Droid Equinox 10

Artificial Intelligence for Autonomous Vehicles

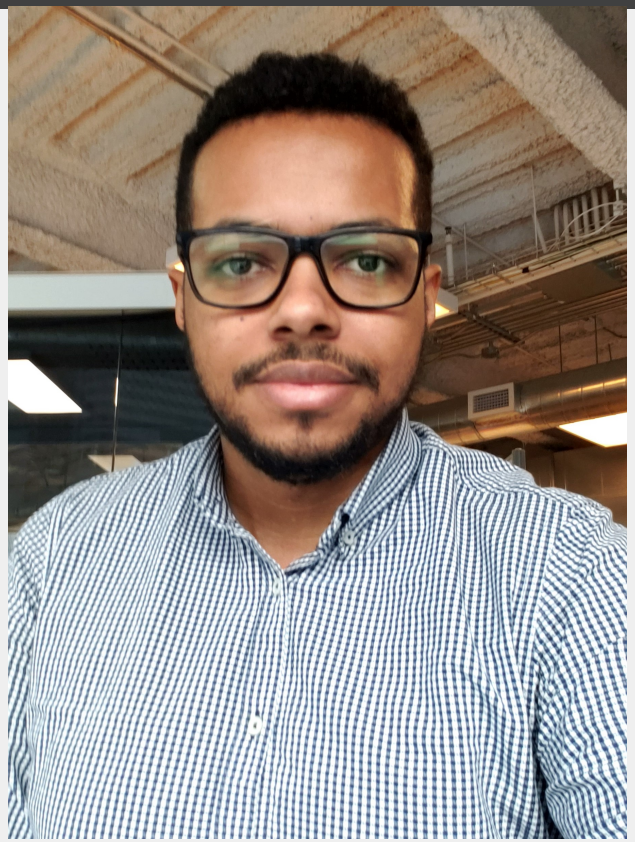
Vinícius Araújo Santos

Machine Learning Engineer **DATA H**

02/06/2020

Speaker

Received his B.S. degree in Computer Engineering from the Universidade Federal de Goiás in 2015. He received MsC. in Computer Science in Computer Science also at UFG in 2018. Researcher in Machine Learning at DATA H Artificial Intelligence developing solutions for autonomous vehicles . He was a member of robotics team Pequi Mecânico, working in the development of soccer robots between 2013 and 2018.



www.linkedin.com/in/viniciusarasantos

vinicius.santos@datah.ai

+1 905 242-7069

Agenda

- **Introduction**
  - Why Autonomous Vehicles?
  - Are Emergent Markets ready for this technology ?
  - Is this a reasonable Career to pursue ?
- **The Software Stack of Autonomous Vehicles**
  - Sensing and Scene Perception
  - External Maps and Localization
  - Resources Management and Decision Making
  - Motion and Path Planning
  - Low Level Control

Agenda

- **The ROS Interface**
  - ROS Basic Structure
  - Topics and Services
  - Using ROS in Python 3
  - Jump Start with ROS Packages
- **Gazebo**
  - Robot simulation
  - Gazebo integration with ROS

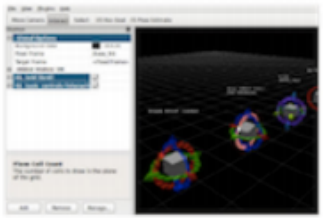
Slides

Please access: https://bit.ly/39dL2hR
Wifi password: rawtal3nt!

The ROS Interface

What is ROS ?

ROS = Robot Operating System


+

+

+


Plumbing	Tools	Capabilities	Ecosystem
▪ Process management	▪ Simulation	▪ Control	▪ Package organization
▪ Inter-process communication	▪ Visualization	▪ Planning	▪ Software distribution
▪ Device drivers	▪ Graphical user interface	▪ Perception	▪ Documentation
	▪ Data logging	▪ Mapping	▪ Tutorials
		▪ Manipulation	

ROS Basic Structure

What is ROS ?

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory.
- Since 2013 managed by OSRF (Open Source Robotics Foundation).
- Today used by many robots, universities and companies.
- De facto standard for robot programming.

The ROS Philosophy

Peer to peer

Individual programs communicate over defined API (ROS messages, services, etc.)

Distributed

Programs can be run on multiple computers and communicate over the network.

ROS TOPICS AND NODES

- Nodes are a single-purpose, executable program.
- Nodes communicate over topics.
- Topic is a name for a stream of messages.
- Subscriber / Publisher

ROS SERVICES

- Code a specific functionality for a robot.
- This functionality is available to call it.
- Example: Do something and then, stop.

Service

Use services when your program can't continue until it receives the result from the service.

ROS SERVICES

- Usually a service is divided in two parts:

Service Server

Provides the functionality to anyone who wants to use it (call it).

Service Client

Calls / Requests the service.

ROS SERVICES

- In services, there two steps:

Start the Service

Make it available for anyone who wants to call it.

Call the service

Execute the service.

ROS Services

- Single-purpose, executable program

Call a service:

```
> rosservice call /the_service_name TAB-TAB
```

See active services with:

```
> rosservice list
```

Retrieve information about a node with:

```
> rosservice info /name_of_your_service
```

ROS Services

- Single-purpose, executable program

Structure of the Service:

```
> rossrv show name_of_the_package/Name_of_Service_message
```

ROS ACTIONS

- Actions are like asynchronous calls to services.
- When you call an action, you are calling a functionality that another node is providing.
- Just the same as with services.
- The difference is that when your node calls a service, it must wait until the service finishes.
- When your node calls an action, it doesn't necessarily have to wait for the action to complete.
- Hence, an action is an asynchronous call to another node's functionality.

ROS ACTIONS

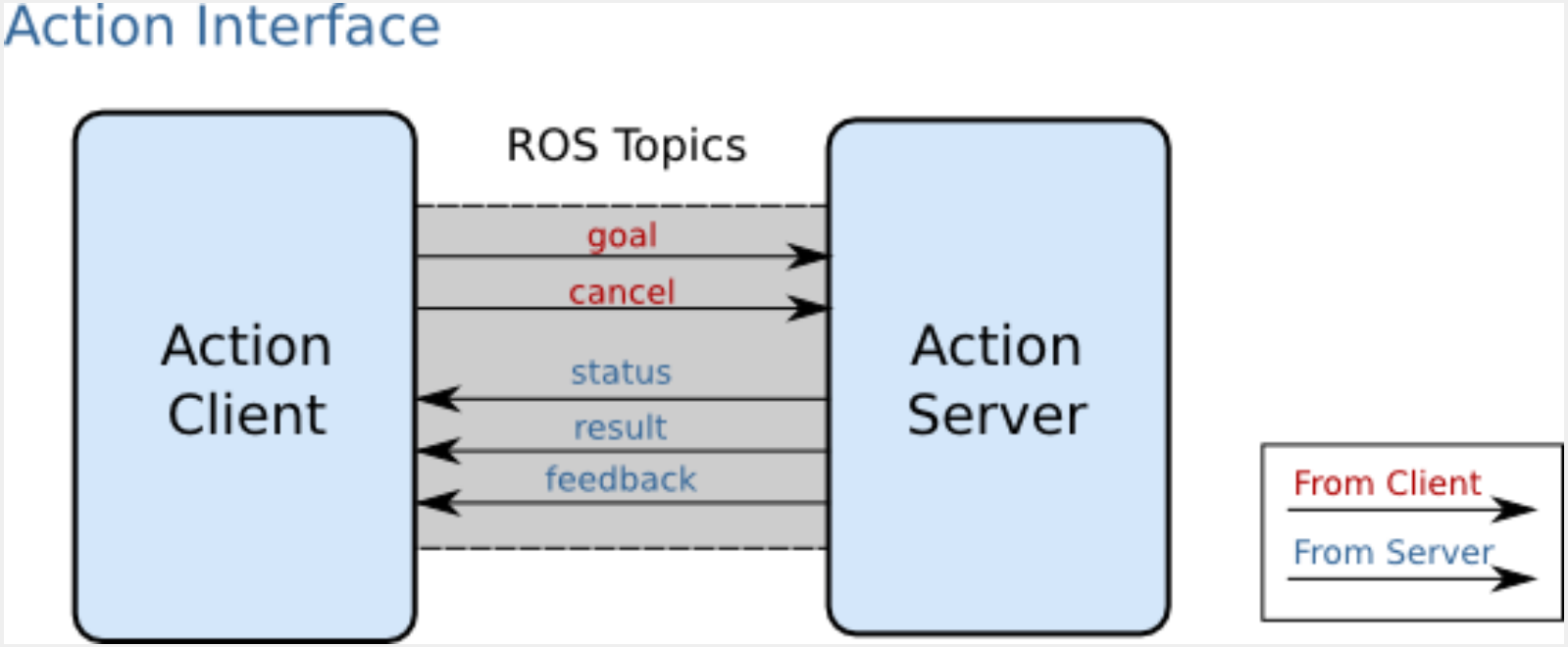
- A action, as in a service, is divided in two parts:

Action Server

The action server allows other nodes to call that action functionality. The node that provides the functionality has to contain an action server.

Action Client

The action client allows a node to connect to the action server of another node. The node that calls to the functionality has to contain an action client.


Action Interface
ROS Topics
goal
cancel
status
result
feedback
Action Client
Action Server
From Client
From Server

Preparing ROS and Gazebo

Preparing the environment:

> sudo apt-get install ros-melodic-turtlebot3
> sudo apt-get install ros-melodic-turtlebot3-gazebo

Setting environment variables:

> export TURTLEBOT3_MODEL=waffle

Preparing the environment:

> sudo apt-get install ros-melodic-joy ros-melodic-teleop-twist-joy
> sudo apt-get install ros-melodic-teleop-twist-keyboard ros-melodic-laser-proc
> sudo apt-get install ros-melodic-rgbd-launch ros-melodic-depthimage-to-laserscan
> sudo apt-get install ros-melodic-amcl ros-melodic-map-server ros-melodic-move-base

Preparing the environment:

> sudo apt-get install ros-melodic-urdf ros-melodic-xacro ros-melodic-compressed-image-transport
> sudo apt-get install ros-melodic-rqt-image-view ros-melodic-gmapping ros-melodic-navigation
> sudo apt-get install ros-melodic-interactive-markers ros-melodic-gazebo*
> sudo apt-get install sudo apt-get install ros-melodic-turtlebot3-teleop

- Download the following file and unpack the file at your home folder: https://bit.ly/2vhx1RN

Clone the Github repository:

```
> git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
> git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
> git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
```

Build the model:

```
> catkin_make
```

Preparing ROS and Gazebo

Launch Gazebo:

```
> roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Run Teleop for controlling:

```
> rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Open Rviz to manage ROS visually:

```
> roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```

Visual Slam:

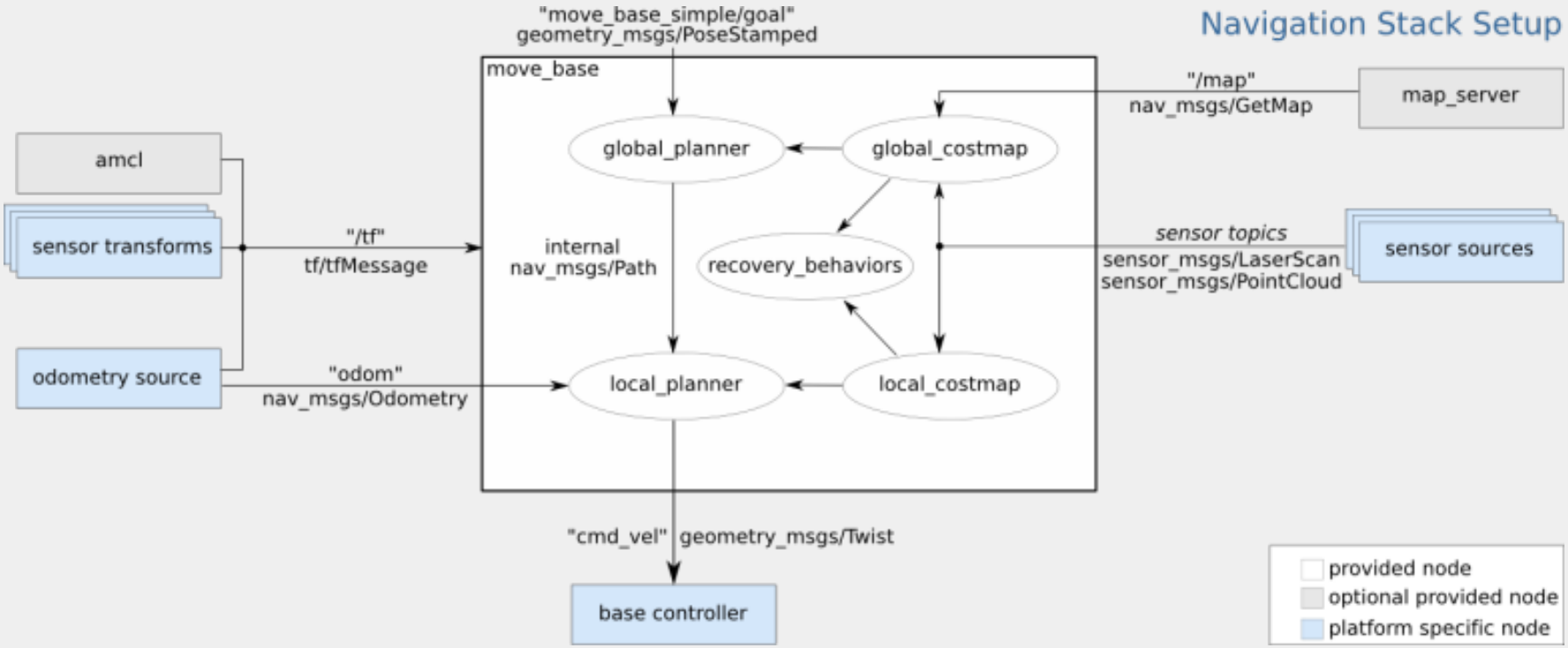
> roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping

Autonomous Navigation:

> roslaunch turtlebot3_navigation move_base.launch

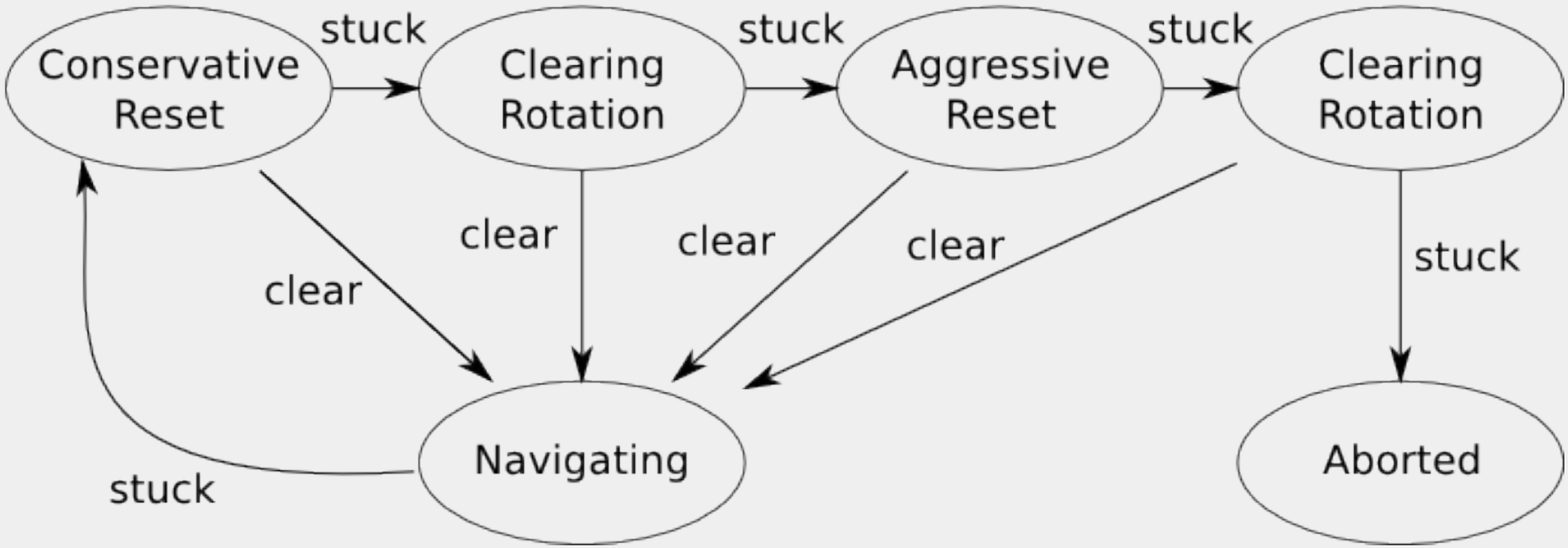
Next Week

The Move Base package implements a full navigation stack.



The recovery_behaviors node expanded:

move_base Default Recovery Behaviors



The Mapping Stack

Usually, one of the two following packages are used in the mapping stack.

AMCL - (Adaptative Monte-Carlo Localization)

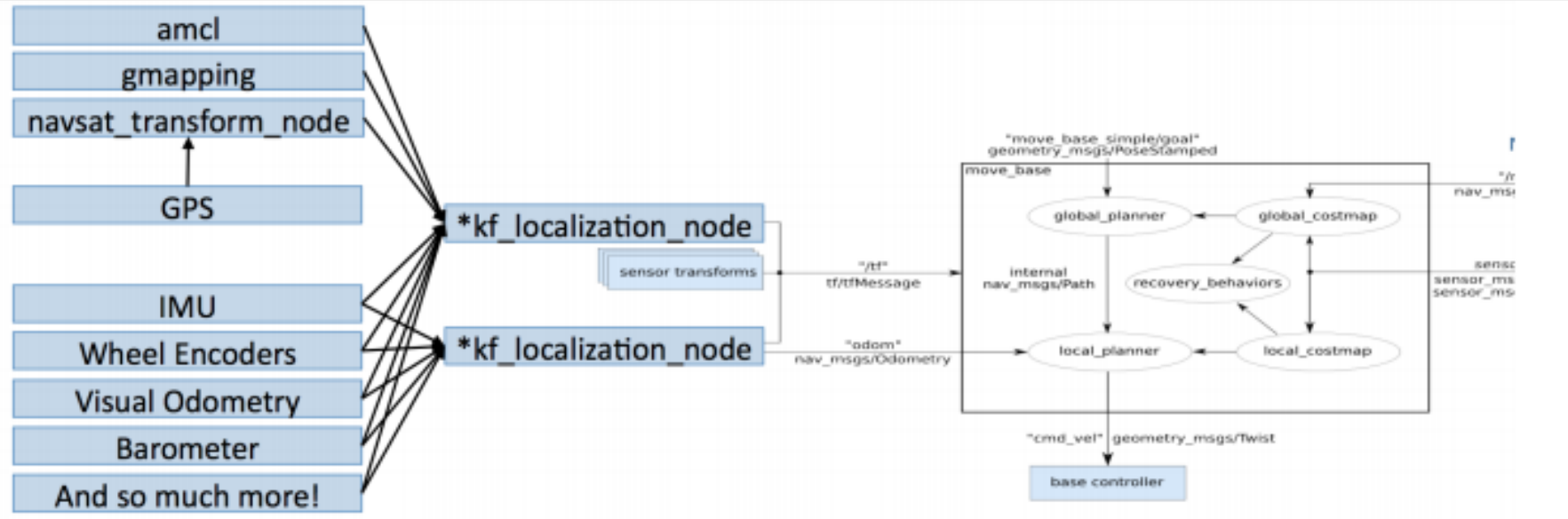
The amcl package is commonly used when a priori maps are known.

Gmapping - (SLAM)

The gmapping package implements a laser-based Simultaneous Localization and Mapping system and is commonly used in unknown or heavily dynamics environments.

The Localization Stack

Finally, the localization aims to fuse sensor readings and deliver a filtered odometry reading. Usually, the robot_localization package is used to provide a simple kalman filter (kf) or an extended kalman filter (ekf).



Deep Learning for Autonomous Vehicles

Supervised learning
Machine learning is a subfield of artificial intelligence.

Intuitively	We want to *learn from* and *make predictions on* data.
Technically	We want to build a model that approximate well (*e.g.* minimize a loss function) an unknown function.

It is important to note that the function we want to approximate may or may not have a closed form.

Application examples

Supervised learning

- Regression

  Polynomial $(x, y, z) \to f(x, y, z)$

  House price (surface, nb rooms, city) $\to$ price

- Classification

  Image classification pixel values $\to$ cat or dog

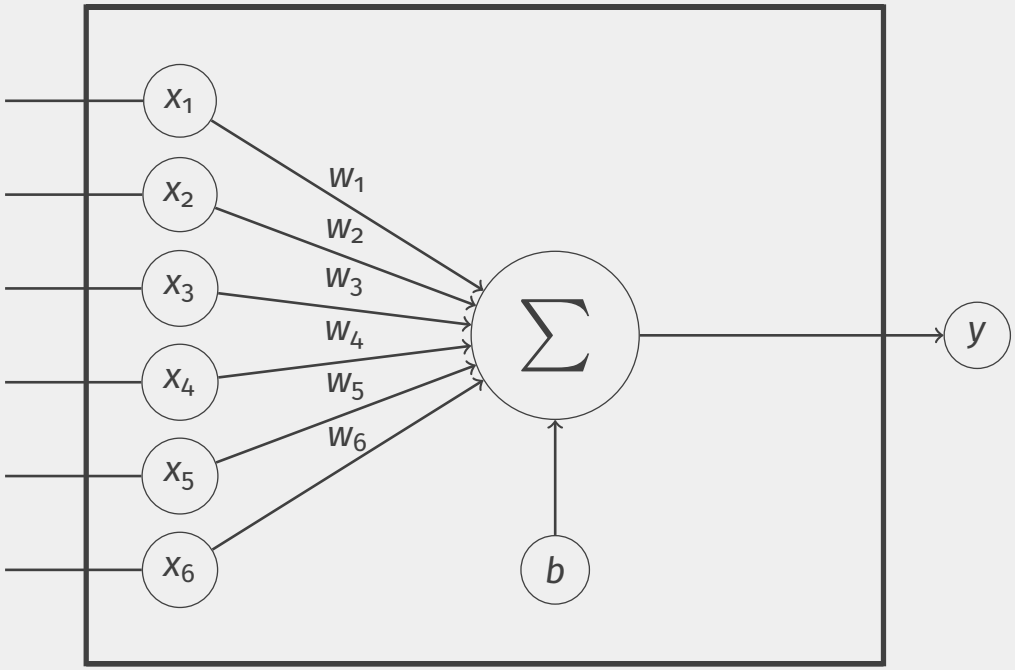
  Text classification list of words $\to$ spam or valid email

Deep learning

Deep learning is a subfield of machine learning in which we use artificial neural networks to make predictions.

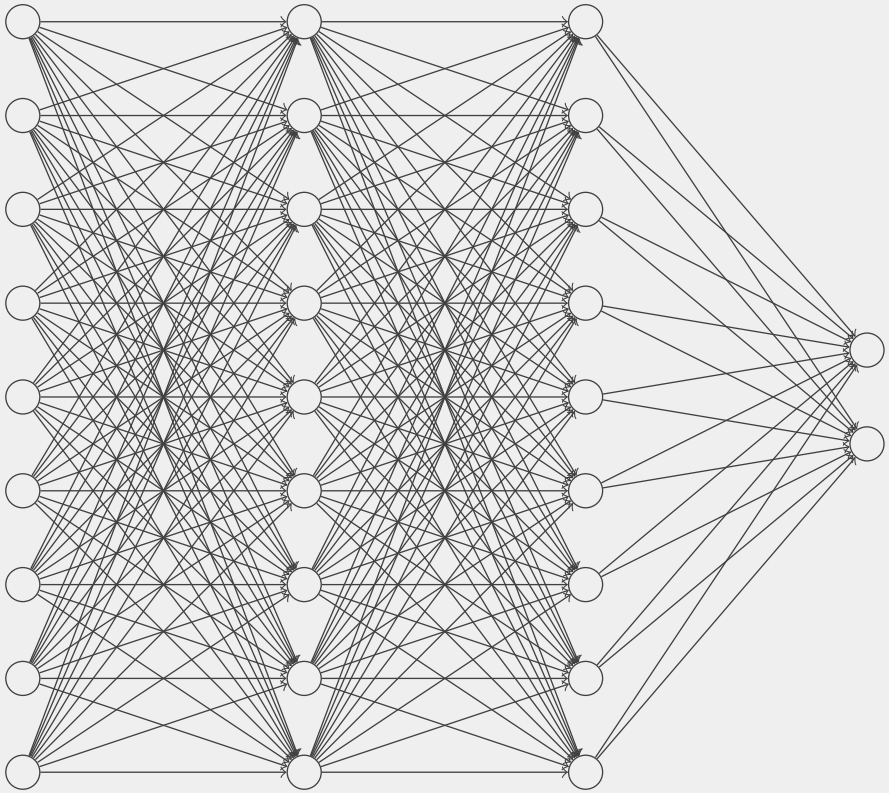
An artificial neural networks is a computation model loosely based on the human brain. It aims to mimic electric signals travelling through neurons in order to make computations.

Neuron

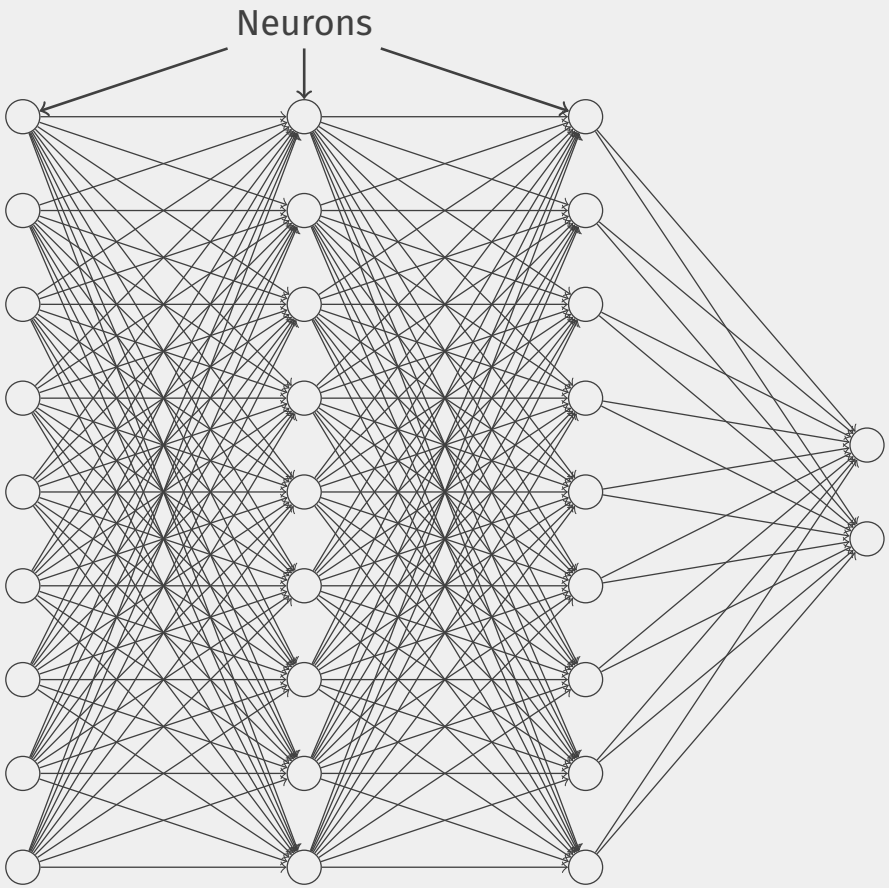


$$y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + b$$

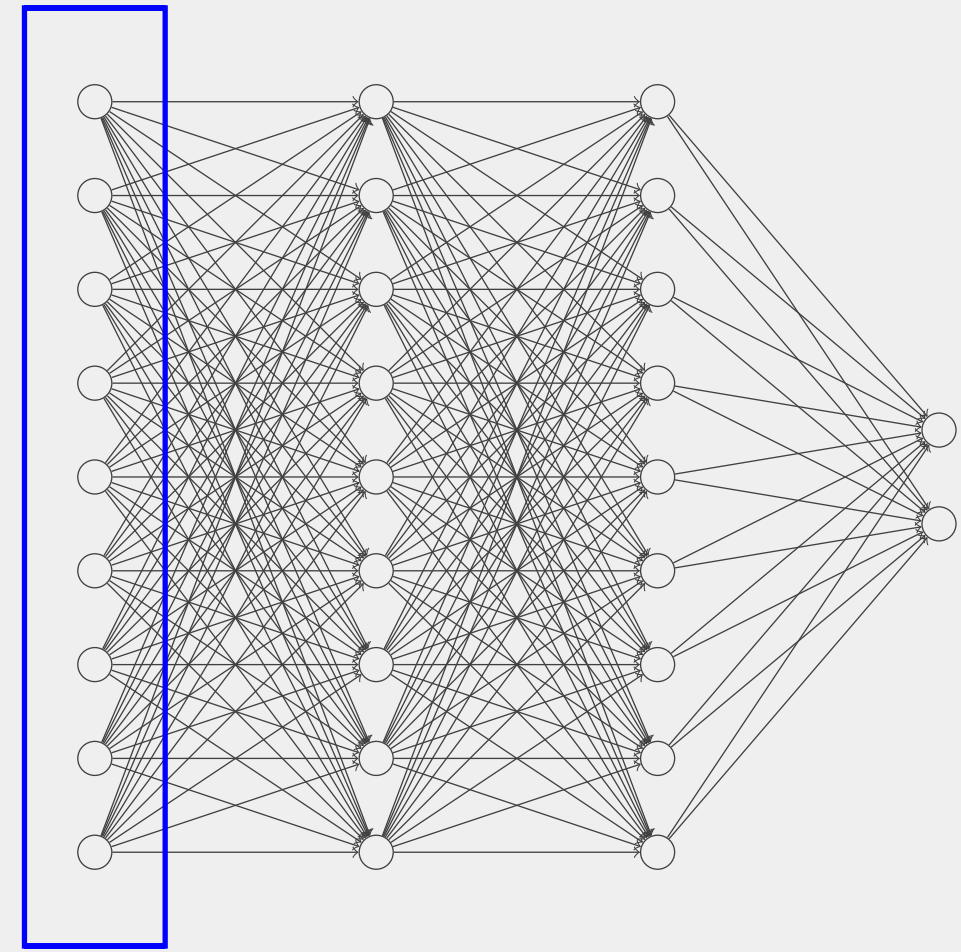
Network



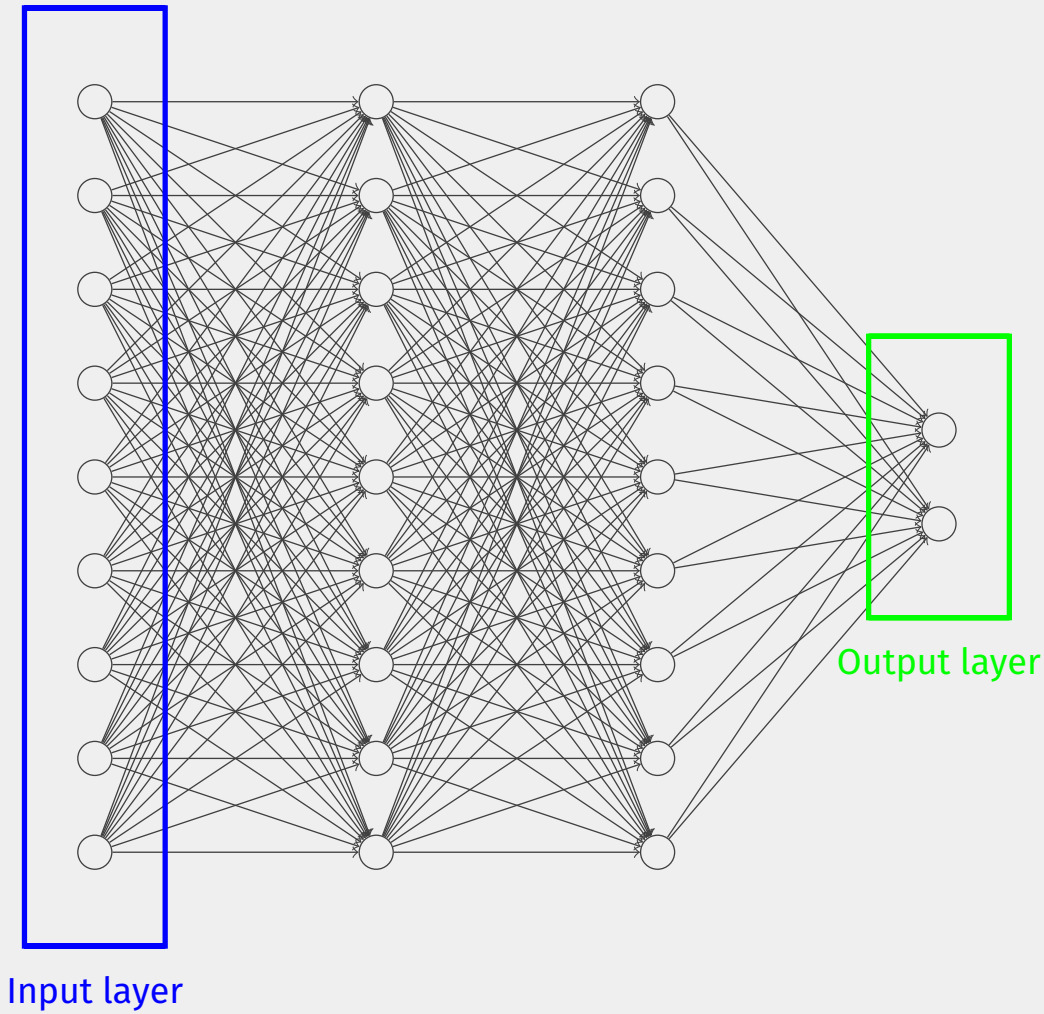
Network



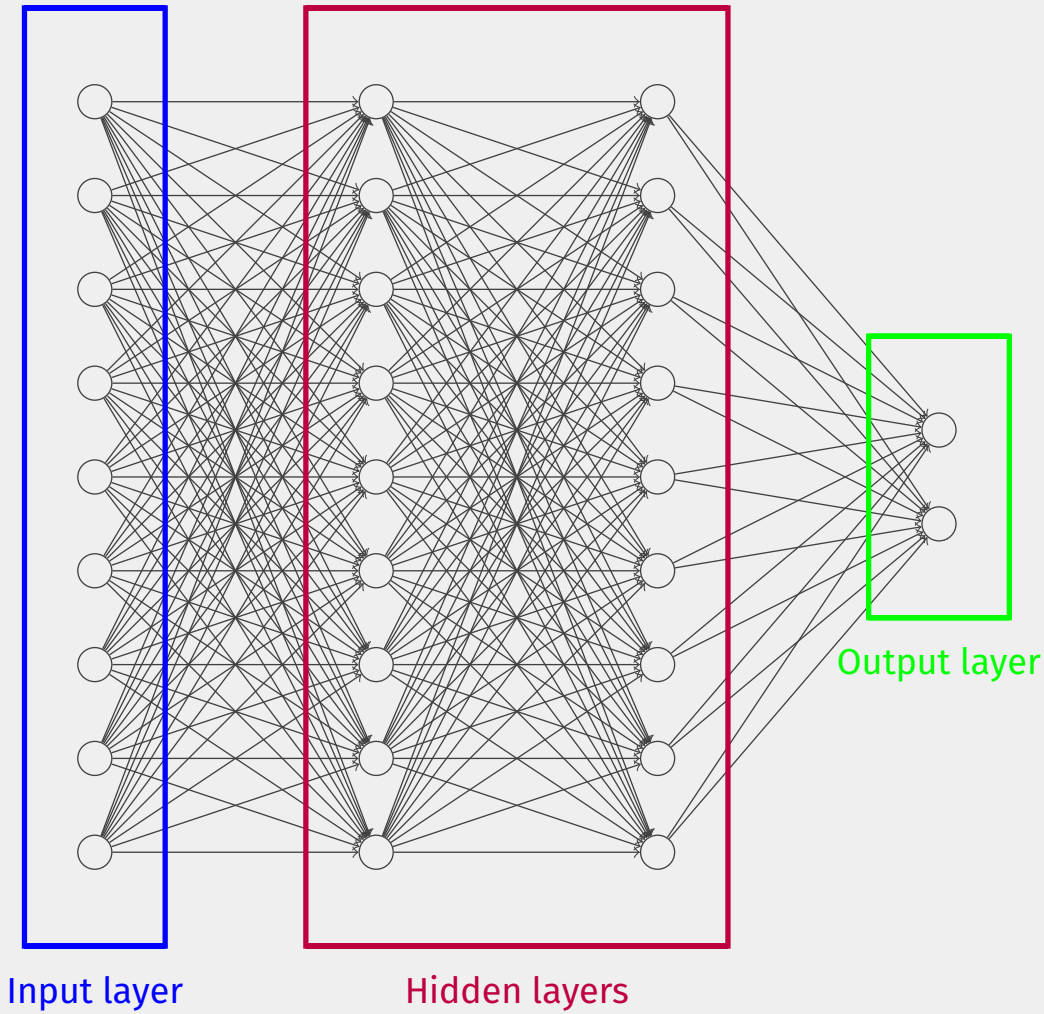
Network



Network



Network



"Problem" with this definition

We now have a quite complicated framework to compute linear functions.

Neuron	linear function
Neural network	linear combination of neuron outputs

To approximate non linear functions, we would like to have a non linear model.

"Problem" with this definition

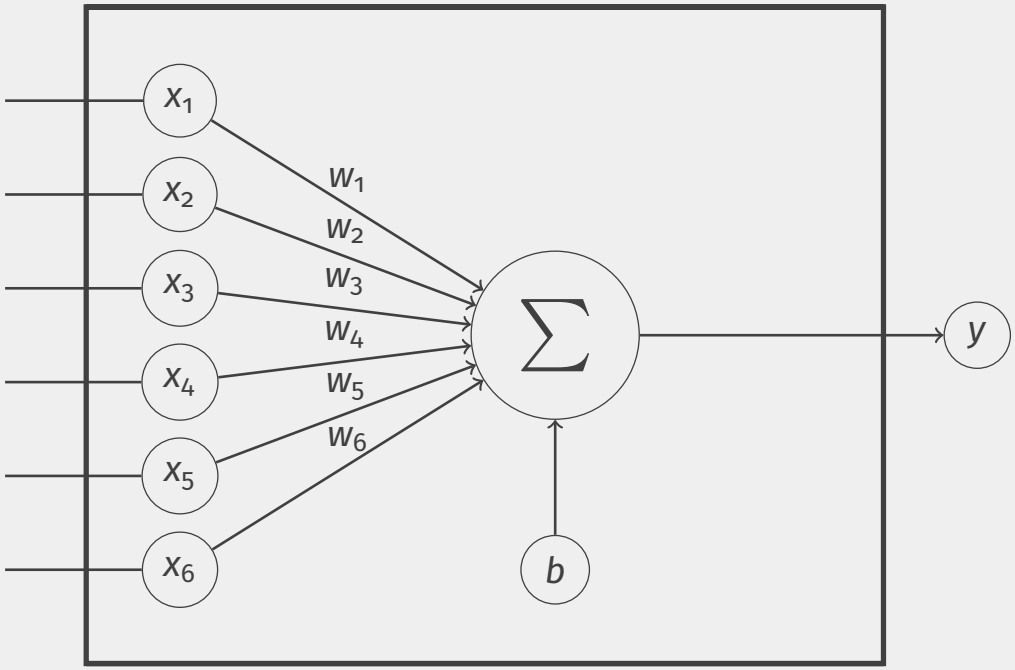
We now have a quite complicated framework to compute linear functions.

Neuron	linear function
Neural network	linear combination of neuron outputs

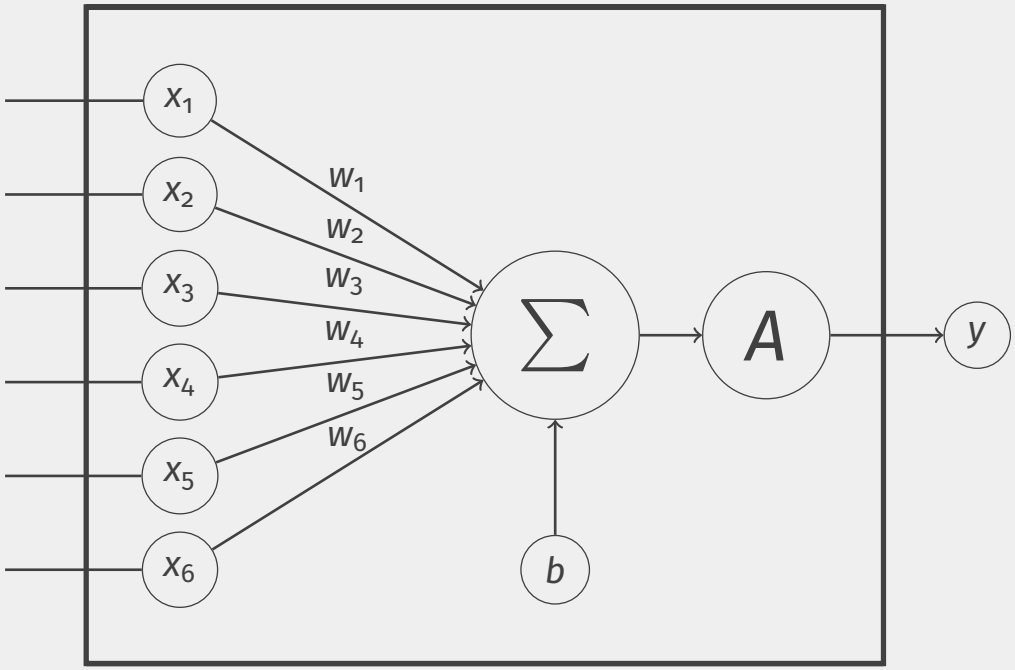
To approximate non linear functions, we would like to have a non linear model.

Solution: add nonlinearity to neurons.

Neuron with activation

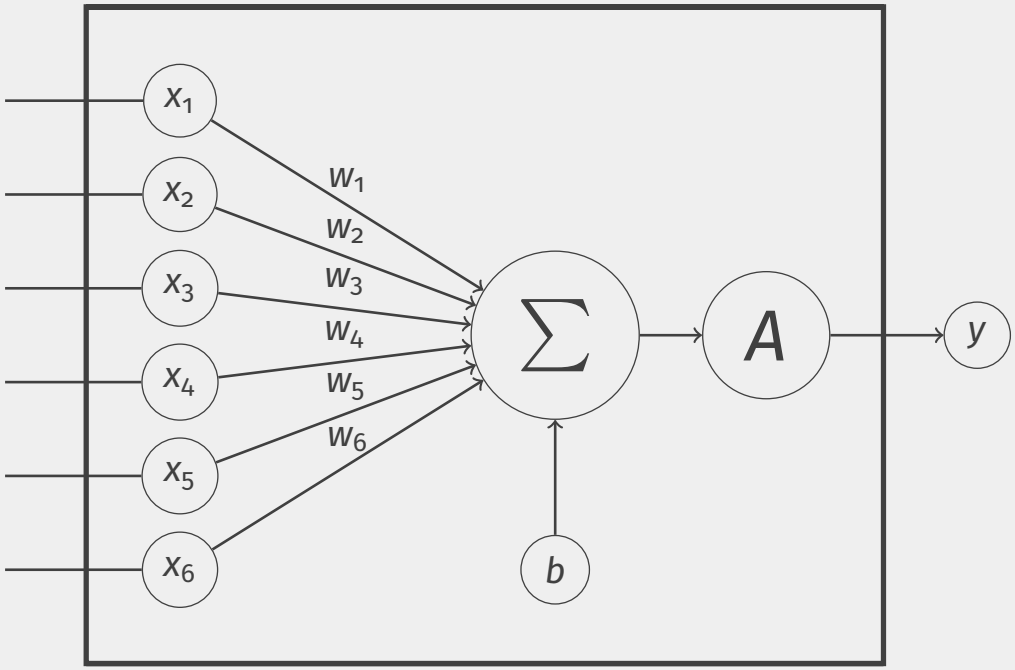


Neuron with activation



$$A(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

Neuron with activation



$$A(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

$$y = A(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + b)$$

Motivation

For now, our networks computes a nonlinear function of the inputs. If we work with images, it has to learn the *spacial structure* of the data by itself, which takes a long time.

A nice way to help it is to build convolution layers into the network.

Kernel

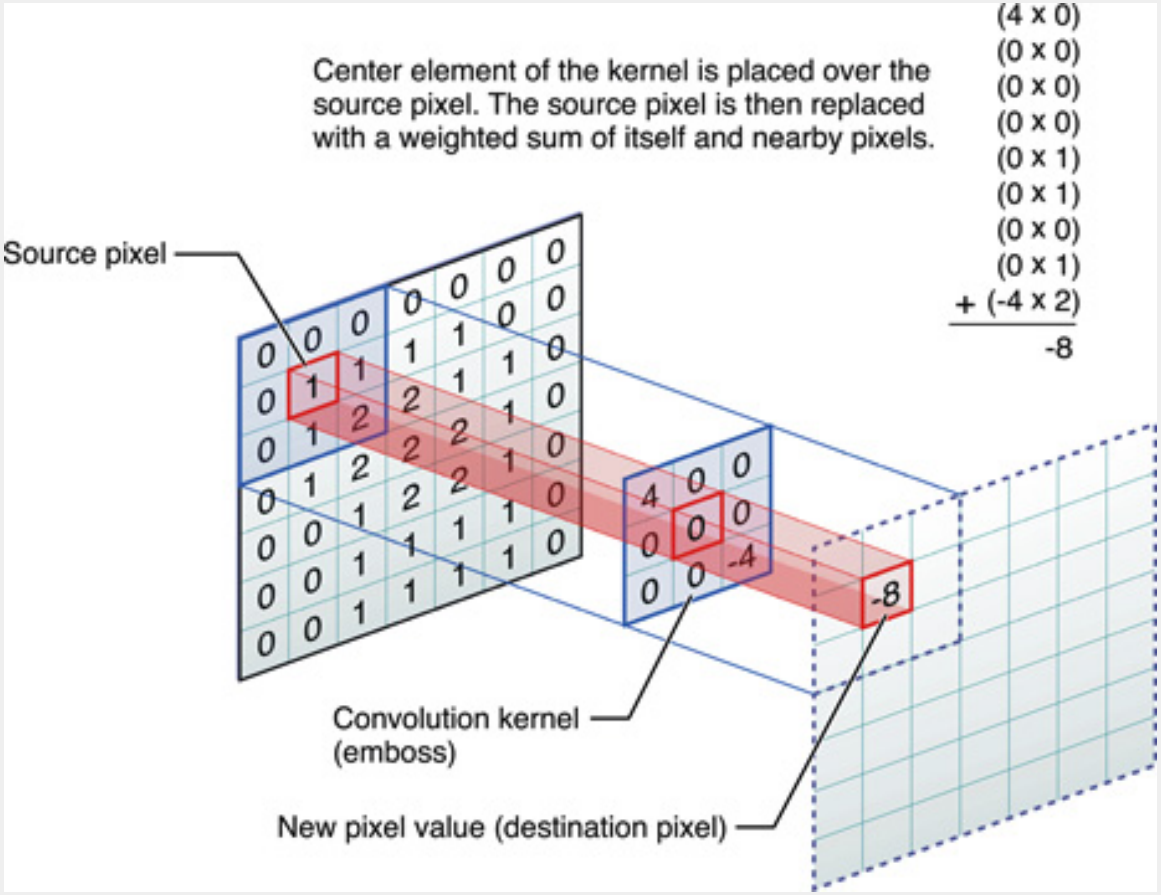
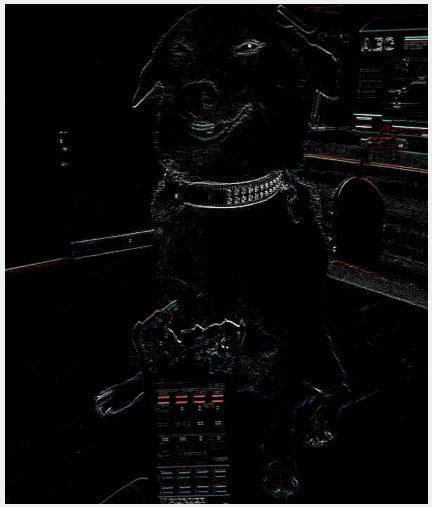


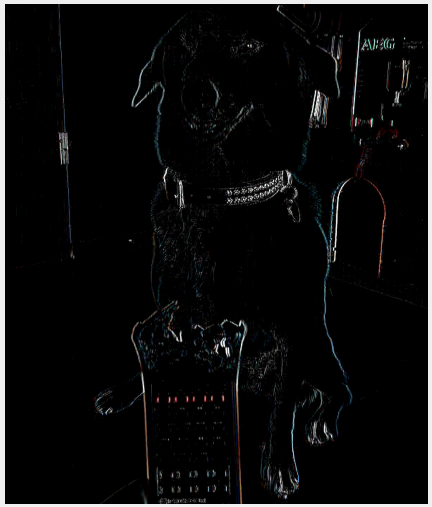
Image from `http://stats.stackexchange.com/`

Convolution

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$



$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$



VGG network (2014)

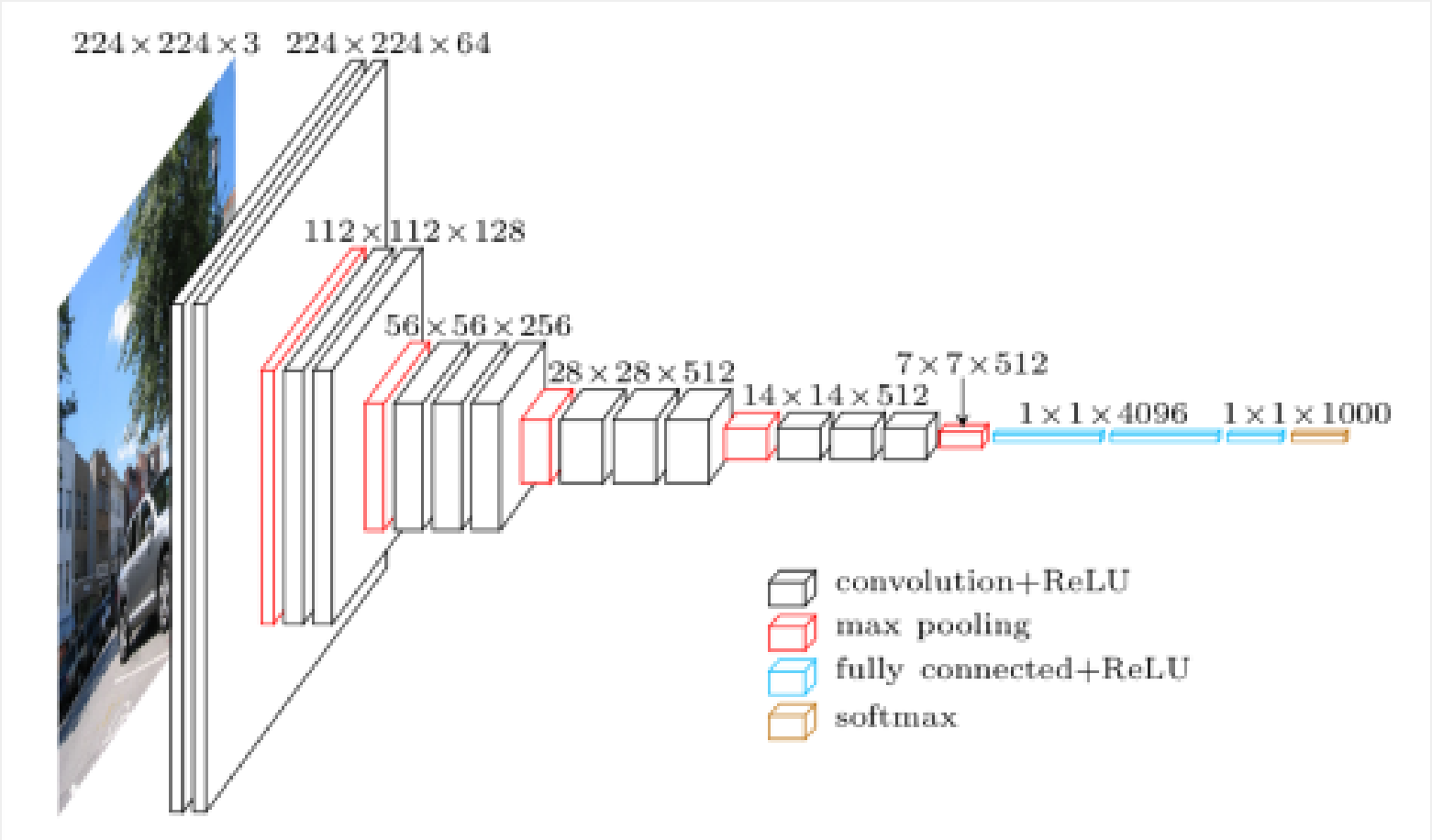


Image from `https://www.cs.toronto.edu/~frossard/post/vgg16/`

What do the filters recognize?

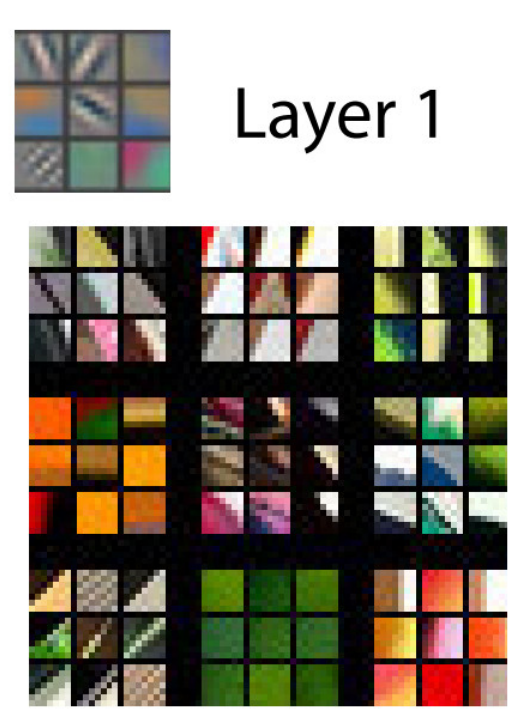


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

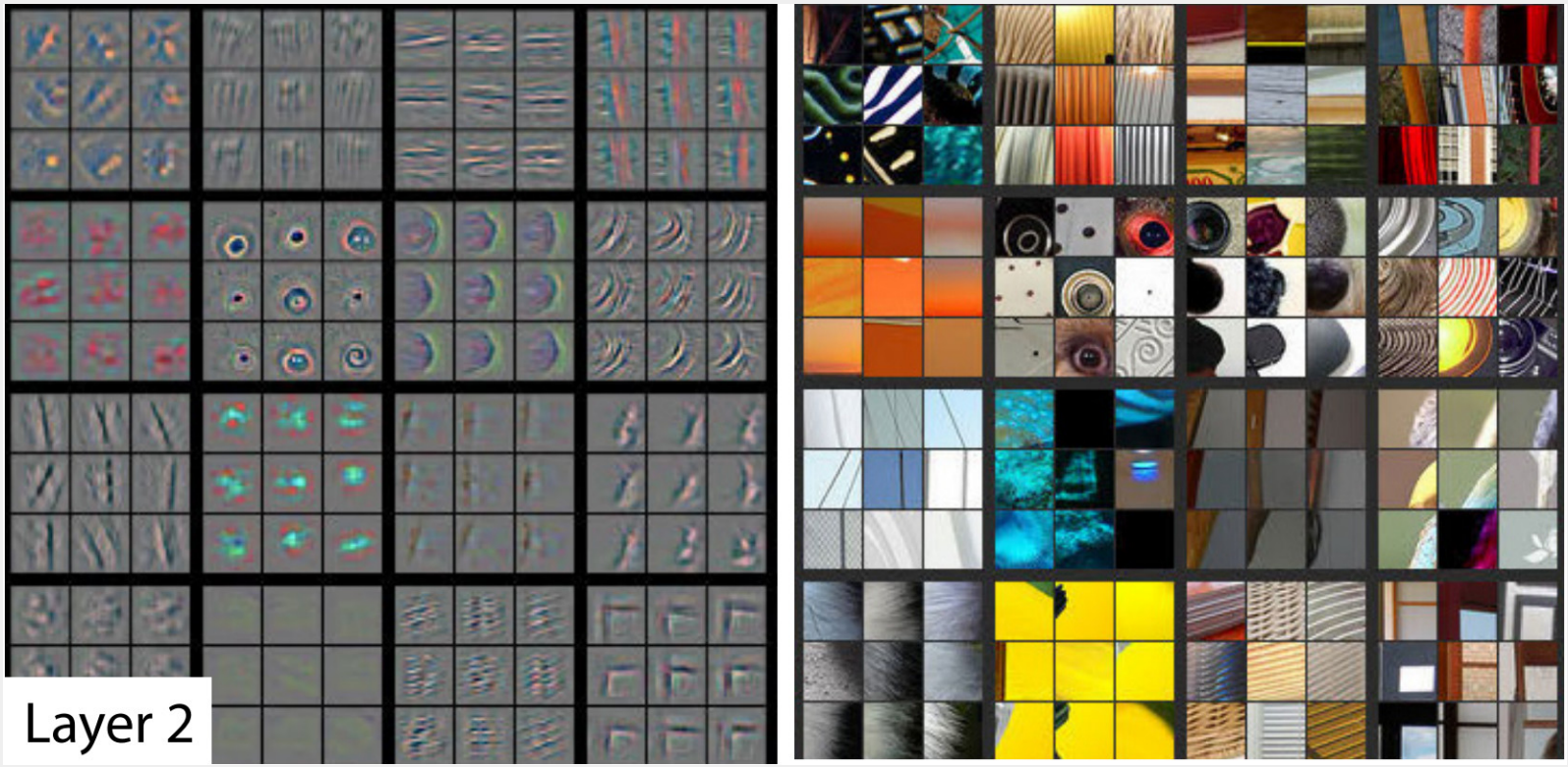


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

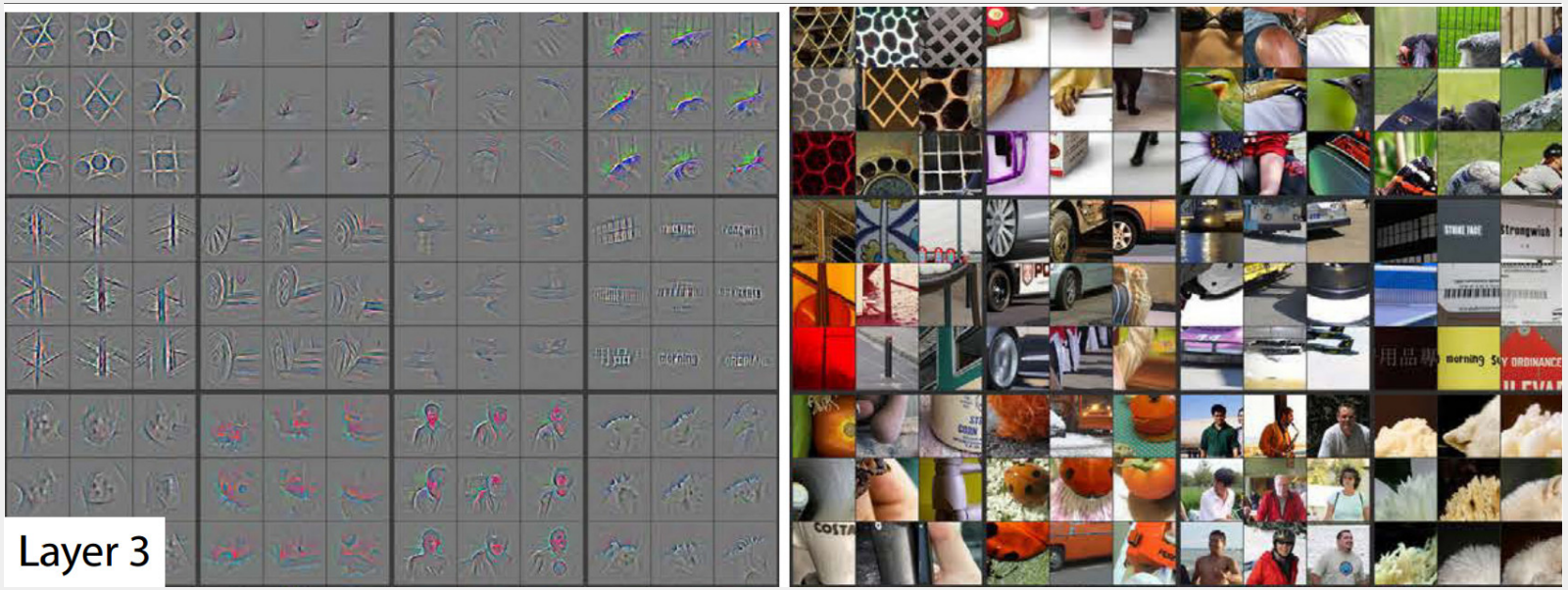


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

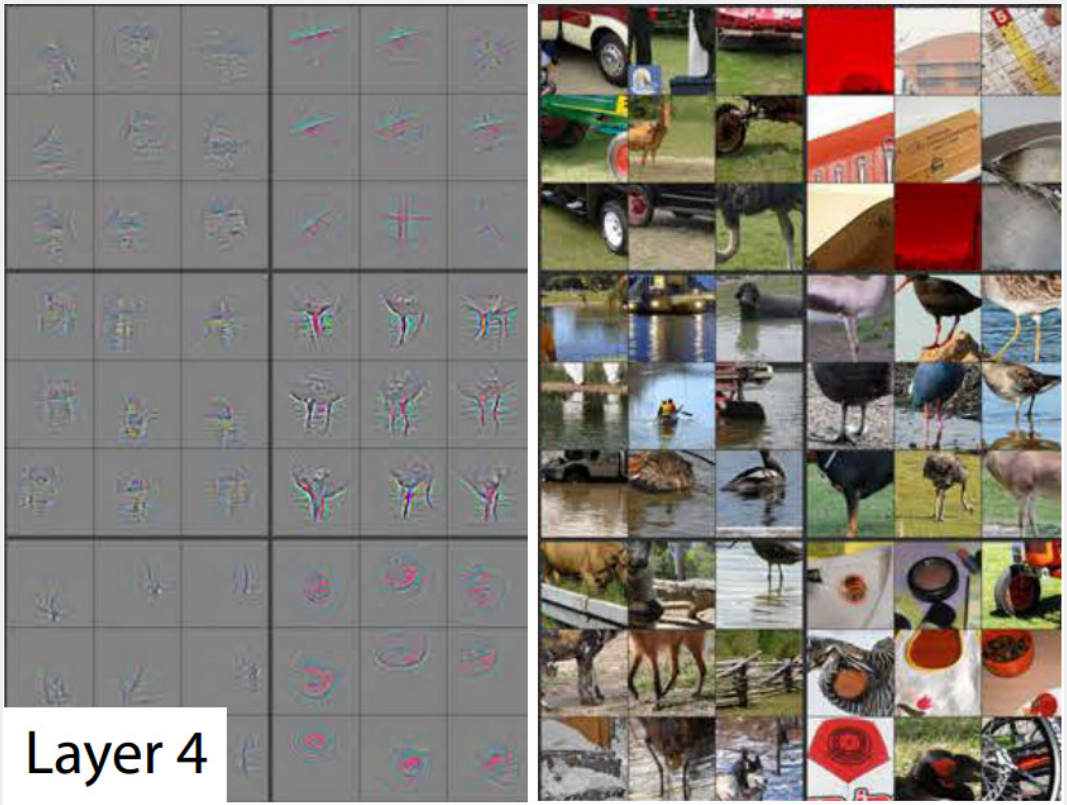


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

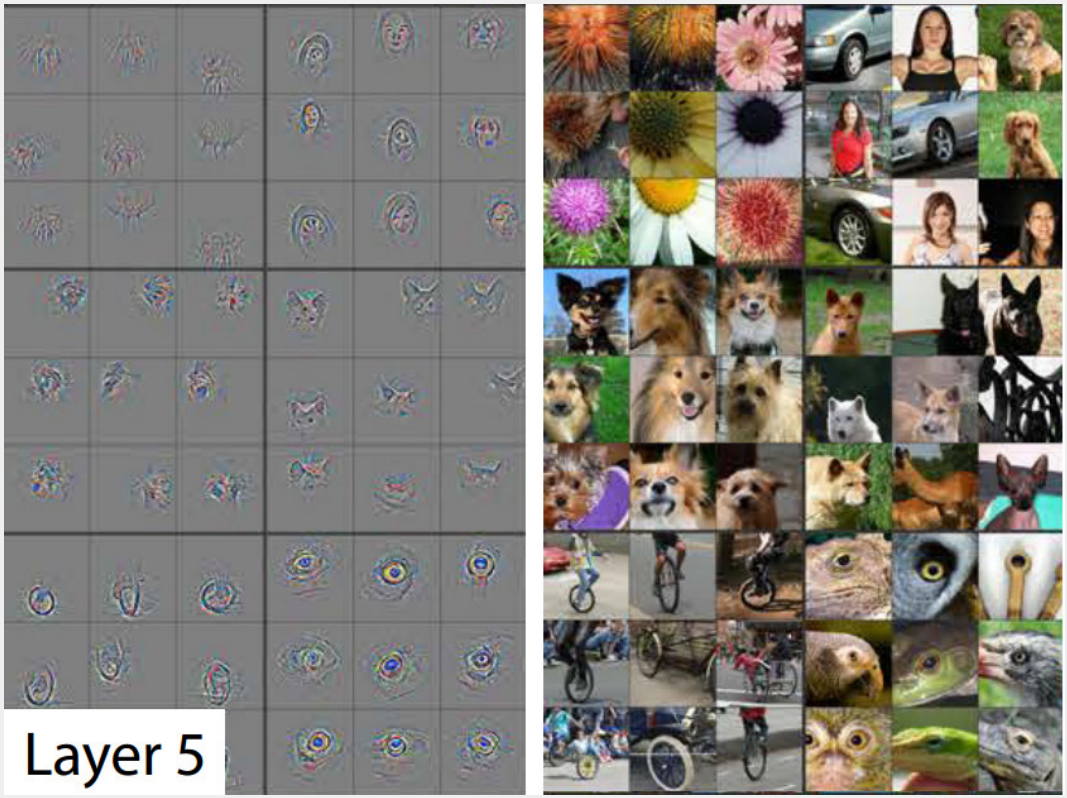


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

VGG network (2014)

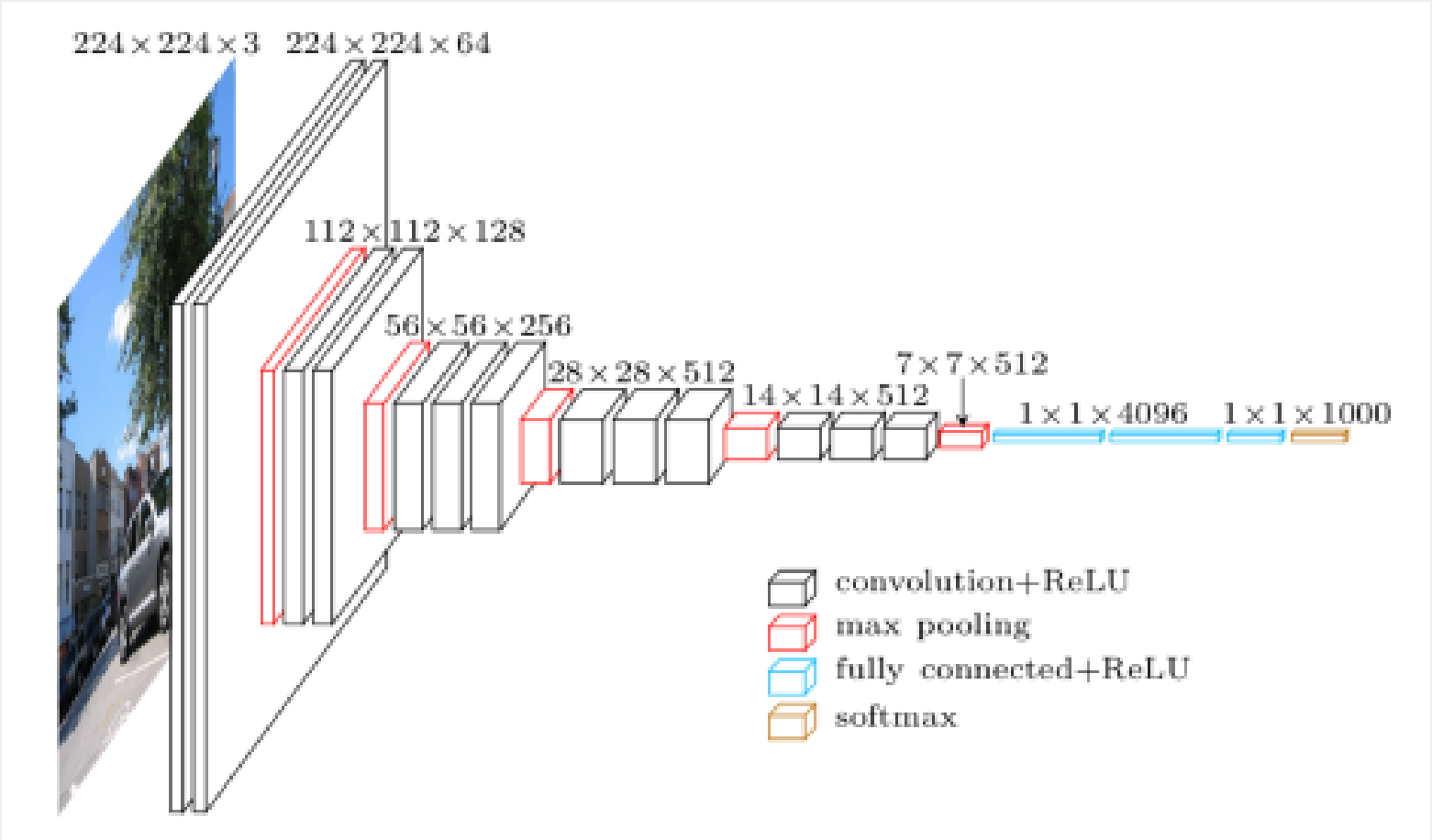
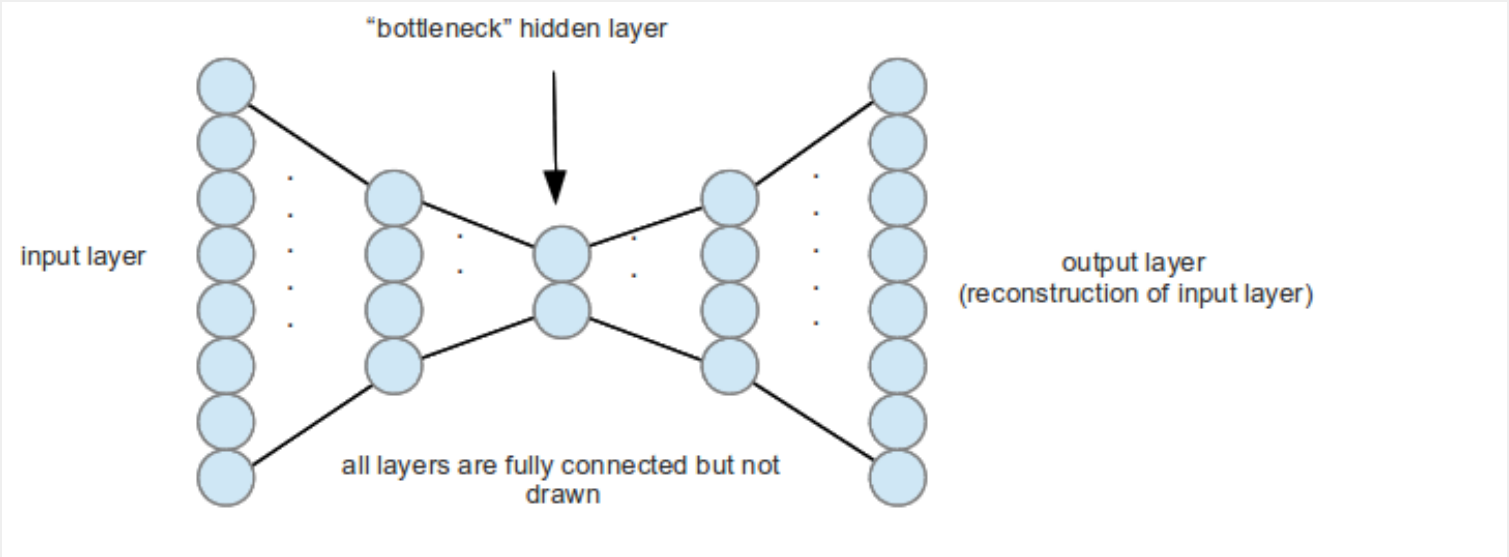


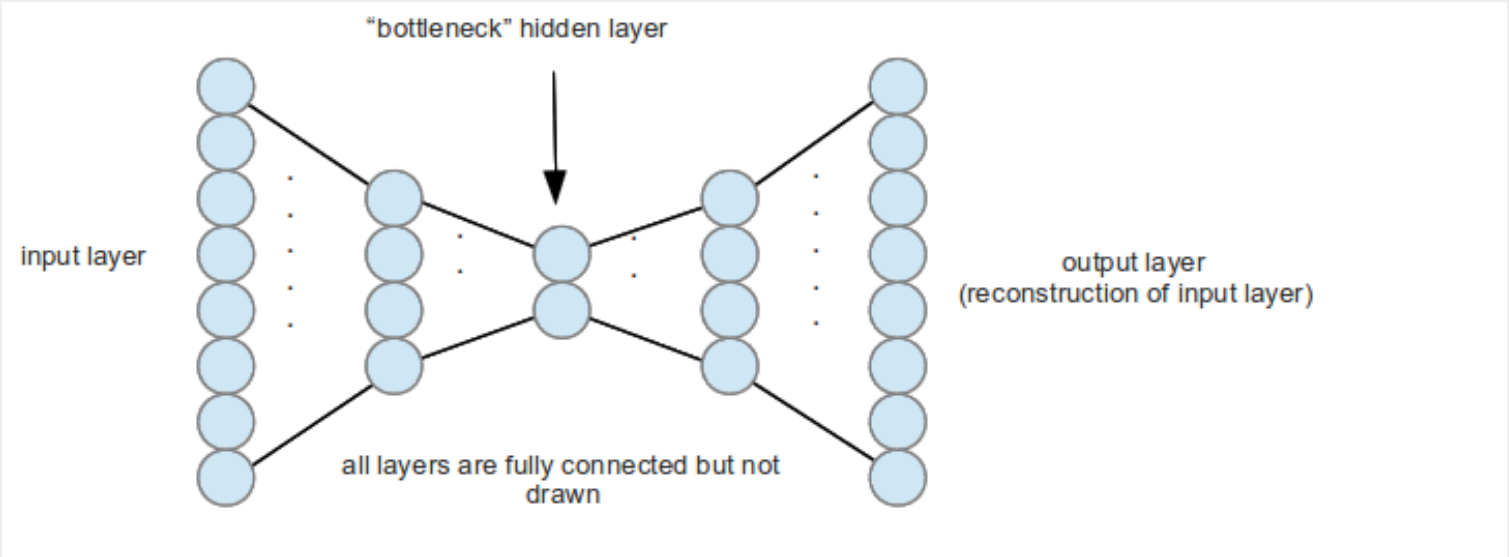
Image from `https://www.cs.toronto.edu/~frossard/post/vgg16/`

Autoencoder: data encoding



Hinton, Salakhutdinov (2006)

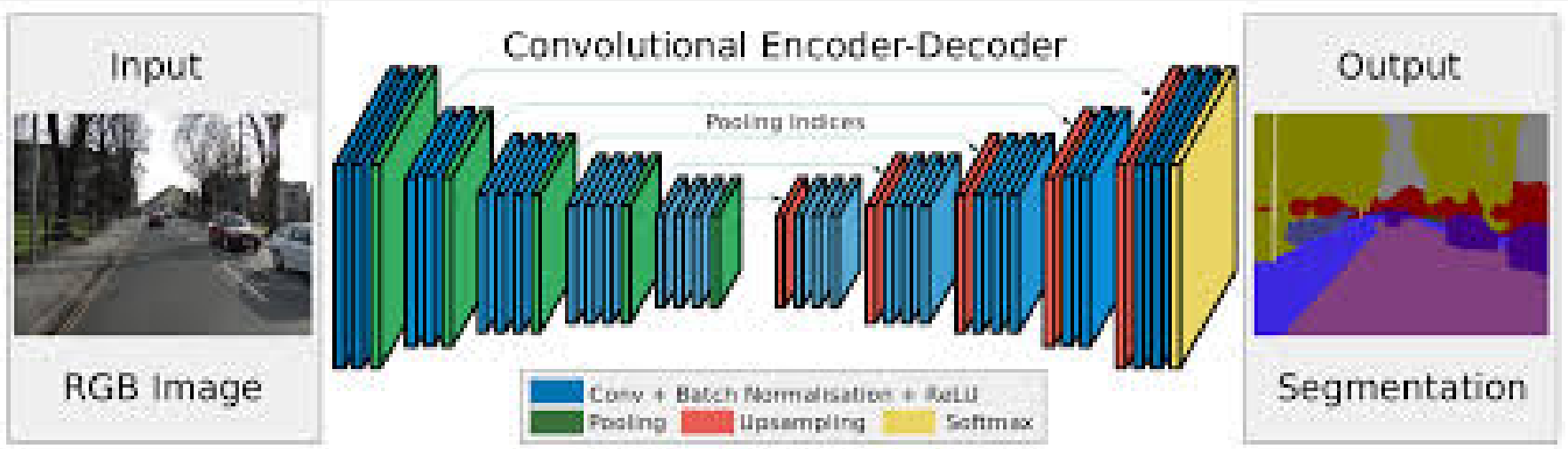
Autoencoder: data encoding



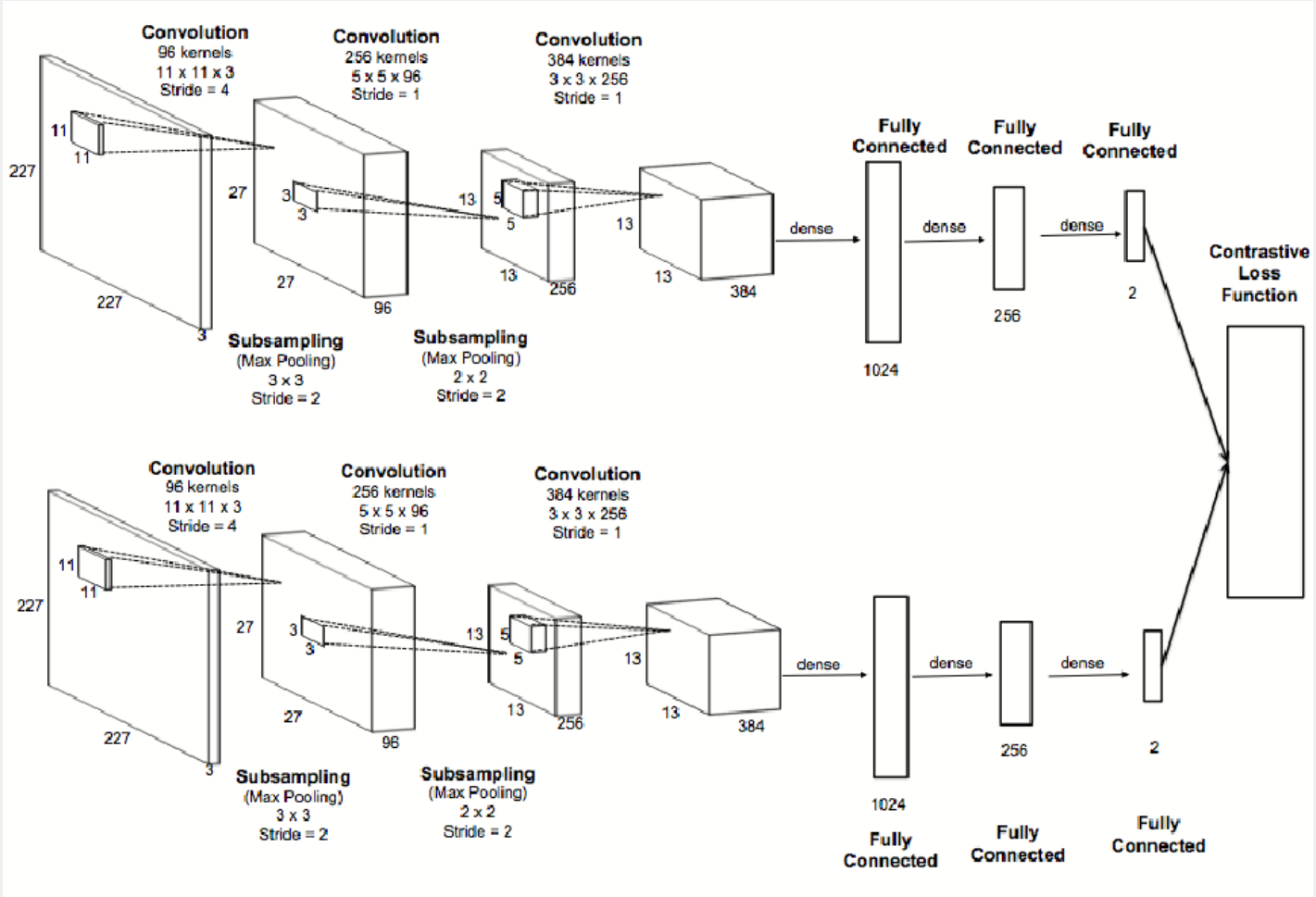
Hinton, Salakhutdinov (2006)

If we cut this autoencoder at the bottleneck, we get two parts: an encoder and a decoder. The encoder is an encoder highly specific to the content the network has been trained with.

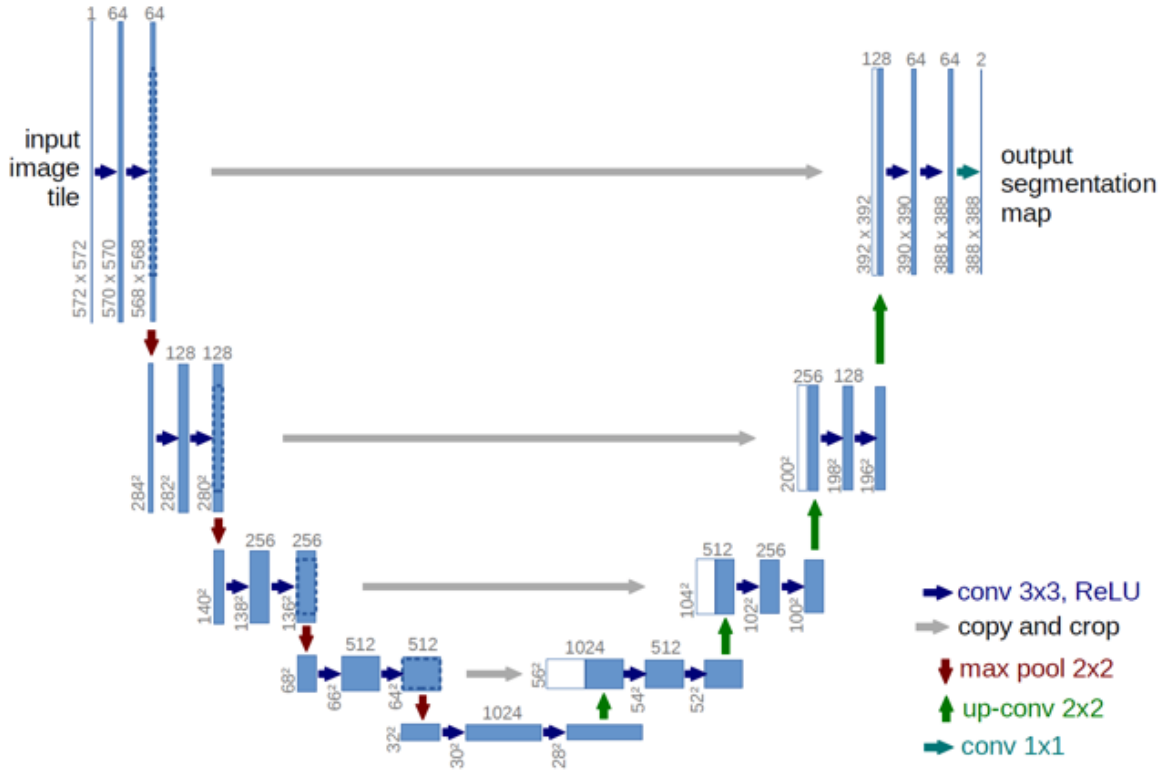
CNN + Autoencoder network: Semantic Segmentation



Siamese Networks



Unet: Semantic Segmentation



Recurrent neural network

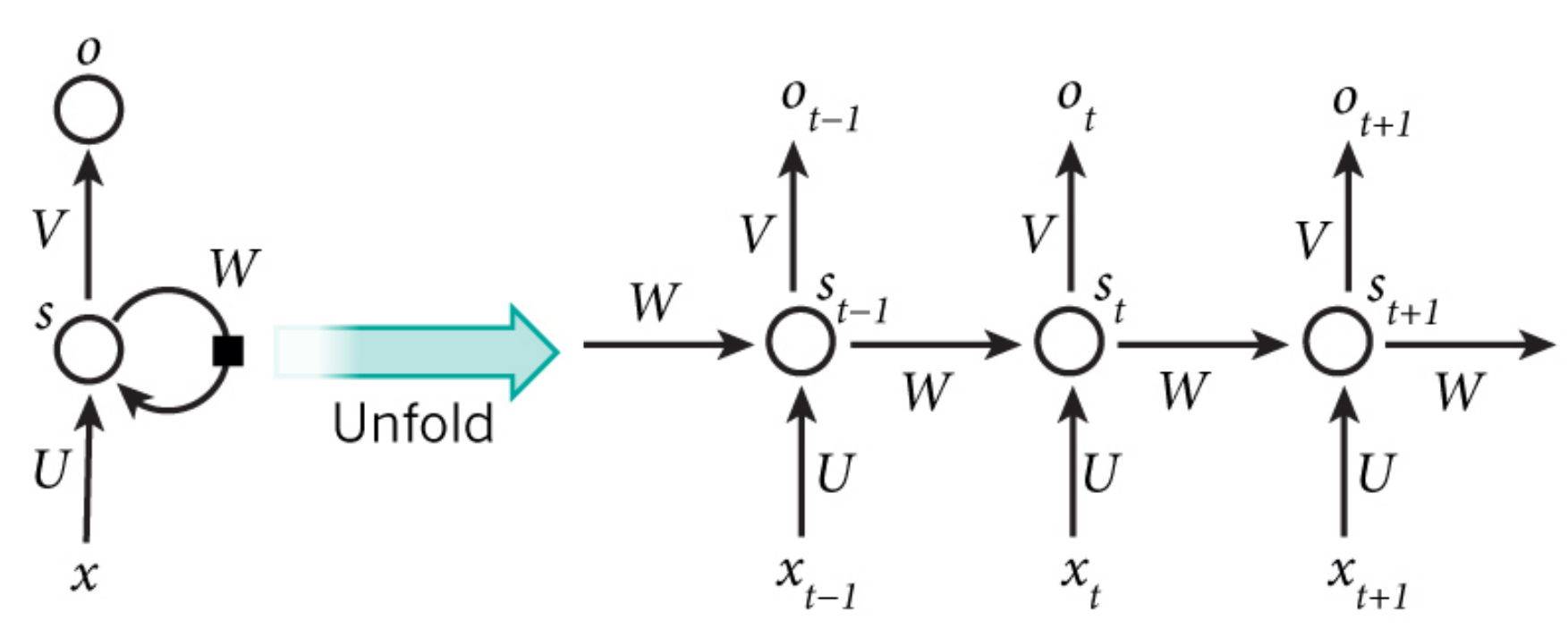


Image from `http://www.wildml.com`

Sequence to class network: text classifier

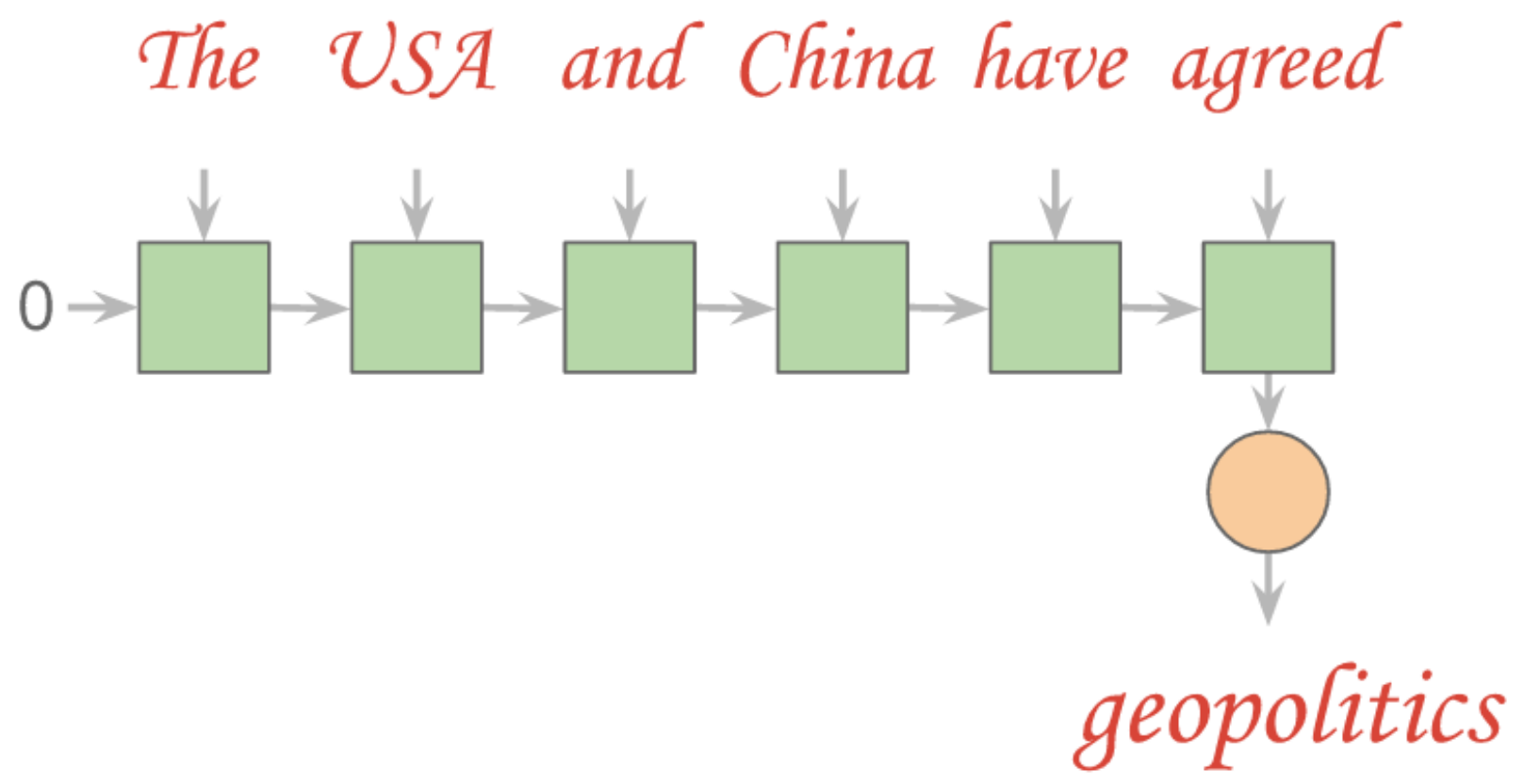


Image from Martin Gorner

Sequence to sequence network: Neural Machine Translation

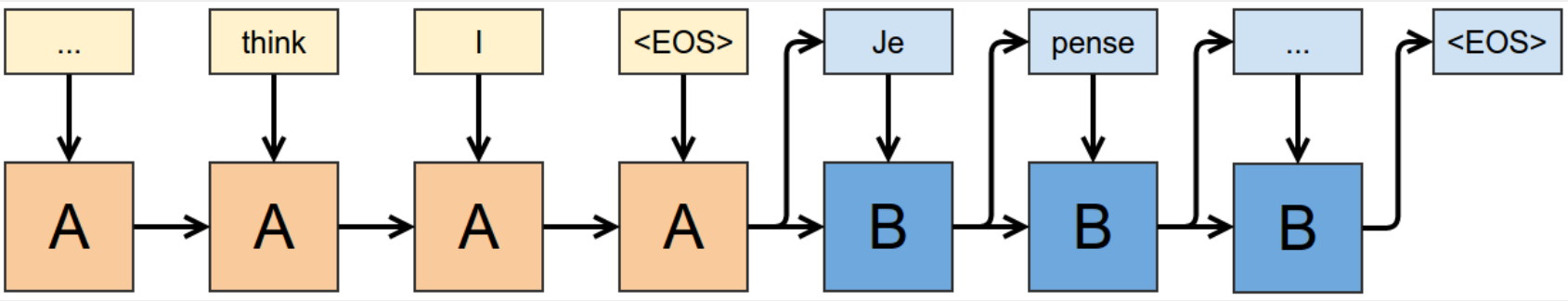


Image from `https://colah.github.io`

Google, September 2016: “The Google Translate mobile and web apps are now using GNMT (Google NMT) for 100% of machine translations from Chinese to English—about 18 million translations per day.”

Sequence to sequence network: Neural Conversation Model

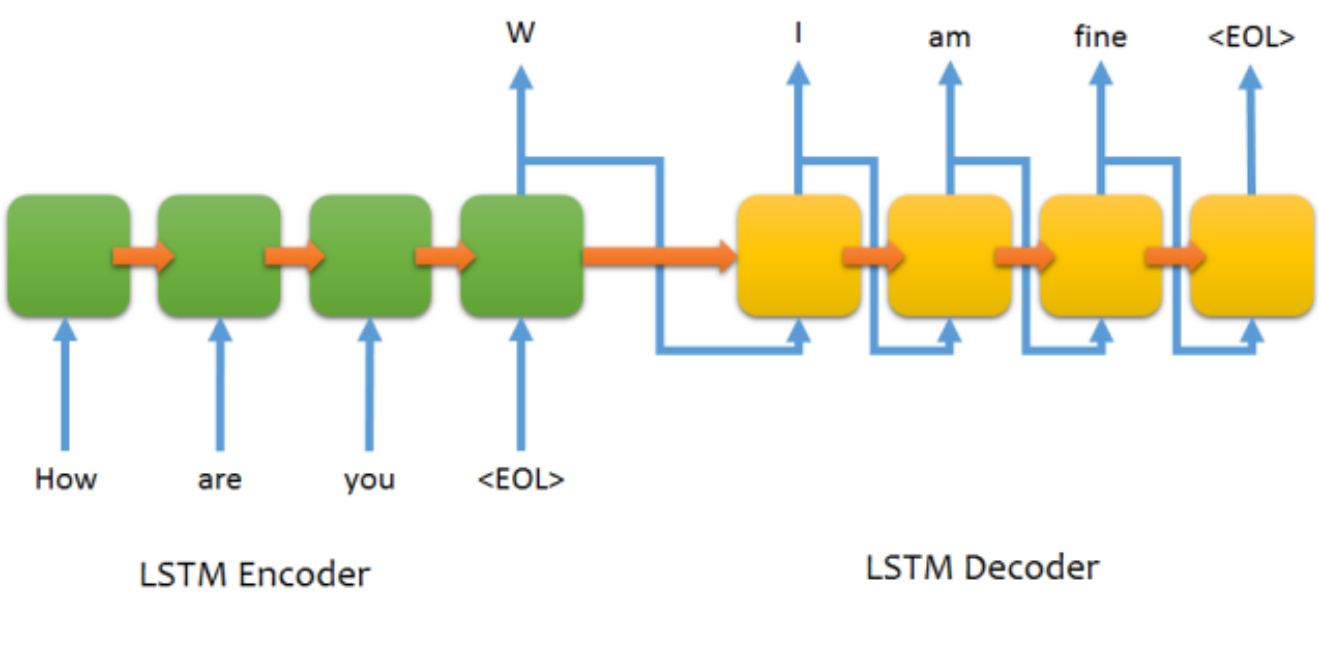


Image from `https://github.com/farizrahman4u/seq2seq`

Really early stage, hard to overcome challenges (context, coherent personality, ...)

Image to sequence: automatic captioning

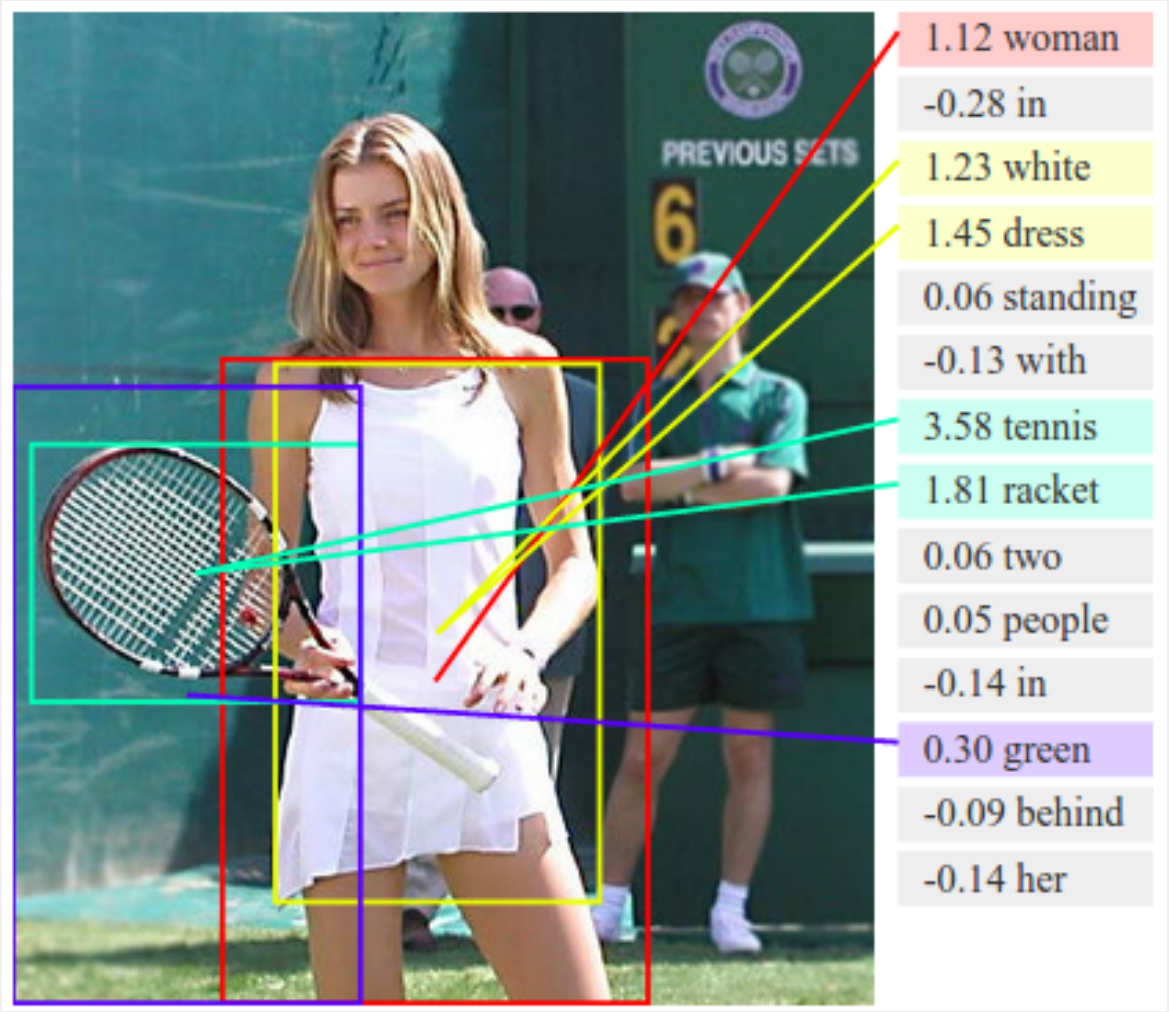


Image from `https://quantumfrontiers.com`

Image to sequence: automatic captioning

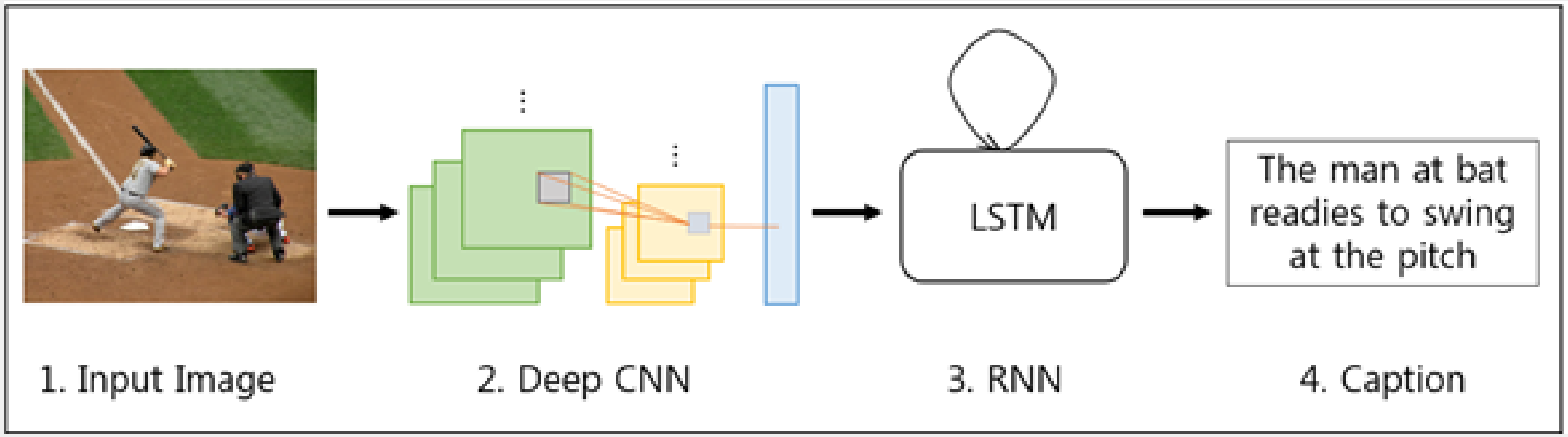


Image from `http://brain.kaist.ac.kr/`

Generative adversarial network

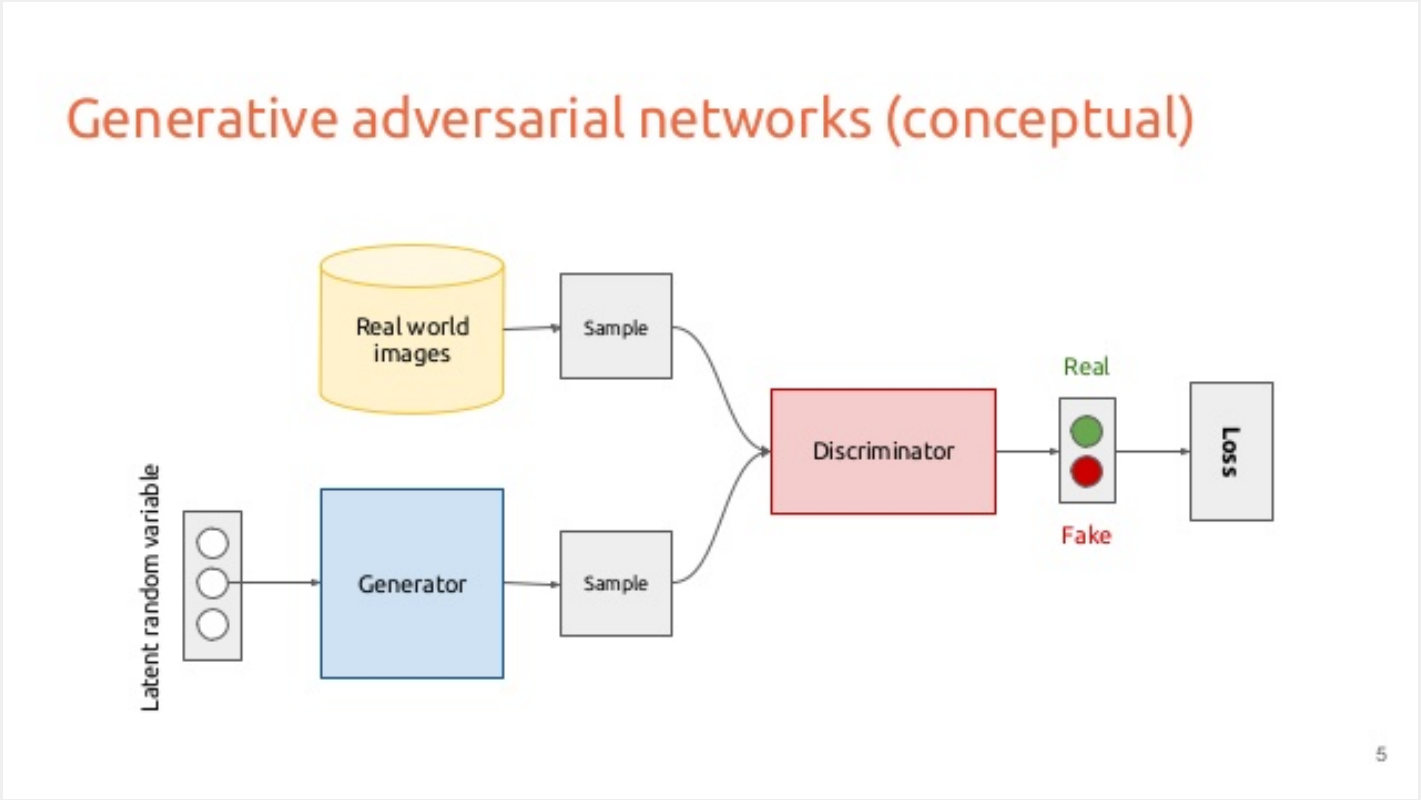


Image from `http://wiki.tum.de/`

Generative adversarial network: text to image
Han Zhang et al. (2016)

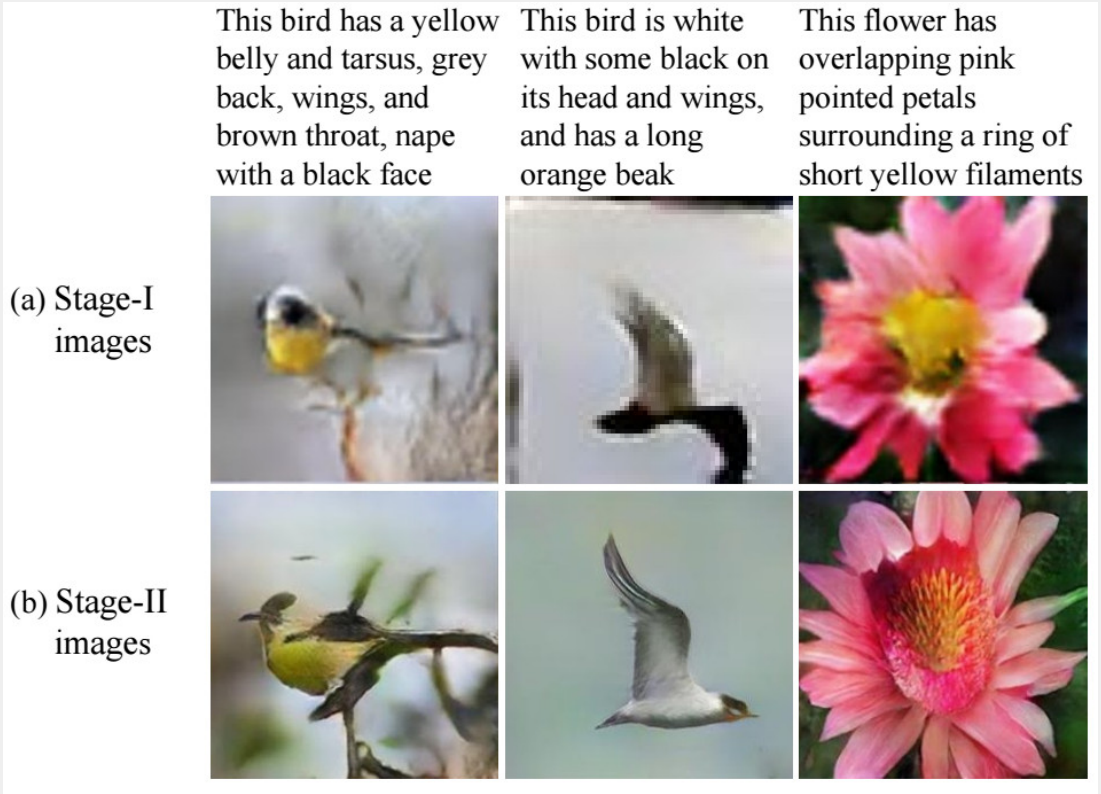
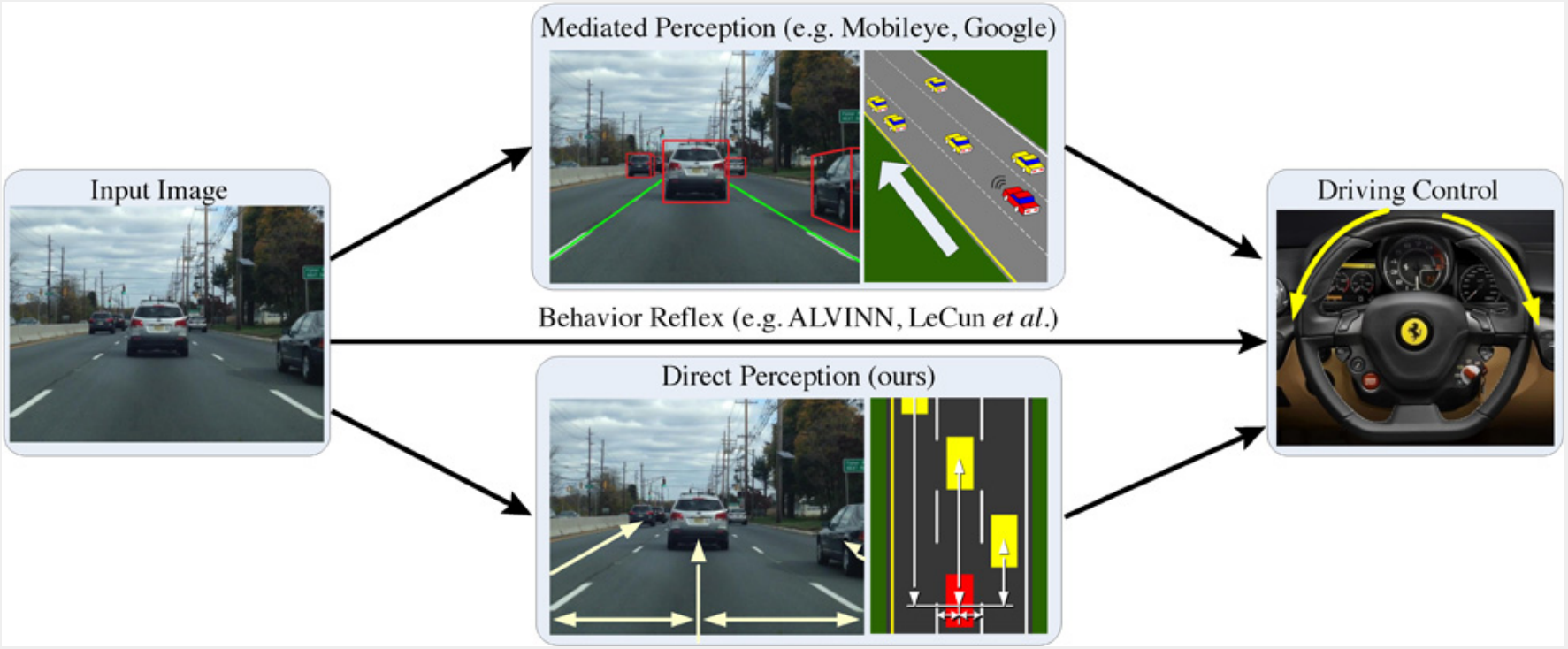


Image from `https://arxiv.org/pdf/1612.03242.pdf`



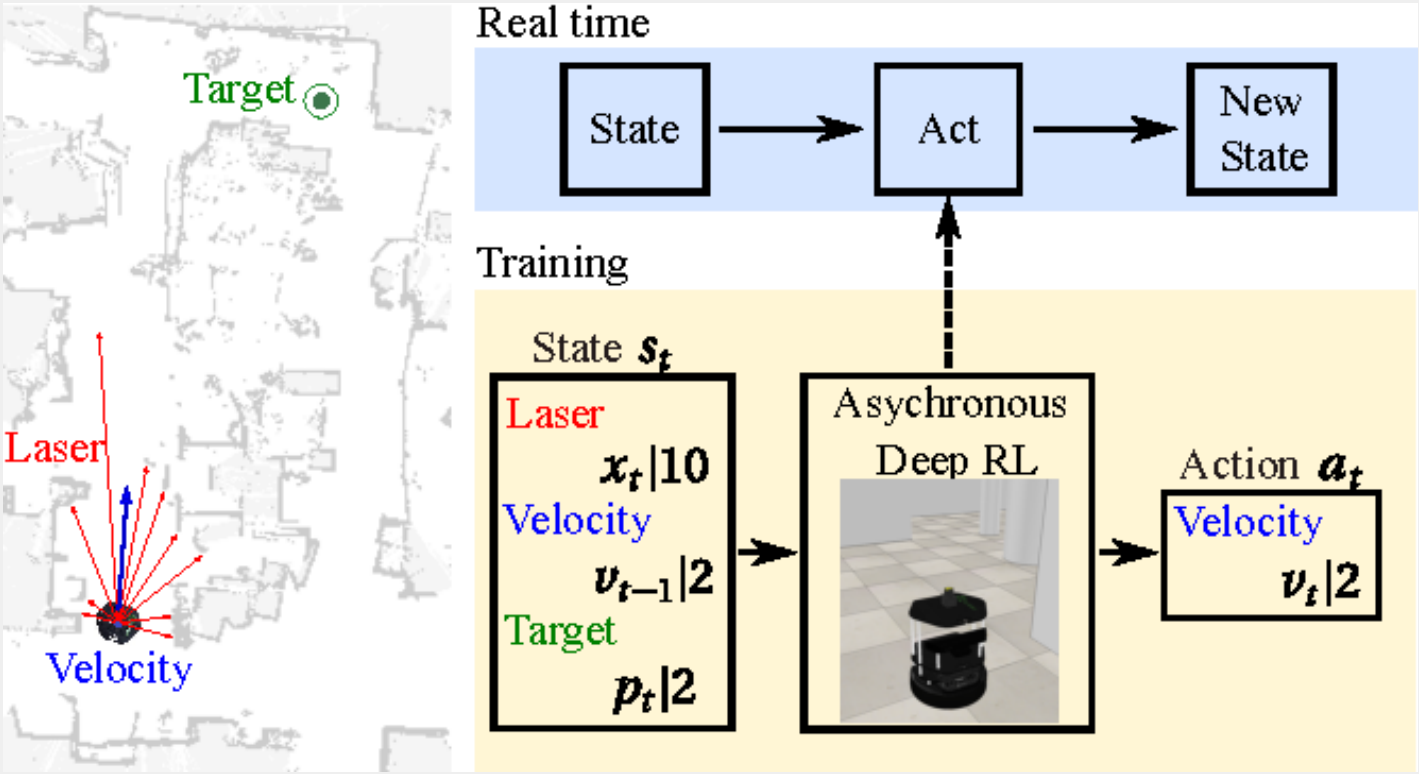
https://arxiv.org/pdf/1801.06734.pdf

http://deepdriving.cs.princeton.edu/

RNNs for Steering through Time

https://www.youtube.com/watch?v=nFTQ7kHQWtc

Deep RL for Motion Planning



https://papers.nips.cc/paper/843-robust-reinforcement-learnin
pdf

Appendix

What is Ubuntu ?

Ubuntu

What is Ubuntu ?

- Ubuntu is an open source software operating system that runs from the desktop, to the cloud, to all your internet connected things.
- Complete desktop Linux operating system, freely available with both community and professional support.
- Currently in the 18.04 LTS (Long Term Support)
- If you’re considering to pursue a career in Robotics, you should definitely use Ubuntu.

Ubuntu

What is Ubuntu ?

- For using ROS and Gazebo we need to use Ubuntu.
- Great opportunity for developing new skills and for our future in I2A2
- **Tutorial:** https://hackernoon.com/installing-ubuntu-18-04-along-with-windows-10-dual-boot-installation-for-deep-learning-f4cd91b58555

Thank You!

www.linkedin.com/in/viniciusarasantos

vinicius.santos@datah.ai

+1 (905) 242-7069