t + K_t(y_t - H(A\hat{x}_{t-1} + Bu_t))$$



$$\hat{x}_t = \hat{x}_t^- + K_t(y_t - H\hat{x}_t^-)$$

A posteriori State
Estimation

A priori State
Estimation

Correction via
Kalman Gain

KALMAN FILTER

Kalman Gain

- A posteriori estimation error:

$$\varepsilon = x_t - \hat{x}_t$$



Is a stochastic variable due to
process noise w_t

- Take the expected value of a posteriori error as an estimation's measure

$$f(K) = E[\varepsilon^T \varepsilon]$$

KALMAN FILTER

Kalman Gain

- Take the expected value of a posteriori error as an estimation measure

$$f(K) = E[\varepsilon^T \varepsilon]$$

- Minimizing f with respect to K will give us the best K

$$\frac{df}{dK} = 0 \longrightarrow K_t = \frac{P_t^- H^T}{H P_t^- H^T + R_t}, \text{ where } P_t^- \text{ is a priori error covariance}$$
$$P_t^- = E[(x_t - \hat{x}_t^-)^T (x_t - \hat{x}_t^-)]$$

KALMAN FILTER

Updating the error covariance

- Starting from the definition of the a posteriori error covariance

$$P_t = E[\varepsilon^T \varepsilon]$$

- Expanding the above equation by using the model of a posteriori estimation we obtain that

$$P_t = (I - K_t H) P_t^-$$

KALMAN FILTER

Estimation of a priori error covariance

- As we estimate a priori the state of the system, we must estimate the a priori error covariance in order to compute the Kalman gain
- Starting from the definition of the a priori error covariance

$$P_t^- = E[(x_t - \hat{x}_t^-)^T (x_t - \hat{x}_t^-)]$$

- By using the system transition model and considering the expected values of the independent signals to be 0 we obtain the a priori error covariance

$$P_t^- = AP_{t-1}A^T + Q_t$$

KALMAN FILTER

Summary

Prediction

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_{t-1}$$

$$P_t^- = AP_{t-1}A^T + Q_t$$

Update

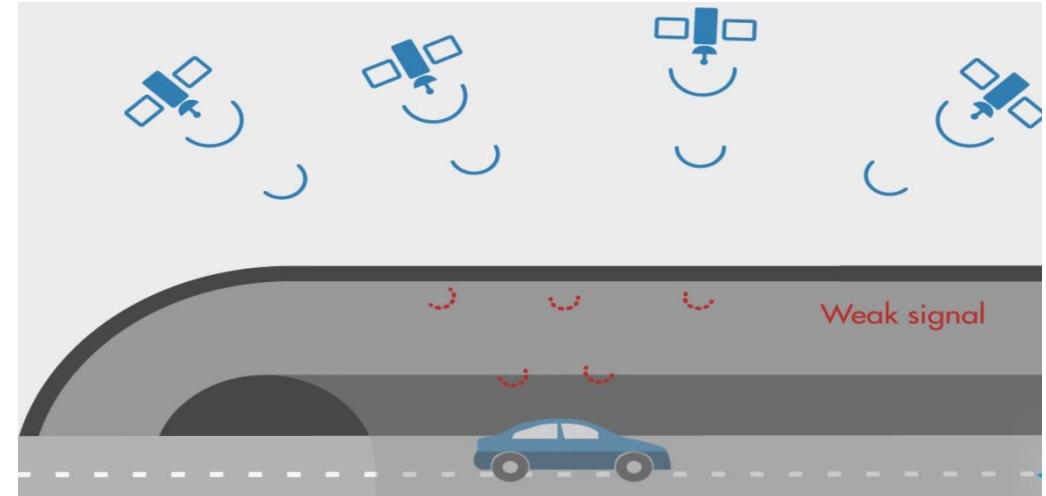
$$\hat{x}_t = \hat{x}_t^- + K_t(y_t - H\hat{x}_t^-)$$

$$K_t = \frac{P_t^- H^T}{H P_t^- H^T + R_t}$$

$$P_t = (I - K_t H)P_t^-$$

KALMAN FILTER

Vehicle tracking

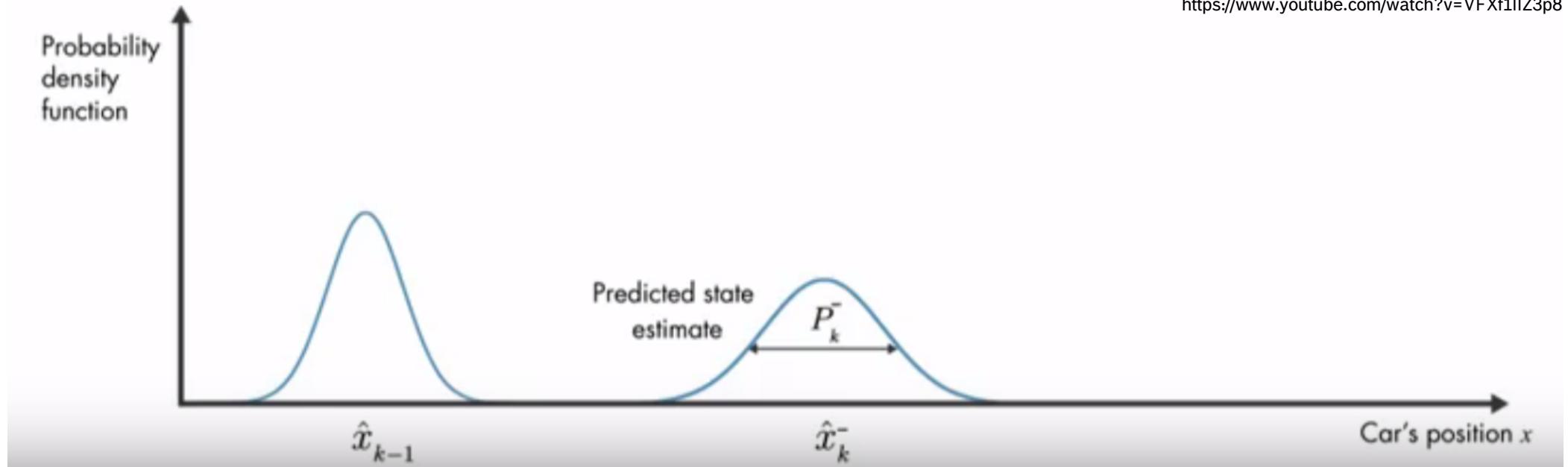


- Assume that the trajectory is a line
- You know the velocity of the car as input to the system
- You measure the position of the car via GPS system

KALMAN FILTER

Vehicle tracking - solution

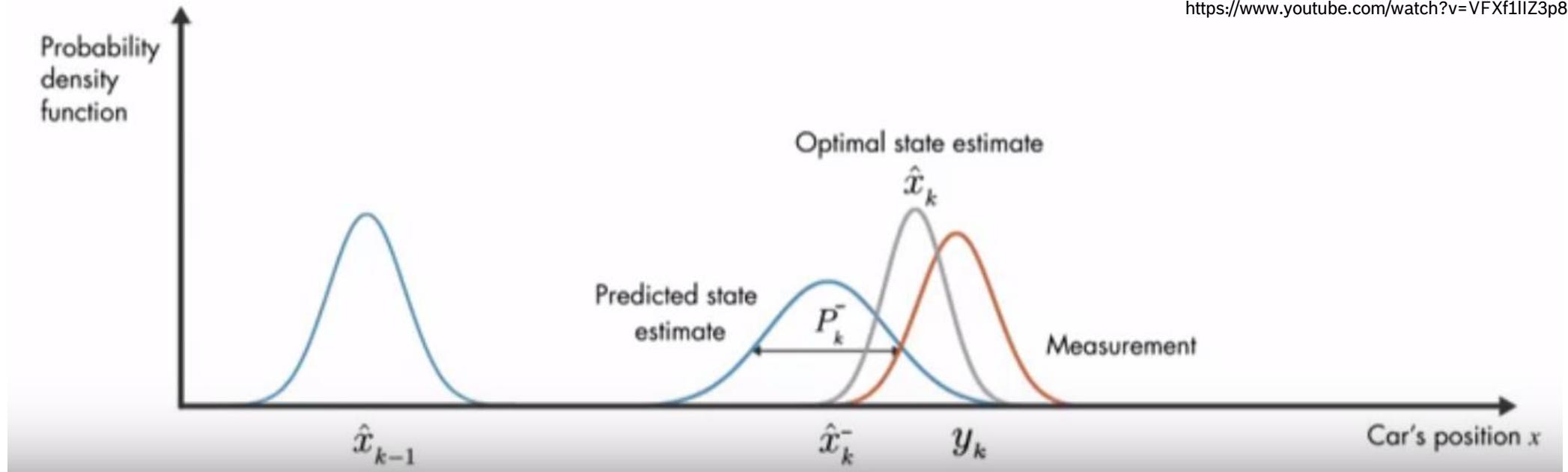
<https://www.youtube.com/watch?v=VFXf1lZ3p8>



KALMAN FILTER

Vehicle tracking - solution

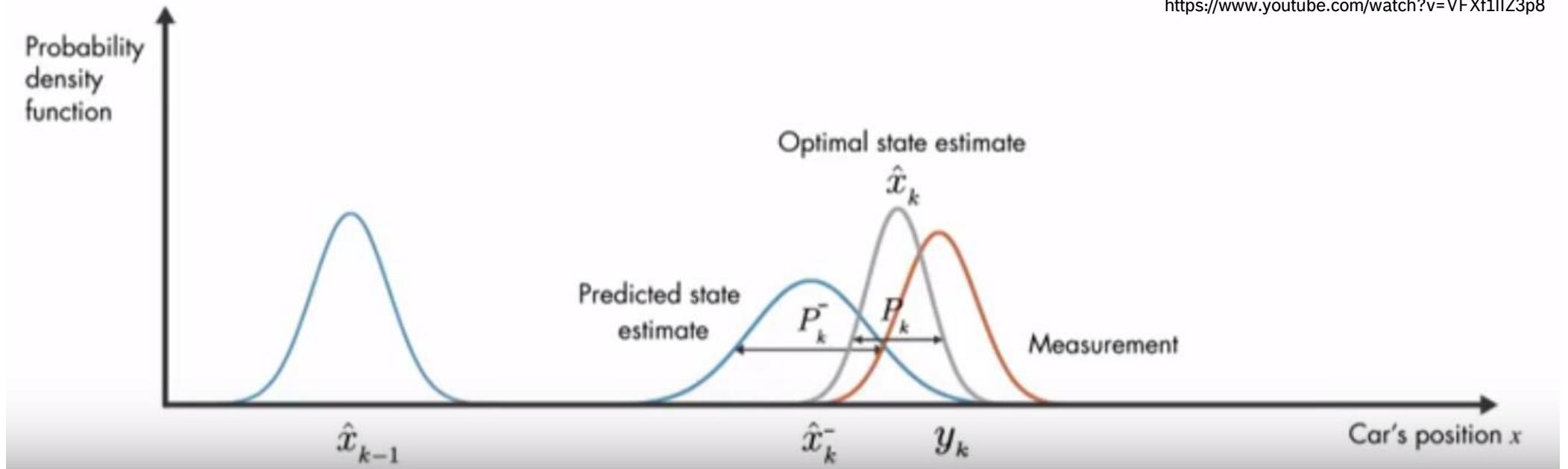
<https://www.youtube.com/watch?v=VFXf1lZ3p8>



KALMAN FILTER

Vehicle tracking - solution

<https://www.youtube.com/watch?v=VFXf1lZ3p8>



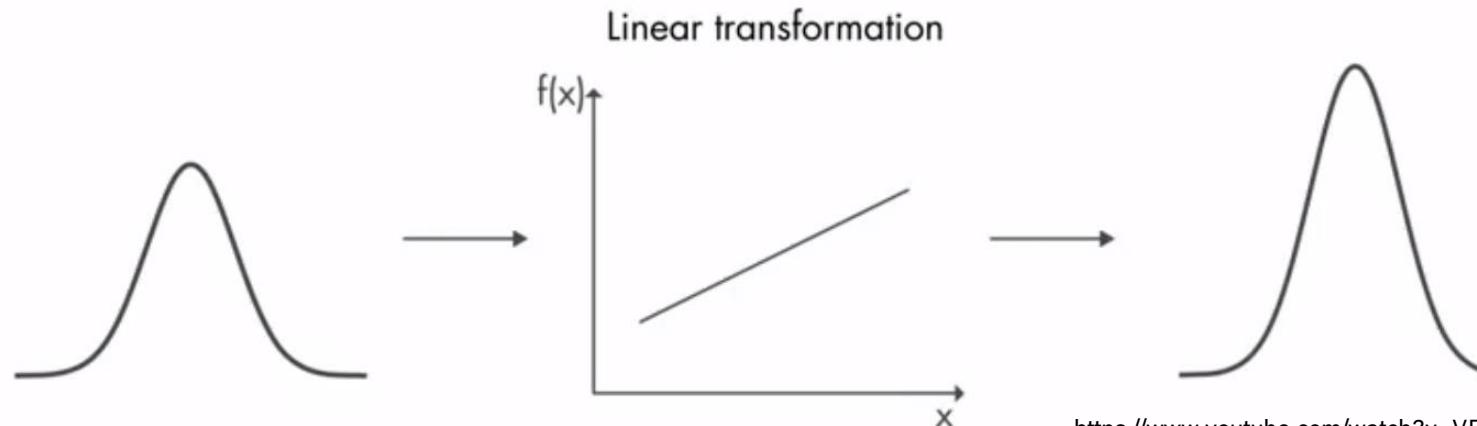
KALMAN FILTER

Principal drawback

- Kalman filter assume that the model of the system is linear

$$x_{t+1} = A_t x_t + B_t u_t$$
$$y_t = H_t x_t$$

- If we map an Gaussian distribution with a linear model we obtain another Gaussian distribution

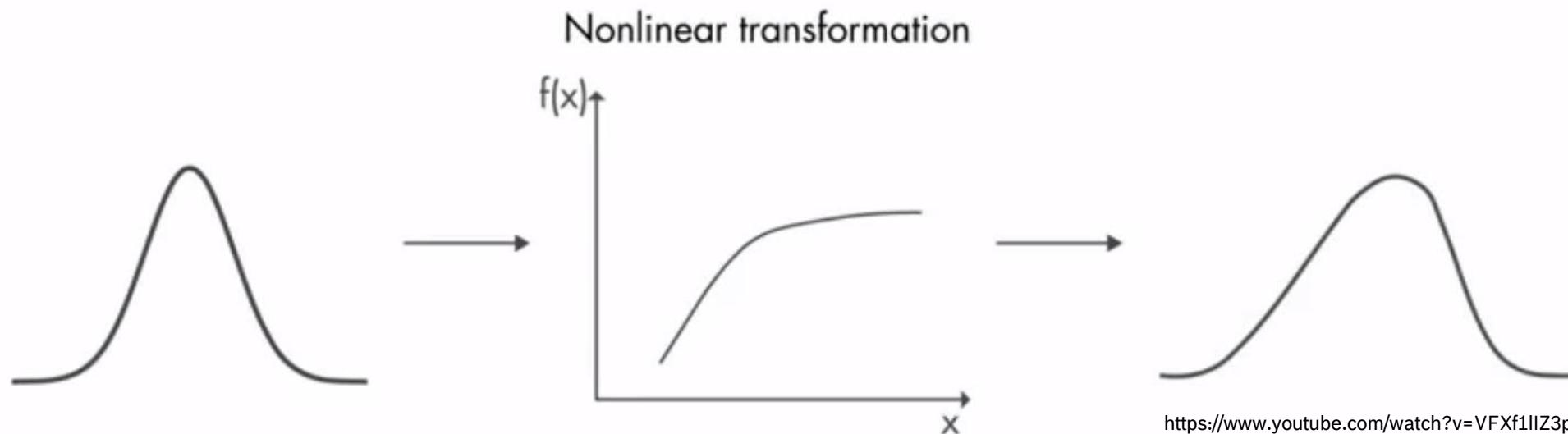


KALMAN FILTER

Principal drawback

- If model is nonlinear the Gaussian distribution is distorted

$$x_{t+1} = f(x_t, u_t)$$
$$y_t = g(x_t)$$



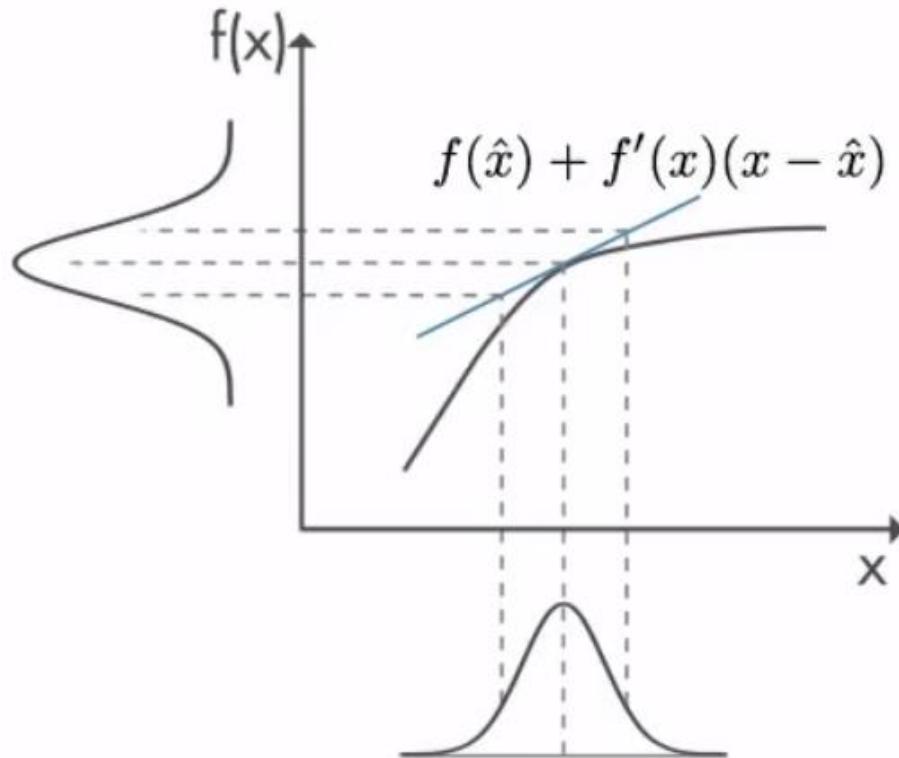
<https://www.youtube.com/watch?v=VFXf1lIZ3p8>

EXTENDED KALMAN FILTER

EXTENDED KALMAN FILTER

Model linearization

Nonlinear transformation

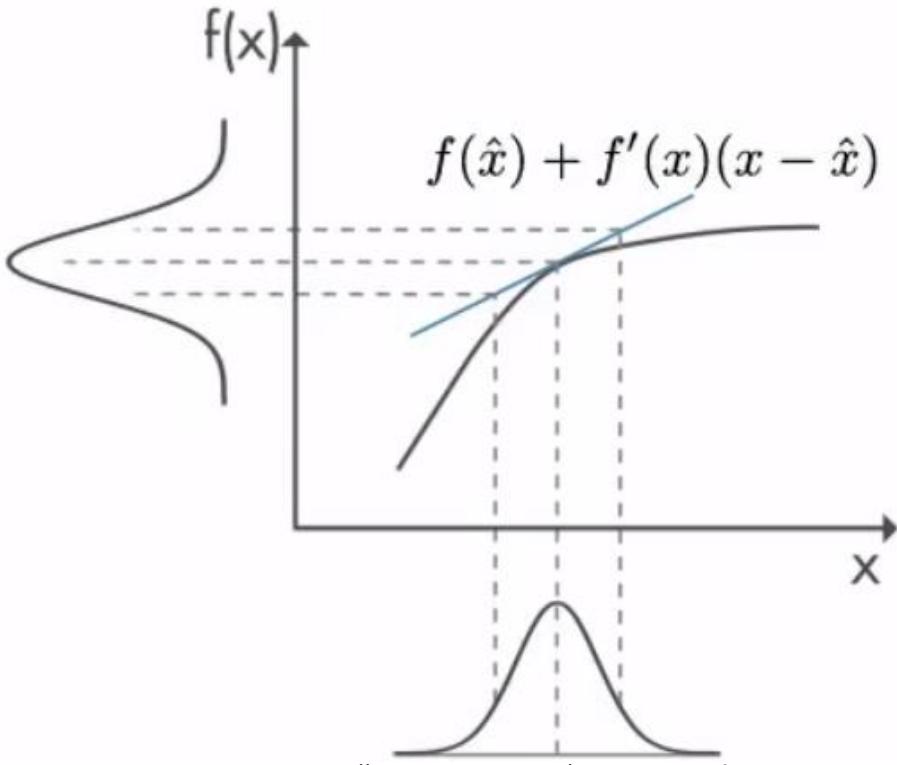


<https://www.youtube.com/watch?v=VFXf1lIZ3p8>

EXTENDED KALMAN FILTER

Model linearization

Nonlinear transformation



<https://www.youtube.com/watch?v=v=VFXf1lZ3p8>

$$f(x) = f(x_k) + \frac{\partial f}{\partial x}(x_k - x)$$

For multidimensional function with multivariable input, here we have the Jacobian matrix

EXTENDED KALMAN FILTER

Model linearization

$$\begin{aligned}x_{t+1} &= f(x_t, u_t) \\y_t &= g(x_t)\end{aligned}$$



$$\left. \begin{aligned}F_K &= \frac{\partial f}{\partial x} \Big|_{\hat{x}_{k-1}, \hat{u}_k} \\G_K &= \frac{\partial g}{\partial x} \Big|_{\hat{x}_k}\end{aligned}\right\}$$

Jacobians

EXTENDED KALMAN FILTER

Jacobian Matrix

$$\left\{ \begin{array}{l} F_K = \frac{\partial f}{\partial x} |_{\hat{x}_{k-1}, \hat{u}_k} \\ G_K = \frac{\partial g}{\partial x} |_{\hat{x}_k} \end{array} \right.$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

EXTENDED KALMAN FILTER

Modified Kalman Algorithm

Prediction

$$\hat{x}_t^- = f(\hat{x}_{t-1}, u_{t-1})$$

$$P_t^- = F P_{t-1} F^T + Q_t$$

Update

$$\hat{x}_t = \hat{x}_t^- + K_t(y_t - h(\hat{x}_t^-))$$

$$K_t = \frac{P_t^- G^T}{G P_t^- G^T + R_t}$$

$$P_t = (I - K_t G) P_t^-$$

At each step, Jacobians must be recomputed !!

EXTENDED KALMAN FILTER

Modified Kalman Algorithm - drawbacks

- Jacobian matrix is difficult to compute analytically
- Numerical computation of Jacobian has a high computational cost
- If system has hard nonlinear parts the first order Taylor approximation fails
- If system has hard nonlinear parts, the kalman filter is not an optimal approach

References

- https://ocw.mit.edu/courses/mechanical-engineering/2-160-identification-estimation-and-learning-spring-2006/lecture-notes/lecture_5.pdf
- https://ocw.mit.edu/courses/mechanical-engineering/2-160-identification-estimation-and-learning-spring-2006/lecture-notes/lecture_6.pdf
- https://ocw.mit.edu/courses/mechanical-engineering/2-160-identification-estimation-and-learning-spring-2006/lecture-notes/lecture_8.pdf

THANK YOU FOR YOUR ATTENTION

STEERING SYSTEMS

Agenda

- ▶ Steering Requirements
- ▶ Steering Systems Design
- ▶ Electric Power Steering
- ▶ Driver Assistance System Functions
- ▶ All Wheel Steering
- ▶ Steer by Wire
- ▶ Superimposed Steering System

Source:

Steering Handbook, Editors: Manfred Harrer, Peter Pfeffer
Springer International Publishing Switzerland 2017

STEERING REQUIREMENTS

Steering Requirements

Steering behaviour, Steering response

- ▶ **Steering behavior** – vehicle response to driver intention
 - ▶ the steer-angle has to correlate to the angle of the wheels by a continuous function
 - steering transmission may not produce any jumps

- ▶ **Steering response** – information transmitted by the steering system
 - ▶ useful information supporting the vehicle control
 - feedback on the limit of adhesion of the wheels
 - ▶ disturbing information
 - fluctuations of the braking power

Steering Requirements

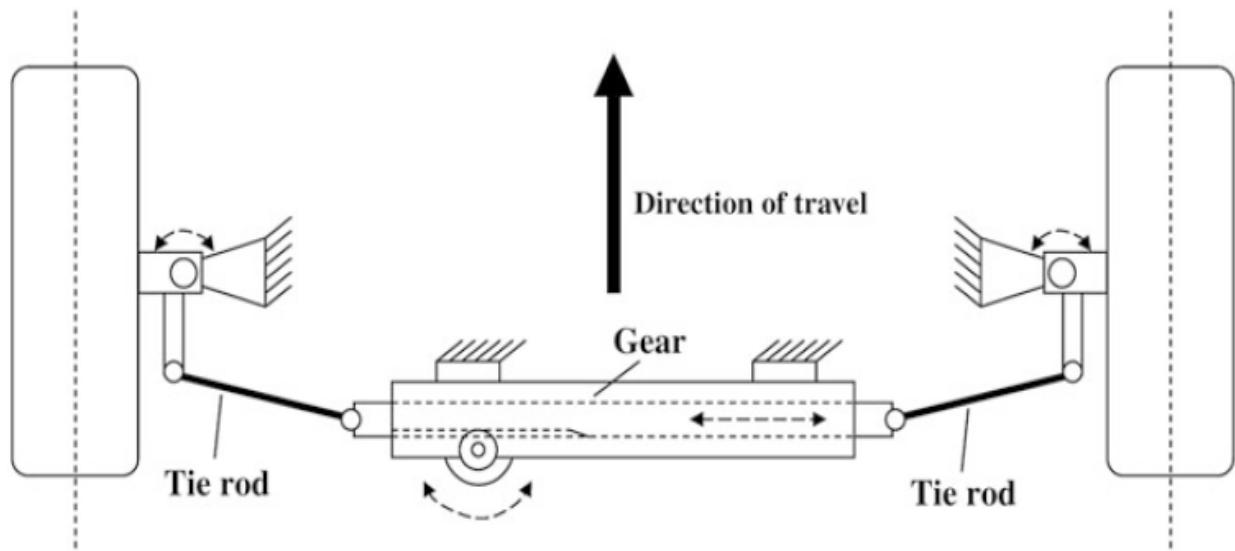
Steering function targets

- ▶ **Steering precision** - instantaneous response to steering input; synchronous behavior of steering angle input, steering torque increase, and vehicle response
- ▶ **Steering comfort** - steering wheel torque adapted to particular driving situations; low steering angle required for parking, cornering, and handling; automatic steering return with an adapted angular velocity of the steering wheel
- ▶ **Steering feedback** of driving state and road information in a balanced relationship with possible interfering variables
- ▶ **Steering dynamic** sufficient for quick maneuvering; a sudden evasive manoeuvre

Steering Requirements

- ▶ Steering behavior and response depend on

- tyres
 - axle kinematics
 - design of the steering system

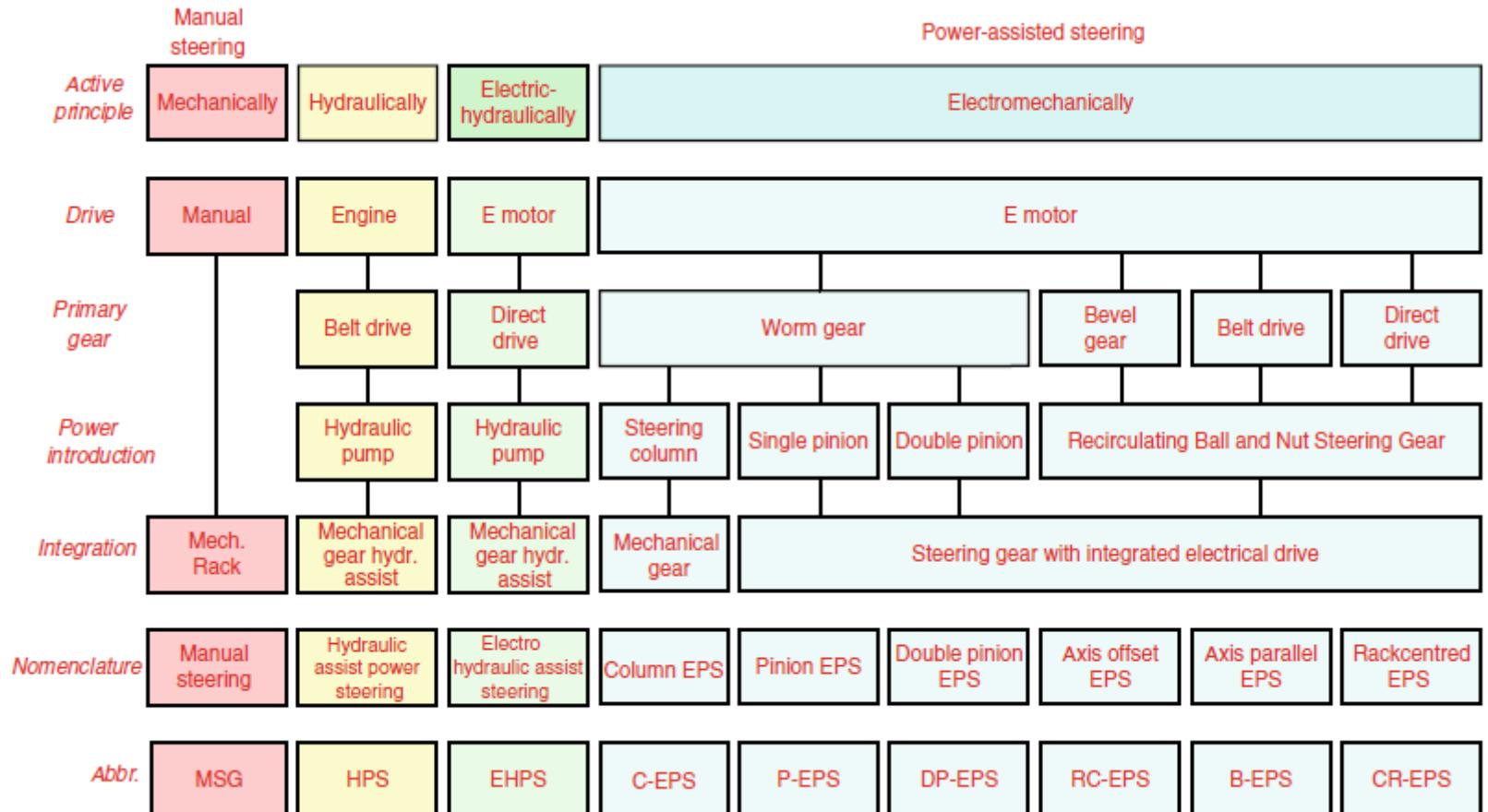


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

STEERING SYSTEMS DESIGN

Steering Systems Design

Classifications of steering systems design

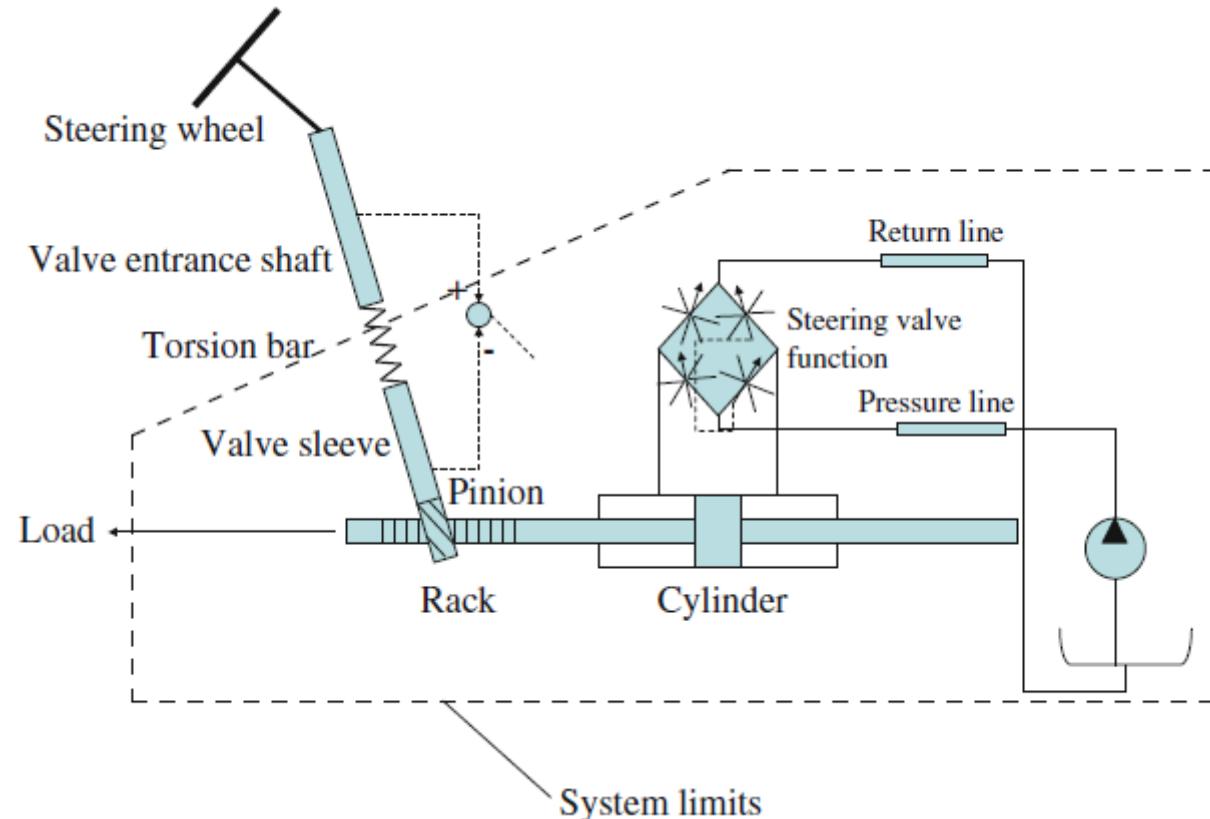


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Systems Design

Hydraulic power steering (HPS)

- ▶ In a HPS system, the power-assist is activated by opening valve connected to the torsion bar
- ▶ the level of the power-assist is a mechanical function of the valve characteristic

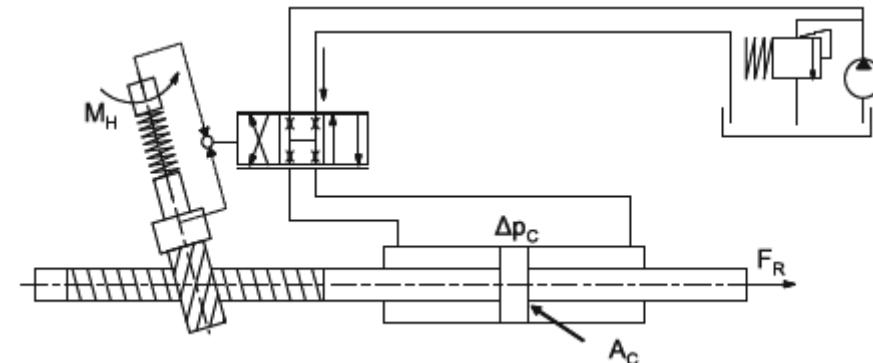
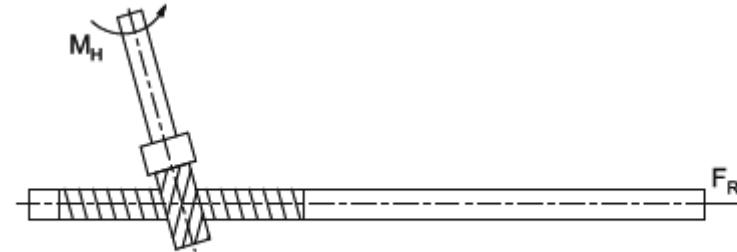


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Systems Design

Comparison between mechanical and hydraulic steering

- ▶ HPS advantages:
 - ▶ reduce steering force
 - ▶ reduce steering ratio
 - ▶ increase the damping of the steering system

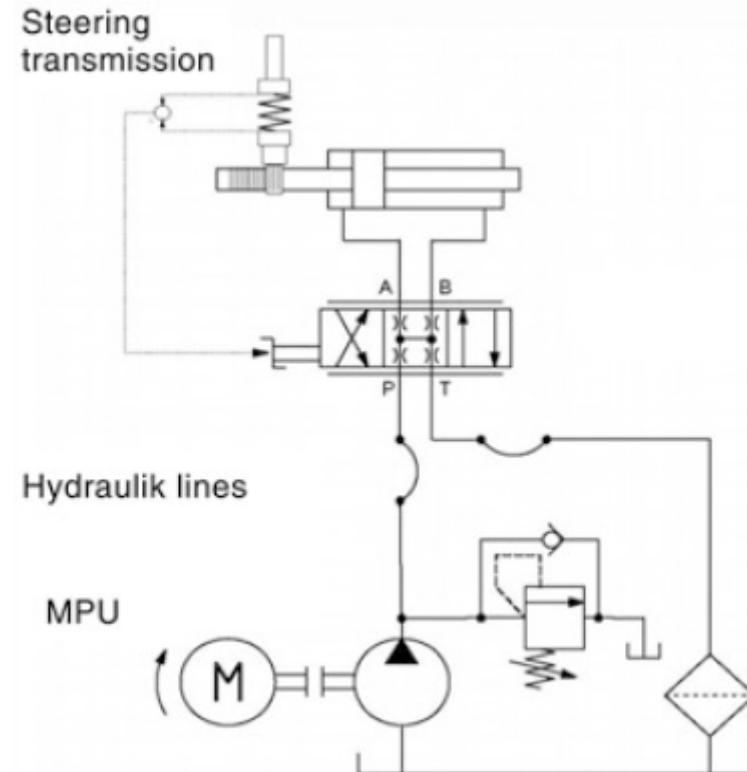


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Systems Design

Electrically powered hydraulic

- ▶ the hydraulic pump is controlled by an electric motor

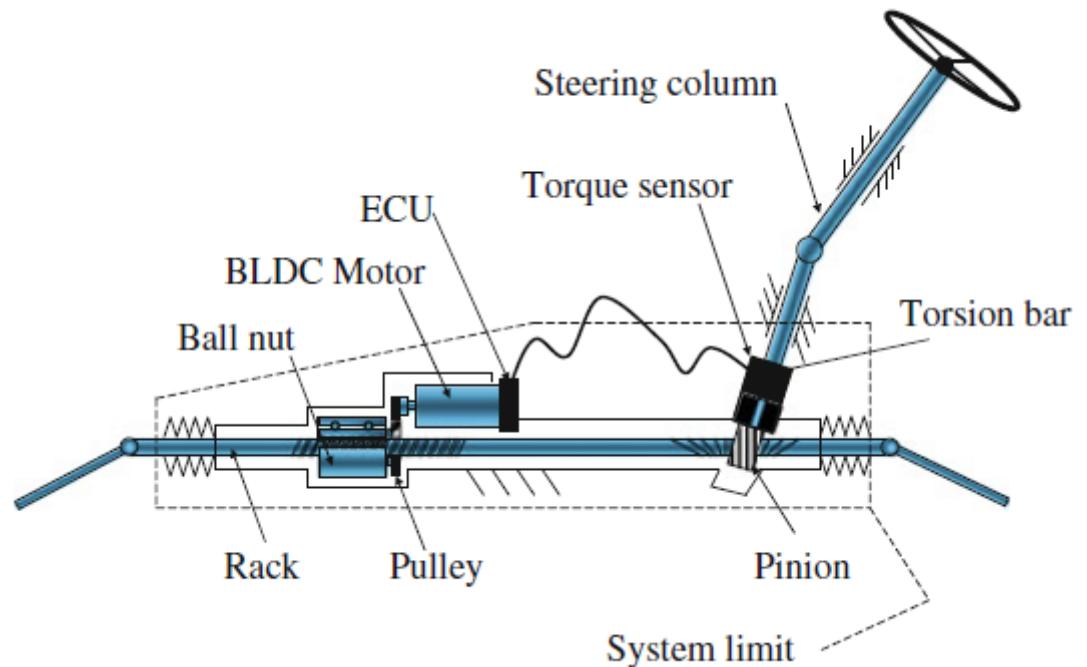


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Systems Design

Electric power steering

- ▶ EPS generates the power-assist by means of an electric motor whose force is fed into the rack or steering column by a servo gear unit
- ▶ EPS uses torque sensors to measure the torsion bar torque
- ▶ the power-assistance is computed in the EPS-ECU using the measured torsion bar torque

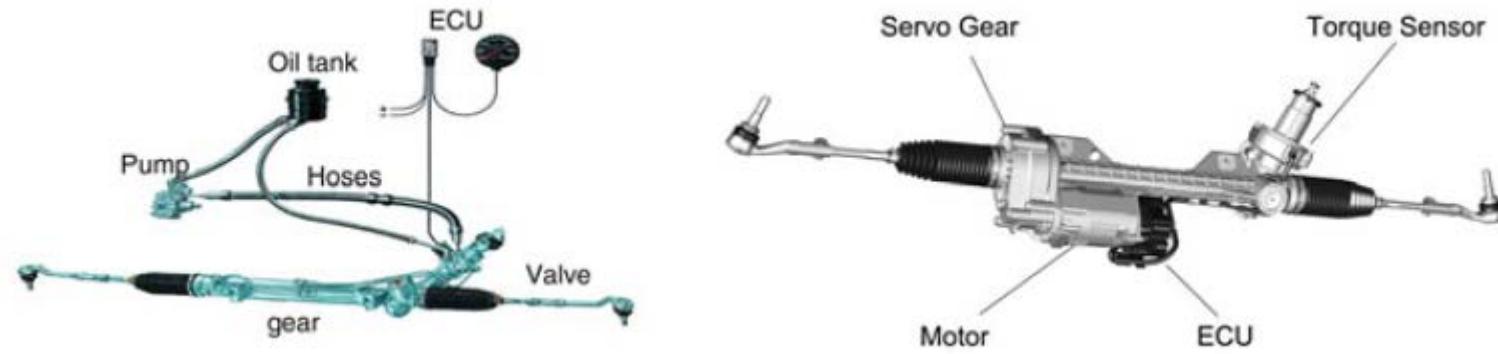


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Systems Design

Comparison between hydraulic and electric power steering

- ▶ HPS has many individual parts that are usually assembled on-board; the system has to be hydraulically filled and the connections tested for leakage
- ▶ EPS is supplied to the vehicle manufacturer as a complete unit
- ▶ EPS has additional functions

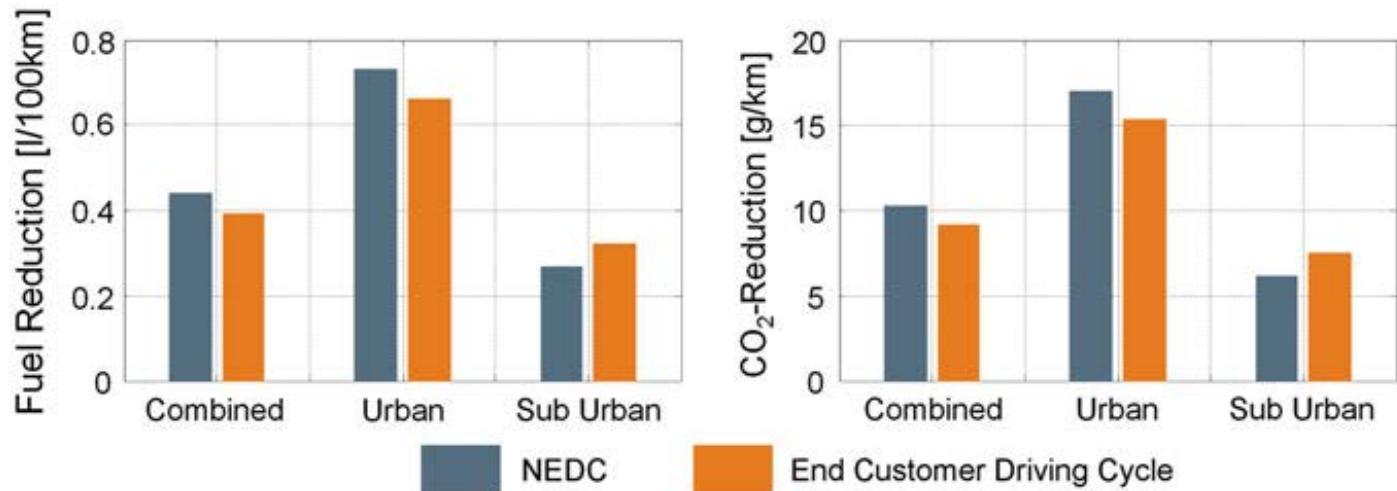


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Systems Design

Comparison between hydraulic and electric power steering

- ▶ EPS is a power on demand system, activated only when the car is steering
 - ▶ fuel and CO₂ reduction
- ▶ Savings on fuel and CO₂ of EPS compared with HPS
 - ▶ NEDC- New European Driving Cycle
 - ▶ measurements on BMW 320i

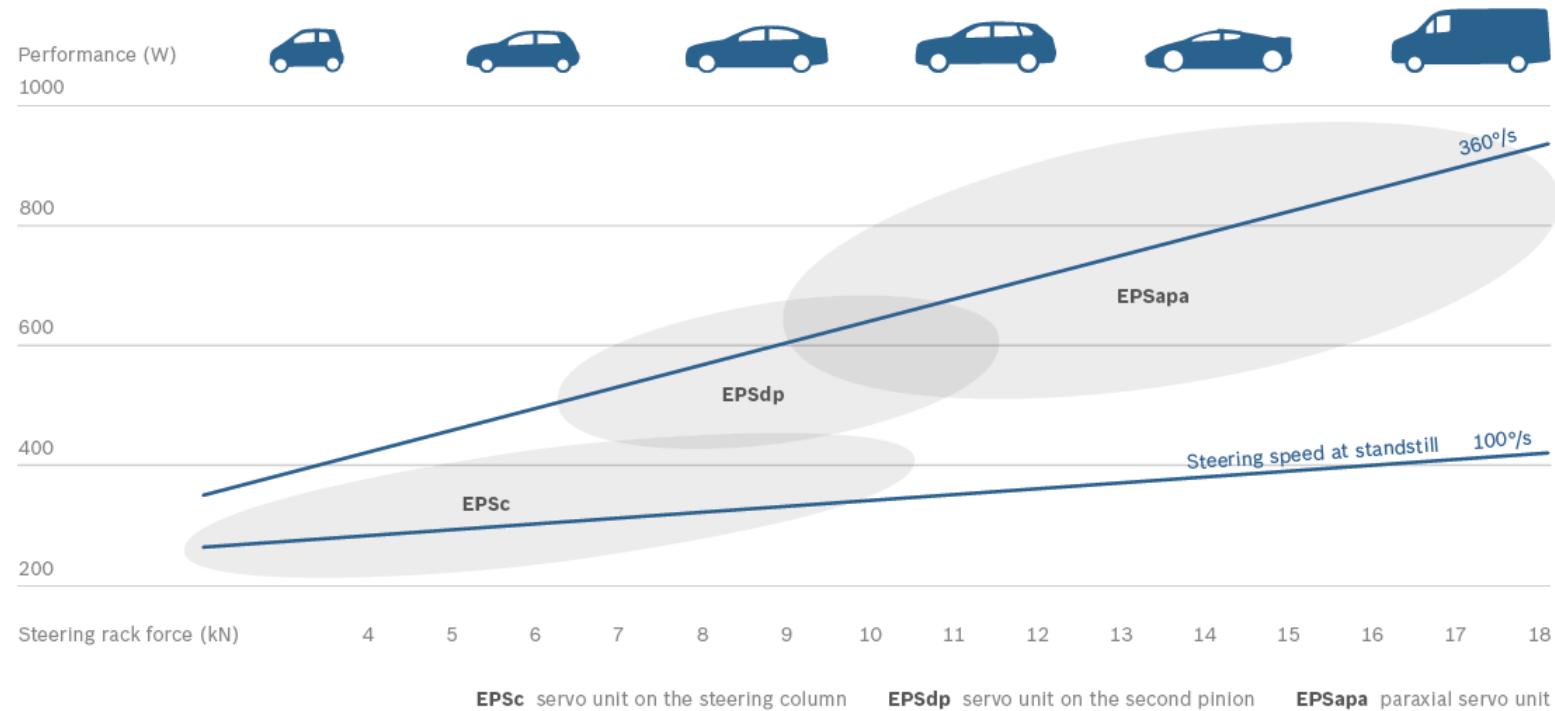


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

ELECTRIC POWER STEERING

Electric Power Steering

Application range of EPS



Electric Power Steering Designs of EPS Systems

- ▶ EPSc: Column
- ▶ EPSp: Pinion
- ▶ EPSdp: Dual Pinion
- ▶ EPSapa: Axle Parallel
- ▶ EPSapa Fail-operational
- ▶ EPSrc: Rack Concentric

Electric Power Steering

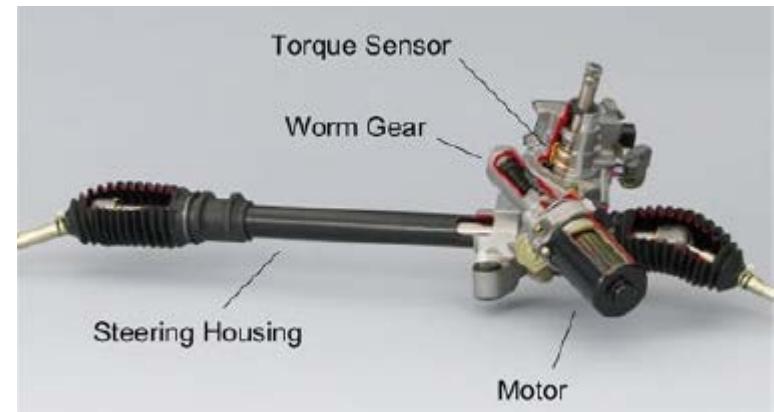
EPSc

- ▶ the forces of the power-assist unit are transferred along steering column
- ▶ high steering forces are not accessible; limiting factors are intermediate steering shaft and pinion
- ▶ for vehicles up to lower mid-range
- ▶ low weight and minimal space requirements



Electric Power Steering EPSp

- ▶ The power-assist unit is placed right at the steering pinion
- ▶ The forces do not need to be transferred along steering column and intermediate steering shaft
- ▶ Higher steering powers than an EPSc



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

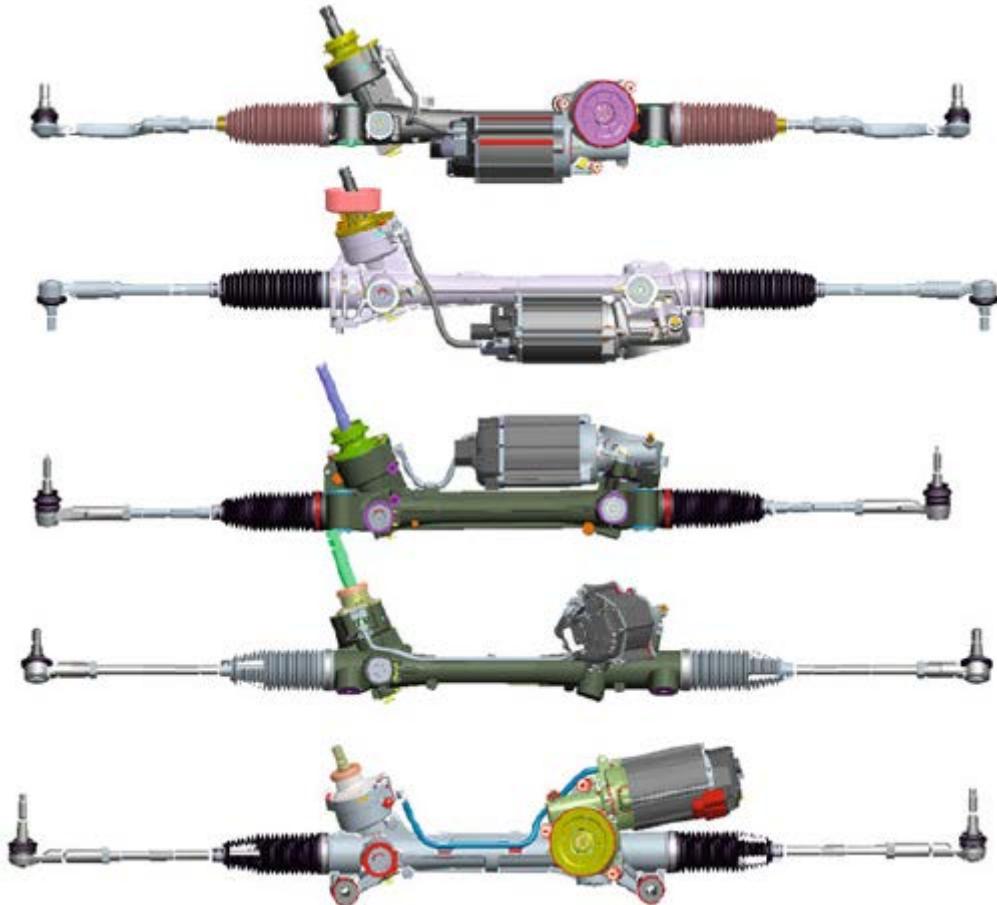
Electric Power Steering EPSdp

- ▶ the power-assist unit is mounted at a second pinion
- ▶ sensor unit and drive unit can be separated
- ▶ system power is 10–15 % higher than that of an EPSc or EPSp
- ▶ for mid-range vehicles
- ▶ versatile installation possibilities



Electric Power Steering EPSdp

- ▶ the servo unit can be positioned to rotate 360 degrees about the axes of the rack and the drive pinion through use of a suitably designed worm gear
- ▶ adapting the steering to very difficult installation space



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Electric Power Steering

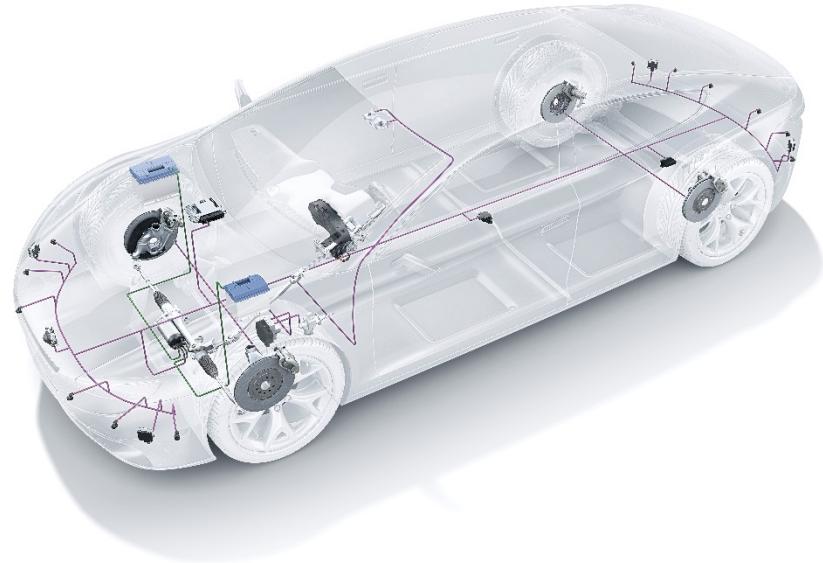
EPSapa

- ▶ the power-assistance is transferred to the rack by a combination of ball screw and timing belt gearbox
- ▶ the ball screw converts the rotation of the engine into a translation of the rack
- ▶ for luxury-class vehicles, sports cars, SUVs and light commercial vehicles
- ▶ high efficiency and low system friction



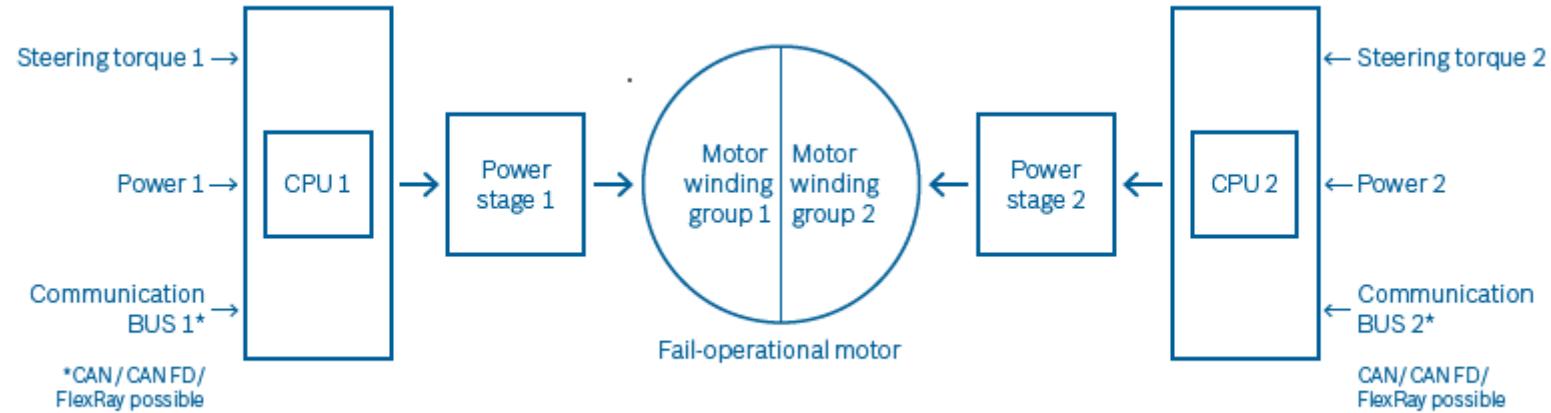
Electric Power Steering EPSapa Fail-operational

- ▶ allows highly automated driving
- ▶ functions and characteristics of standard steering are retained in the non-automated mode
- ▶ based on paraxial electromechanical steering
- ▶ redundant system design
- ▶ in case of a fault in the electronics, at least 50% of steering assistance is retain

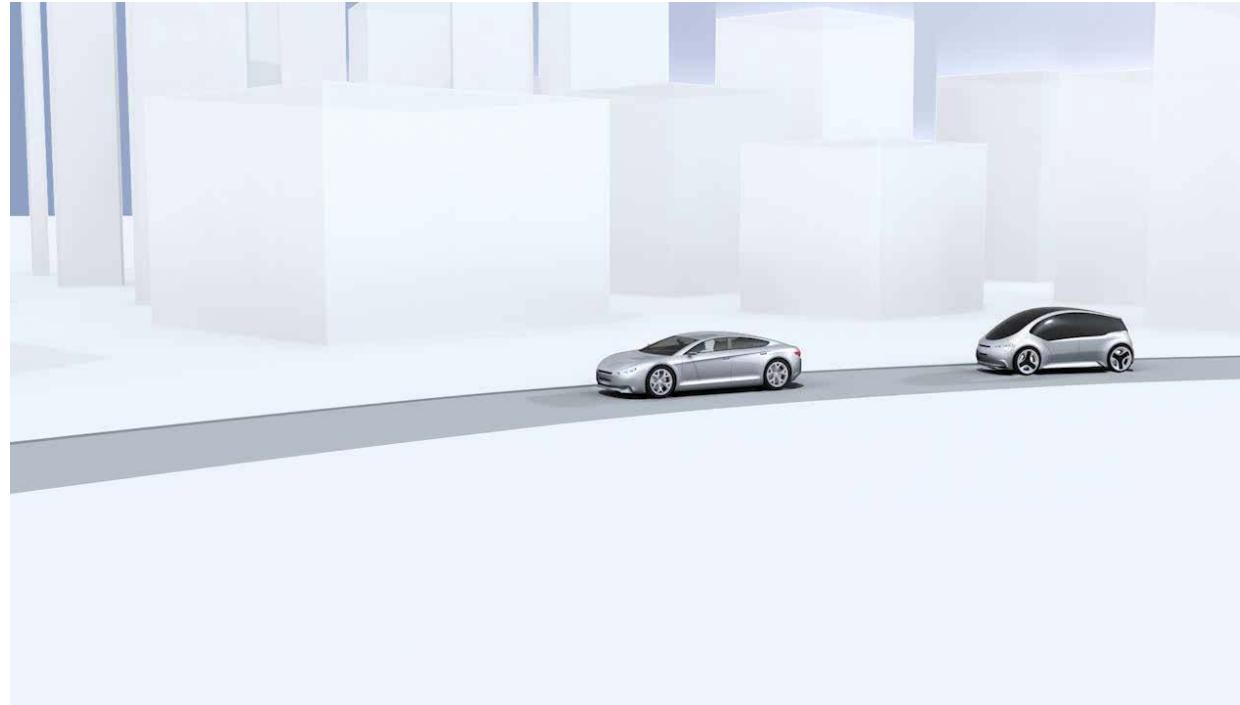


Electric Power Steering

Structure of a Fail-operational steering system



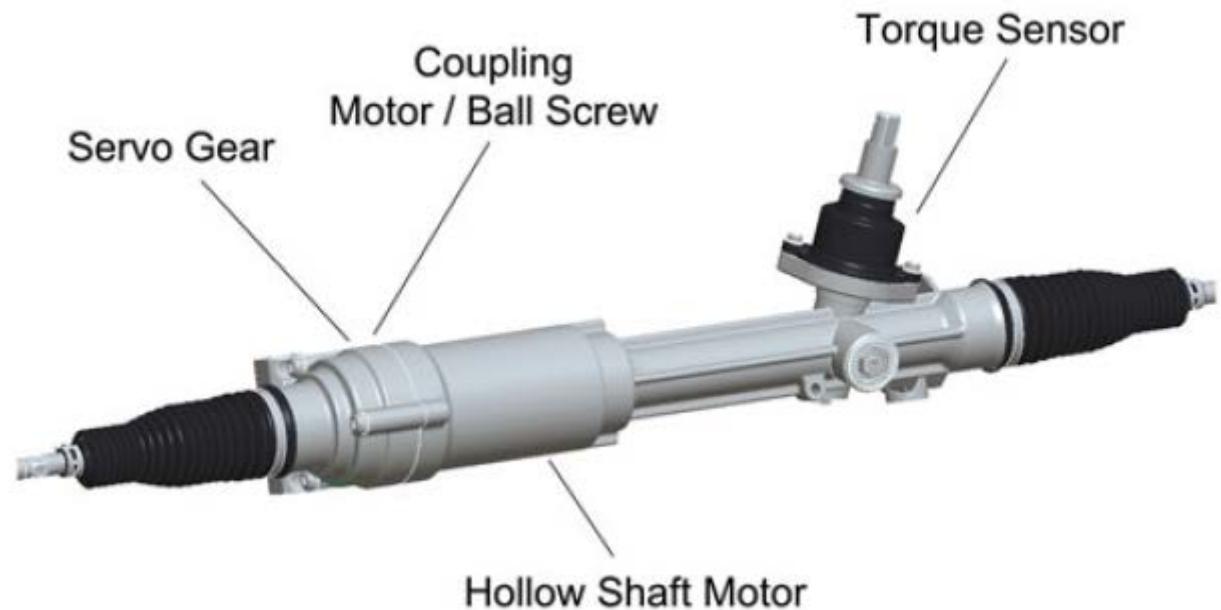
Electric Power Steering EPSapa Fail-operational



Electric Power Steering

EPSrc

- ▶ concentric configuration requires a special servo motor with hollow shaft rotor; the rack of the steering passes through the motor
- ▶ uses a ball screw as a gear to convert the rotation of the engine into a translation of the rack
- ▶ the ball screw is here directly driven by an electric motor



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

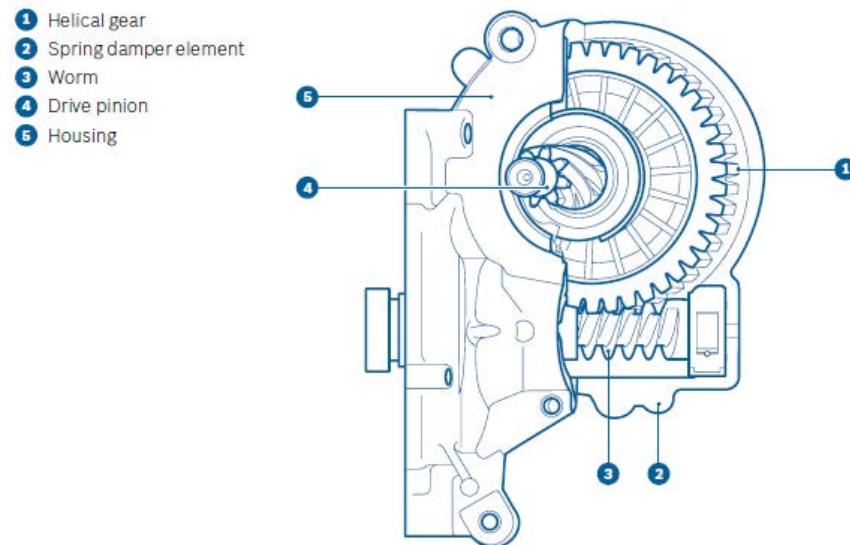
Electric Power Steering

Power-Assisted Gear

- ▶ establishes the connection between steering wheel, drive unit and wheels
- ▶ convert the rotation of the power-assist into a translation of the rack
- ▶ high transmission ratios can be achieved by a combination of two gearbox layers.
- ▶ power assisted gears used for EPS:
 - Worm Gear
 - Ball Screw Drive
 - Toothed Belt Drive

Electric Power Steering Worm Gear

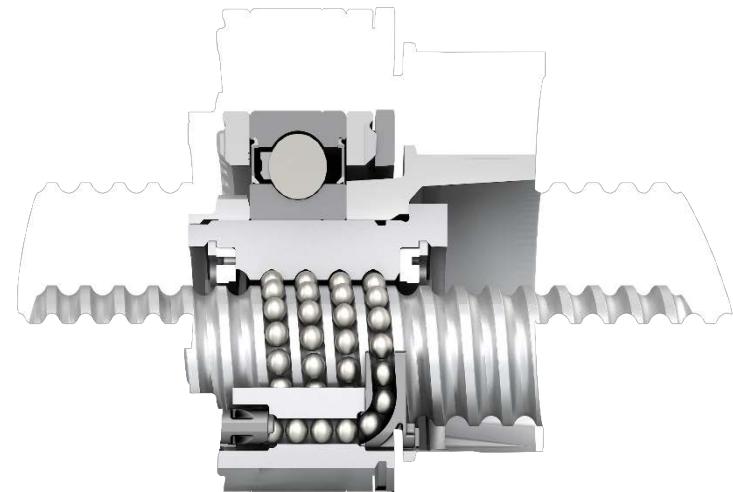
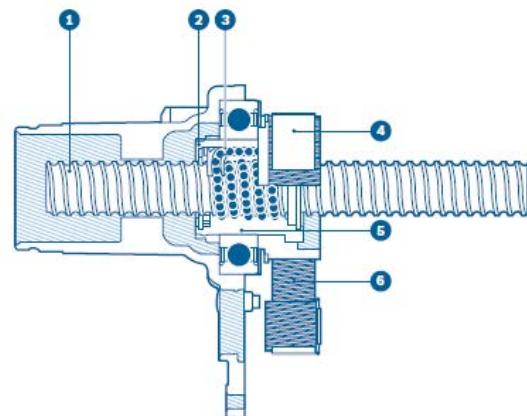
- ▶ used in the EPSdp to transmit power between electric motor and main drive pinion
- ▶ used in EPSc and ESPp to transfer the motor power to the steering column



Electric Power Steering Ball Screw Drive

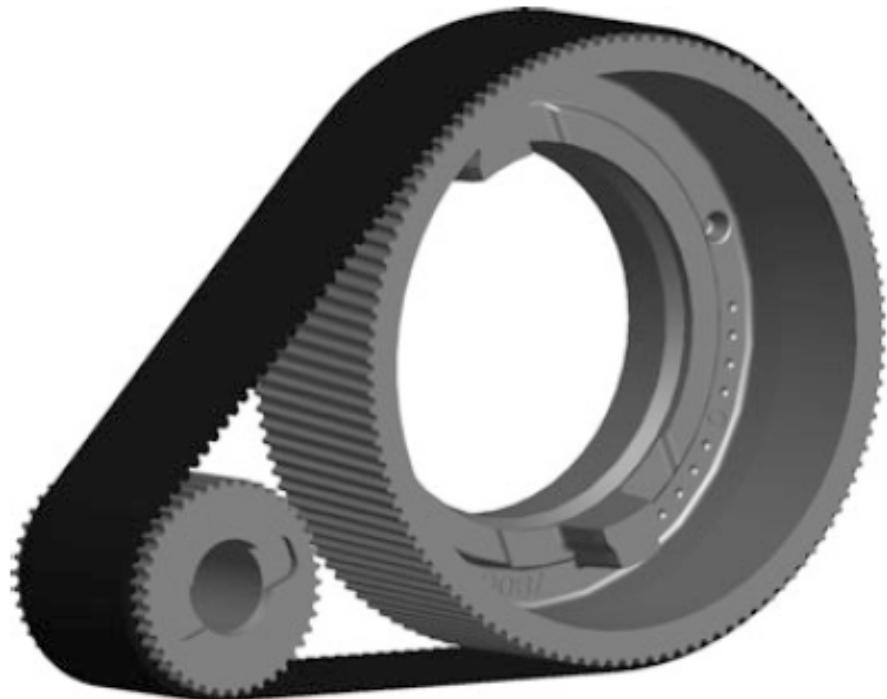
- ▶ transform the rotation of the electric motor in an EPSSpa or an EPSSrc system into a translation of the rack
- ▶ the drive comes either directly (EPSSrc) or via a belt gearbox (EPSSpa) from the motor

- ➊ Steering rack
- ➋ Ball return channel
- ➌ Ball chain
- ➍ Toothed disc
- ➎ Ball recirculating nut
- ➏ Toothed belt



Electric Power Steering Toothed Belt Drive

- ▶ used in ESPapa, transfer the motor power to the rack
- ▶ motor axle and rack are axle-parallel
- ▶ the toothed belt drives consist of a belt and two serrated pulleys



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Electric Power Steering

Electric Motor

- ▶ power assist is supplied by the electric motor
- ▶ it converts electric energy into mechanical energy
- ▶ the operating range of an electric motor for EPS can be divided into a
 - ▶ speed range with constant torque: parking maneuver (high steering force up to a defined steering speed)
 - ▶ speed range with almost constant output power: evasive manoeuvres (high steering speed are required)



Electric Power Steering

Steering Functions

- ▶ define the force that the driver perceives while guiding the steering wheel
- ▶ response of the free steering wheel (automatic parking)
- ▶ steering functions classifications:
 - ▶ basic steering functions
 - power-assistance, friction compensation, inertia compensation, damping
 - ▶ extended steering functions (EPS specific)
 - Active Return
 - ▶ functions at vehicle level (Driver Assistance System Functions)
 - Park Steering Assistant, Lane Departure Warning, Lane Keeping System, Driver Steering Recommendation(DSR)

DRIVER ASSISTANCE SYSTEM FUNCTIONS

Driver Assistance System Functions

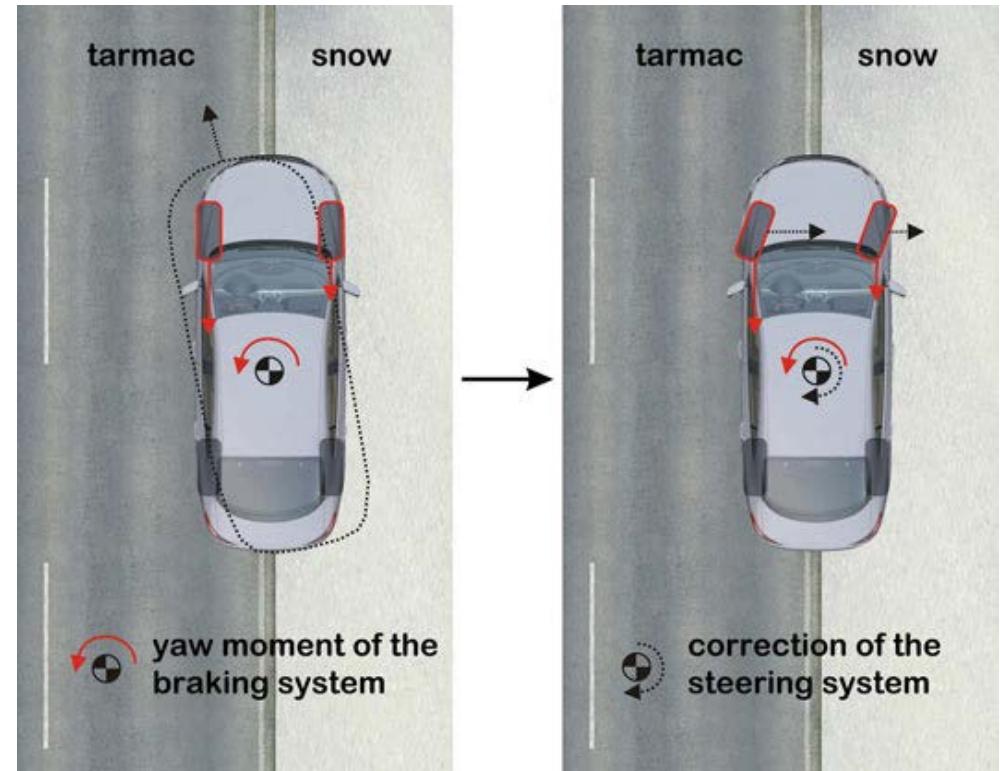
Overview

- ▶ determines the current driving situation on the basis of steer-angle, steering torques, lateral acceleration, yaw rate sensors and ESP-computed values
- ▶ identifies the best steer-angle for the present driving situation
- ▶ is computed and superimposed a steering torque to the steering wheel
- ▶ the additional torque should motivate the driver to adapt the self-chosen steering wheel angle to the ideal one
- ▶ increase the efficiency of ESP/ABS systems

Header of section

Driver Steering Recommendation(DSR)

- ▶ improvement of handling and braking distance
- ▶ support the driver in braking maneuvers when there are asymmetrical friction values on the road (μ -split)
 - ▶ based on the yaw rate of the vehicle and the brake pressure difference at the front wheels, DSR computes the corrective steering wheel
 - ▶ the driver has to steer towards the low friction value to counteract the pull towards higher friction

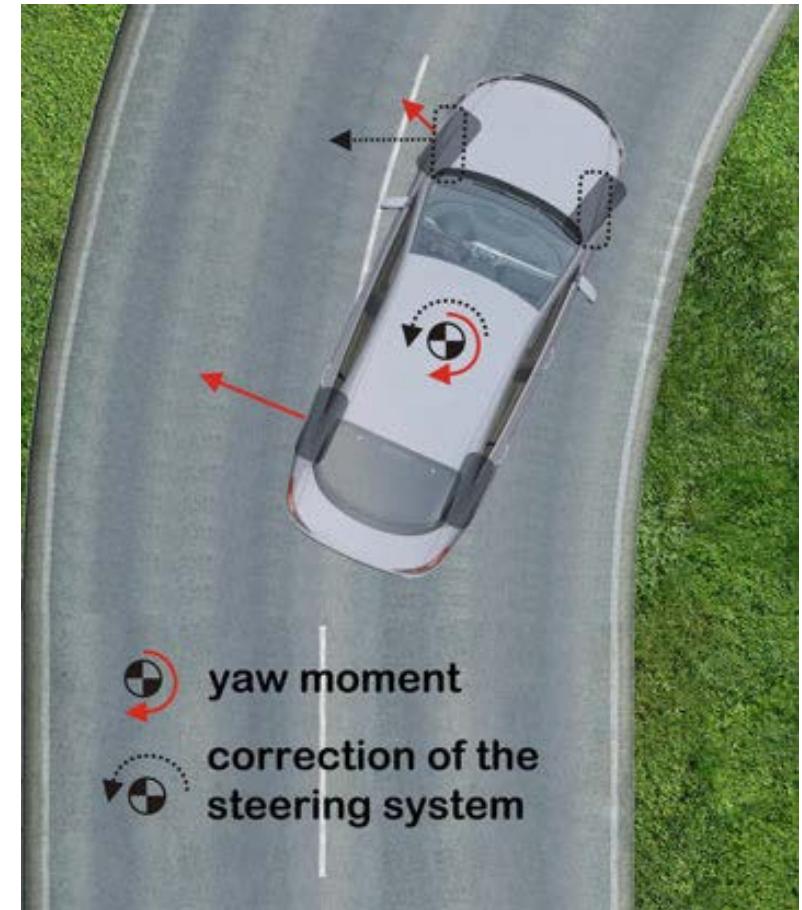


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Driver Steering Recommendation(DSR)

- ▶ support the driver in oversteering conditions
 - ▶ in a oversteering situation yaw moment is generated pushing the car rear outward
 - ▶ the driver has to compensate the yawing movement by counter steering
 - ▶ DSR provides a superimposed steering wheel torque to the driver, pointing where to steer
 - ▶ the driver's counter steering movement is optimized and ESP interventions can be avoided or reduced.



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Driver Steering Recommendation(DSR)

- ▶ Ergonomics Requirements
 - ▶ steering wheel torque interventions have to be always continuous (may not include any jumps)
 - ▶ possibility to control the steering characteristics by parameters
 - ▶ limitation of the highest additional steering wheel torques, the function has to be deactivated if the driver does not agree with the recommended steering wheel torques.
- ▶ Safety Requirements
 - ▶ the driver should be able to control the driving situation whenever an additional steering wheel torque is applied
 - ▶ monitoring and limitation of the steering wheel torques and their gradients

Driver Assistance System Functions

Lane Keeping Systems (LKS)

- ▶ support the driver in keeping the lane by intervening with a correcting wheel torque
- ▶ components of the lane keeping system:
 - ▶ camera and ECU (1)
 - ▶ electromechanical steering (2)
 - ▶ multi function steering wheel (3)
 - ▶ instrument cluster (4)

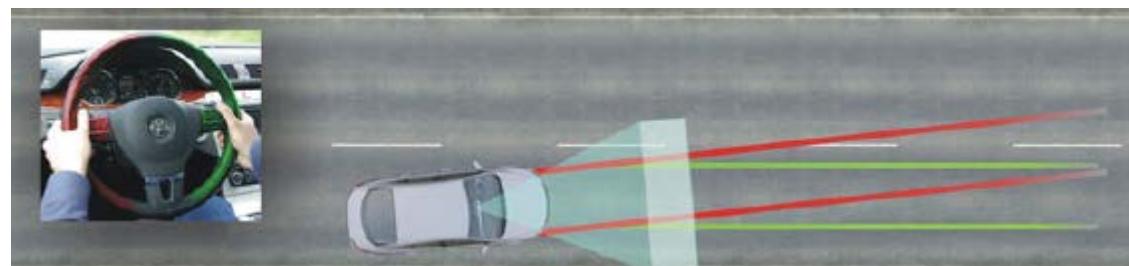


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Lane Keeping Systems (LKS)

- ▶ the lane keeping system is designed for use on well developed country roads and highways
- ▶ the system only switches to the ‘active’ state when the following criteria are fulfilled:
 - ▶ lane recognised
 - ▶ lane is wide enough
 - ▶ curvature of the lane is small enough
 - ▶ speed is faster than 65 km/h
- ▶ If the vehicle deviates from his lane, the lane assist will countersteer

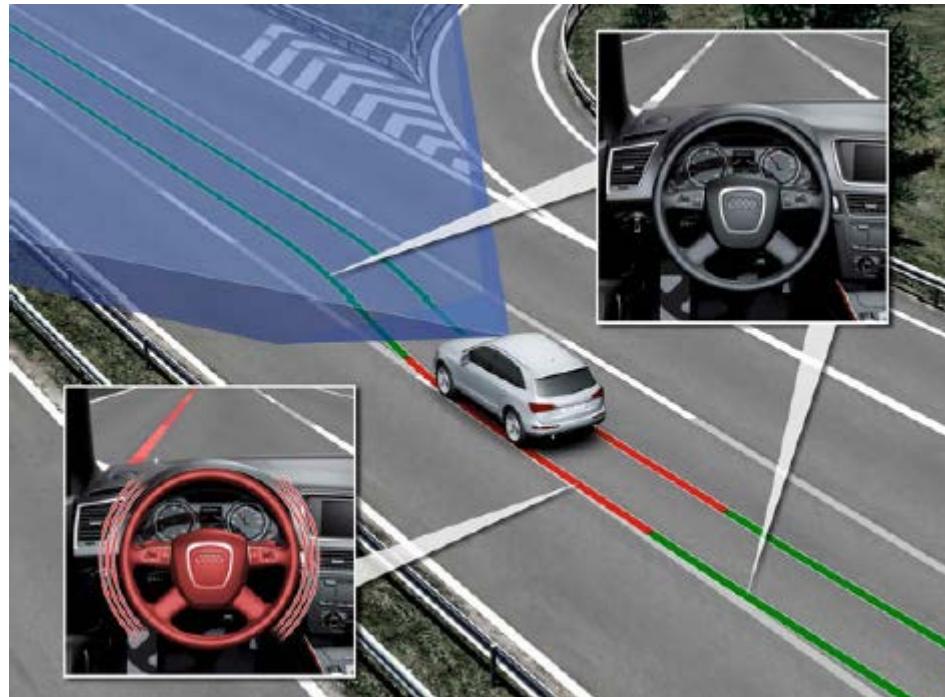


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Lane Departure Warning (LDW)

- ▶ warn the driver by an optical, acoustic or haptic signal that the lane was left, but they do not interfere into the lane guidance
- ▶ activation limit 65km/h
- ▶ function status displayed in the instrument cluster

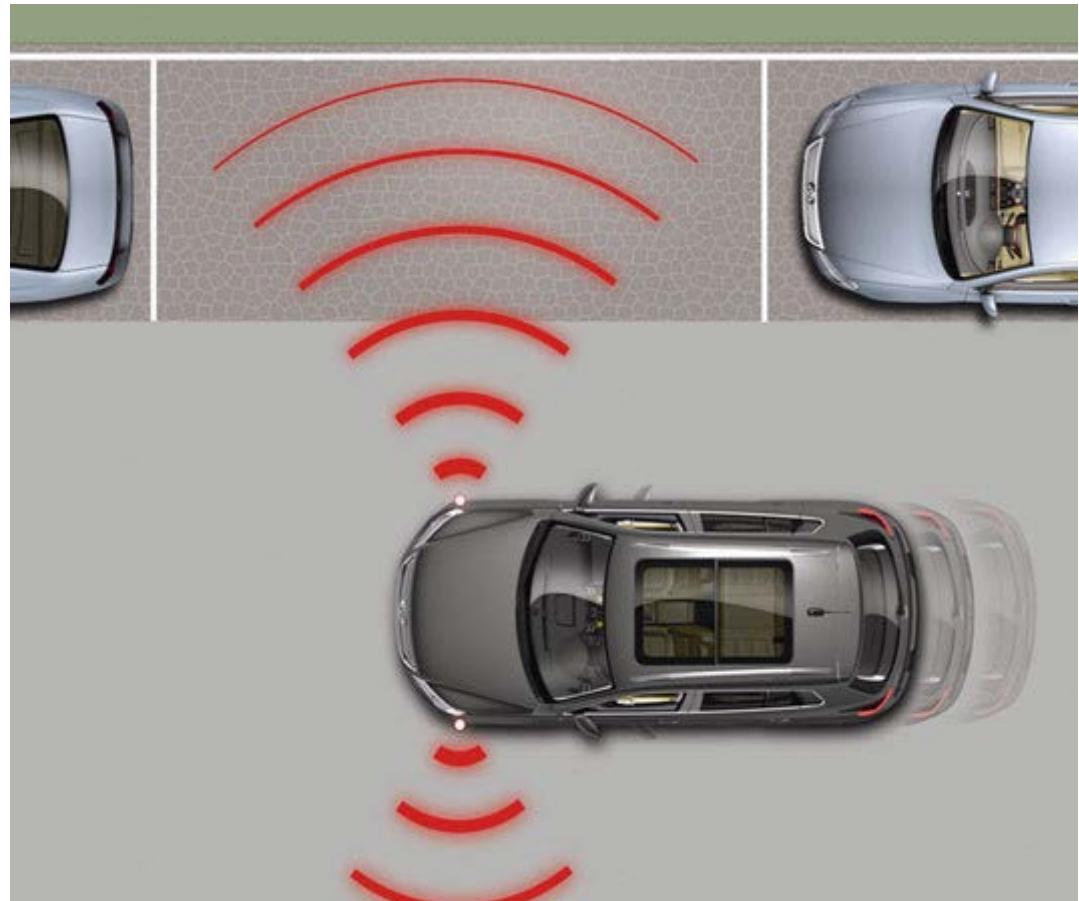


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Parking Assist

- ▶ support the driver by independent steering into a parking spot
- ▶ parking spot detection is made by ultrasonic sensors



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Parking Assist

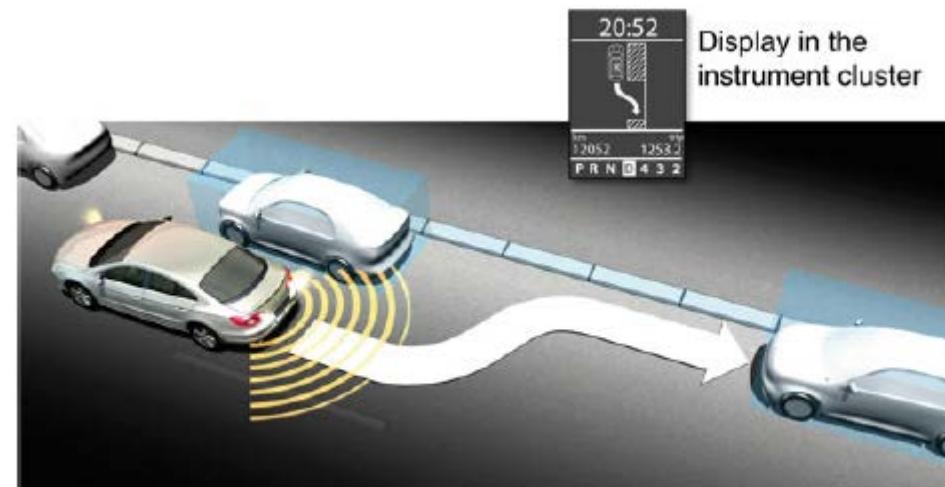
- ▶ parking assist systems:
 - ▶ informing systems
 - inform the driver by acoustic and/or visual indication how far away the driver still is from an object within the driving space
 - ▶ controlled parking assist systems
 - suggest to the driver specific measures based on evaluated information on the surroundings
 - ▶ semiautomatic parking
 - full adoption of a function (automatic steering)
 - ▶ fully automatic parking
 - the driver gives a command to park when a parking spot has been detected
 - the vehicle parks automatically in the parking spot

Driver Assistance System Functions

Parking Assist (Semiautomatic parking)

- semiautomatic parking requirements:

- the vehicle has to achieve a suitable final position, matching the parking situation
- the time needed for parking should be very short
- the vehicle may not collide with any object, otherwise the driver has to be warned during the manual longitudinal guidance



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Driver Assistance System Functions

Parking Assist (Semiautomatic parking)

- ▶ automatically carry out the best steering wheel movements to park the car on the ideal line, in one backward pull
- ▶ survey of the parking spot and the steering movements
- ▶ the driver remains responsible for clutch, gas and brake
- ▶ the driver may, at any time and by deliberate action, override the function
- ▶ the control action shall be automatically disabled if the vehicle speed exceeds the set limit of 10 km/h by more than 20 % or the signals to be evaluated are no longer being received

ALL WHEEL STEERING

All Wheel Steering Overview

- ▶ all-wheel steering provides the possibility to make the back wheels steerable, in addition to the steering at the front axle
- ▶ improves of the lateral dynamic driving characteristics of the respective vehicle

- ▶ basic concepts of rear-wheel steering
 - ▶ mechanical systems
 - ▶ hydraulic systems
 - ▶ electromechanical systems

$$\delta_h = f(\delta_v)$$

$$\delta_h = f(v_x, \delta_v)$$

δ_h - rear wheel steer angle

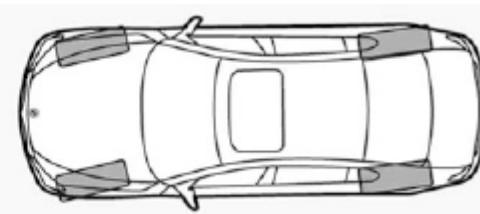
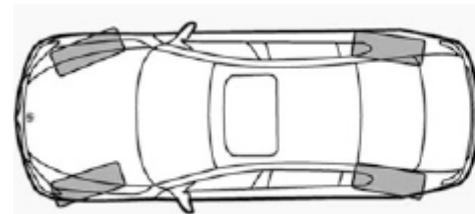
δ_v - front wheel steer angle

v_x - vehicle forward speed

All Wheel Steering

All-wheel steering principles

- ▶ parallel direction
 - the rear and front wheels are steered in the same direction
- ▶ opposite direction
 - the rear wheels are turned against the steering direction of the front wheels



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

opposite direction

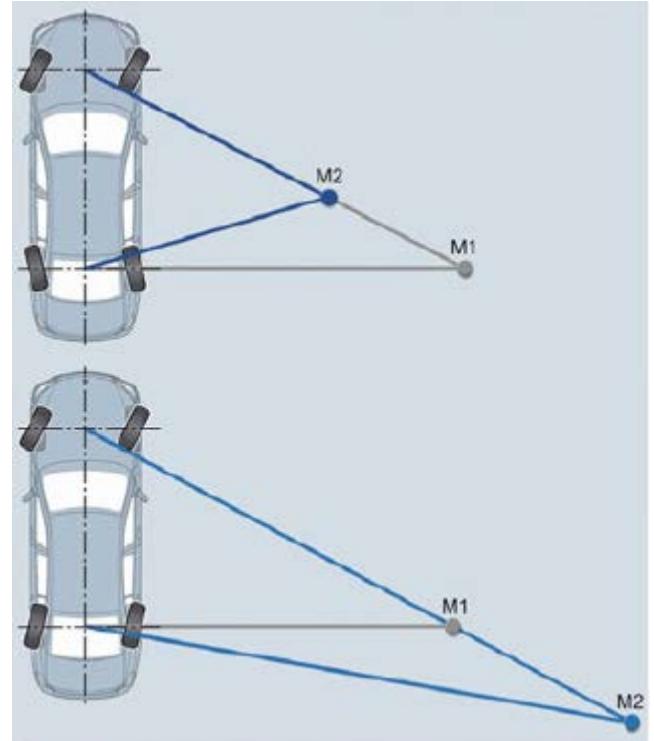
parallel direction

All Wheel Steering

Change of the cornering circle and virtual change of the wheelbase

- ▶ opposite direction (upper picture)
 - the instantaneous center of the vehicle moves forwards, this has an effect as if the wheelbase was shortened
 - the cornering circle shrinks, the vehicle is more agile

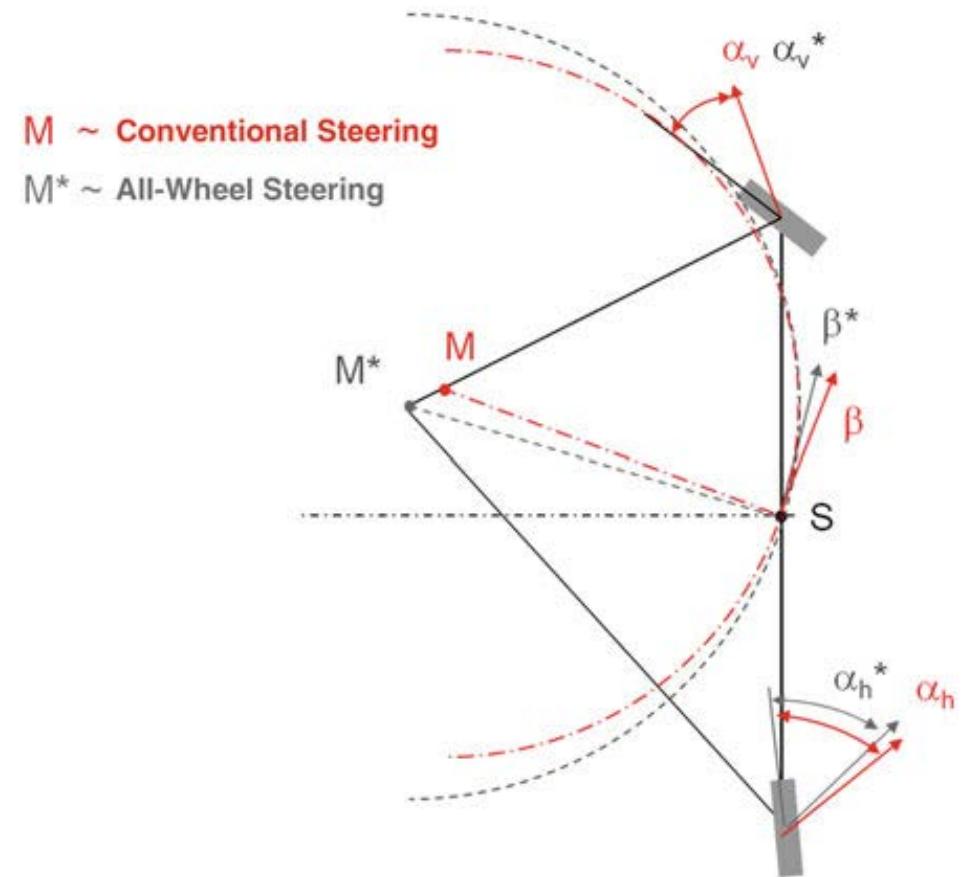
- ▶ parallel direction (lower picture)
 - the instantaneous center moves to the back
 - the virtual extension of the wheelbase increases the stability



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

All Wheel Steering Stationary Vehicle Characteristic (All-Wheel Steering)

- ▶ Comparison between vehicle with front and all-wheel steering
 - ▶ the same highest lateral acceleration
 - ▶ different side slip angles
 - ▶ with all-wheel steering the side slip angle of the vehicle can be compensated to improve vehicle stability

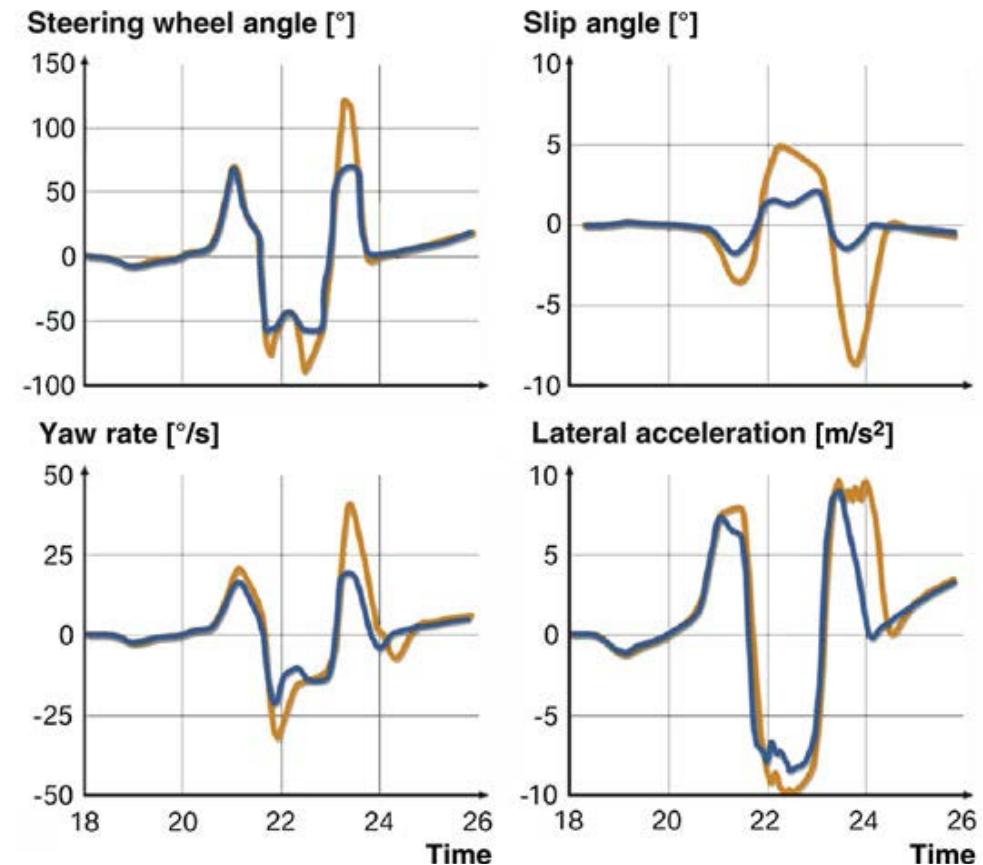


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

All Wheel Steering

Nonstationary Vehicle Characteristics (All-Wheel Steering)

- ▶ ISO lane change
 - ▶ all-wheel steering with parallel direction response (blue)
 - ▶ Front wheel steering (yellow)
- ▶ higher driveability for all-wheel steering vehicle
 - ▶ lower side slip angle
 - ▶ better correlation between steering angle and yaw rate/lateral acceleration

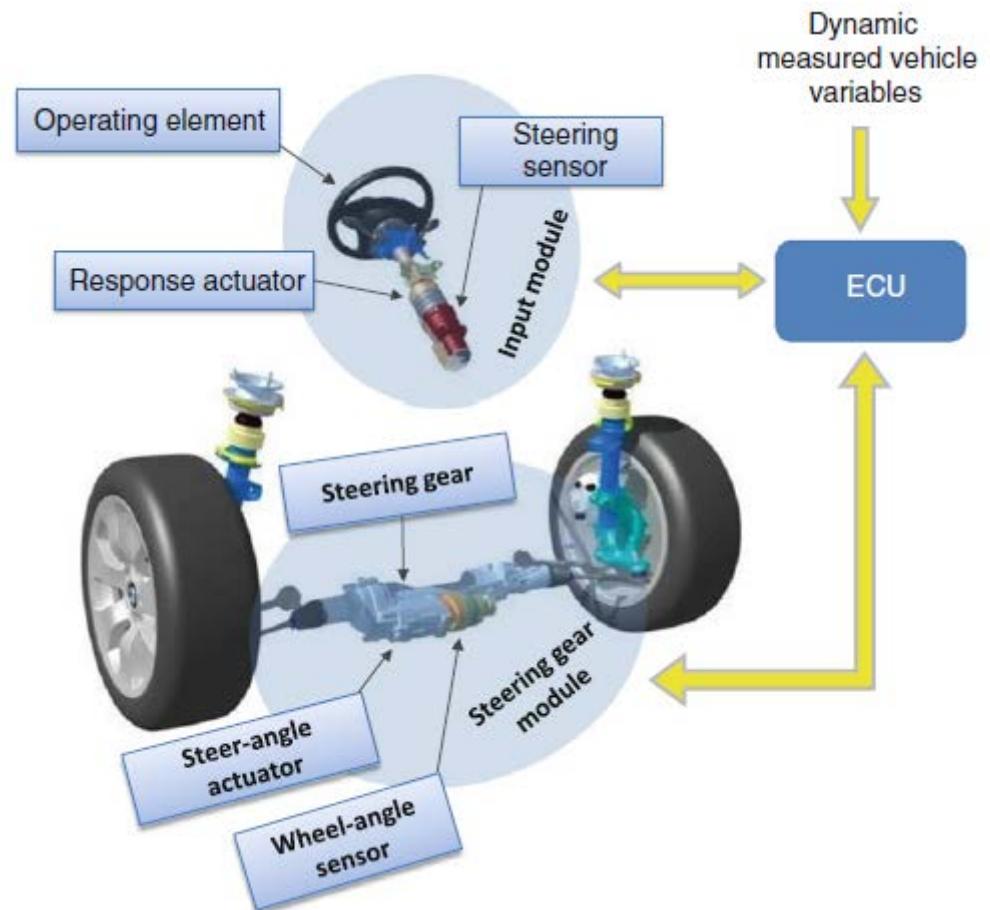


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

STEER BY WIRE

Steer by Wire Overview

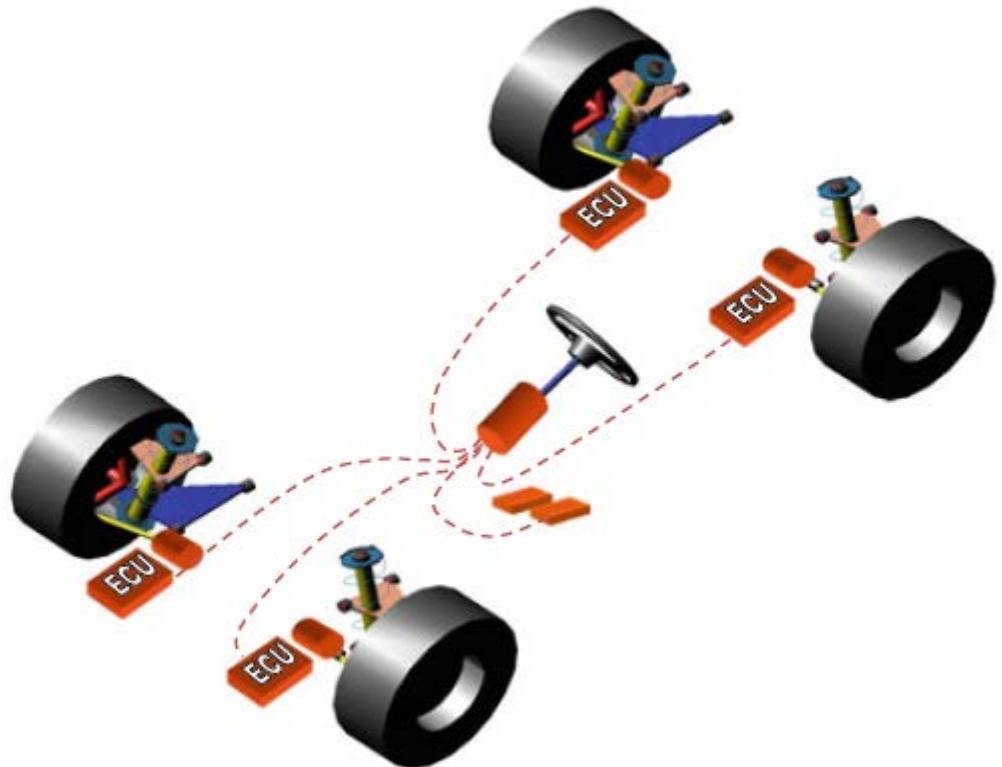
- ▶ a system that electrically transmits a steering command from an operating element (steering wheel) by an ECU to an actuator executing the steering command at the driven wheels
- ▶ biggest challenge: to meet the safety and reliability requirements



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steer by Wire Corner module concept

- ▶ performs the chassis functions (steering, driving, braking, vertical dynamics) at the individual wheels



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

SUPERIMPOSED STEERING SYSTEM

Superimposed Steering System Overview

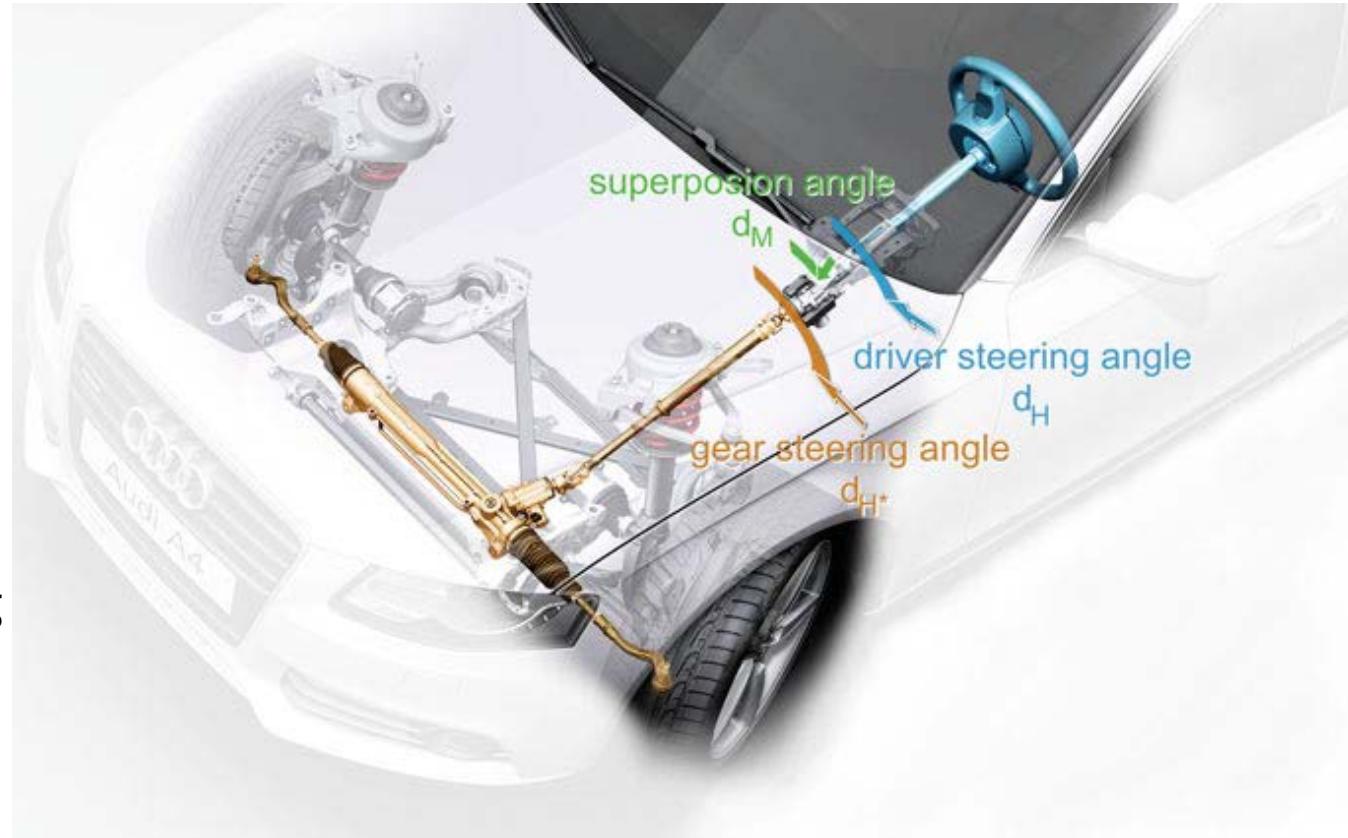
- ▶ introduce an additional angle to the driver's steering input

$$\delta_{H^*} = \delta_H + \delta_M$$

δ_M - freely controllable engine angle (added angle)
 δ_H - steer wheel angle
 δ_{H^*} - driven shaft

- ▶ additional steering functions: steering dynamics and steering stabilization functions

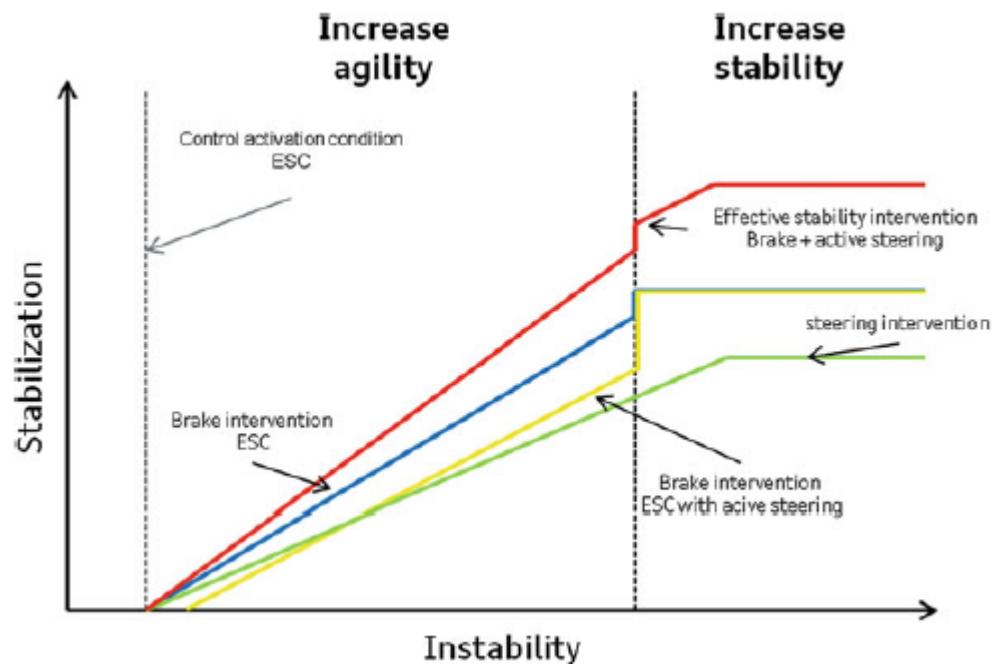
M. Harrer and P. Pfeffer (eds.),
Steering Handbook



Superimposed Steering System

Steering Stabilisation

- ▶ active stabilizing steering corrections that are almost independent from the driver
- ▶ stabilize the vehicle in dynamically critical situations
 - ▶ overall stability of the vehicle is improved by concurrent braking and steering interventions
 - ▶ less critical driving situations, vehicle stability could be achieved only with the superimposed steering system:
 - ▶ to achieve the best agility and concurrent stability, the best distribution of the stabilization torque on brake and steering is made by an arbitrating concept.

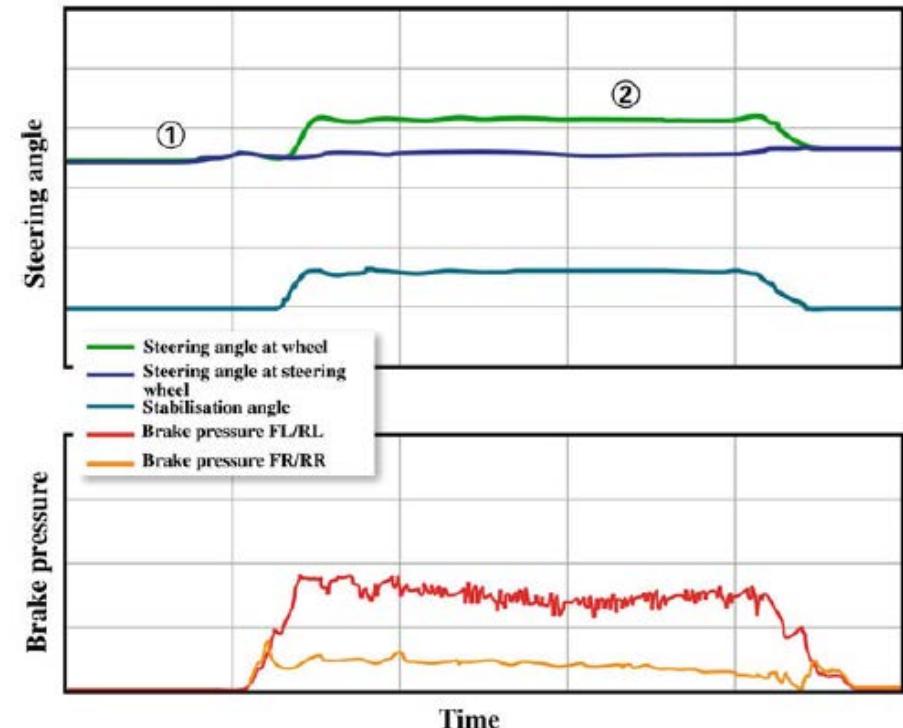
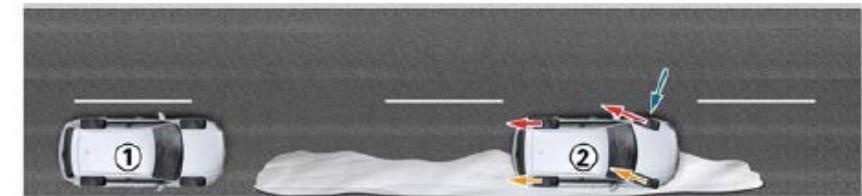


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Superimposed Steering System

Steering Stabilization During Braking on Roads with Different Friction Values (μ -split)

- ▶ braking on such a road generates a yaw torque from the higher braking powers on the side with more friction
- ▶ to continue driving straight, the stabilisation system set a steering wheel angle that compensates the interfering yaw torque

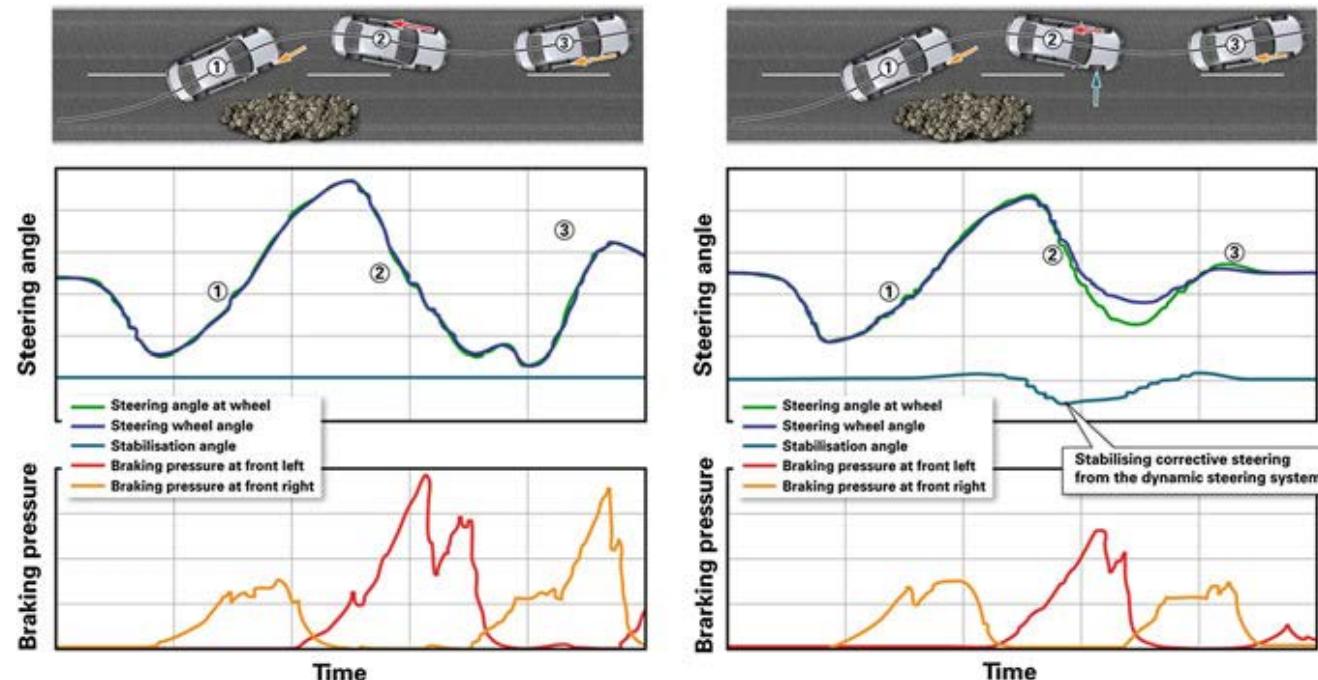


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Superimposed Steering System

Steering Stabilisation at Oversteering

- reduce or fully compensate the too high yaw response of the vehicle
- lower number of brake interventions that make the overall stabilization look very harmonious



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Bosch EPS

BASIC PRINCIPLES OF STEERING

Agenda

1. Vehicle Cornering
2. Steering Assistance Torque
3. Motor Torque Characteristics
4. Basic Steering Functions
5. Lateral Vehicle Dynamics

Source:

Steering Handbook, Editors: Manfred Harrer, Peter Pfeffer

Springer International Publishing Switzerland 2017

D. Schramm et al., Vehicle Dynamics, DOI: 10.1007/978-3-540-36045-2_10,
Springer-Verlag Berlin Heidelberg 2014

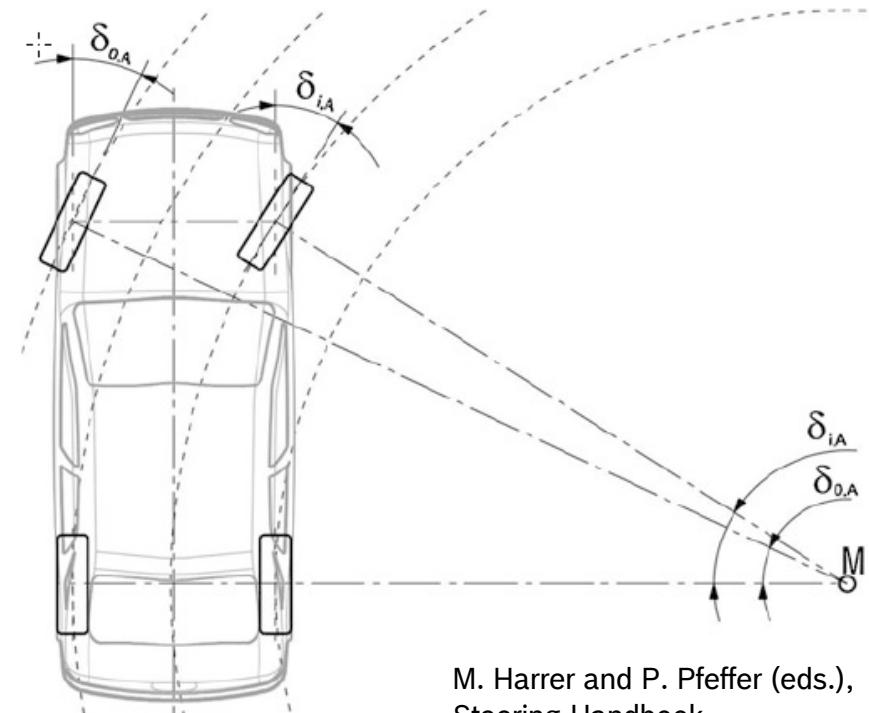
VEHICLE CORNERING

Vehicle Cornering

Slow Cornering – vehicle turning without any lateral force

- ▶ all tires have to be oriented tangentially to concentric arcs (centre plane of the wheel)
- ▶ the instantaneous center of the car M will be located on the rear axle

$\delta_{i,A}$ - the steering wheel angles inside
 $\delta_{o,A}$ - the steering wheel angles outside



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Vehicle Cornering

Slip angle

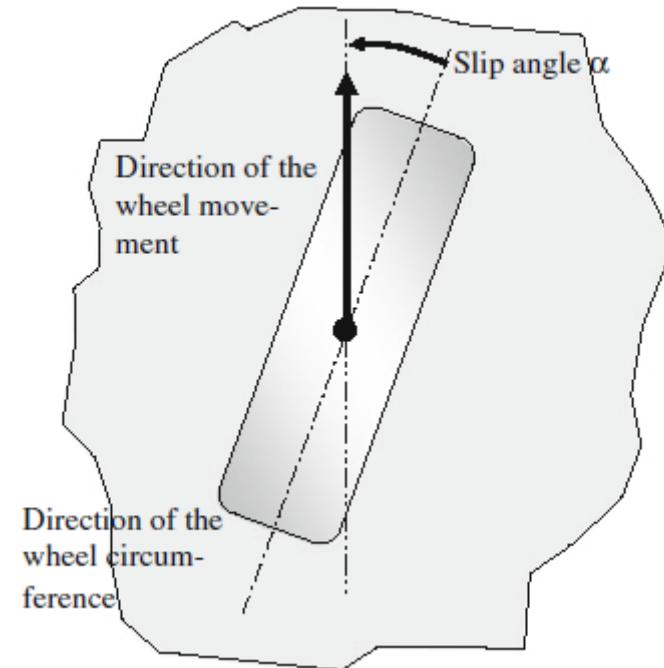
- ▶ the angle between the direction of the wheel circumference and the direction of the movement of the wheel

$\alpha_{F,i}$ - front slip angle inside

$\alpha_{F,o}$ - front slip angle outside

$\alpha_{R,i}$ - rear slip angle inside

$\alpha_{R,o}$ - rear slip angle outside

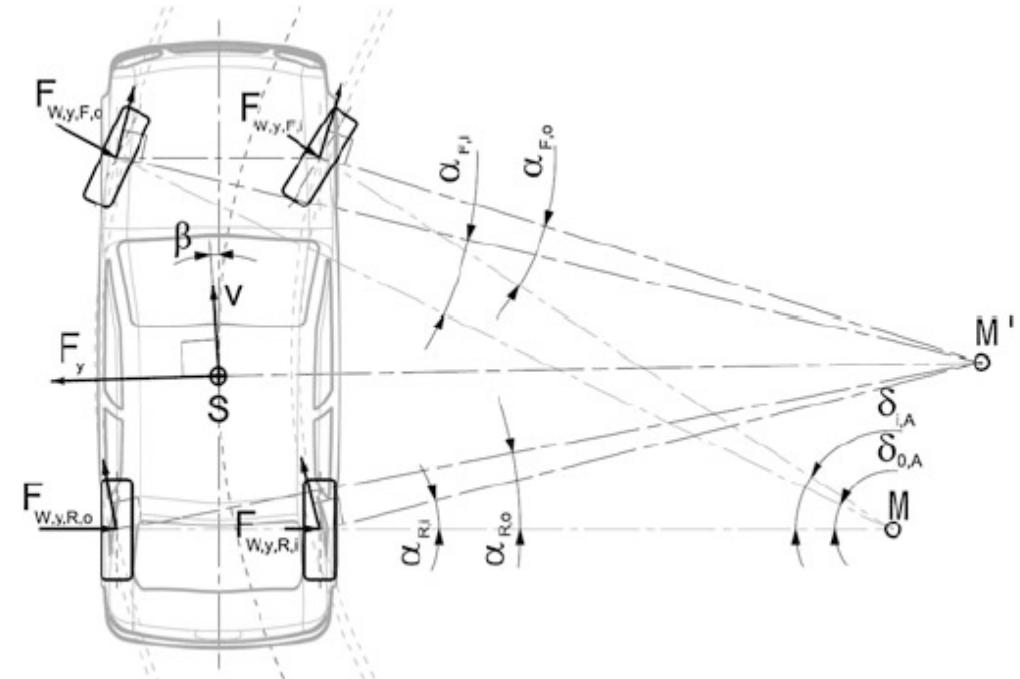


M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Vehicle Cornering

Fast Cornering – vehicle turning with lateral acceleration

- ▶ lateral forces occurs at front and rear wheels
- ▶ the center on which the car is cornering results from the intersection of the perpendicular line to the actual path of the moving wheels
- ▶ the actual instantaneous center of the car M' moves towards the front axle



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Vehicle Cornering

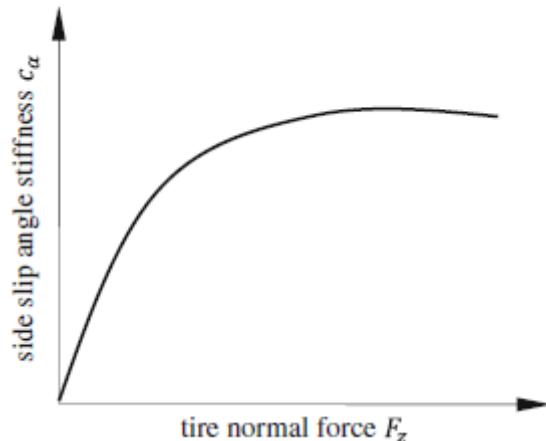
Lateral force of the tire (F_Y)

- lateral forces of the tire are produced by the lateral deformation of the rubber

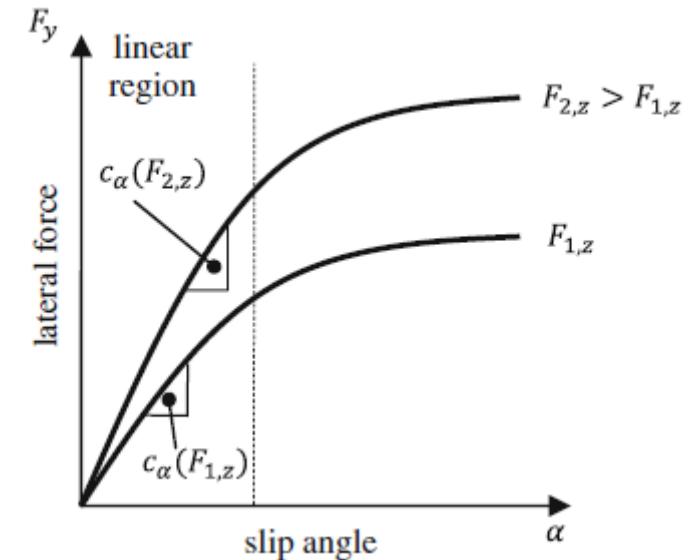
$$F_Y = C_\alpha \alpha$$

C_α - cornering stiffness

α - slip angle



- linear operating range of the tire - lateral acceleration up to 3-4 m/s² (on dry road)
- constant wheel load



D. Schramm, M. Hiller, R. Bardini
Vehicle Dynamics Modeling and Simulation

Vehicle Cornering

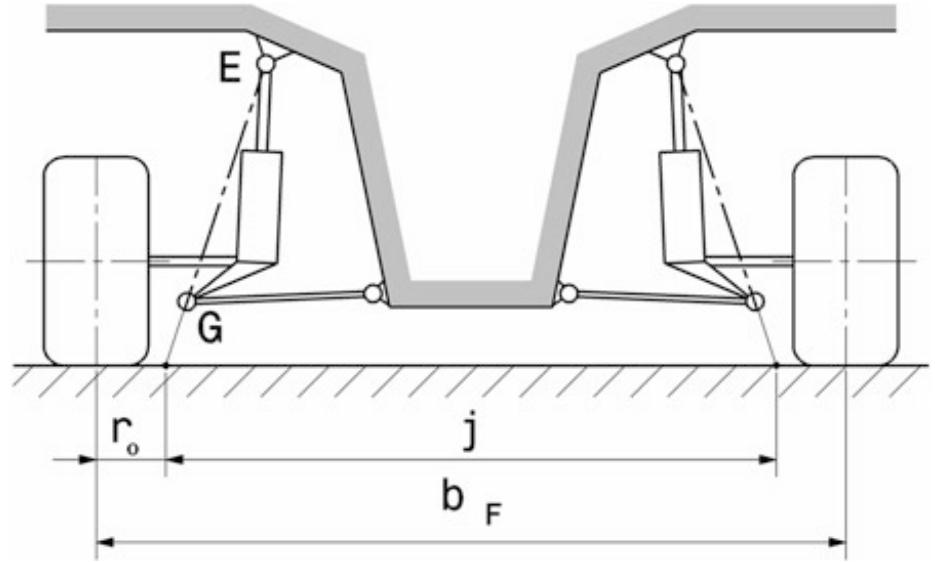
Characteristics of the steering geometry

EG – steering axis (kingpin axis)

j – distance of the steering axes on the road

b_F – front track

r_0 – scrub radius



Vehicle Cornering

Steering torque (M_S)

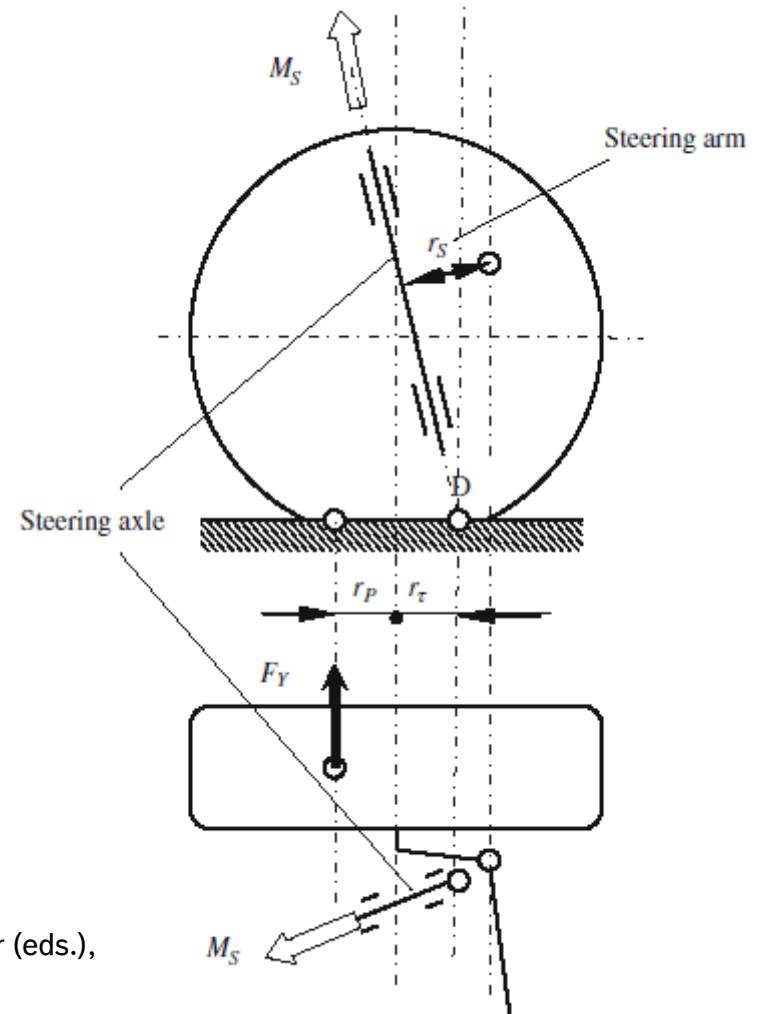
- ▶ total steering torque around of the steering axle of the front wheels

$$M_S = F_Y(r_\tau + r_P)$$

r_τ - mechanical trail (distance between the centre of contact patch and the point where the steering axis intersect the ground)

r_P - pneumatic trail (offset between the centre of contact patch and the effective acting point of lateral force)

M. Harrer and P. Pfeffer (eds.),
Steering Handbook

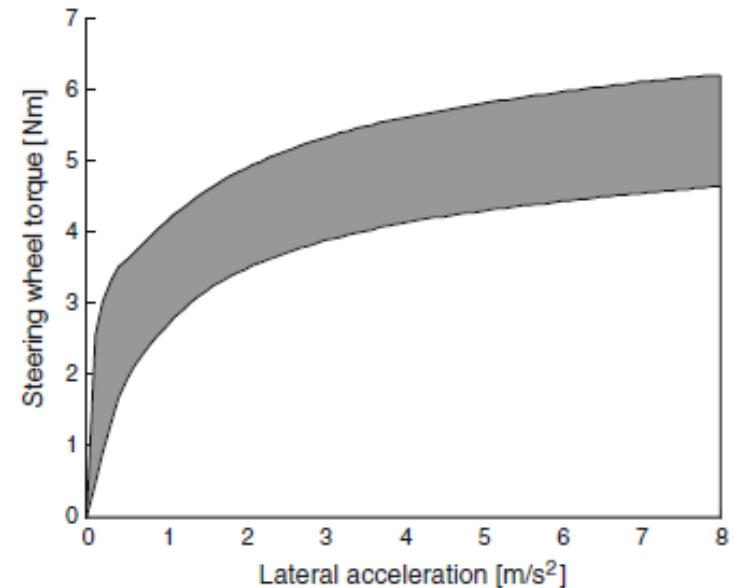


STEERING ASSISTANCE TORQUE

Steering Assistance Torque

Requirements on steering wheel torque

- ▶ range of measured steering wheel torques for various sports cars (figure)
- ▶ Input
 - ▶ steering wheel torque curve in relation to the lateral acceleration of the vehicle
- ▶ Output
 - ▶ calculation of the vehicle torque for stationary cornering



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

Steering Assistance Torque

Steering assistance ratio

$$A_S = \frac{M_S}{i_S M_H}$$

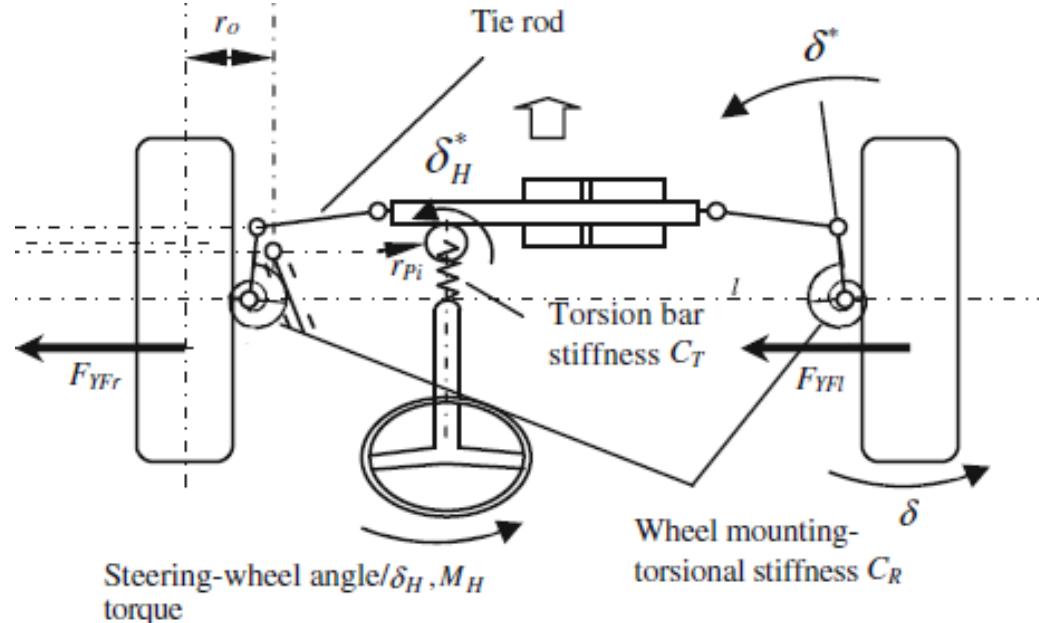
$$M_H = C_T(\delta_H - \delta_H^*)$$

$$M_S = C_R(\delta^* - \delta)$$

► no elasticity between pinion and steering arm

$$\delta_H^* = \delta^* i_S$$

$$i_S = \frac{\delta_H^*}{\delta^*} = \frac{\delta_H - \frac{M_H}{C_T}}{\delta + \frac{M_S}{C_R}}$$



M. Harrer and P. Pfeffer (eds.),
Steering Handbook

i_S	- steering ratio
δ_H	- steering wheel angle
δ_H^*	- pinion angle
δ	- steering angle
δ^*	- steering arm angle
M_H	- steering wheel torque
C_T	- torsion bar stiffness
C_R	- axle support stiffness (elasticity of the tie road and the axle mounting)

Steering Assistance Torque

Steering wheel angle and steering angle

$$\delta_H = \delta i_S + M_S i_S \left(\frac{1}{C_R} + \frac{1}{C_T i_S^2 A_S} \right)$$

$$\delta_H = \delta i_S + \frac{F_Y r i_S}{C_S}$$

- for steering without distortion of the torsion bar or for infinite total steering stiffness

$$\delta_H = \delta i_S$$

- effective cornering stiffness - steering stiffness output on cornering stiffness

$$\frac{1}{C_{\alpha,eff}} = \frac{1}{C_\alpha} + \frac{r}{C_S}$$

C_S - total steering stiffness

$C_{\alpha,eff}$ - effective cornering stiffness

$r = r_\tau + r_P$ - total trail

Steering Assistance Torque

Steering wheel torque (M_H)

- optimum steering reinforcement increase linearly to the lateral acceleration

$$A_S = C_A(D_A + K_A a_Y)$$

- constant total trail
- constant steering ratio

- steering wheel torque

$$M_H = \frac{M_S}{i_S A_S} = \frac{F_Y(r_\tau + r_P)}{i_S A_S} = \frac{m_F r}{i_S A_S} a_Y$$

$$M_H = \frac{C_A}{A_S} a_Y = \frac{1}{D_A + K_A a_Y} a_Y$$

$$C_A = \frac{m_F r}{i_S}$$

$$F_Y = m_F a_Y \quad \begin{aligned} &\text{- lateral force at the front axle} \\ m_F & \quad \begin{aligned} &\text{- mas of the vehicle at the front axle} \\ a_Y & \quad \begin{aligned} &\text{- lateral acceleration} \end{aligned} \end{aligned} \end{aligned}$$

Steering Assistance Torque

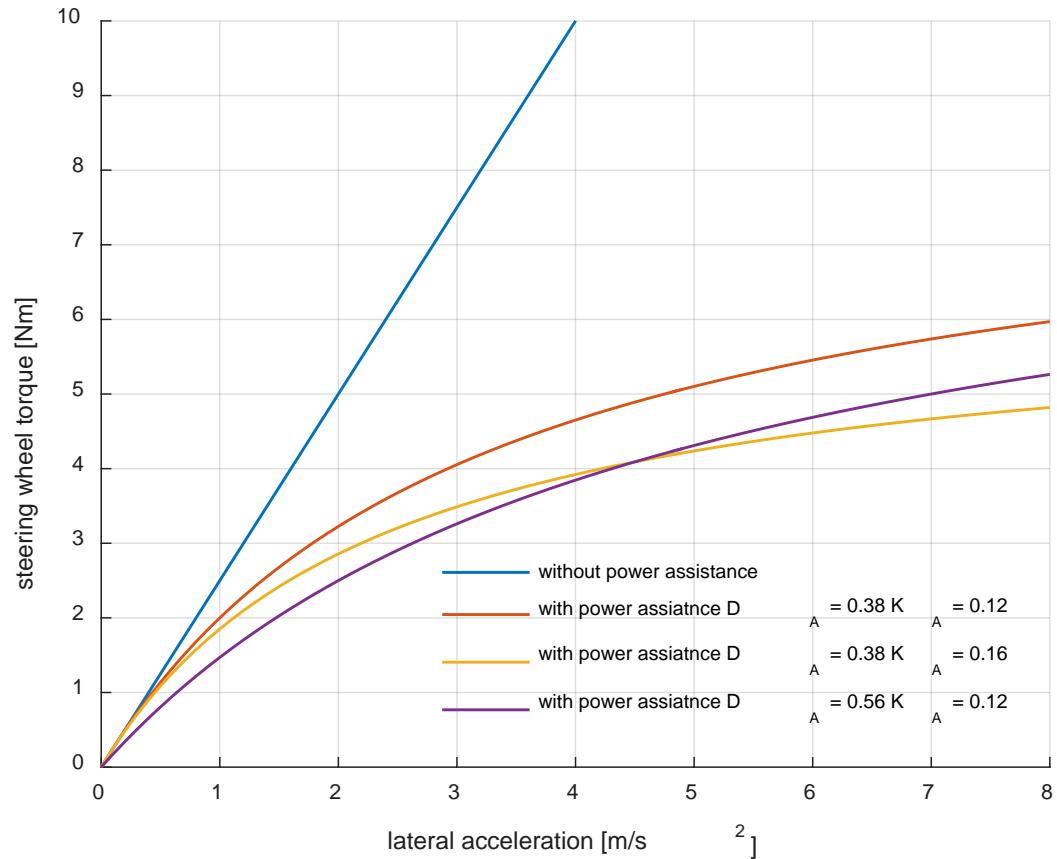
Steering wheel torque (M_H)

- ▶ the steering wheel torque rises degressively

$$M_H = \frac{1}{\frac{D_A}{a_Y} + K_A}$$

- ▶ for vehicles without power steering $A_S = 1$
 - ▶ constant lateral acceleration gradient of the steering wheel torque

$$M_H = C_A a_Y$$



Steering Assistance Torque

Steering assistance torque (M_A)

- ▶ steering assistance torque is the difference between the steering torque at the wheels and the torque applied by the driver

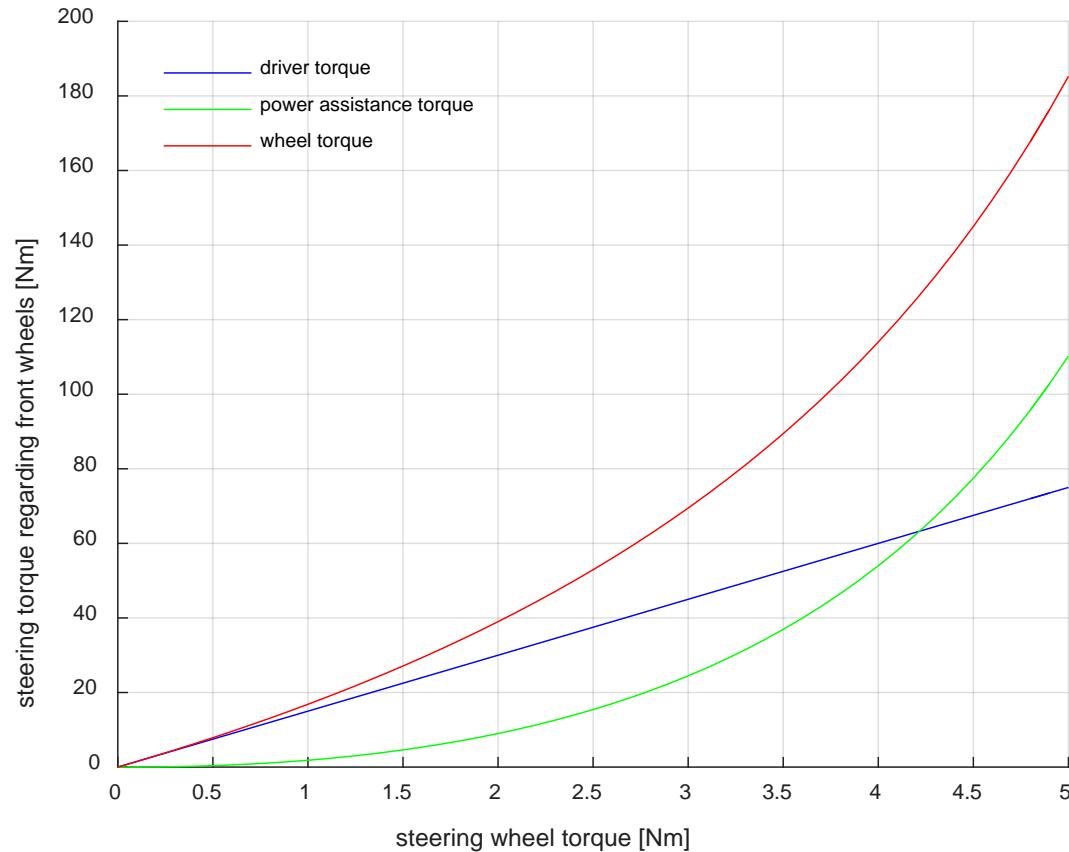
$$M_A = M_S - M_H i_S$$

$$\begin{aligned} M_A &= M_H i_S A_S - M_H i_S \\ &= M_H (m_F r (D_A + K_A a_Y) - i_S) \\ &= \frac{a_Y (m_F r D_A + m_F r K_A a_Y - i_S)}{D_A + K_A a_Y} \\ &= \frac{M_H (m_F r D_A + i_S K_A M_H - i_S)}{1 - K_A M_H} \end{aligned}$$

- ▶ parameterization of the steering assistance torque based on gradient factor D_A and degressivity factor K_A in relation to the steering wheel torque or lateral acceleration

Steering Assistance Torque

Steering torque regarding front wheels



MOTOR TORQUE CHARACTERISTICS

Motor Torque Characteristics

Steering rack force on EPS

- ▶ EPSc; EPSp
 - ▶ power assist unit is placed on the steering column
 - ▶ power assist torque is transmitted to the steering column
 - ▶ steering wheel torque and power assist torque is transferred to rack force by a steering gear
- ▶ EPSdp; EPSapa; EPSrc
 - ▶ power assist unit is placed on the rack
 - ▶ power assist torque is transmitted to the rack by a second pinion, belt, ball screw
- ▶ Steering gear ratio
 - ▶ the ratio between rack path and steering wheel angle
 - ▶ displacement of the rack during one turn of the pinion

Motor Torque Characteristics

Steering rack force on EPS

- steering rack force on EPSc and EPSp

$$F_r v_r = \left(M_H + \frac{\omega_P}{\omega_H} M_P \right) \omega_H$$

$$F_r = \frac{2\pi}{i_G} (M_H + i_P M_P) \quad \omega_P = i_P \omega_H$$

- steering rack force on EPSdp, EPSapa, EPScr

$$F_r \cdot v_r = M_H \cdot \omega_H + M_{dP} \cdot \omega_{dP}$$

$$F_r = \frac{2\pi}{i_G} M_H + i_{dP} M_{dP} \quad \omega_{dP} = i_{dP} v_r$$

$$\omega_H = \frac{2\pi}{i_G} v_r$$

i_G - steering gear ratio [m/rev]

i_P - motor pinion ratio [-]

i_{dP} - rack pinion ratio [rad/m]

ω_H - steering wheel velocity [rad/s]

ω_P - motor velocity (column pinion) [rad/s]

ω_{dP} - motor velocity (rack pinion) [rad/s]

F_r - rack force [N]

v_r - rack velocity [m/s]

Motor Torque Characteristics

Steering assistance / motor torque

► EPSdp, EPSSapa, EPScr

$$M_S \omega_S = M_H \omega_H + M_{dP} \omega_{dP}$$

$$M_S \omega_S = \left(M_H + \frac{\omega_{dP}}{\omega_H} M_{dP} \right) \omega_H$$

$$M_S = \left(M_H + \frac{\omega_{dP}}{\omega_H} M_{dP} \right) i_S$$

$$M_A = M_S - M_H i_S$$

$$= \frac{\omega_{dP}}{\omega_H} i_S M_{dP}$$

► EPSc, EPSp

$$M_S = \left(M_H + \frac{\omega_P}{\omega_H} M_P \right) i_S$$

$$M_A = \frac{\omega_P}{\omega_H} i_S M_P$$

Motor Torque Characteristics

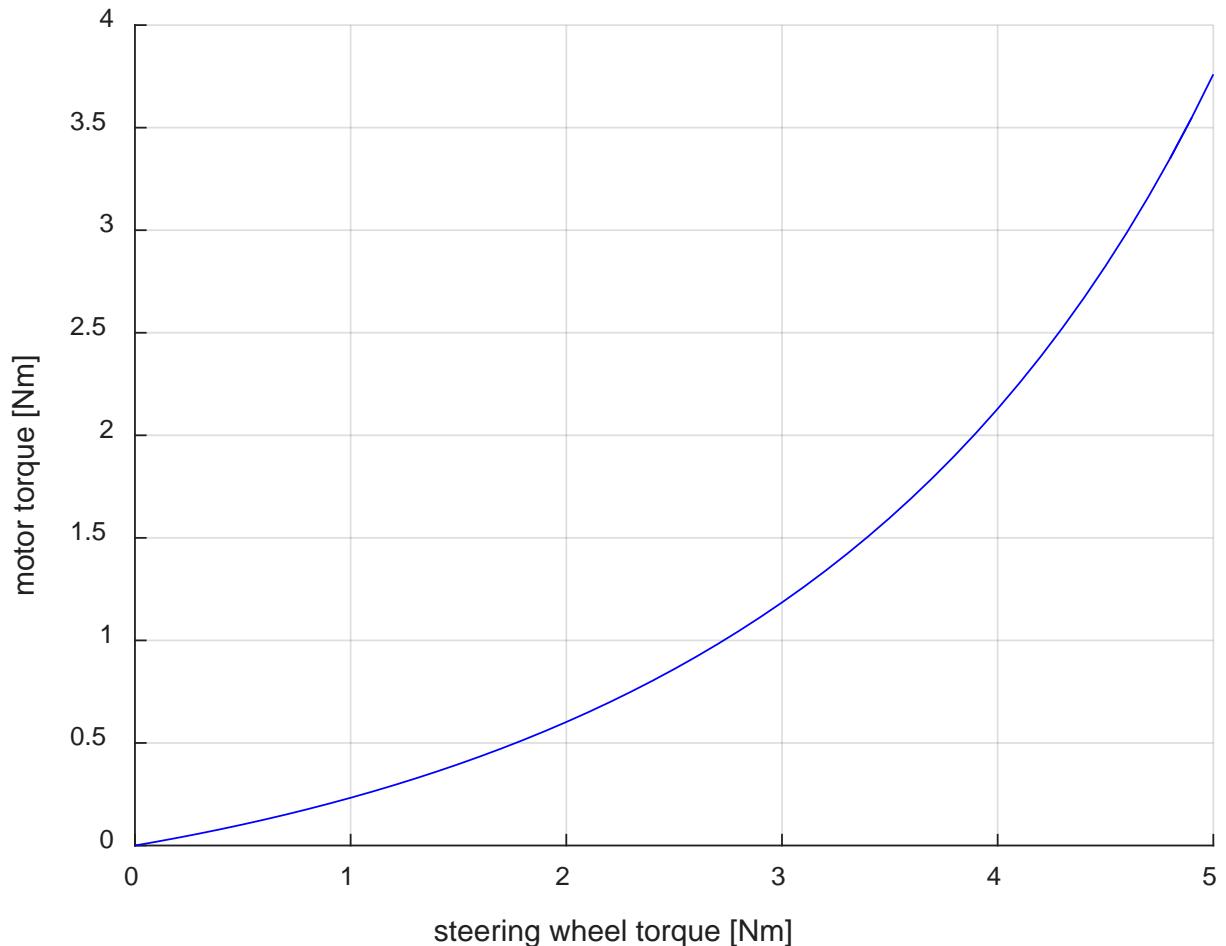
Motor torque characteristics

- Motor torque on EPSc and EPSp

$$M_P = \frac{1}{i_P i_S} M_A$$

- Motor torque on EPSdp, EPSapa, EPScr

$$M_{dP} = \frac{2\pi}{i_{dP} i_G i_S} M_A$$

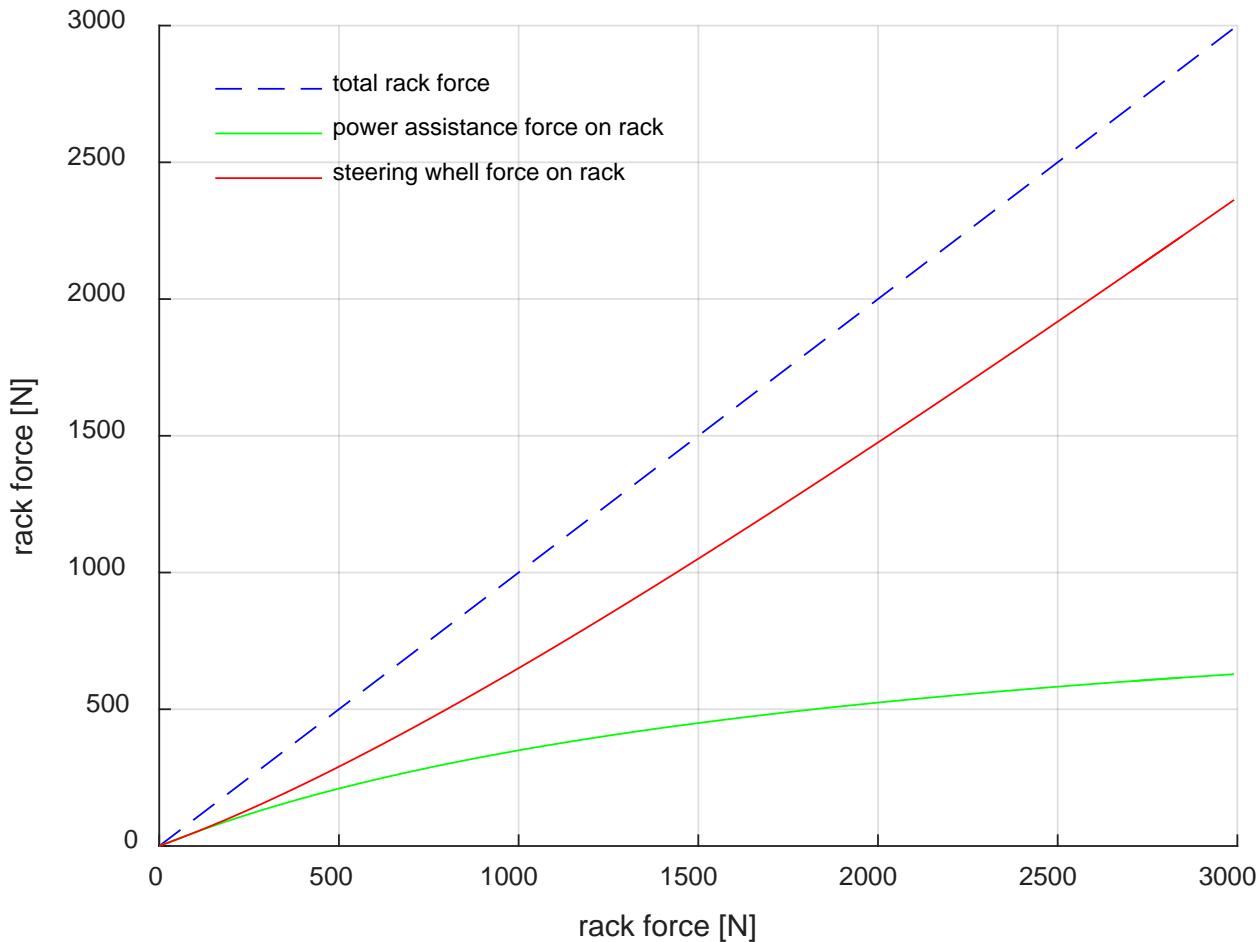


Motor Torque Characteristics

Steering rack force on EPS

- steering wheel torque and power assistance torque distribution on forces acting on the rack

$$F_r = \frac{2\pi}{i_G} (M_H + i_P M_P)$$



BASIC STEERING FUNCTIONS

Basic Steering Functions

Friction compensation

- ▶ power steering system generates a system friction which is higher than other steering systems
- ▶ the feedback of the steering system is affected by higher friction
- ▶ useful information about the current driving situation and road condition is accordingly reduced by friction
- ▶ a high friction coefficient in the steering system will support the suppression of interferences
- ▶ any disturbances in the steering wheel (wheel imbalances, fluctuations of the braking forces) can be reduced by higher friction;
- ▶ in steady cornering steering friction will generate a higher torque when the steering angle increase and a lower torque when the steering angle decrease

- ▶ reduce the effect of the friction in the steering with regard the torque requested by the power assistance

Basic Steering Functions

Friction compensation

- ▶ steady rack displacement with no load and motor off

$$b_H \omega_H = M_H - K_{TB}(\delta_H - i_H x_r)$$

$$b_{dP} \omega_{dP} = -K_{dP}(\delta_{dP} - i_{dP} x_r)$$

$$b_r v_r = i_H K_{TB}(\delta_H - i_H x_r) + i_{dP} K_{dP}(\delta_{dP} - i_{dP} x_r)$$

$$b_r v_r = i_H(M_H - b_H \omega_H) - \frac{i_{dP}^2}{i_H} b_{dP} \omega_H$$

$$M_H = \left(b_H + \frac{1}{i_H^2} b_r + \frac{i_{dP}^2}{i_H^2} b_{dP} \right) \omega_H$$

$$\omega_{dP} = \frac{i_{dP}}{i_H} \omega_H$$

$$\delta_{dP} = \frac{i_{dP}}{i_H} \delta_H$$

$$i_H = \frac{2\pi}{i_G}$$

$$v_r = \frac{\omega_H}{i_H}$$

Basic Steering Functions

Inertia compensation

- ▶ EPS systems have high inertia, the steering movements initiated by the driver have to act against the torque generated by inertia
- ▶ a function for inertia compensation has to reduce the inertia effect on the steering torque characteristics; additional torque must be requested from the EPS motor

$$m_r \frac{d\nu_r}{dt} = i_H K_{TB} (\delta_H - i_H x_r) + i_{dP} K_{dP} (\delta_{dP} - i_{dP} x_r)$$

$$\frac{d\nu_r}{dt} = \frac{1}{i_{dP}} \frac{d\omega_{dP}}{dt}$$

$$J_H \frac{d\omega_H}{dt} = M_H - K_{TB} (\delta_H - i_H x_r)$$

$$\frac{d\omega_H}{dt} = \frac{i_{dP}}{i_H} \frac{d\omega_{dP}}{dt}$$

$$\frac{m_r}{i_{dP}} \frac{d\omega_{dP}}{dt} = i_H \left(M_H - J_H \frac{i_{dP}}{i_H} \frac{d\omega_{dP}}{dt} \right) + i_{dP} \left(M_{dP} - J_{dP} \frac{d\omega_{dP}}{dt} \right)$$

$$M_{dP} = \left(\frac{1}{i_{dP}^2} m_r + J_{dP} \right) \frac{d\omega_m}{dt}$$

$$J_{dP} \frac{d\omega_{dP}}{dt} = M_{dP} - (\delta_{dP} - i_{dP} x_r)$$

Basic Steering Functions

Active damping

- ▶ a friction- and inertia-compensated steering system responds very sensitively to disturbances in the force balance
- ▶ bumpy road could lead to high acceleration of the steering system which is perceived by the driver as a kickback
- ▶ small change on applied steering wheel torque lead to powerful system movements
- ▶ a damping function must be introduced to compensate for these undesirable characteristics

- ▶ this function has to request an torque from EPS motor that is oriented against the steering direction, proportional to the current steering speed and parameterised as a function of the vehicle speed

Basic Steering Functions

Active return

- ▶ EPS basic steering functions: power assistance, friction compensation, inertia compensation and active damping displays a steering response that is comparable to that of an hydraulic power steering
- ▶ the active return function is design to improve the runback response of the front axle
- ▶ EPS motor adds torque to guide the free wheel and driver controlled wheel into the straight ahead position
- ▶ it is a function of steering wheel angle, applied steering torque and vehicle speed

LATERAL VEHICLE DYNAMICS

Lateral Vehicle Dynamics

Linear single track model

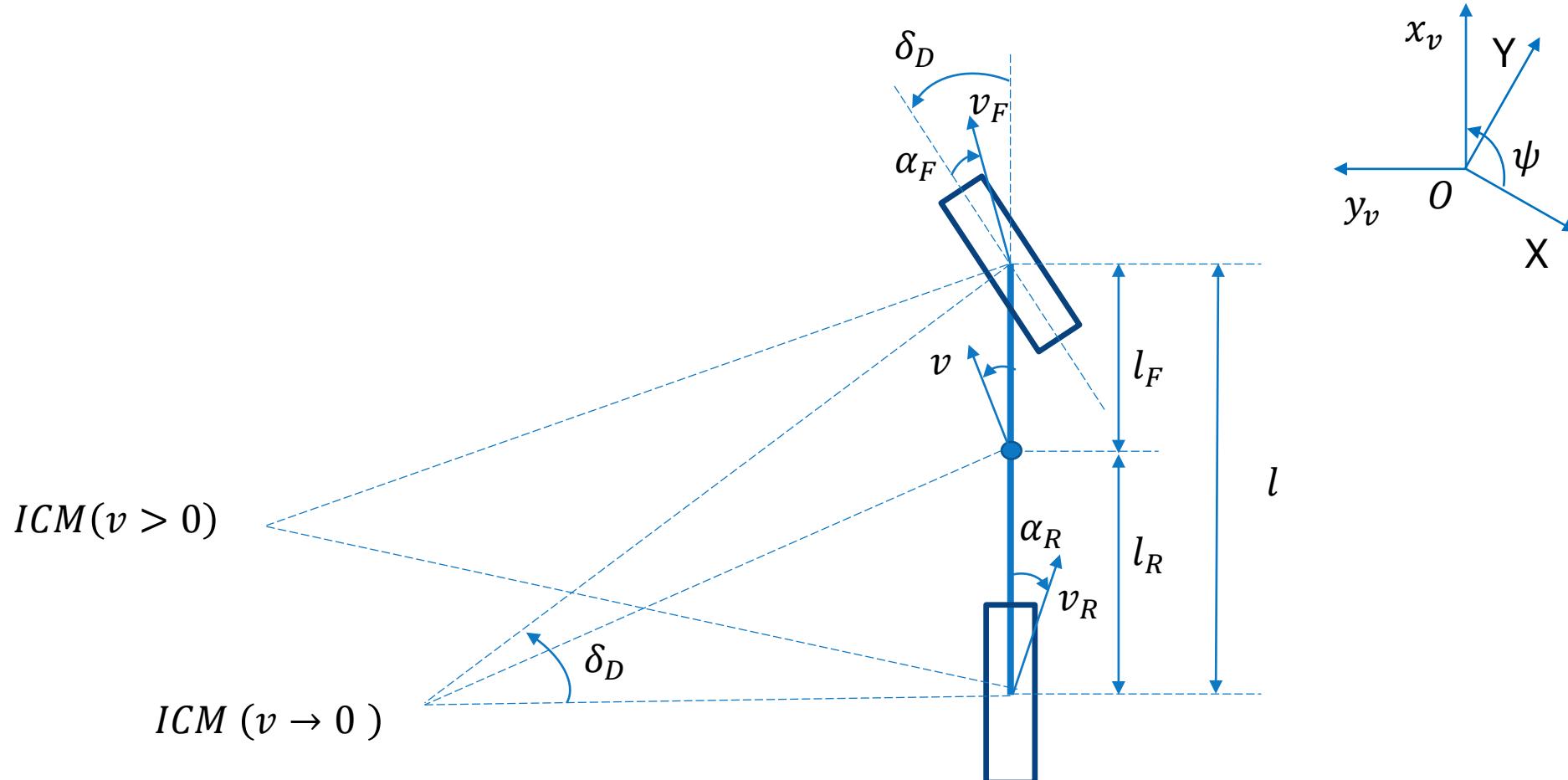
- ▶ the linear single track allows to approximate the lateral vehicle dynamics
- ▶ following simplification where assumed:
 - all the forces act on a plane flat road; the left an right tyre of each axle is exposed at the same load
 - the equations of the system are linearized; the tyre force is assumed proportional to the slip angle; the trigonometric functions are linearized
 - constant longitudinal velocity
- ▶ the model is suitable for lateral accelerations up to 4m/s^2 on dry roads

$$a_y \leq 0.4g \approx 4\text{ m/s}^2$$

- ▶ the vehicle is described by a moving coordinate system located at the vehicle center of gravity ($O_v x_v y_v z_v$) and an inertial coordinate system ($O_{XYZ} XYZ$)

Lateral Vehicle Dynamics

Linear single track model



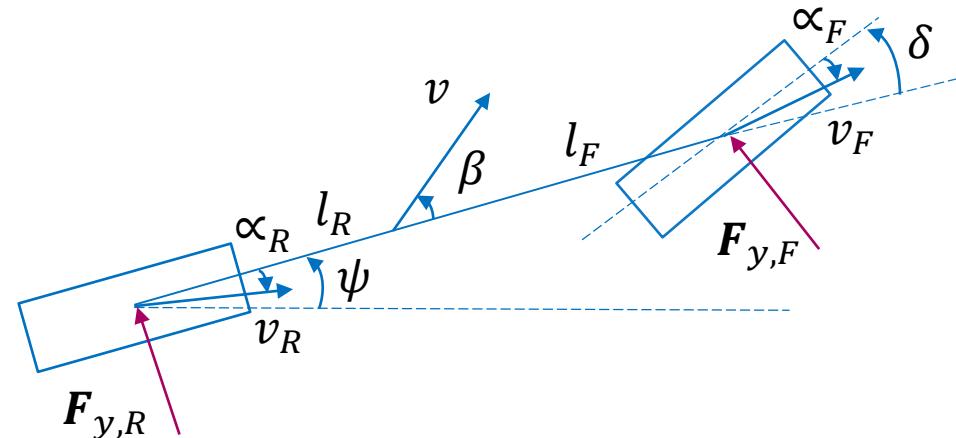
Lateral Vehicle Dynamics

Vehicle velocity

- the vehicle velocity

$$\boldsymbol{v} = \begin{bmatrix} v \cos \beta \\ v \sin \beta \\ 0 \end{bmatrix}$$

- the front/rear axle vehicle velocity



$$\boldsymbol{v}_F = \boldsymbol{v} + \boldsymbol{\omega} \times \boldsymbol{r}_F$$

$$\boldsymbol{v}_R = \boldsymbol{v} + \boldsymbol{\omega} \times \boldsymbol{r}_R$$

$$\boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$

$$\boldsymbol{r}_F = \begin{bmatrix} l_F \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{r}_R = \begin{bmatrix} -l_R \\ 0 \\ 0 \end{bmatrix}$$

Lateral Vehicle Dynamics

Vehicle velocity

- ▶ the longitudinal component of the speed is equal for every point of the vehicle
- ▶ the lateral component of the speed changes by the rotating part of the yaw velocity multiplied by the distance to the front/rear axle

$$\boldsymbol{v}_F = \begin{bmatrix} v \cos \beta \\ v \sin \beta + \dot{\psi} l_F \\ 0 \end{bmatrix} = \begin{bmatrix} v_F \cos(\delta - \alpha_F) \\ v_F \sin(\delta - \alpha_F) \\ 0 \end{bmatrix}$$

$$\boldsymbol{v}_R = \begin{bmatrix} v \cos \beta \\ v \sin \beta - \dot{\psi} l_R \\ 0 \end{bmatrix} = \begin{bmatrix} v_R \cos(-\alpha_R) \\ v_R \sin(-\alpha_R) \\ 0 \end{bmatrix}$$

Lateral Vehicle Dynamics

Front / rear slip angles representations

- ▶ from the front/rear vehicle velocity:

$$\tan(-\alpha_R) = \frac{v \sin \beta - \dot{\psi} l_R}{v \cos \beta}$$

$$\tan(\delta - \alpha_F) = \frac{v \sin \beta + \dot{\psi} l_F}{v \cos \beta}$$

- ▶ for small angles, front and rear slip angles have following representation:

$$\alpha_F = \delta - \beta - \frac{\dot{\psi} l_F}{v} \quad \sin \beta = \beta; \cos \beta = 1$$

$$\alpha_R = -\beta + \frac{\dot{\psi} l_R}{v}$$



Lateral Vehicle Dynamics

Vehicle acceleration

- ▶ the vehicle acceleration

$$\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt} + \boldsymbol{\omega} \times \boldsymbol{v}$$

- ▶ with the assumption of a constant longitudinal velocity

$$\boldsymbol{a} = \begin{bmatrix} -v\dot{\beta} \sin \beta \\ v\dot{\beta} \cos \beta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} v \cos \beta \\ v \sin \beta \\ 0 \end{bmatrix} = \begin{bmatrix} -v(\dot{\beta} + \dot{\psi}) \sin \beta \\ v(\dot{\beta} + \dot{\psi}) \cos \beta \\ 0 \end{bmatrix}$$

- ▶ acceleration is perpendicular on velocity: $\boldsymbol{a} \cdot \boldsymbol{v} = 0$
- ▶ acceleration magnitude

$$a_n = v(\dot{\beta} + \dot{\psi})$$

Lateral Vehicle Dynamics

Vehicle acceleration

- ▶ the acceleration projection on Oy_v axis

$$a_y = a_n \cos \beta$$

- ▶ with the assumption of small slip angle, $\cos \beta = 1$

$$a_y = v(\dot{\beta} + \dot{\psi})$$

- ▶ the vehicle center of gravity describe a path given by a function of yaw angle, slip angle and radius of curvature

$$v = R(\dot{\psi} + \dot{\beta})$$

- ▶ the acceleration projection on Oy_v axis is described by velocity and radius of curvature of the path of the centre of gravity

$$a_y = \frac{v^2}{R}$$

Lateral Vehicle Dynamics

Dynamic equations

- the principle of linear momentum in the lateral direction

$$ma_y = C_{\alpha F} \alpha_F \cos \delta + C_{\alpha R} \alpha_R$$

- the principle of angular momentum around the car axis Oy_v

$$I_z \ddot{\psi} = C_{\alpha F} \alpha_F \cos \delta l_F - C_{\alpha R} \alpha_R l_R$$

- for small steering angle, $\cos \delta = 1$

$$mv(\dot{\beta} + \dot{\psi}) = C_{\alpha F} \left(\delta - \beta - \frac{\dot{\psi} l_F}{v} \right) + C_{\alpha R} \left(-\beta + \frac{\dot{\psi} l_R}{v} \right)$$
$$I_z \ddot{\psi} = C_{\alpha F} \left(\delta - \beta - \frac{\dot{\psi} l_F}{v} \right) l_F - C_{\alpha R} \left(-\beta + \frac{\dot{\psi} l_R}{v} \right) l_R$$

Lateral Vehicle Dynamics

Dynamic equations in state space representation

$$\dot{x} = Ax + Bu$$

- input vector
 - steering angle
- output vector
 - yaw velocity
 - slip angle
- system matrix
- control matrix

$$u = [\delta]$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \dot{\psi} \\ \beta \end{bmatrix}$$

$$A = \begin{bmatrix} -\frac{1}{v} a_{11} & -a_{12} \\ -1 - \frac{1}{v^2} a_{21} & -\frac{1}{v} a_{22} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{C_{\alpha,F} l_F}{\psi} \\ \frac{C_{\alpha,F}}{m v} \end{bmatrix}$$

$$a_{11} = \frac{C_{\alpha,F} l_F^2 + C_{\alpha,R} l_R^2}{\psi}$$

$$a_{12} = \frac{C_{\alpha,F} l_F - C_{\alpha,R} l_R}{\psi}$$

$$a_{21} = \frac{C_{\alpha,F} l_F - C_{\alpha,R} l_R}{m}$$

$$a_{22} = \frac{C_{\alpha,F} + C_{\alpha,R}}{m}$$

Lateral Vehicle Dynamics

Vehicle stability during straight line driving

- ▶ assume steering angle equal to zero

$$\delta = 0$$

- ▶ the linear state space system becomes

$$\dot{x} = Ax$$

- ▶ the linear system is stable when the polynomial for the characteristic equation of the matrix system has positive coefficients

$$\det(\lambda I - A) = \lambda^2 + a_1\lambda + a_2$$

$$a_1 = \frac{1}{v}(a_{11} + a_{22})$$

$$\begin{aligned} a_2 &= -a_{12} + \frac{1}{v^2}(a_{11}a_{22} - a_{12}a_{21}) \\ &= \frac{C_{\alpha,F}C_{\alpha,R}l^2}{m\psi v^2} \left(1 + \frac{C_{\alpha,R}l_R - C_{\alpha,F}l_F}{C_{\alpha,F}C_{\alpha,R}l^2} mv^2 \right) \end{aligned}$$

Lateral Vehicle Dynamics

Vehicle stability during straight line driving

- ▶ $a_1 > 0$ for any velocity
- ▶ $a_2 > 0$ for any velocity if $C_{\alpha,R}l_R > C_{\alpha,F}l_F$
- ▶ the vehicle becomes unstable if $C_{\alpha,R}l_R < C_{\alpha,F}l_F$ and

$$v^2 > \frac{l^2}{m} \frac{C_{\alpha,F}C_{\alpha,R}}{C_{\alpha,F}l_F - C_{\alpha,R}l_R}$$

Lateral Vehicle Dynamics

Stationary steering

- ▶ for stationary steering, the steering angle, the yaw rate and slip angle are constant

$\delta, \dot{\psi}, \beta$ - constant

- ▶ the linear/angular momentum principle

$$m \frac{v^2}{R} = C_{\alpha F} \alpha_F + C_{\alpha R} \alpha_R$$

$$C_{\alpha F} \alpha_F l_F = C_{\alpha R} \alpha_R l_R$$

- ▶ the driving behaviour for a specific vehicle could be established based on the difference between front and rear slip angle obtained from the linear/angular momentum principle

$$\alpha_F - \alpha_R = \frac{mv^2}{Rl} \left(\frac{l_R}{C_{\alpha F}} - \frac{l_F}{C_{\alpha R}} \right)$$

Lateral Vehicle Dynamics

Stationary steering

- ▶ the driving behaviour for a specific vehicle and for a specific steering motion is characterized by the self-steering gradient

$$EG = \frac{m}{l} \left(\frac{l_R C_{\alpha R} - l_F C_{\alpha F}}{C_{\alpha F} C_{\alpha R}} \right)$$

- ▶ the difference between front and rear slip angle is described by the self-steering gradient, vehicle velocity and the circle radius the vehicle is driving on

$$\alpha_F - \alpha_R = EG \frac{v^2}{R}$$

Lateral Vehicle Dynamics

Stationary steering

- ▶ which is the steering angle for a vehicle with a velocity v to follow a circle with radius R ?
- ▶ the steering angle is obtained from front and rear slip angle (geometrical and velocity representation)

$$\delta = \frac{\dot{\psi}l}{v} + \alpha_F - \alpha_R$$

- ▶ recalling the yaw rate $\dot{\psi} = \frac{v}{R}$, lateral acceleration $a_y = \frac{v^2}{R}$ and self-steering gradient EG

$$\delta = \frac{l}{R} + EG a_y$$

Lateral Vehicle Dynamics

Stationary steering

- ▶ the ratio between wheel base and the radius of the vehicle path is called **Ackerman steering angle**

$$\delta_D = \frac{l}{R}$$

- ▶ the steering angle is the Ackerman steering angle and dynamic component depending on vehicle velocity

$$\delta = \delta_D + EG \frac{v^2}{R}$$

- ▶ the steering angle increase or decrease depending on velocity and the sign of the self steering gradient

Lateral Vehicle Dynamics

Steering driving behaviour

- neutral steering $EG = 0$; steering angle is equal with the Ackerman angle

$$\delta = \delta_D$$

- understeering $EG > 0$; steering angle is greater than the Ackerman angle

$$\delta > \delta_D$$

- oversteering $EG < 0$; steering angle is smaller than the Ackerman angle

$$\delta < \delta_D$$

Lateral Vehicle Dynamics

Yaw amplification factor

- for constant steering angle, yaw rate takes on different values depending on steering gradient EG

$$\dot{\psi} = \frac{v}{l + EGv^2} \delta_{st} \quad \delta = \delta_{st} - \text{constant}$$

- yaw amplification factor for a given velocity

$$\frac{\dot{\psi}}{\delta} = \frac{v}{l + EGv^2}$$

- the yaw amplification factor is small for understeering vehicles $EG > 0$ and large for oversteering vehicles $EG < 0$

Lateral Vehicle Dynamics

Critical velocity

- If the self steering gradient $EG = -\frac{l}{v^2}$ the vehicle becomes instable, small steering inputs lead to infinite yaw rotation

$$\frac{\dot{\psi}}{\delta} \rightarrow \infty$$

- critical velocity v_{cr} is the velocity at which the yaw amplification factor strives towards an infinite values; it is defined for $EG < 0$

$$v_{cr} = \sqrt{-\frac{l}{EG}}$$

$$v_{cr} = \sqrt{\frac{l^2}{m} \frac{C_{\alpha F} C_{\alpha R}}{l_F C_{\alpha F} - l_R C_{\alpha R}}}$$

Lateral Vehicle Dynamics Characteristics velocity

- characteristic velocity v_{ch} is velocity at which the yaw amplification factor reaches its maximum

$$\frac{d}{dv} \left(\frac{\dot{\psi}}{\delta} \right) = \frac{l - EGv^2}{(l + EGv^2)^2} = 0$$

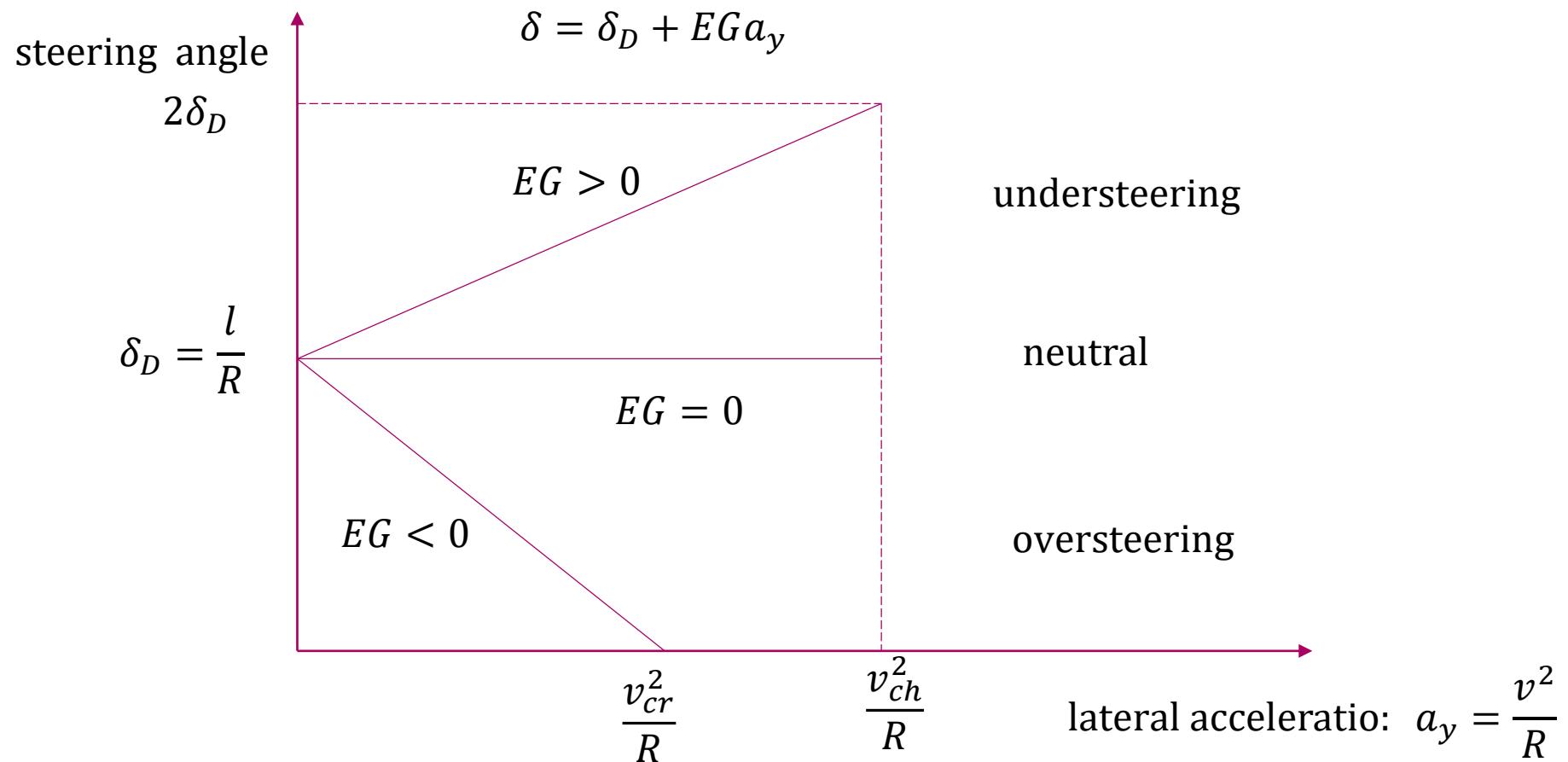
$$v_{ch}^2 = \frac{l}{EG}$$

$$v_{ch} = \sqrt{\frac{l^2}{m} \frac{C_{\alpha F} C_{\alpha R}}{l_R C_{\alpha R} - l_F C_{\alpha F}}}$$

- typical values for the characteristic velocity are between 65 and 100km/h

Lateral Vehicle Dynamics

Steering driving behaviour depending on steering gradient



POWER TRAIN OVERVIEW

Agenda

1. Car ECU architecture
2. Power Train: What is it and what for?
3. Vehicle communication
4. ECU SW Architecture (AUTOSAR)

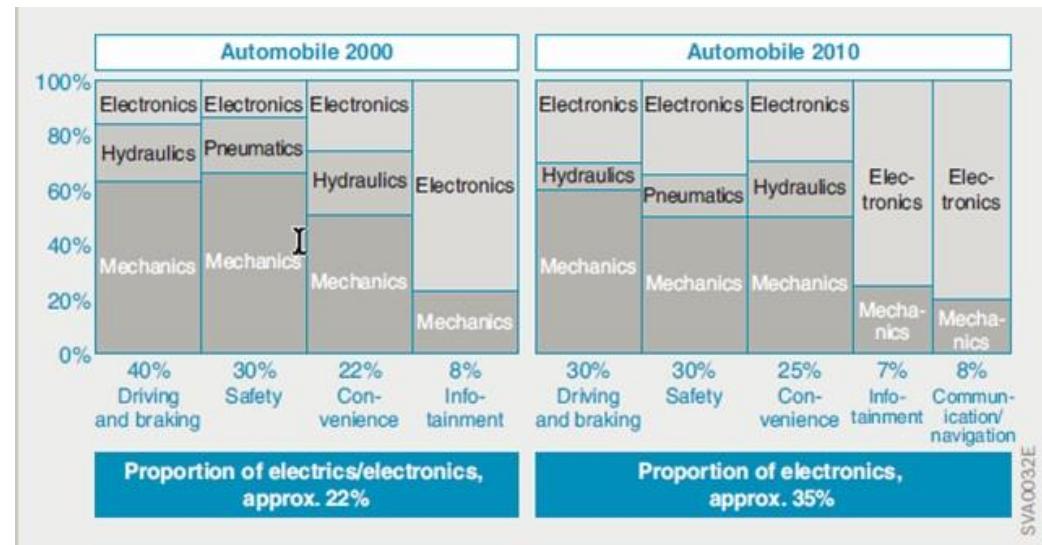
Agenda

- 1. Car ECU architecture**
2. Power Train: What is it and what for?
3. Vehicle communication
4. ECU SW Architecture (AUTOSAR)

1. Car ECU architecture

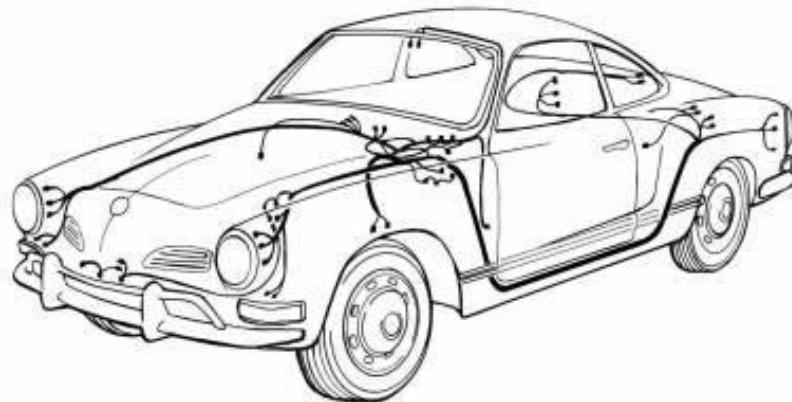
History

- In 1950 electrical network comprised approx. 40 lines (battery, starter, ignition and the lighting and signalling systems)
- Today in a premium-class vehicle, there may be up to 80 control units performing their duties.
- A modern car can have over 200 microcontrollers



1. Car ECU architecture

Feature comparison



Vintage

- Aprox. 40 wires
- Headlights and backlights
- Horn
- Electric wipers
- Radio
- Seatbelts

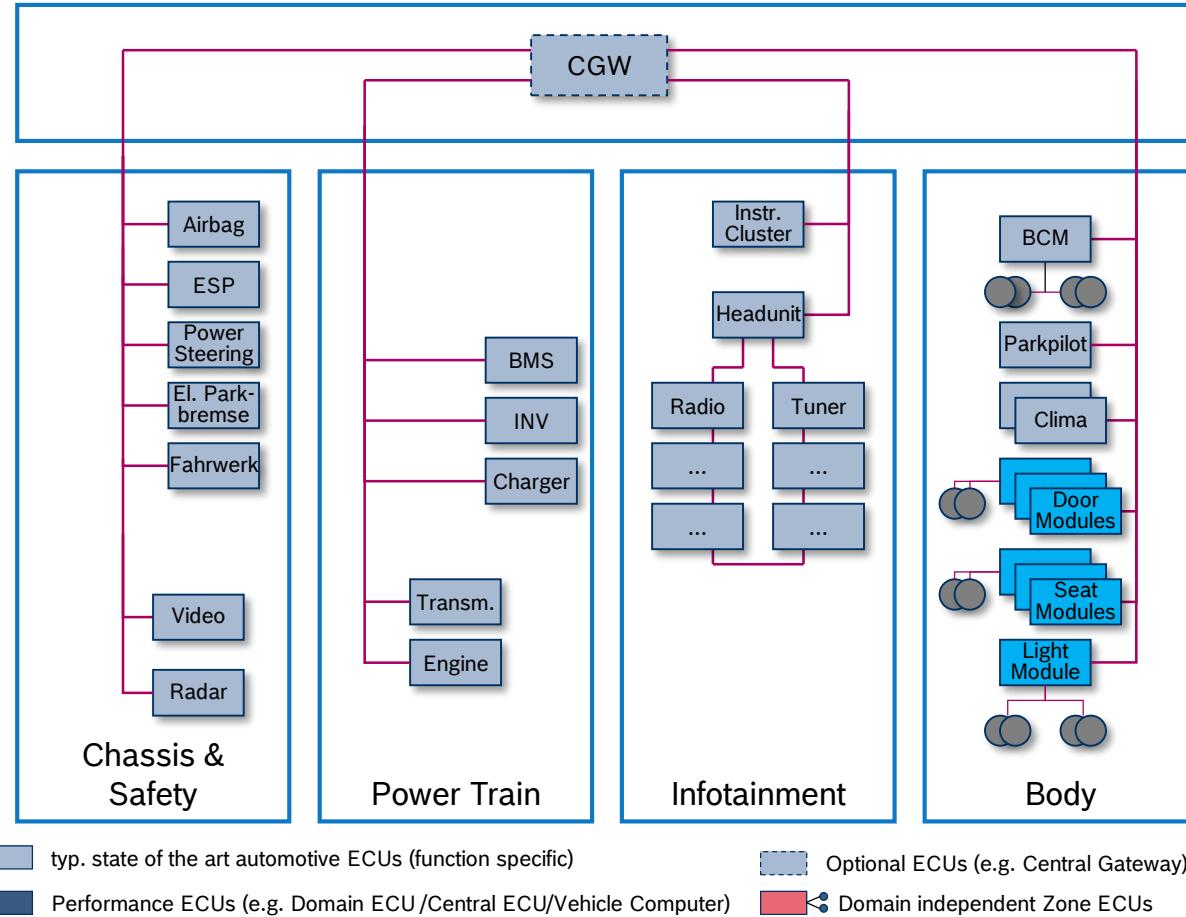


Modern

- 800 to 1000 wires.
- Traction control
- Dynamic torque distribution
- Hill decent assist
- Hill start assist
- Launch control
- Regenerative braking
- Start-stop
- Pre-tensioning seatbelts
- ABS
- ESP
- Park pilot
- ACC
- Lane assist
- Adaptive headlights
- 4 zone AC
- Infotainment
- Internet connectivity
- Tire pressure monitoring
- Heated and cooled seats
- Heated steering wheel
- Voice commands
- Airbags
- Emergency braking
- Soft close doors
- Automatic boot
- ...

1. Car ECU architecture

ECU Domains



Main properties & characteristics

- ▶ mainly encapsulated E/E architecture structure
- ▶ each function has his “own” ECU
- ▶ partly merge of ECUs
 - examples:
 - Parking-Assist ECU into Body Control Module (BCM)
 - APB into ESP

Nomenclature

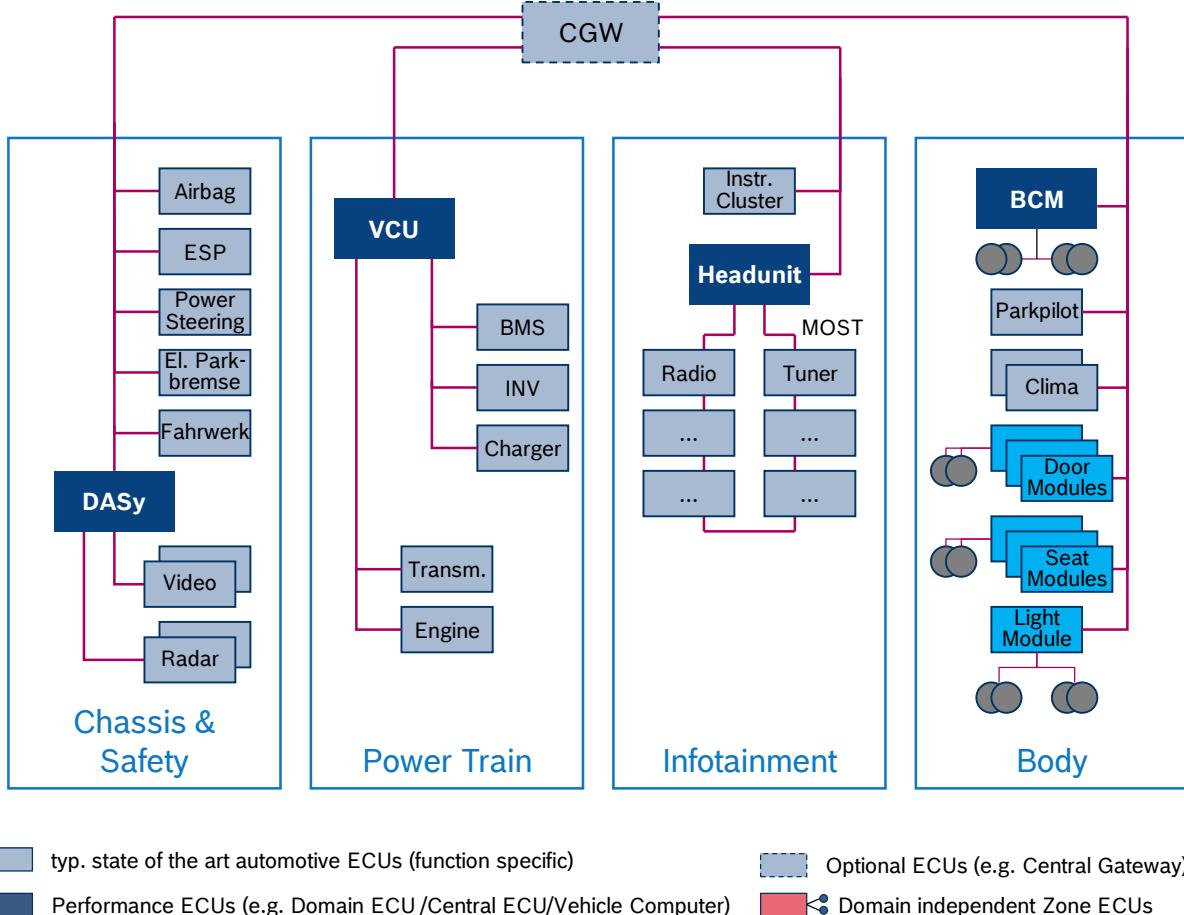
- ▶ “Function Specific ECUs”
e.g. ESP = stability functions, Engine ECU = engine functions
- ▶ “Domain Specific Zone ECUs”
e.g. Body-Domain Door/Roof/Light-ECU

Options / Variants

- ▶ CGW as stand alone ECU or pot. integrated in e.g. BCM

ECU architecture

Domains 2



Main properties & characteristics

- ▶ Mainly domain centralized E/E architecture
- ▶ Domain centralized ECUs as “master” of the domain for intra-domain functions (integration platform for each domain)
- ▶ Motivation: Higher performance requirements and standardization of sensor/actuator ECUs

Nomenclature

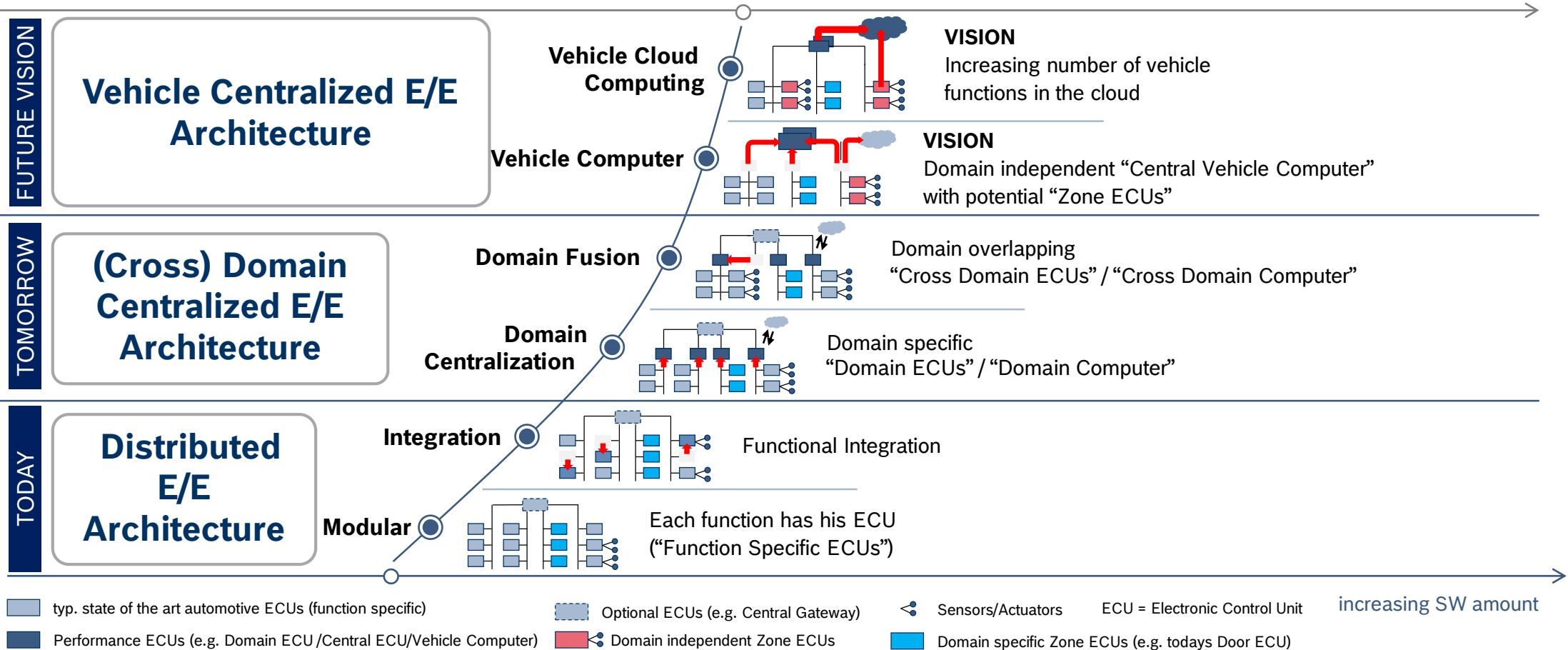
- ▶ “Domain specific Central ECUs”
e.g. DASy, VCU, HU
- ▶ “Function Specific ECUs”
e.g. ESP = stability functions, Engine ECU = engine functions
- ▶ “Domain Specific Zone ECUs”
e.g. Body-Domain Door/Roof/Light-ECU

Options / Variants

- ▶ Not every domain will have a domain specific “Central ECUs”
- ▶ Not every domain specific “Central ECUs” will have the same performance requirements

ECU architecture

Next steps



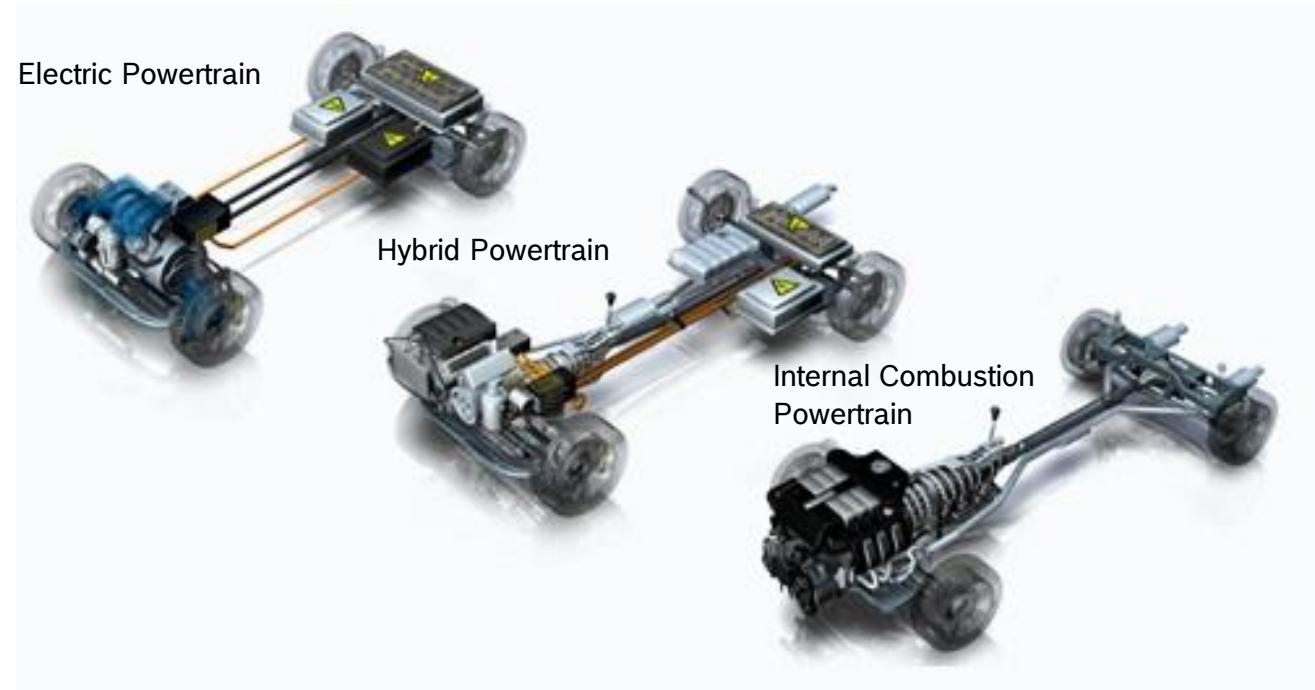
Agenda

- 1. Car ECU architecture**
2. Power Train: What is it and what for?
3. Vehicle communication
4. ECU SW Architecture (AUTOSAR)

2. Power train

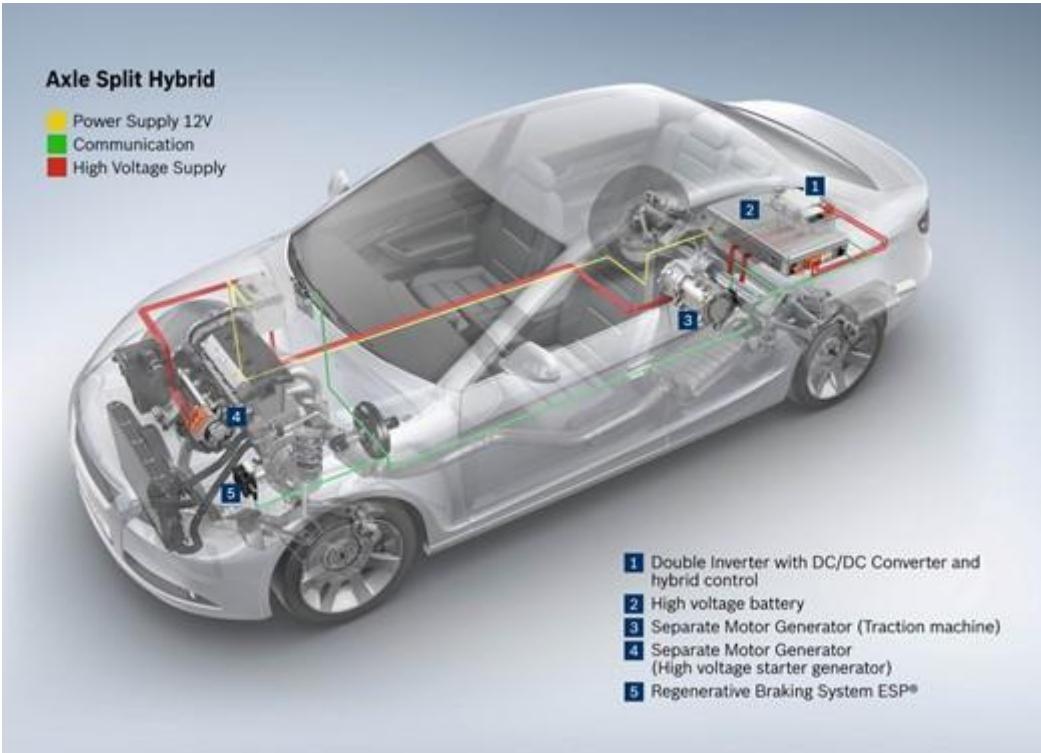
Overview

- The term **power train** or **power plant** describes the main components that generate power and deliver it to the road surface, water, or air.
- This includes the engine, transmission, drive shaft, differentials, and the final drive (drive wheels, continuous track as in caterpillar tractors).
- Types of powertrain available:
 - Internal combustion
 - Hybrid
 - Plug-in hybrids
 - Non Plug-in hybrids
 - Full Electric

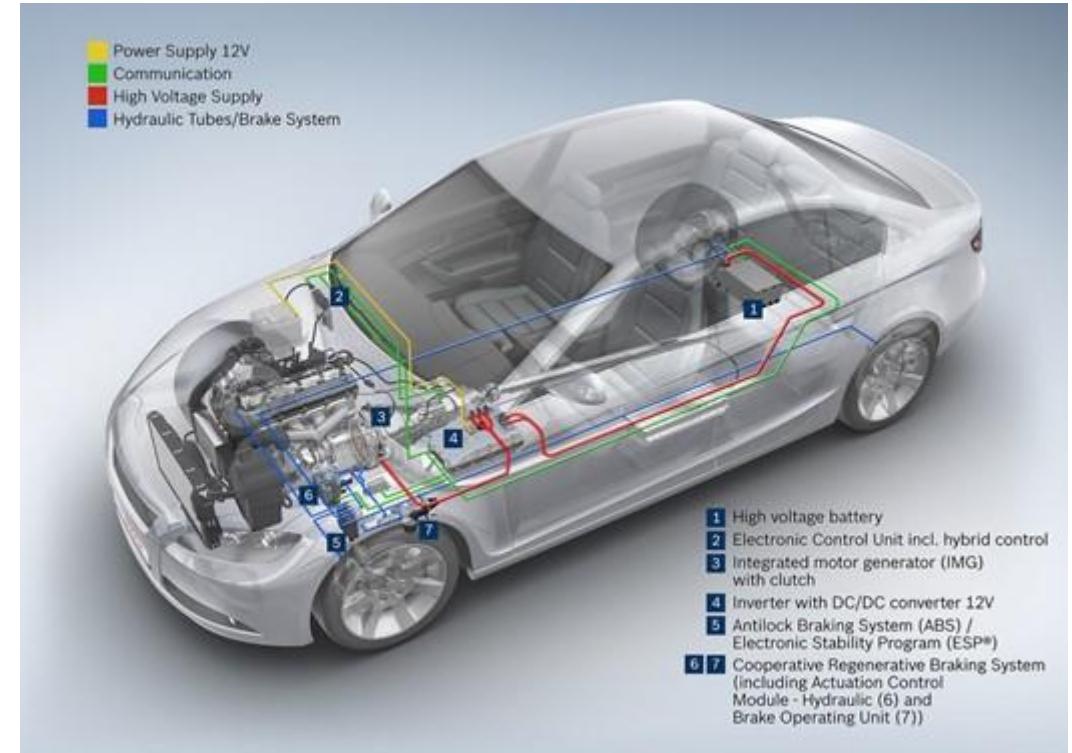


2. Power train

Hybrid drive



Plug-in Hybrid

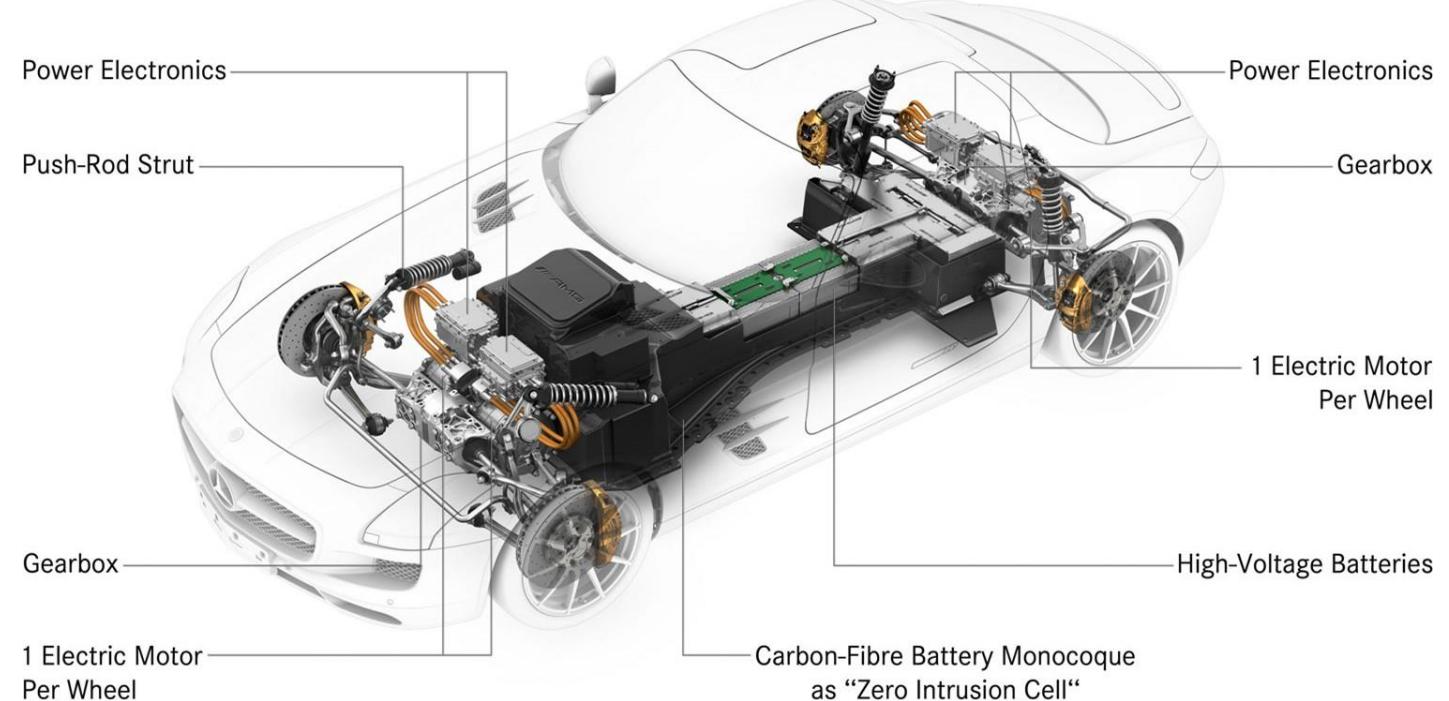


Hybrid

2. Power train

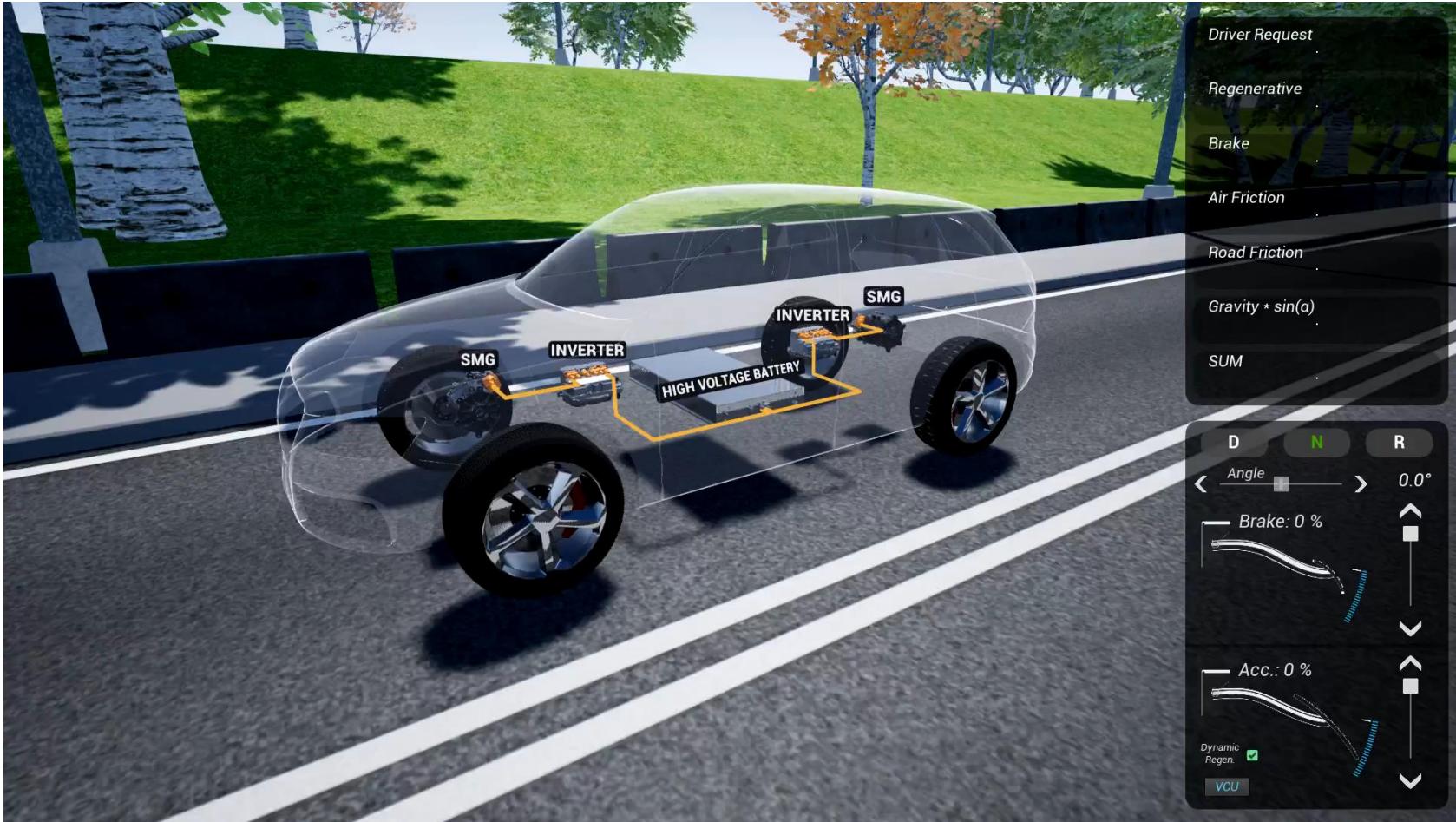
Electric drive

- Main components:
 - Electric motor
 - Invertor
 - Battery management
 - Battery (20 – 100 kWh)
- Functionalities
 - Traction control
 - Torque vectoring
 - Dynamic power distribution
 - Regenerative braking



2. Power train

Regenerative Braking



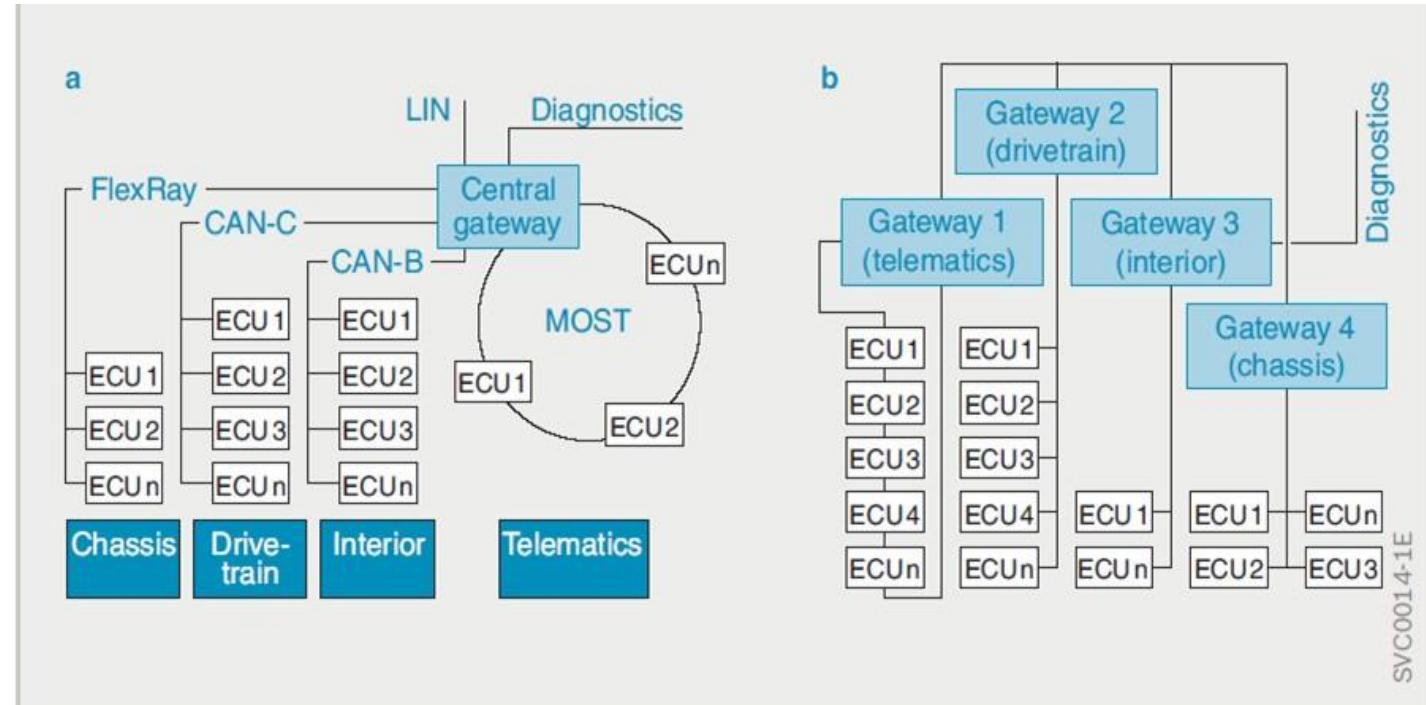
Agenda

1. Car ECU architecture
2. Power Train: What is it and what for?
- 3. Vehicle communication**
4. ECU SW Architecture (AUTOSAR)

3. Vehicle communication

Overview

- Chassis/ Safety domain:
 - Bus type: Flexray
 - Topology: Star topology
 - Data rates: Max. 20 Mbit/s
 - Standard: Flexray consortium
 - SAR Classification: Drive-by-wire
- Powertrain/ Drivetrain domain:
 - Bus type: High Speed CAN
 - Topology: Linear Bus
 - Data rates: 10kbit/s – 1Mbit/s
 - Standard: ISO 1198
 - SAR Classification: Class C
- Body/ Interior domain:
 - Bus type: Low Speed CAN
 - Topology: Linear Bus
 - Data rates: Max. 125 kbit/s
 - Standard: ISO 11519-2
 - SAR Classification: Class B
- Infotainment / Telematics domain:
 - Bus type: MOST
 - Topology: Ring topology
 - Data rates: Max. 22.5 Mbit/s
 - Standard: MOST cooperation
 - SAR Classification: Mobile Media



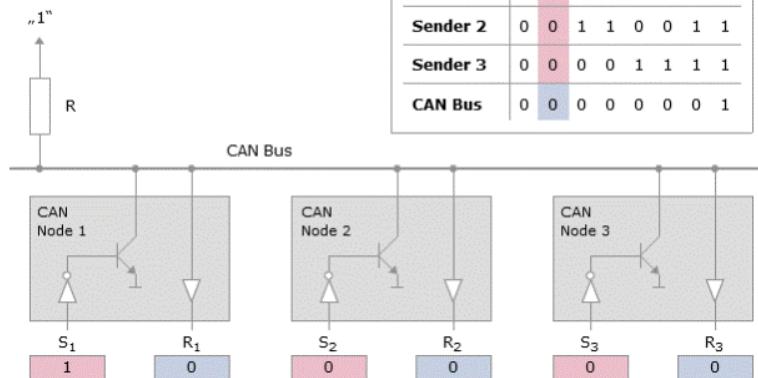
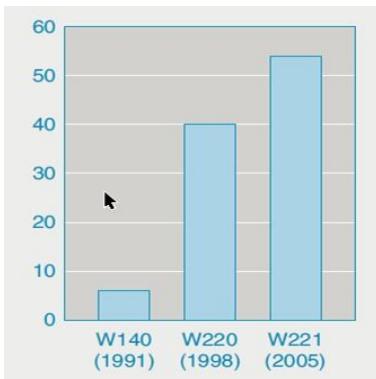
a. Central Gateway topology

b. Distributed Gateway topology

3. Vehicle communication

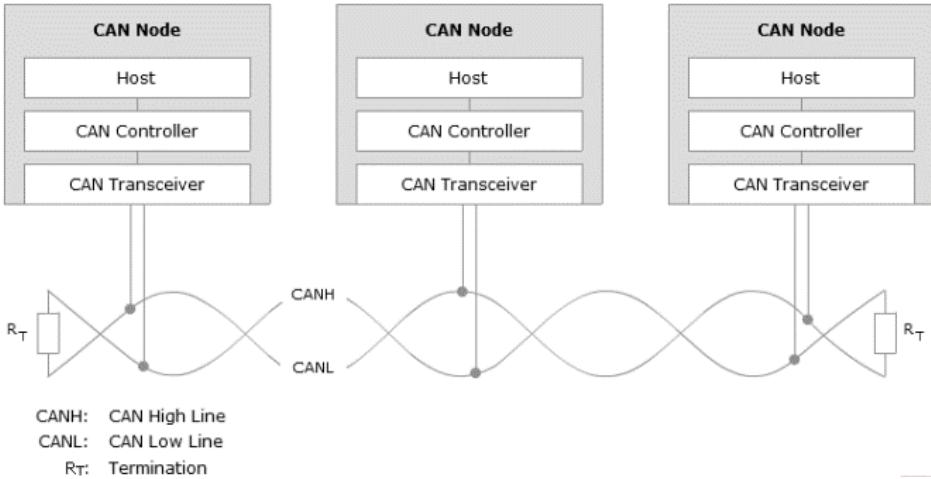
CAN Overview

- Development of the CAN bus started in 1983 at Robert Bosch GmbH.
- The first CAN controller chips, produced by Intel and Philips, came on the market in 1987
- In 1991 the CAN bus (Controller Area Network) was the first bus system to be introduced to a motor vehicle in mass production

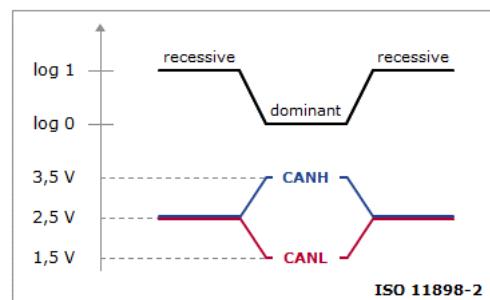


CAN Arbitration

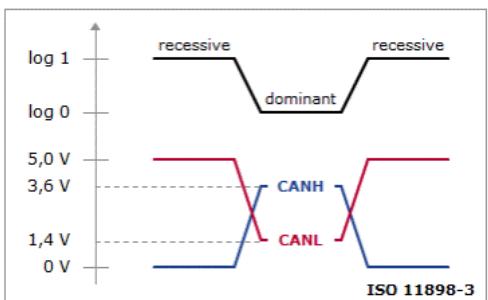
- ISO 11898 specifies
 - 1 MBit/s for 40 m (specified)
 - 500 kBit/s up to 100 m (recommendation)
 - 250 kBit/s up to 250 m
 - 125 kBit/s up to 500 m
 - 40 kBit/s up to 1,000 m



ECU connection



High speed CAN



Low speed CAN

3. Vehicle communication

CAN Frame

- Standardized in accordance with ISO 11898
- Prioritized communication
- Data transfer rates: up to 1 MBits/s
- Data capacity: up to 8 bytes per message
- Real-time response
- Very high reliability of data transfer
- Fault detection and signaling

Short-circuit resistance

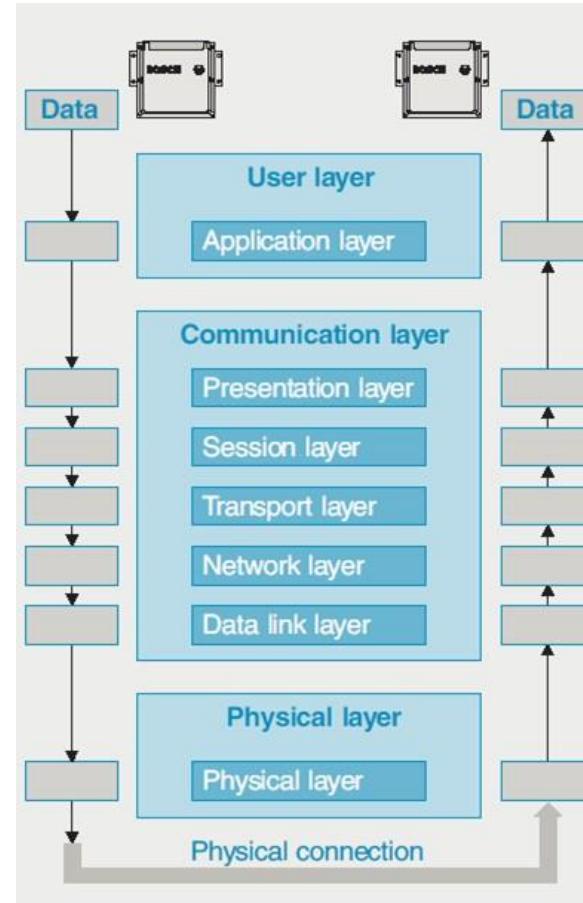
Frame types:

- | | |
|----------------|----------------------------------|
| • Data frame | Error detection |
| • Remote frame | • <i>Cyclic redundancy check</i> |
| • Error frame | • <i>Frame check</i> |
| | • <i>ACK check</i> |
| | • <i>Monitoring</i> |

ISO OSI reference model (Open Systems

Interconnection) used to describe the protocol communicate:

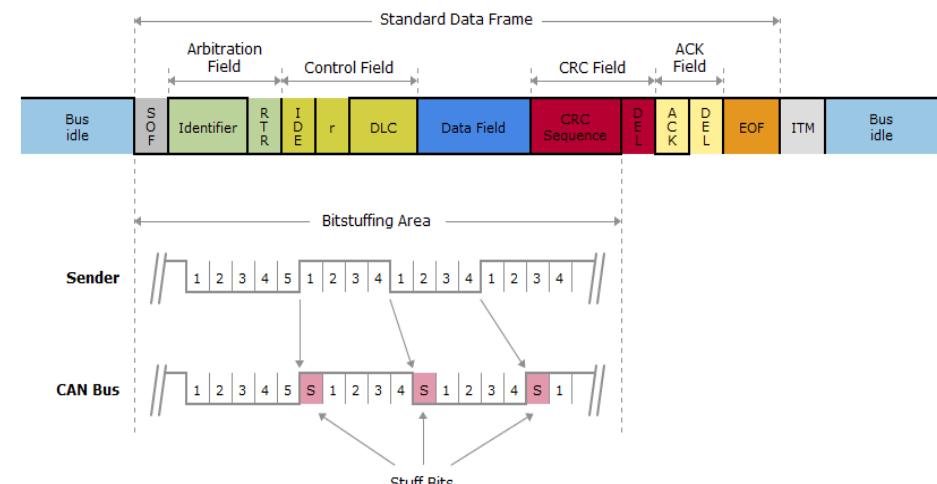
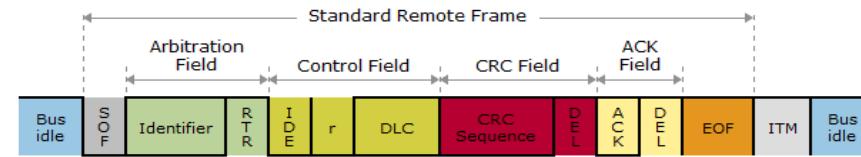
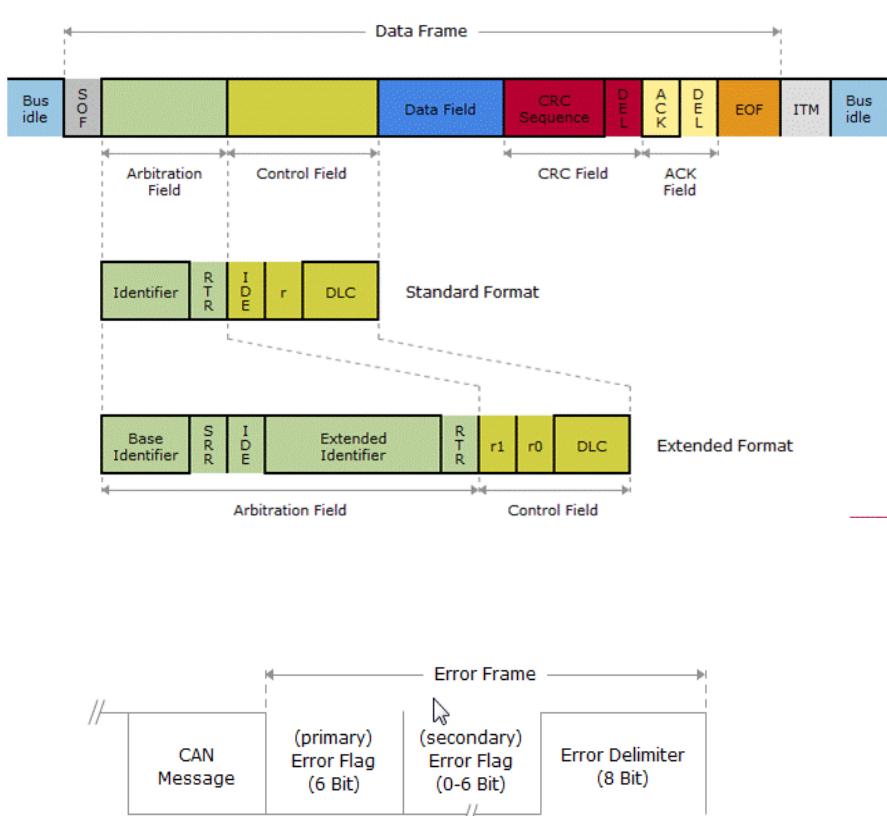
- Application layer
- Communication Layer
- Physical layer



CAN OSI Model

3. Vehicle communication

CAN Frame Types

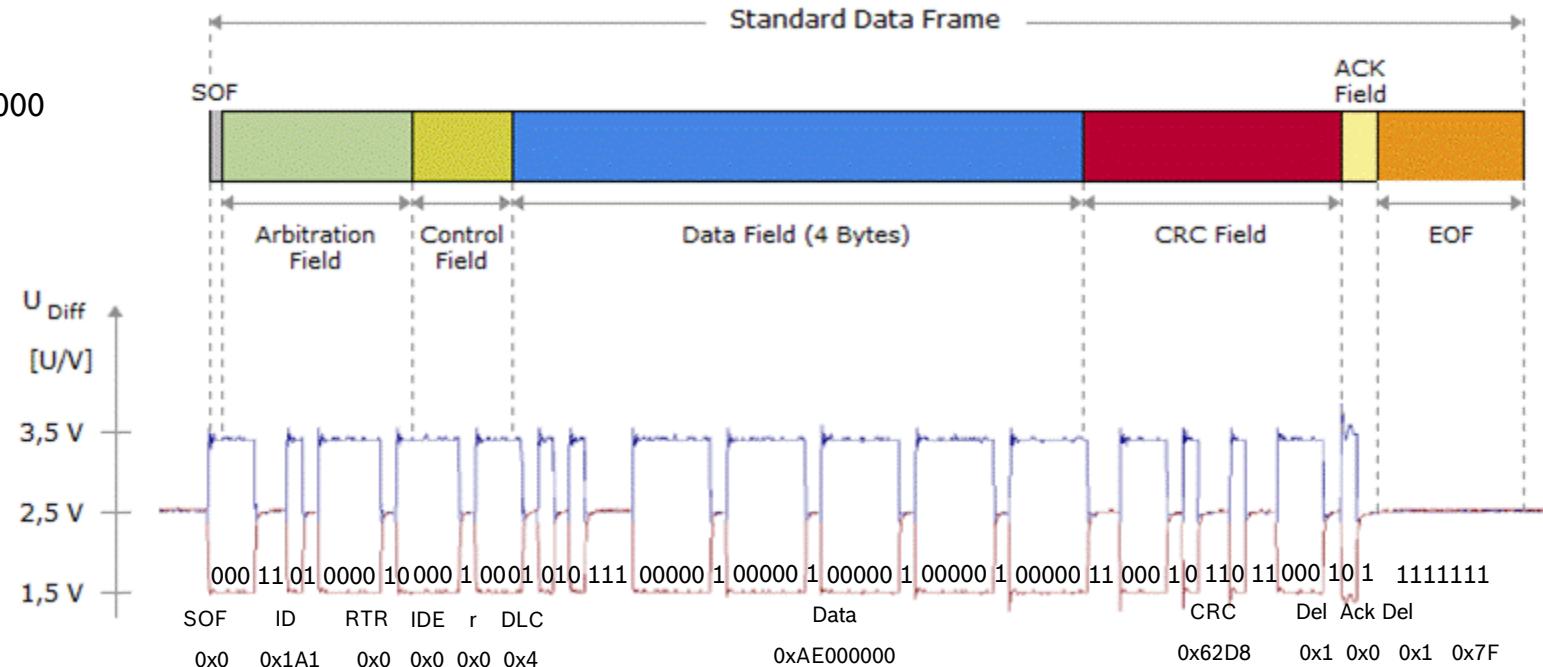


3. Vehicle communication

CAN Example

Send standard data frame:

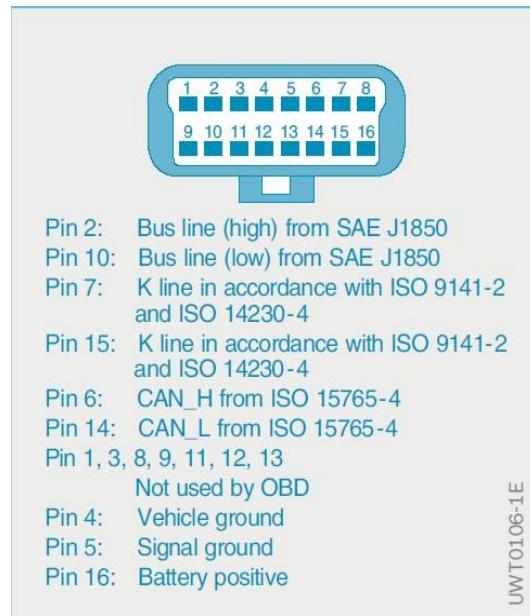
- ID – 0x1A1
- DLC – 0x4
- Data – 0xAE000000



3. Vehicle communication

Diagnostics protocols

- Emissions diagnostics functionality is mandatory by legislation CARB (California Ari Resource Board) and EOBD (European On board diagnostics)
- CAN baud rates are 500 kBd or 1 MBd.
- Addressing and message types Tester communication on the CAN bus is defined by ISO 15765.
- The communication services of ISO 15765-3 or 14 229-1 are defined in a similar way to those of ISO 14230-3
- OEM and workshops use the OBD port for troubleshooting, calibration and maintenance



OBD connector

Layer	CARB			
	K line	CAN		
7	ISO 15031-5	ISO 15031-5	ISO 15031-5	ISO 15031-5
6				
5				ISO 15765-4
4				
3				ISO 15765-2 ISO 15765-4
2	ISO 9141-2	ISO 14230-2 ISO 14230-4	SAE J1850	ISO 11898 ISO 15765-4
1	ISO 9141-2	ISO 14230-1 ISO 14230-4	SAE J1850	ISO 11898 ISO 15765-4

Layer	Manufacturer-specific	
	K line: KWP 2000	CAN / UDS
7	ISO 14230-3	ISO 15765-3 ISO 14229-1
6		
5		ISO 15765-3
4		
3		ISO 15765-2
2	ISO 14230-2	ISO 11898-1
1	ISO 14230-1	ISO 11898

Diagnostic protocols

- a CARB communication
b Customer-specific communication

Layers of the OSI reference model

- 7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data link
1 Physical

UDS Unified Diagnostic Services

3. Vehicle communication

UDS (Unified diagnostics services)

- A diagnostic communication protocol in the electronic control unit (ECU) environment within the automotive electronics, which is specified in the ISO 14229-1. It is derived from ISO 14230-3 (KWP2000) and ISO 15765-3 (Diagnostic Communication over Controller area network (DoCAN))

SID (hex)	Service
0x10	Diagnostic Session Control service
0x11	ECU Reset service
0x27	Security Access service
0x28	Communication Control service
0x3E	Tester Present service
0x83	Access Timing Parameter service
0x84	Secured Data Transmission service
0x85	Control DTC Setting service
0x86	Response On Event service
0x87	Link Control service

UDS Diagnostic services

Hex	Sub-function
0x00	Not defined
0x01	Hard reset
0x02	Key Off On Reset
0x03	Soft reset
0x04	Enable Rapid Power Shutdown
0x05	Disable Rapid Power Shutdown
0x06 – 0x3F	Not defined
0x40 – 0x5F	Vehicle Manufacturer Specific
0x60 – 0x7E	System Supplier Specific
0x7F	Not defined

Available subservices for
ECU Rest Service

Tx (transit message)

0.	1.	2.	3.	4.	5.	6.	7.
Length	SID	sub-service	Padding	Padding	Padding	Padding	Padding

Rx (received message)

Case 1: Positive response

0.	1.	2.	3.	4.	5.	6.	7.
Length	SID+0x40	sub-service	Padding	Padding	Padding	Padding	Padding

Case 2: Negative response

0.	1.	2.	3.	4.	5.	6.	7.
Length	0x7F	SID	NRC	Padding	Padding	Padding	Padding

Request and response frames

3. Vehicle communication

CAN UDS request example

- Using service ID 0x10 – Diagnostics session control
- Change diagnostics session to Extended Diagnostics Session
(Sub service 0x03)

Tx

0.	1.	2.	3.	4.	5.	6.	7.
0x02	0x10	0x03	x	x	x	x	x

Rx Positive response

0.	1.	2.	3.	4.	5.	6.	7.
0x02	0x50	0x03	x	x	x	x	x

Rx Negative response:

0.	1.	2.	3.	4.	5.	6.	7.
0x03	0x7F	0x10	NRC	x	x	x	x

NRC	Description	Mnemonic
0x12	sub Function Not Supported	SFNS
0x13	incorrect Message Length Or Invalid Format	IMLOIF
0x22	conditions Not Correct	CNC
0x24	request Sequence Error	RSE
0x31	request Out Of Range	ROOR
0x33	security Access Denied	SAD
0x35	invalid Key	IK
0x36	exceeded Number Of Attempts	ENOAA
0x37	required Time Delay Not Expired	RTDNE
0x38 – 0x4F	reserved By Extended Data Link Security Document	RBEDLSD
0x70	upload Download Not Accepted	UNDNA
0x71	transfer Data Suspended	TDS
0x72	general Programming Failure	GFP
0x73	wrong Block Sequence Counter	WBSC
0x93	voltage Too High	VTH
0x93	voltage Too Low	VTL

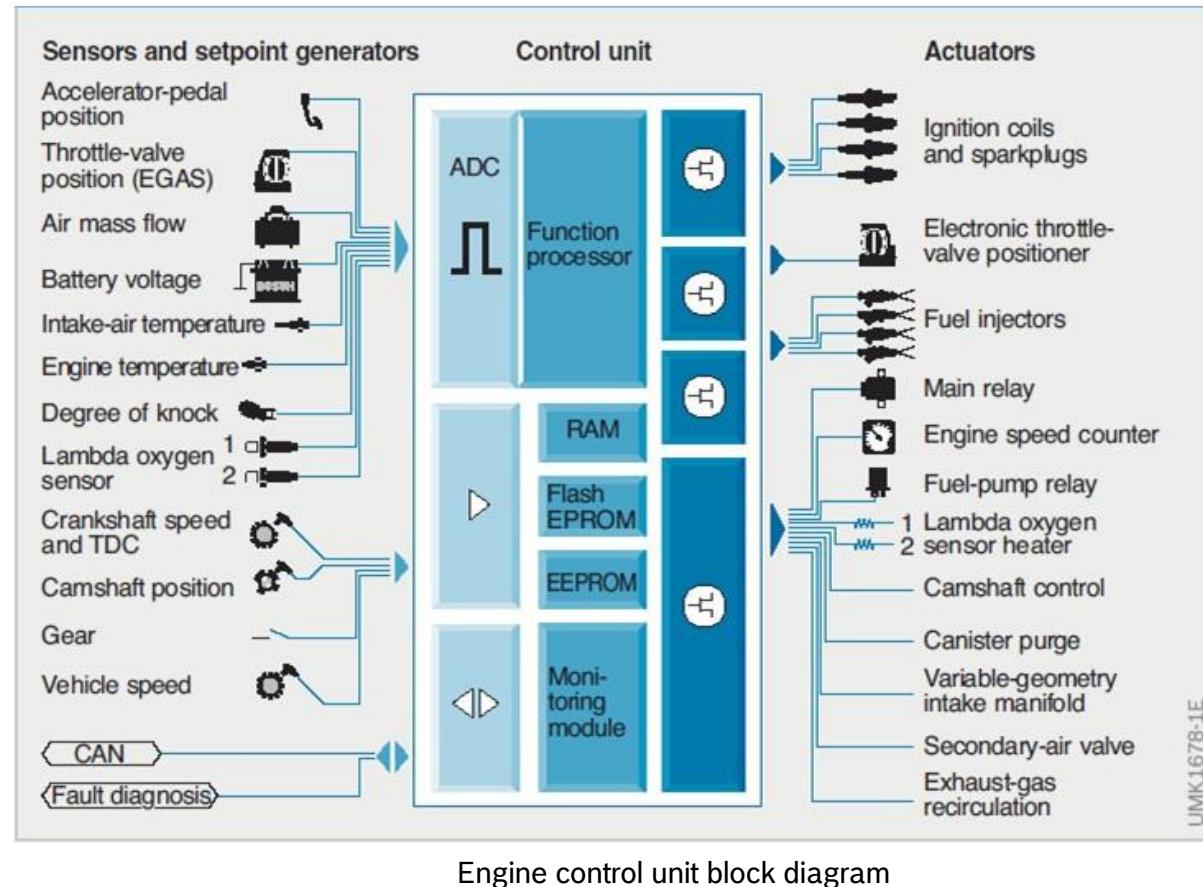
Negative response code (NRC) Table

Agenda

1. Car ECU architecture
2. Power Train: What is it and what for?
3. Vehicle communication
- 4. ECU SW Architecture (AUTOSAR)**

4. ECU SW Architecture

Why a standard SW Arch.?



4. ECU SW Architecture

AUTOSAR Overview

► AUTomotive Open System Architecture



AUTOSAR – Core Partners and Members

Status: 30th September 200



Courtesy of



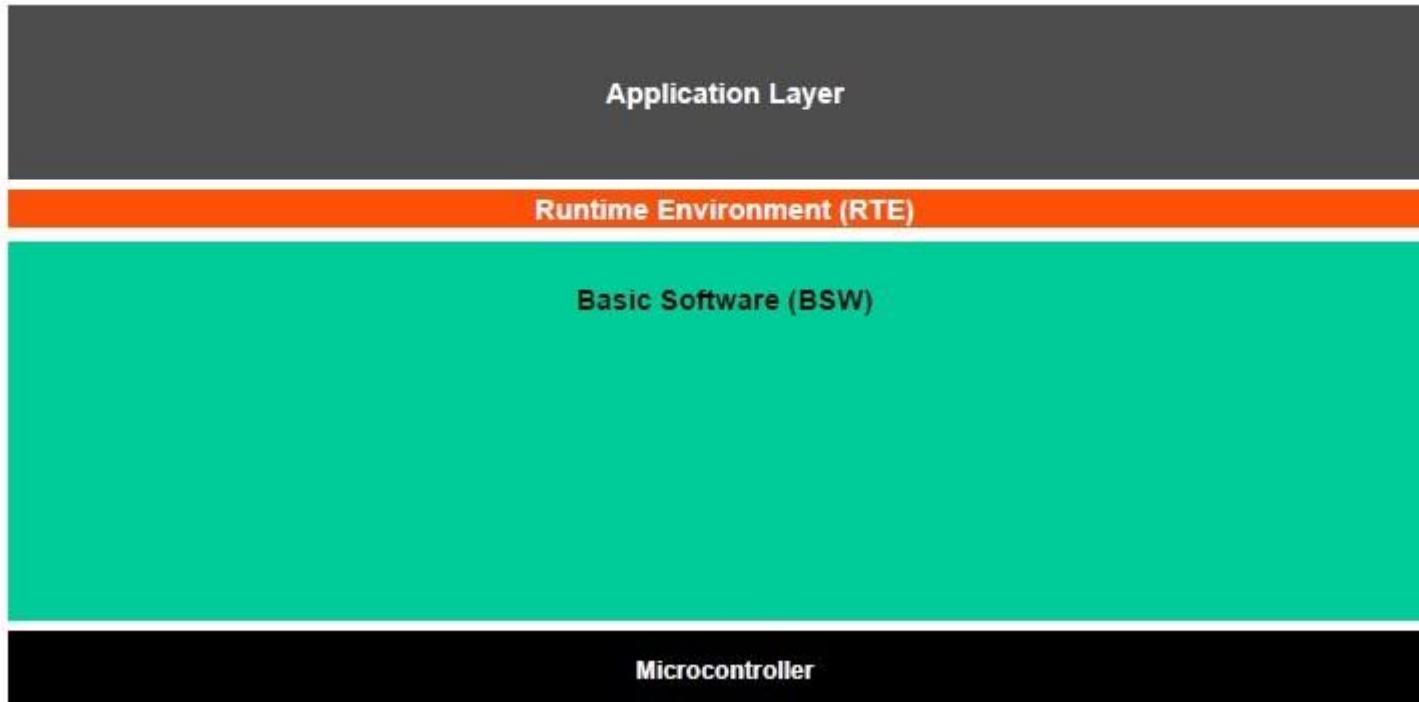
4. ECU SW Architecture

Motivation and Goals

- ▶ Scalability and re-use of Basic SW across OEMs and vehicle platforms
- ▶ Support integration of 3rd party Basic SW during design time
- ▶ Support integration of 3rd party Application SW Components during design time
- ▶ Support relocate ability of Application SW Components during design time
- ▶ Validation of Application SW Interfaces on vehicle level

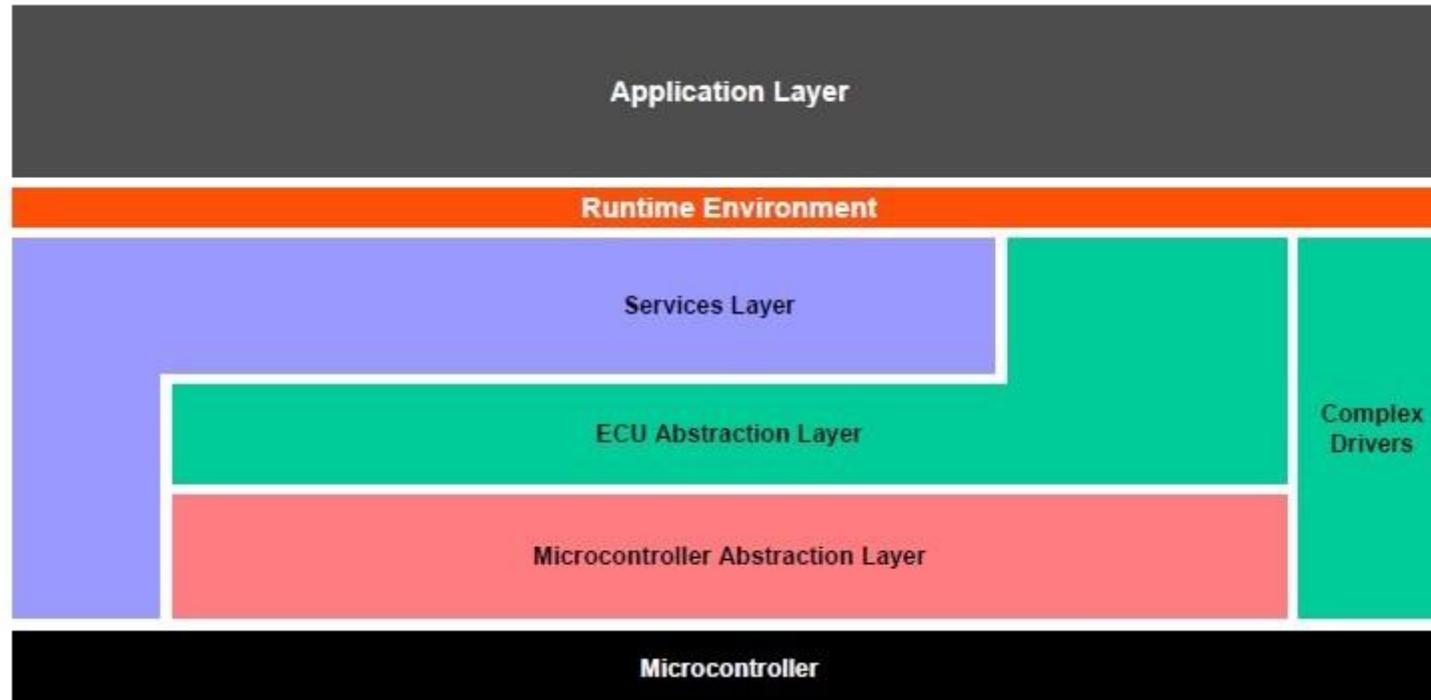
4. ECU SW Architecture

SW Structure



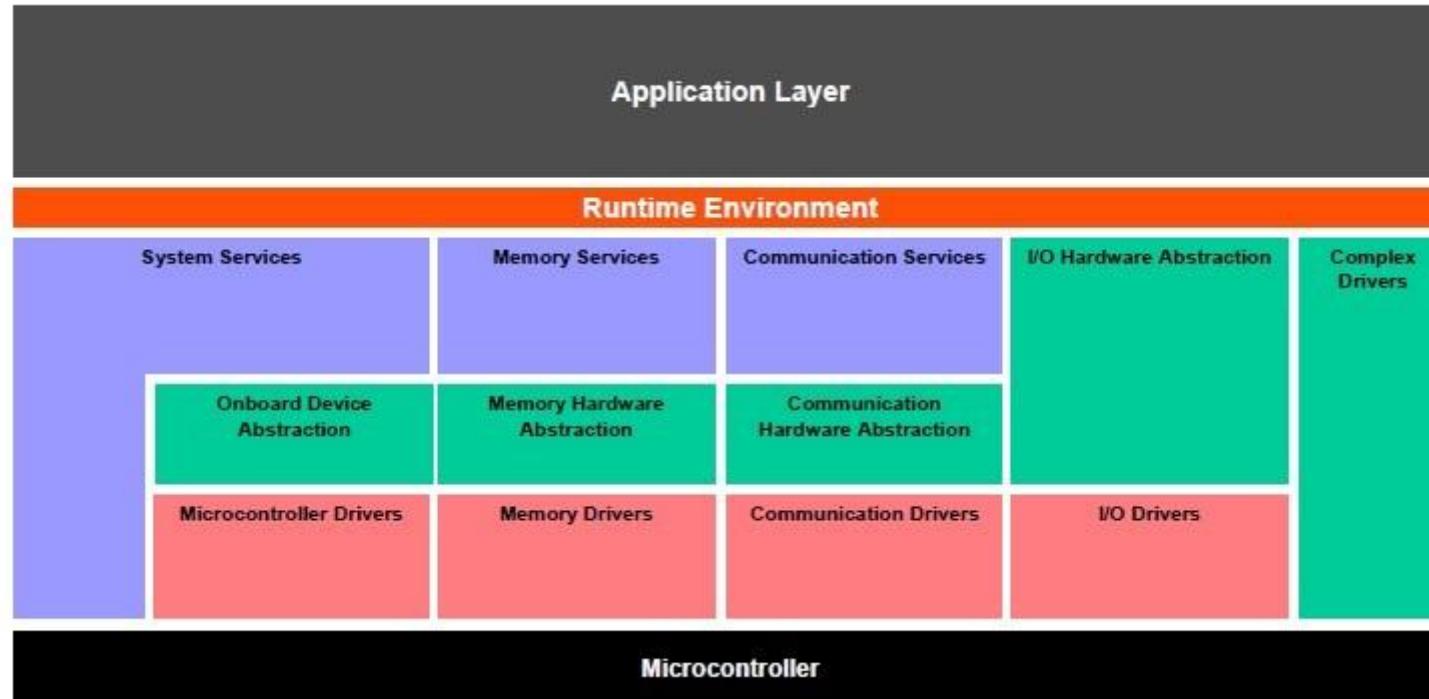
4. ECU SW Architecture

SW Structure



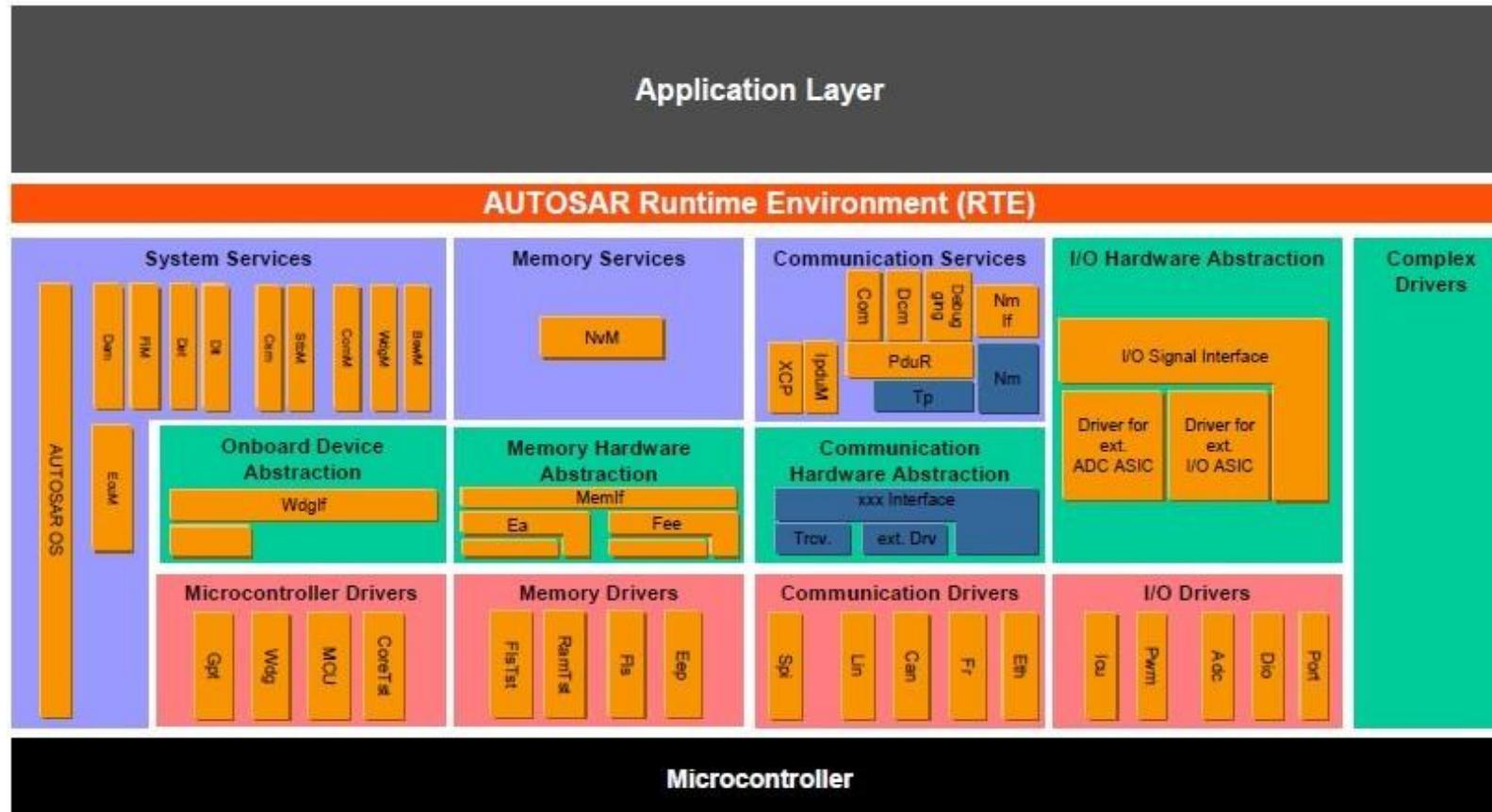
4. ECU SW Architecture

SW Structure



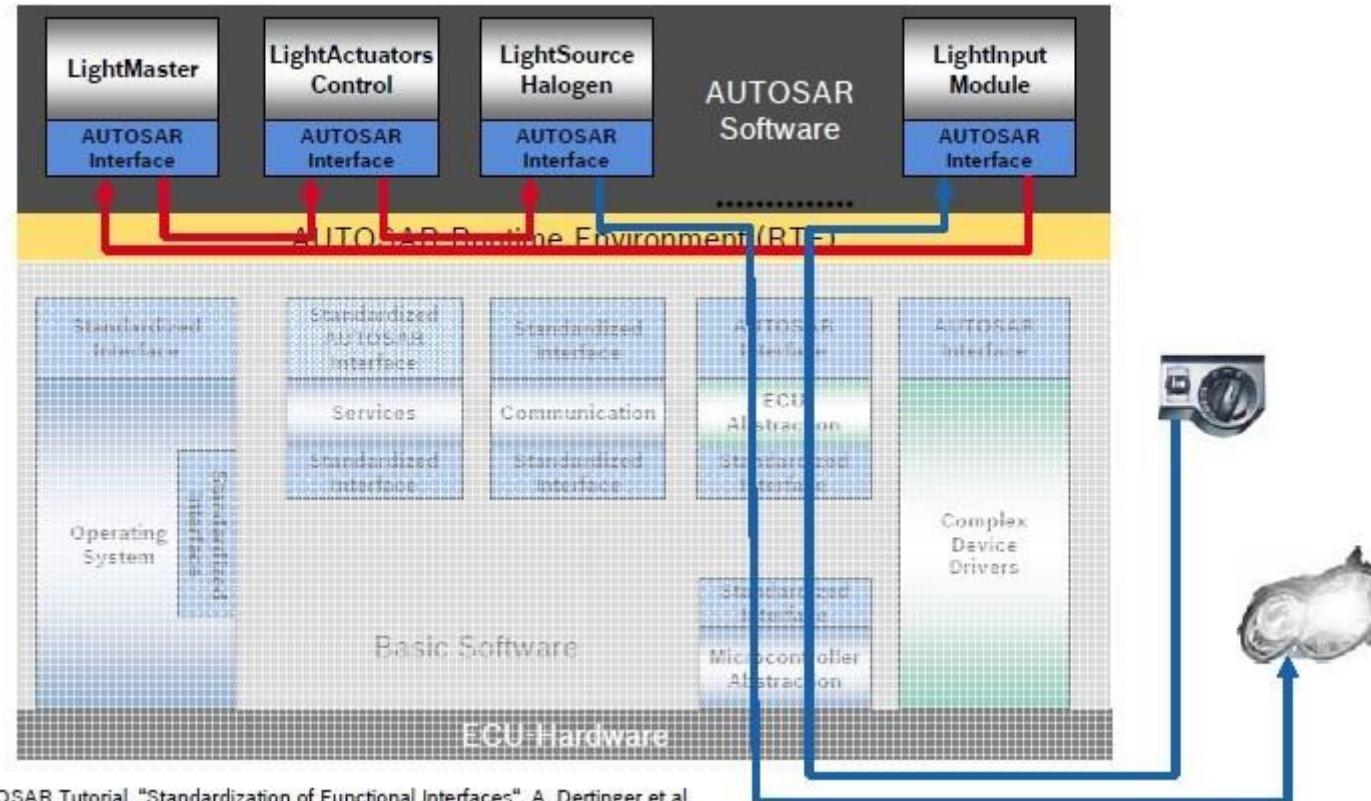
AUTOSAR Overview

SW Structure



4. ECU SW Architecture

SW Structure



Bibliography

- *Bosch Automotive Electrics and Automotive Electronics. Systems and Components, Networking and Hybrid Drive, 5th Edition, Robert Bosch GmbH Ed.*
- *Automotive Mechatronics Automotive Networking · Driving Stability Systems · Electronics, Konrad Reif Ed.*
- <http://www.bosch.co.jp/en/press/group-1306-04.asp>
- <https://www.autosar.org/standards/classic-platform/>
- https://www.researchgate.net/figure/Automotive-wire-harness_235699269

THANK YOU

