Droid Equinox 09

Artificial Intelligence for Autonomous Vehicles

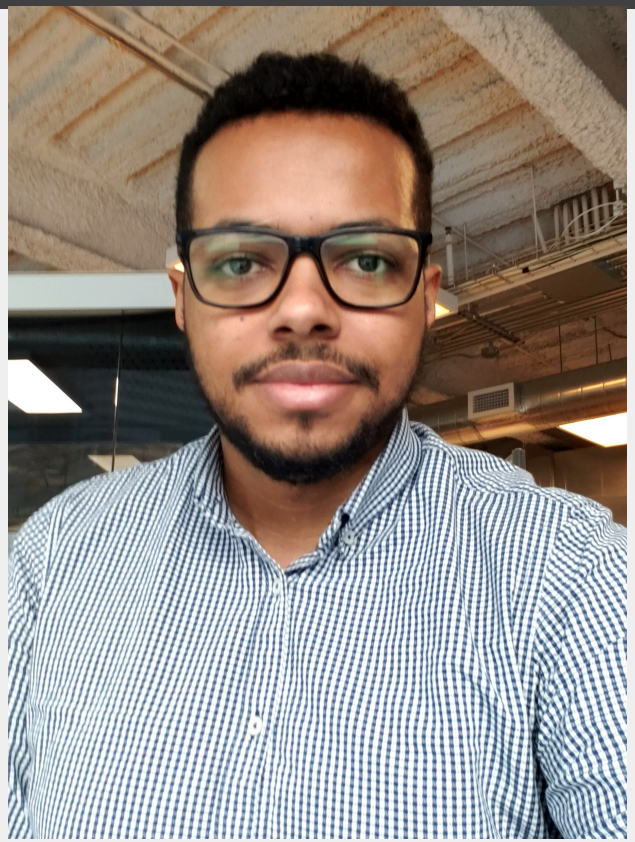
Vinícius Araújo Santos

Machine Learning Engineer **DATA H**

01/30/2020

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

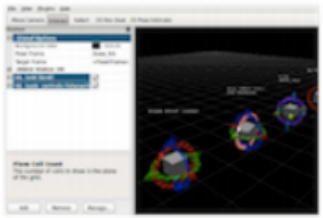
Slides

Please access: https://bit.ly/38TEvc1
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.

ROS Basic Structure

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 Basic Structure

Multi-language

ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).

Light-weight

Stand-alone libraries are wrapped around with a thin ROS layer.

Free and open-source

Most ROS software is open-source and free to use.

ROS MASTER

- Manages the communication between nodes.
- Every node registers at startup with the master node.

Start the master node with:

```
> roscore
```

ROS Nodes

- Single-purpose, executable program
- Individually compiled, executed, and managed
- Organized in packages

Run a node with :

```
> rosrun package_name node_name
```

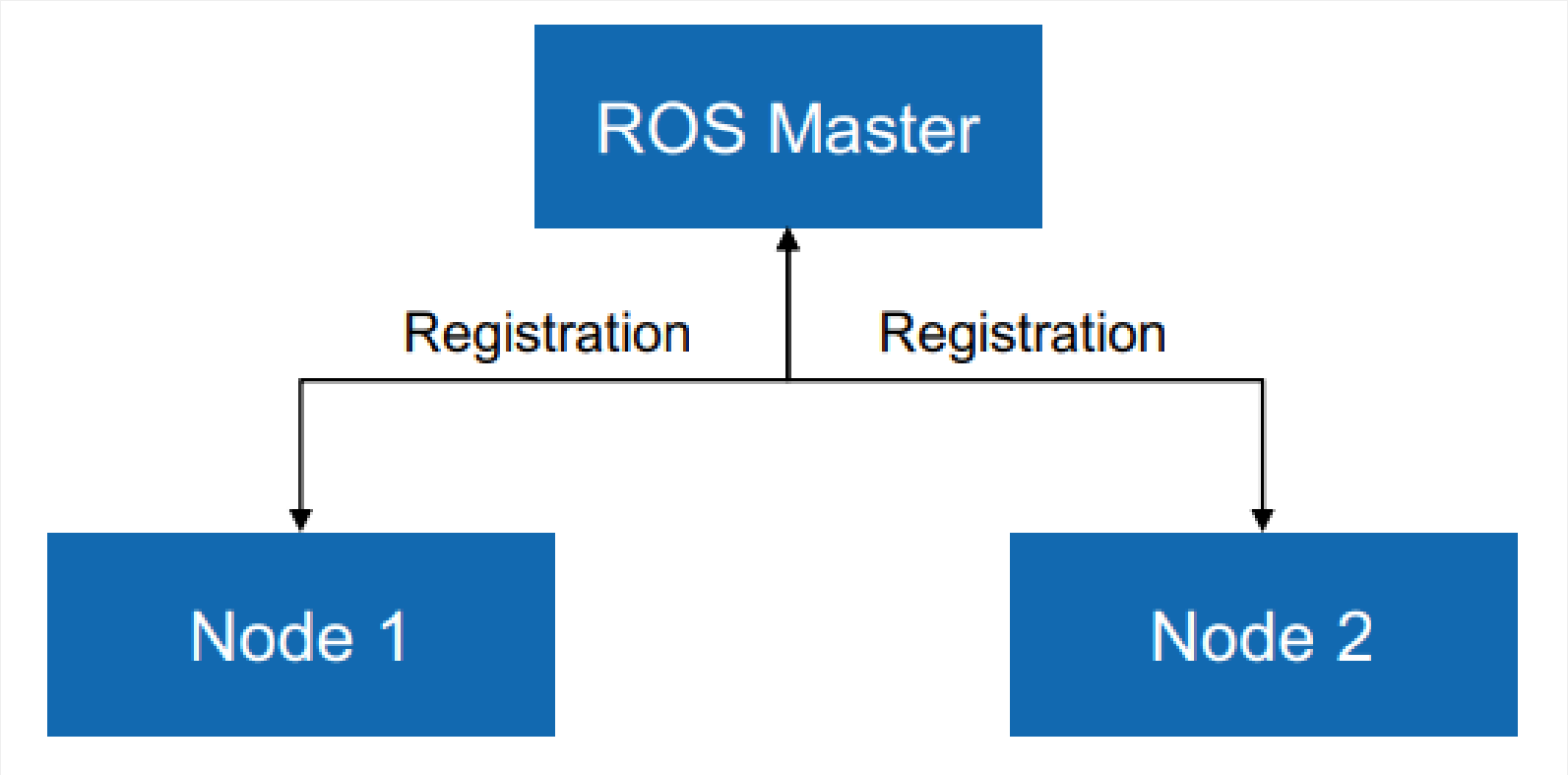
See active nodes with:

```
> rosnode list
```

Retrieve information about a node with:

```
> rosnode info node_name
```

ROS Nodes



ROS Topics

- Nodes communicate over topics
  - They can publish or subscribe to a topic
  - Typically, **1** publisher and **n** subscribers
- Topic is a name for a stream of messages

List active topics with :

```
> rostopic list
```

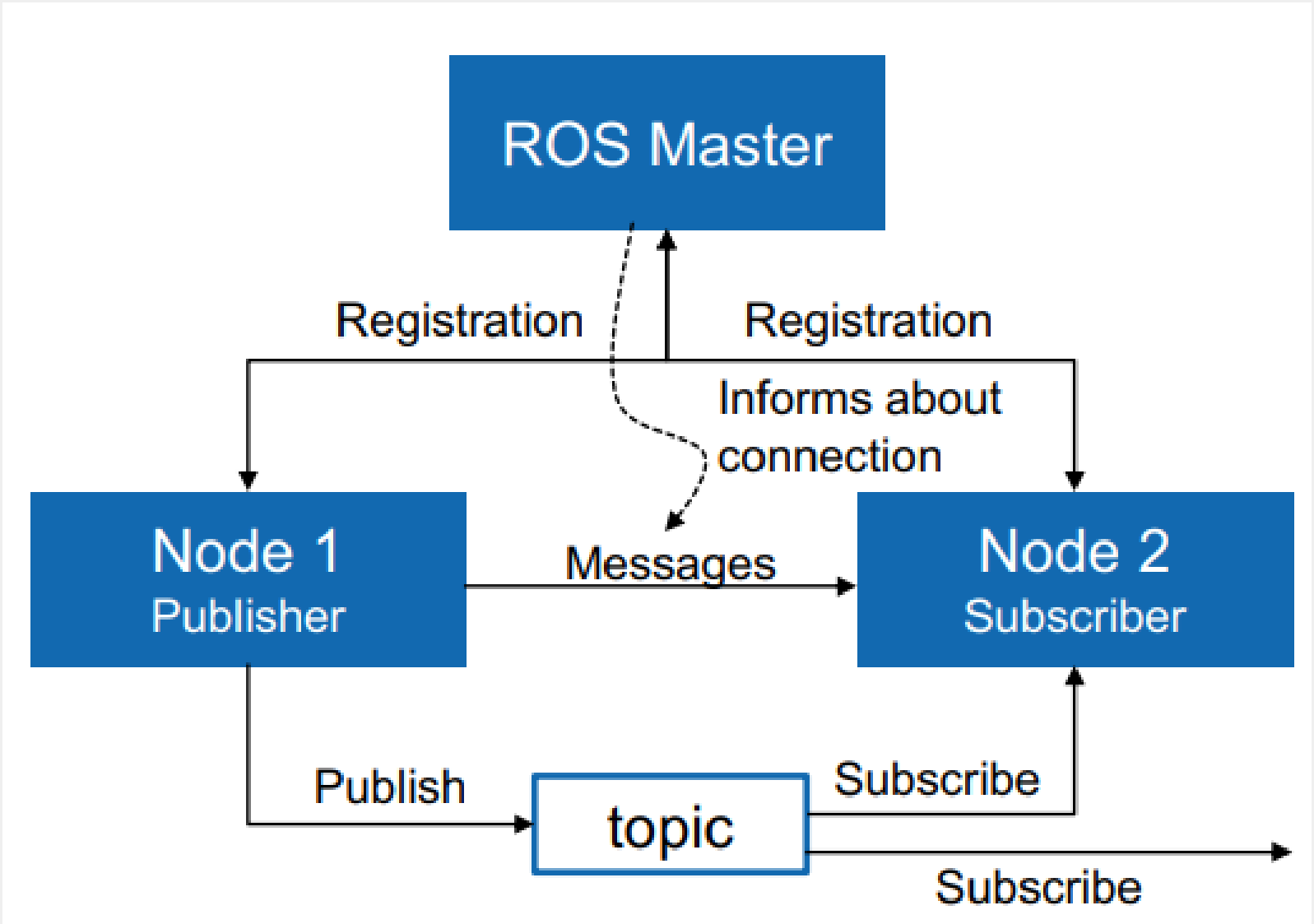
Manually Subscribe and print the contents of a topic with :

```
> rostopic echo /topic_name
```

Show information about a topic with :

```
> rostopic info /topic_name
```

ROS Nodes


ROS Master
Registration
Registration
Informs about
connection
Node 1
Publisher
Messages
Node 2
Subscriber
Publish
topic
Subscribe
Subscribe

ROS Messages

- Data structure defining the type of a topic
- Compromised of a nested structure of integers, floats, booleans, strings etc. and arrays of objects
- Defined in *.msg files

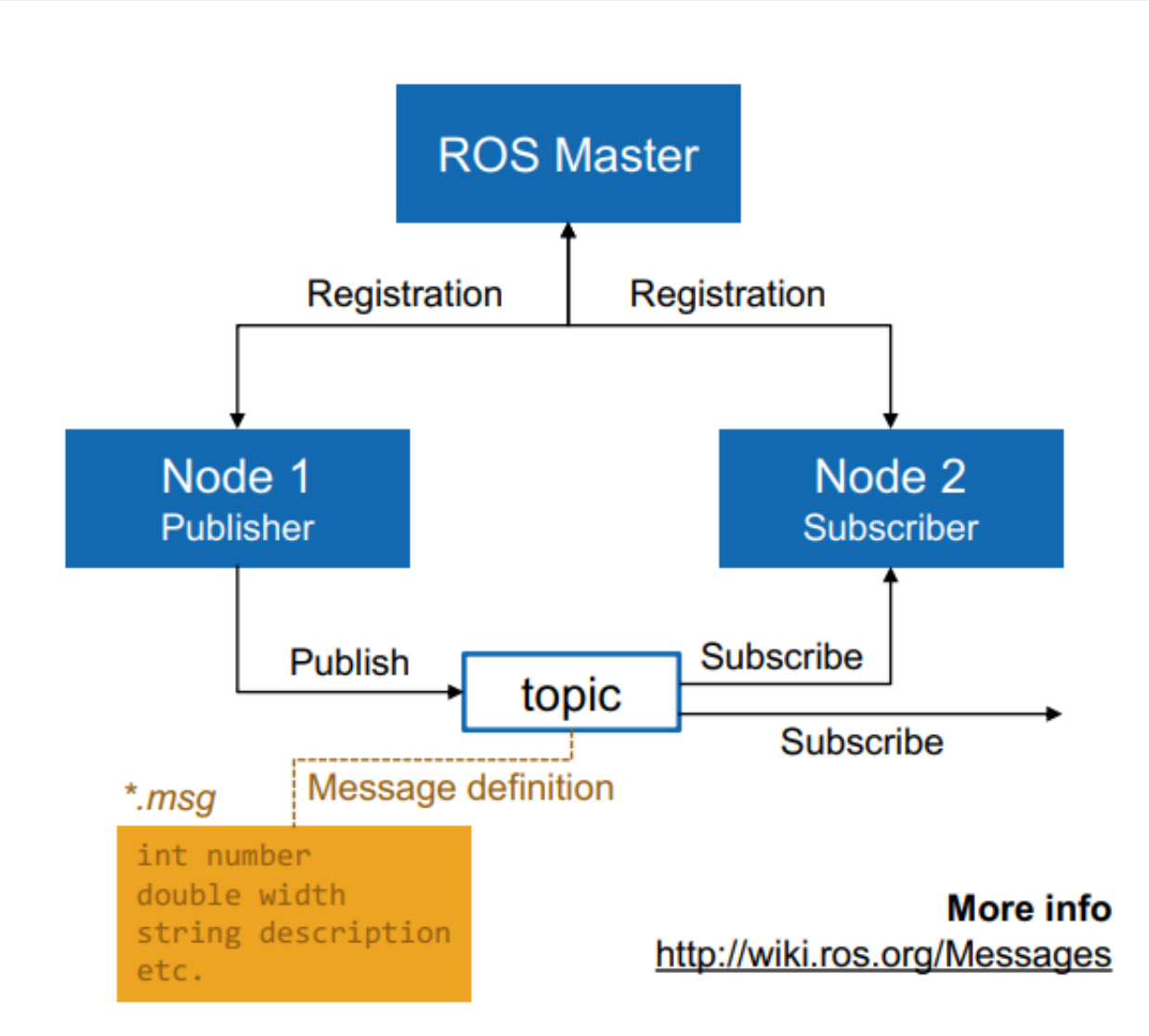
See the type of a topic message with :

```
> rostopic type /topic_name
```

Manually publish to a topic with :

```
> rostopic pub /topic_name type args
e.g
> rostopic pub /my_string_topic std_msgs/String "Hello World"
```

ROS Messages



ROS Messages

geometry_msgs/Point.msg

```
float64 x
float64 y
float64 z
```

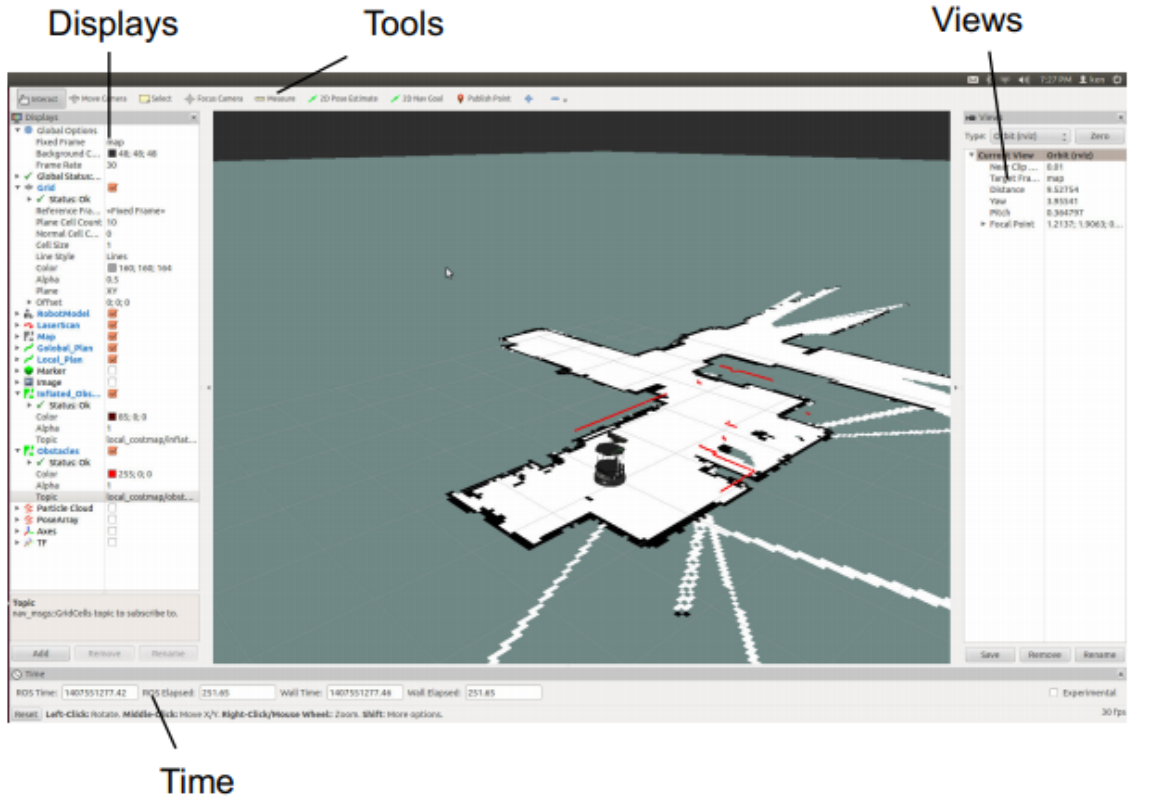
sensor_msgs/Image.msg

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

geometry_msgs/PoseStamped.msg

```
std_msgs/Header header
 uint32 seq
 time stamp
 string frame_id
geometry_msgs/Pose pose
  geometry_msgs/Point position
    float64 x
    float64 y
    float64 z
  geometry_msgs/Quaternion orientation
    float64 x
    float64 y
    float64 z
    float64 w
```

Rviz - Visualization Tool for ROS systems

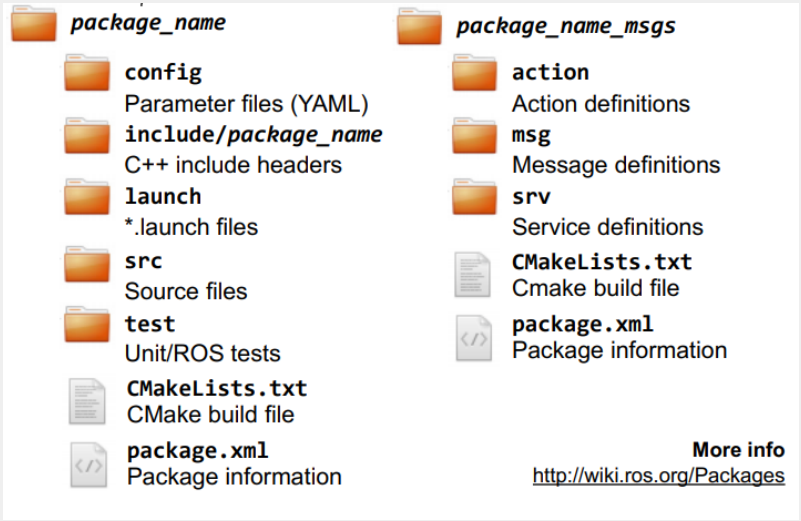


To launch rviz, use (after launching the roscore):

```
> rosrun rviz rviz
```

Jump Start with ROS Packages

- ROS software is organized into packages, which can contain source code, launch files, configuration files, message definitions, data, and documentation
- A package that builds up on/requires other packages (e.g. message definitions), declares these as dependencies.



To create a new package, use:

> catkin_create_pkg package_name {dependencies}

Using ROS in Python 3

The rospy is a ROS **client library** for python.

- The rospy enable Python programmers to quickly interface with ROS Topics, Services, and Parameters.
- The design of rospy favors implementation speed (i.e. developer time) over runtime performance so that algorithms can be quickly prototyped and tested within ROS.
- It is also ideal for non-critical-path code, such as configuration and initialization code.
- Many of the ROS tools are written in rospy to take advantage of the type introspection capabilities.

Using ROS in Python 3 - Example

Create a new package for this example using:

```
> catkin_create_pkg droid_equinox_09 std_msgs rospy
```

Go to the src folder

```
> cd droid_equinox_09/src
```

Using ROS in Python 3 - Example

Preparing the environment:

> sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
> sudo apt install python-pip

Installing some packages:

> pip install rospkg
> pip install empy

Using ROS in Python 3 - Example

Download sample and change permissions:

> wget
`https://raw.github.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py`
> chmod +x talker.py

Let's Run!

> roscore
In another window ...
> python talker.py

Testing ...

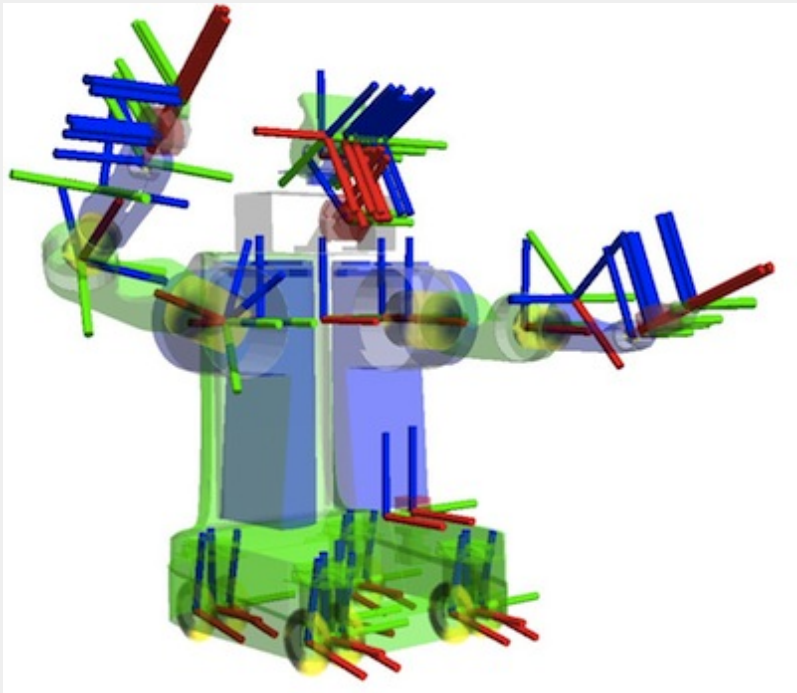
In another window ...
> rostopic echo /chatter

Using ROS in Python 3 - Example Code

```
import rospy
from std_msgs.msg import String
def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()
if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

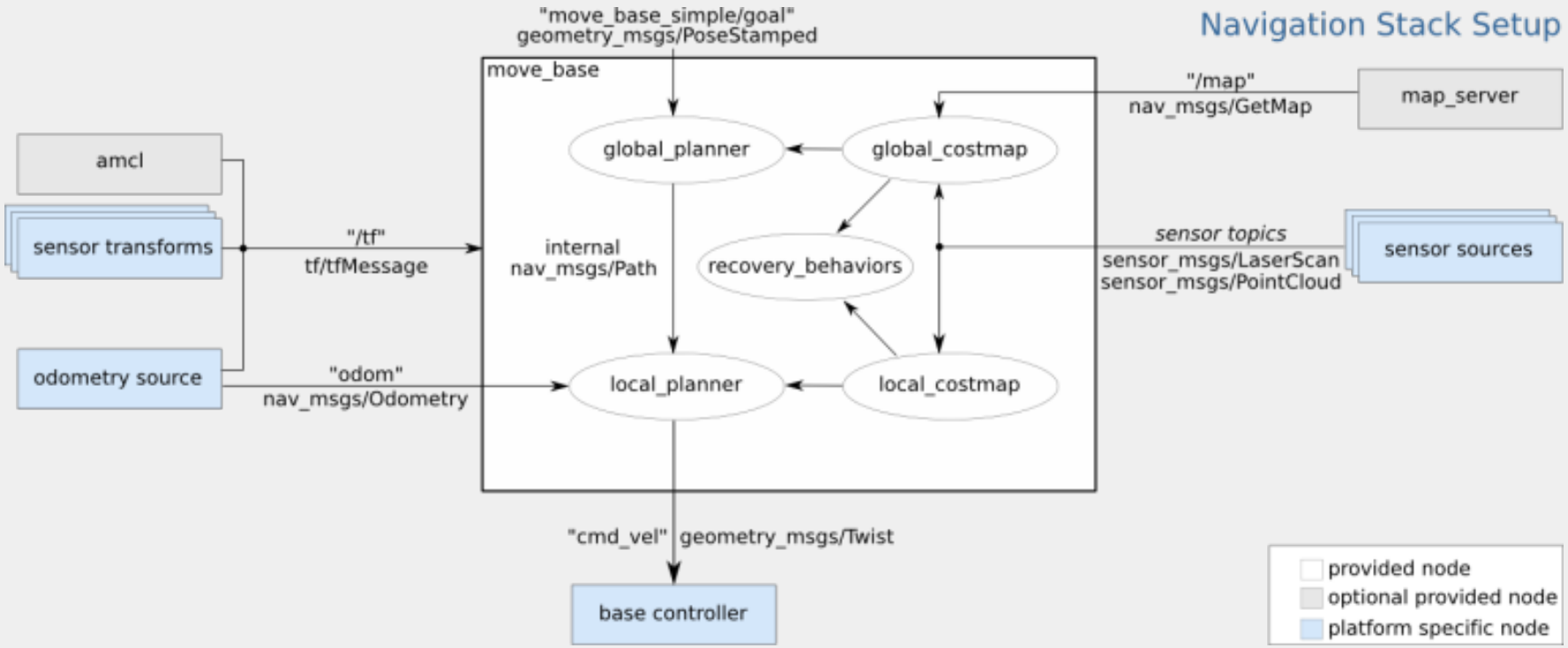
The TF Transformation System

- Tool for keeping track of coordinate frames over time.
- Maintains relationship between coordinate frames in a tree structure buffered in time.
- Lets the user transform points, vectors, etc. between coordinate frames at desired time.
- Implemented as publisher/subscriber model on the topics /tf and /tf_static.



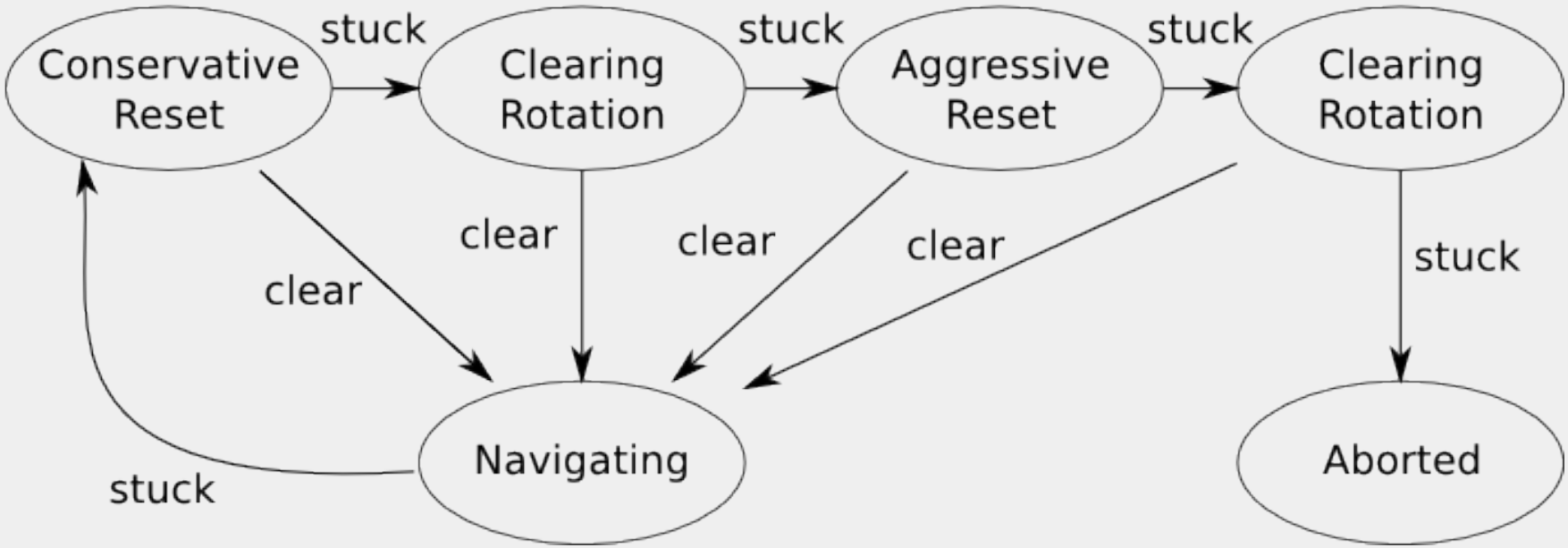
Next Week

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)

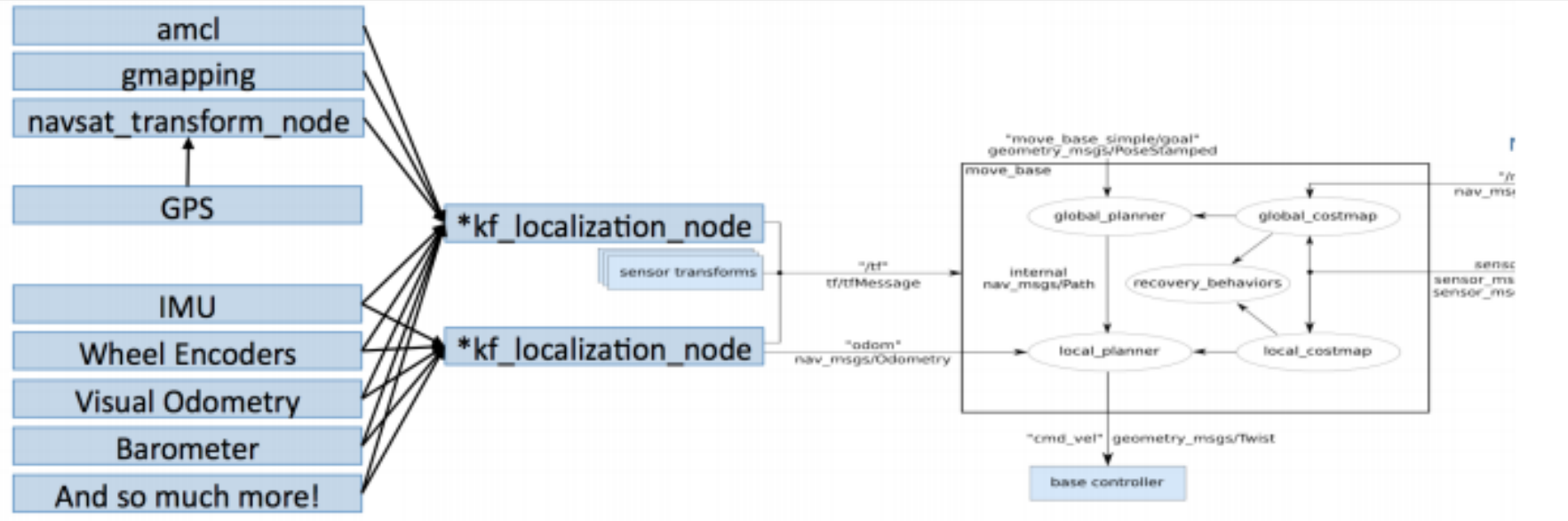
The amcl package is commonly used when a priori maps are known.

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

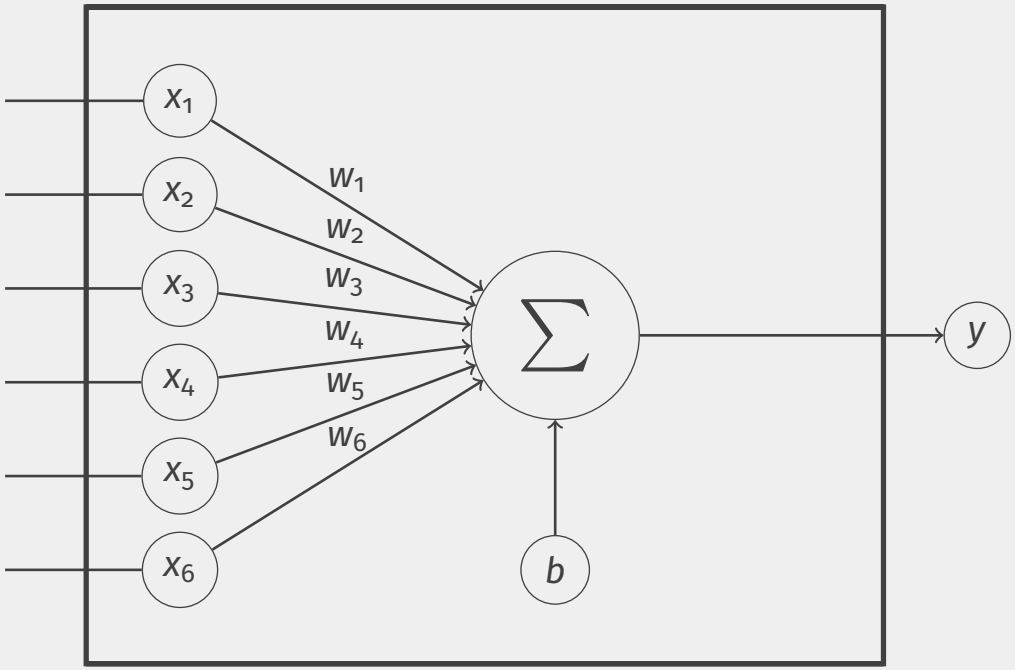
  Text classification list of words $\to$ spam or valid email

Deep learning

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

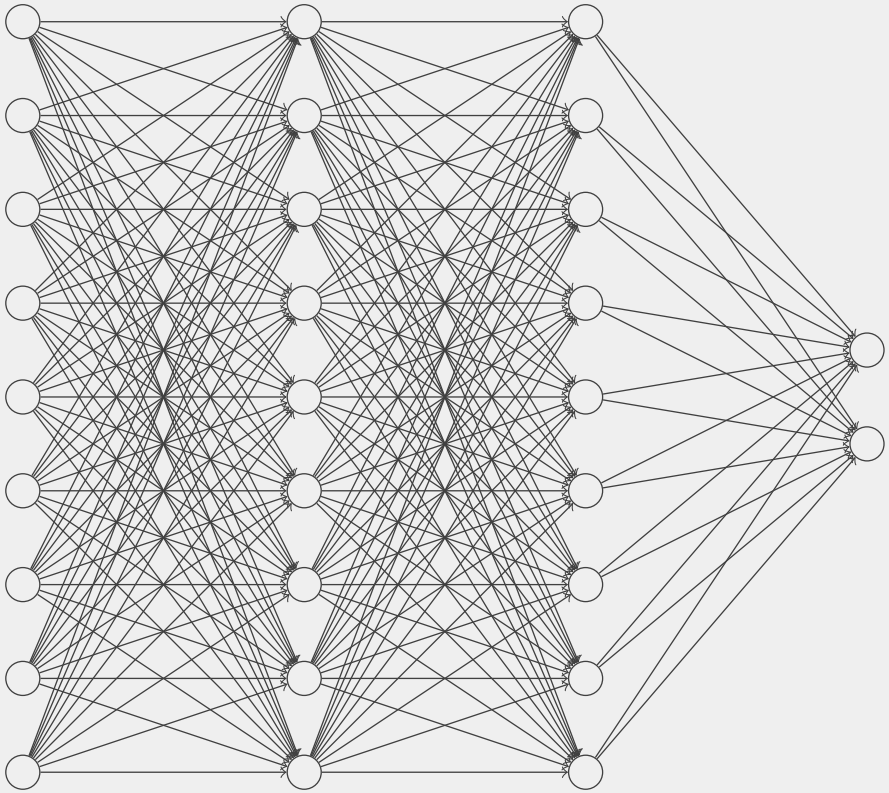
An artificial neural networks is a computation model loosely based on the human brain. It aims to mimic electric signals travelling through neurons in order to make computations.

Neuron

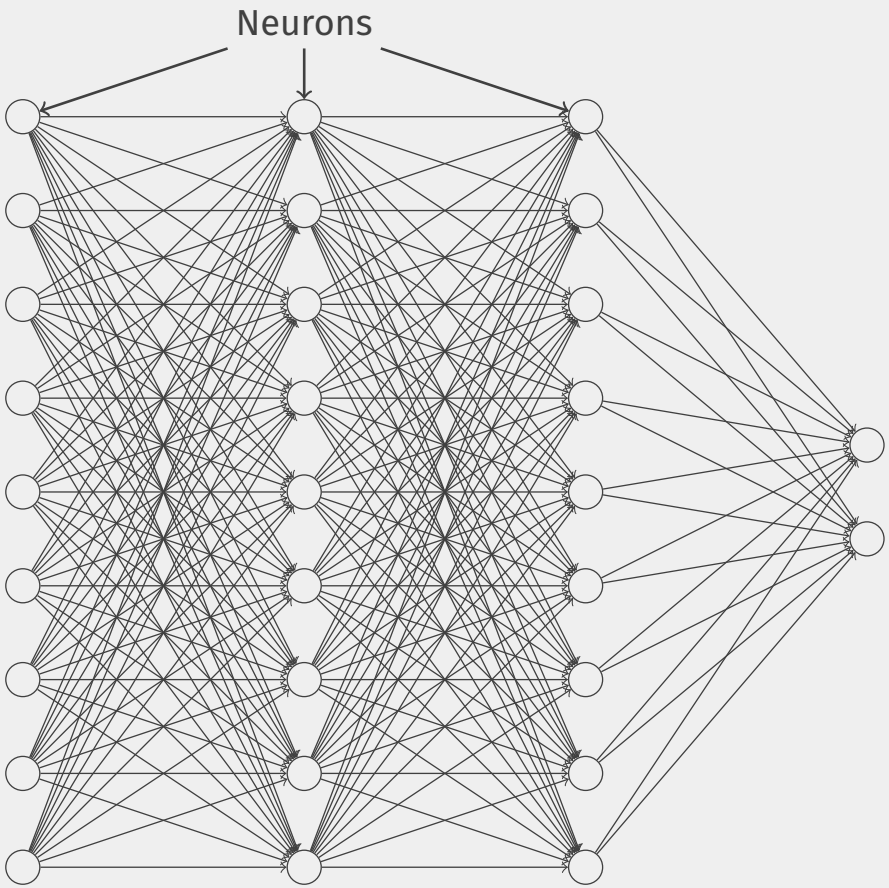


$$y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + b$$

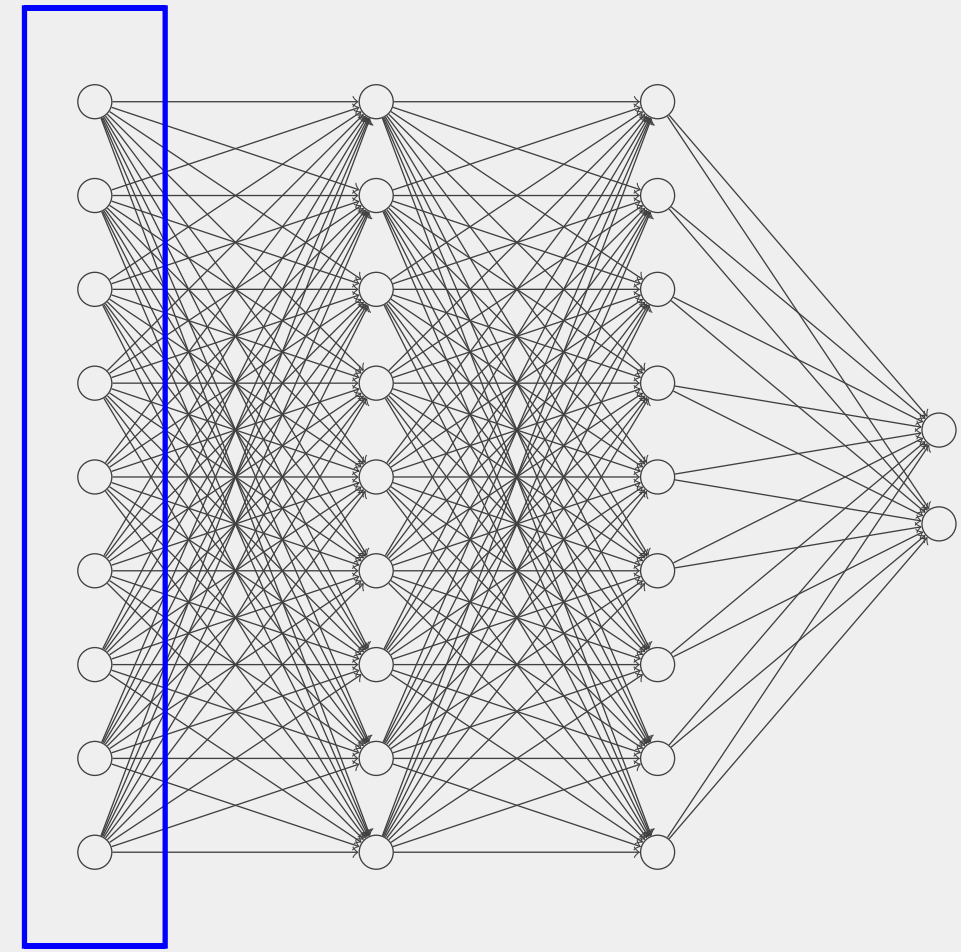
Network



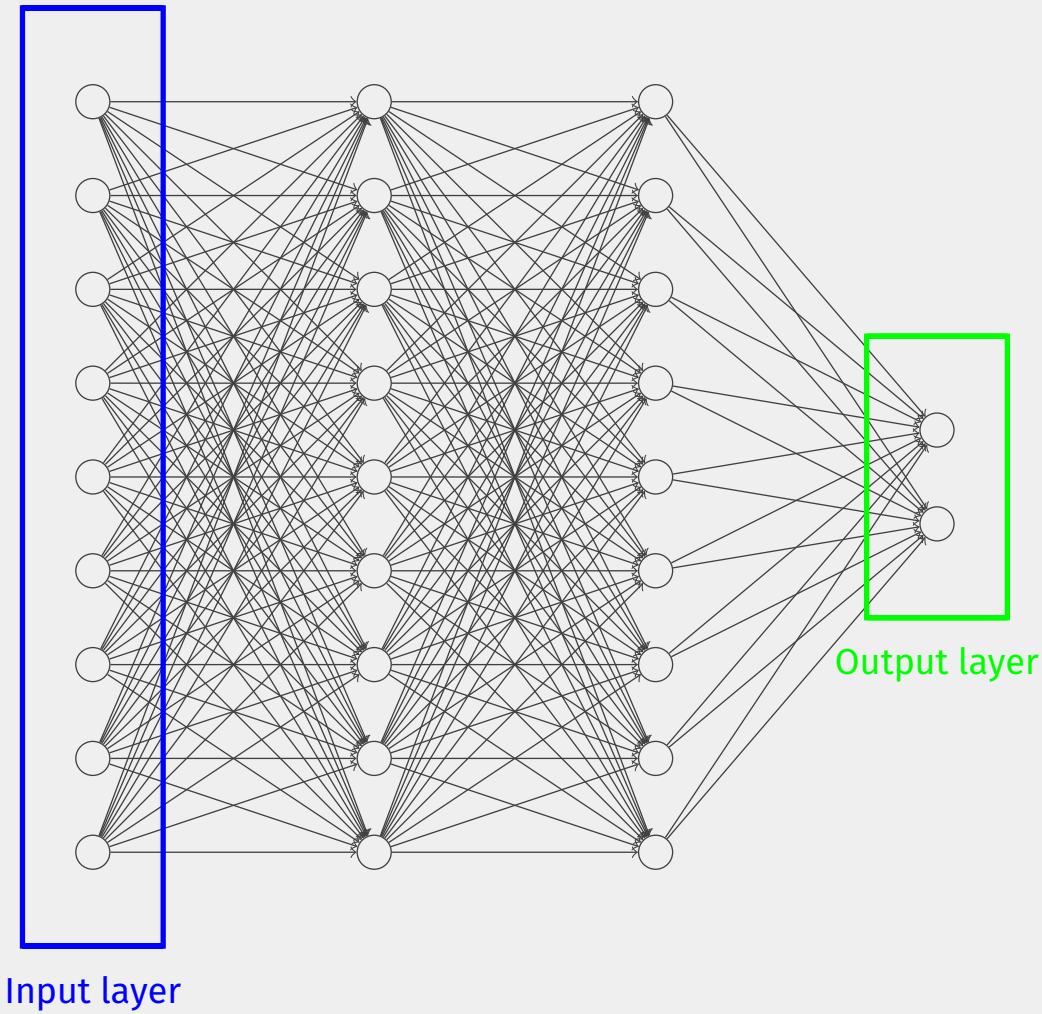
Network



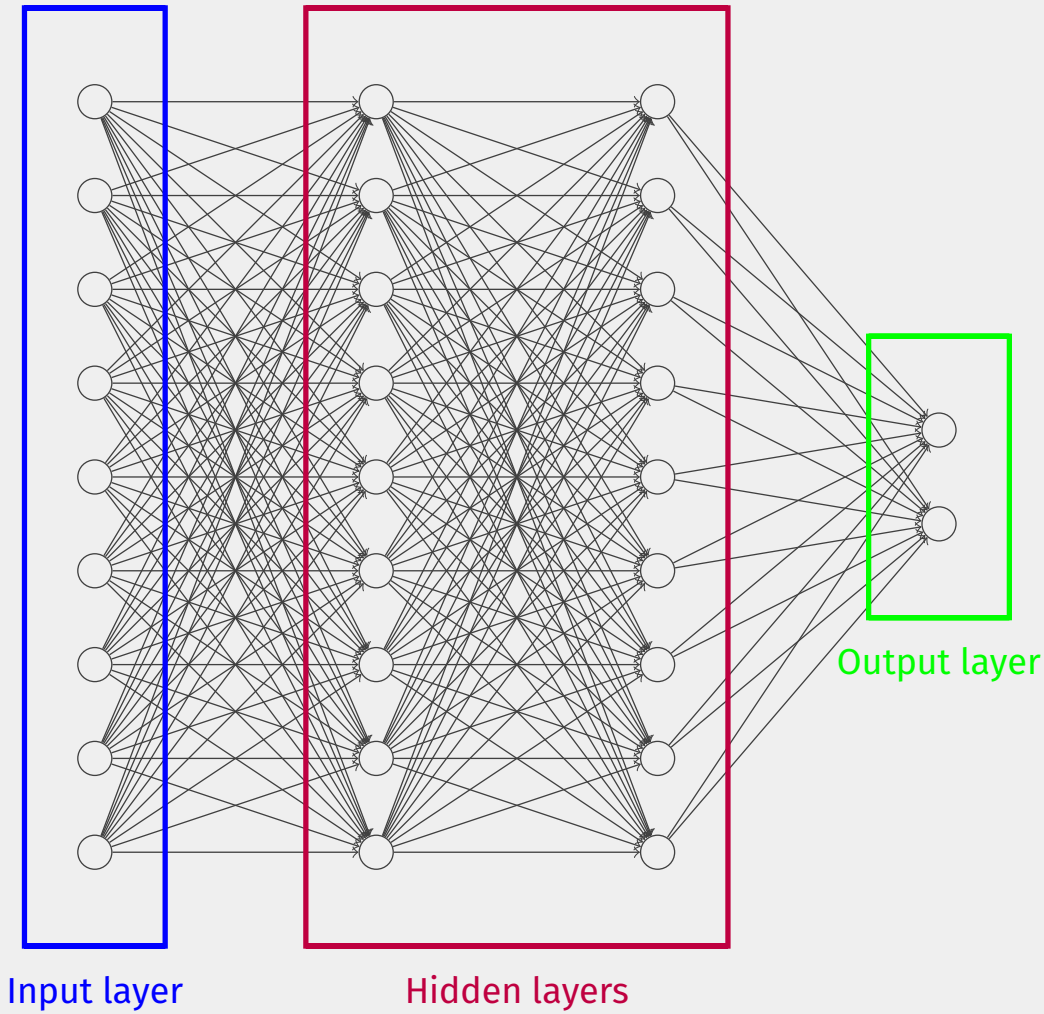
Network



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

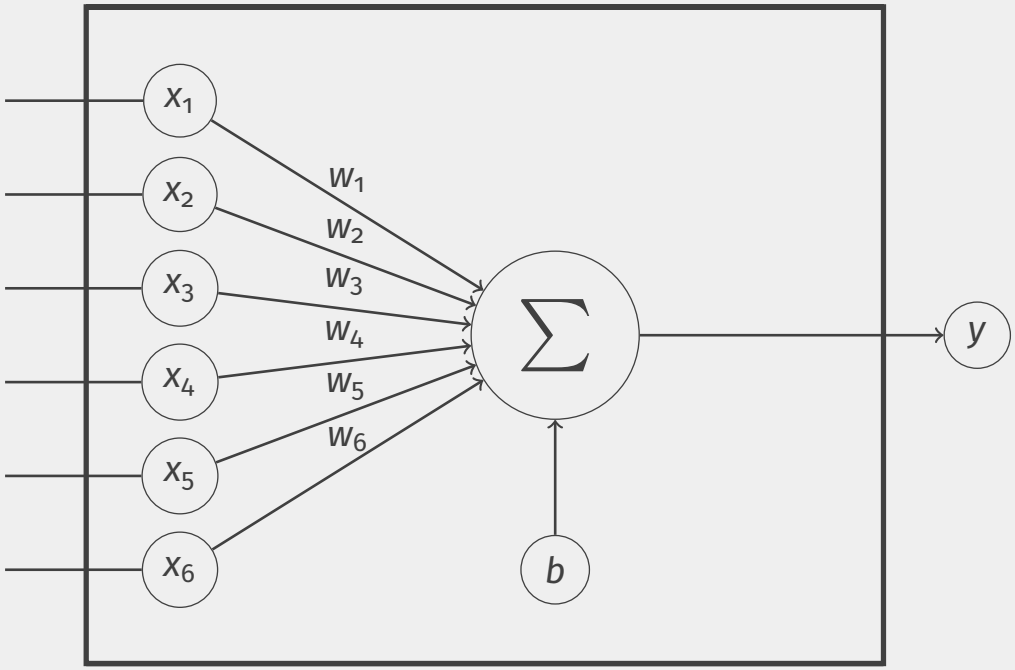
We now have a quite complicated framework to compute linear functions.

Neuron	linear function
Neural network	linear combination of neuron outputs

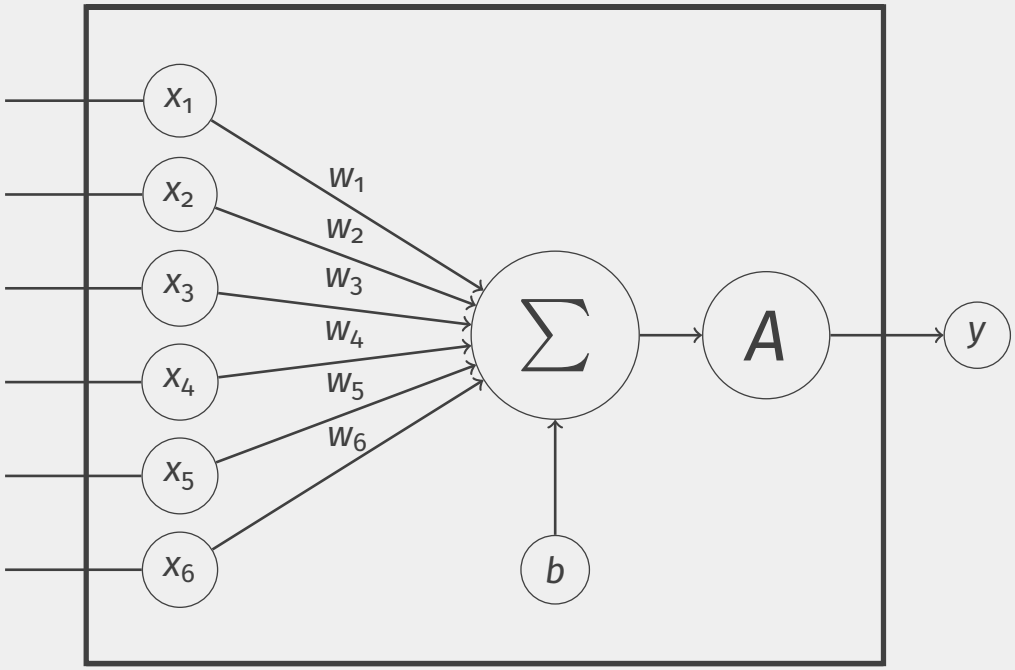
To approximate non linear functions, we would like to have a non linear model.

Solution: add nonlinearity to neurons.

Neuron with activation

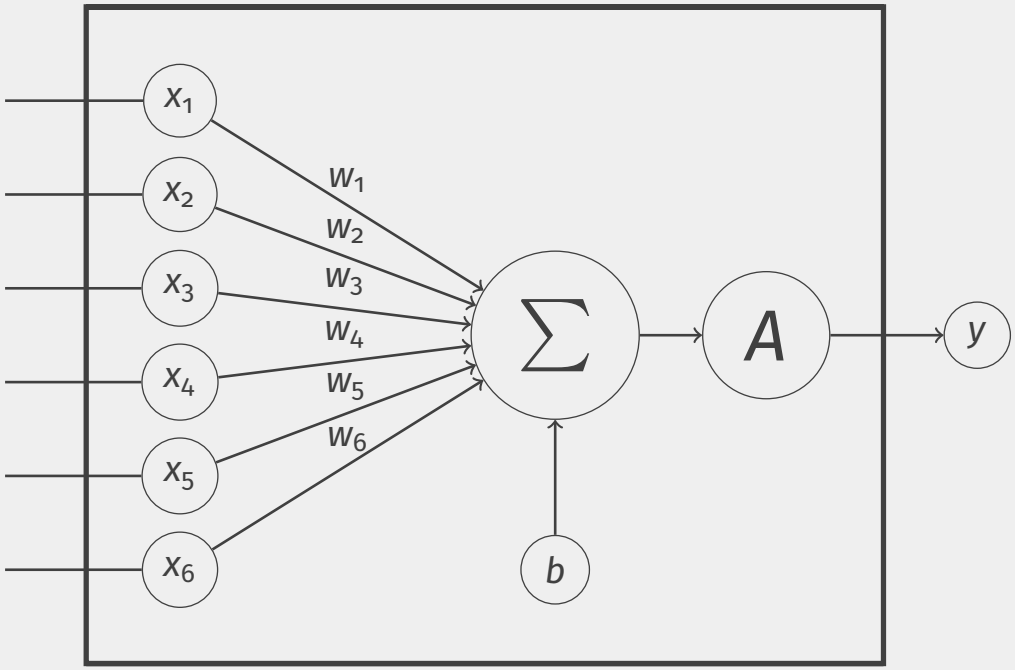


Neuron with activation



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

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

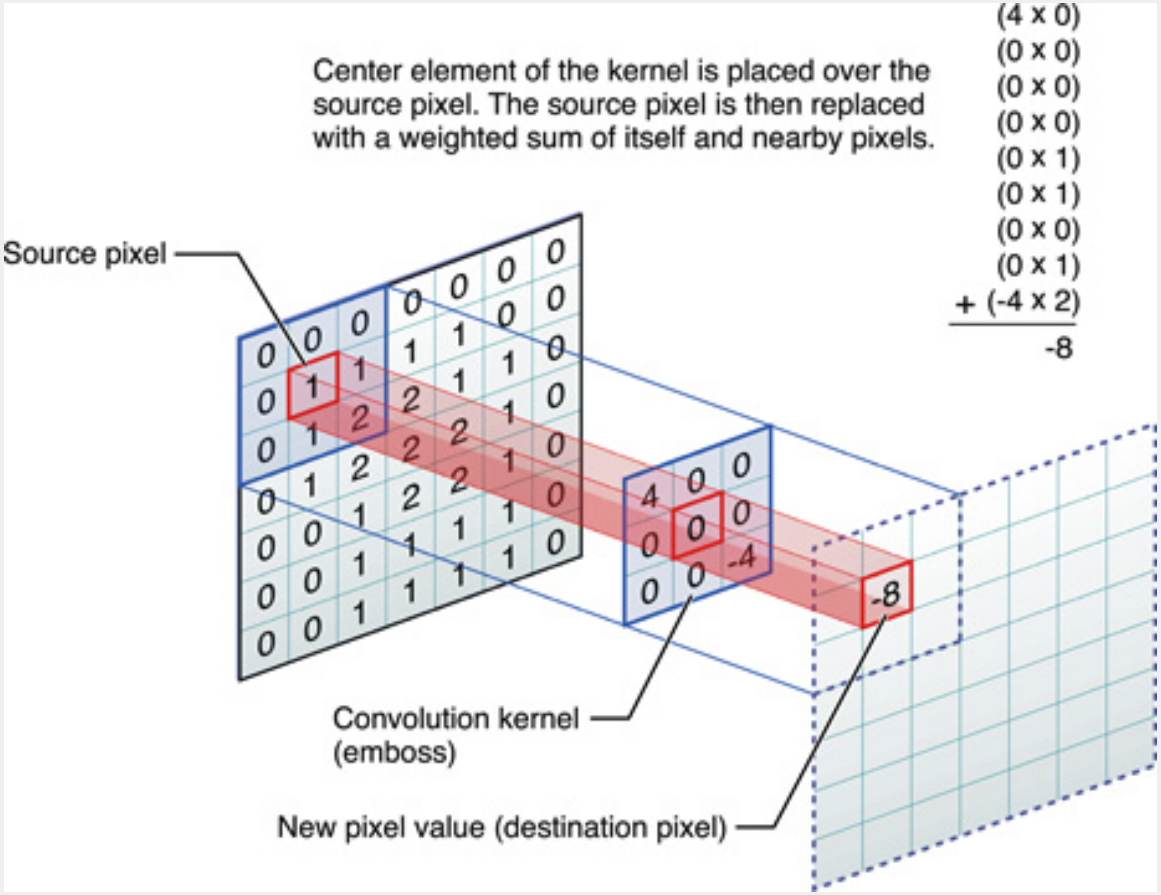
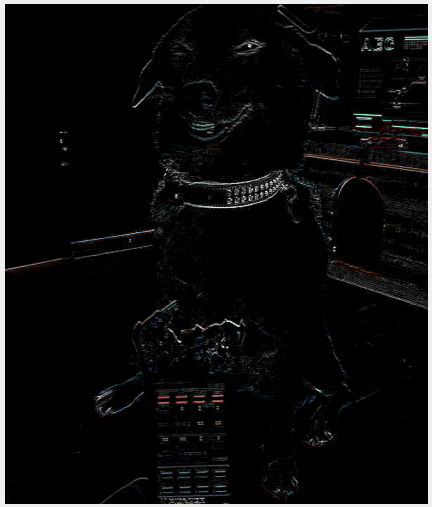


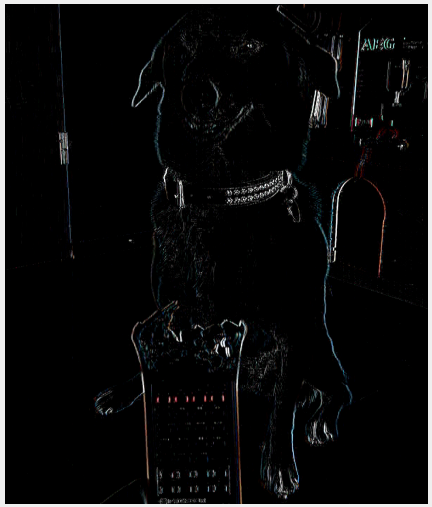
Image from `http://stats.stackexchange.com/`

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)

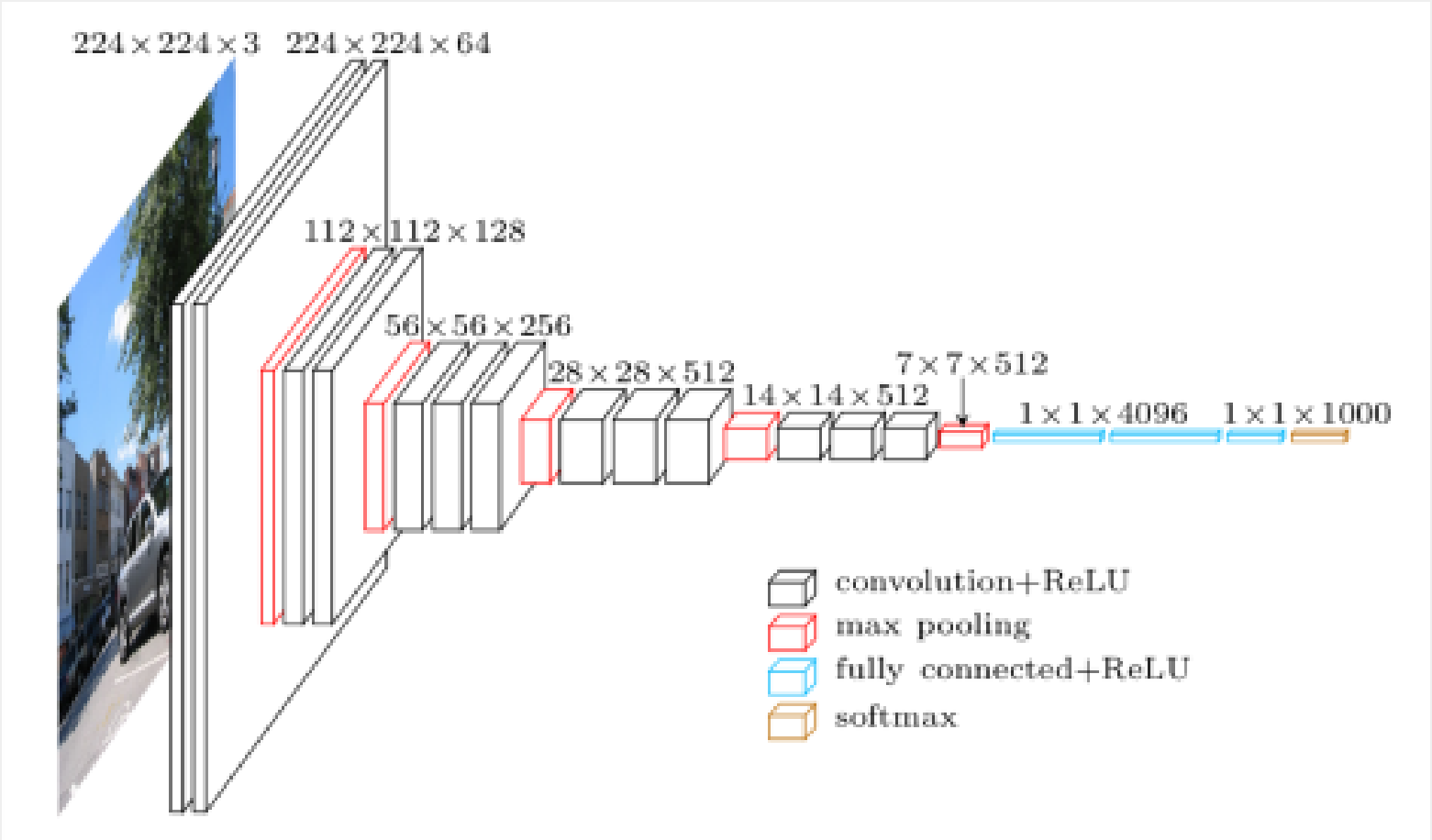


Image from `https://www.cs.toronto.edu/~frossard/post/vgg16/`

What do the filters recognize?

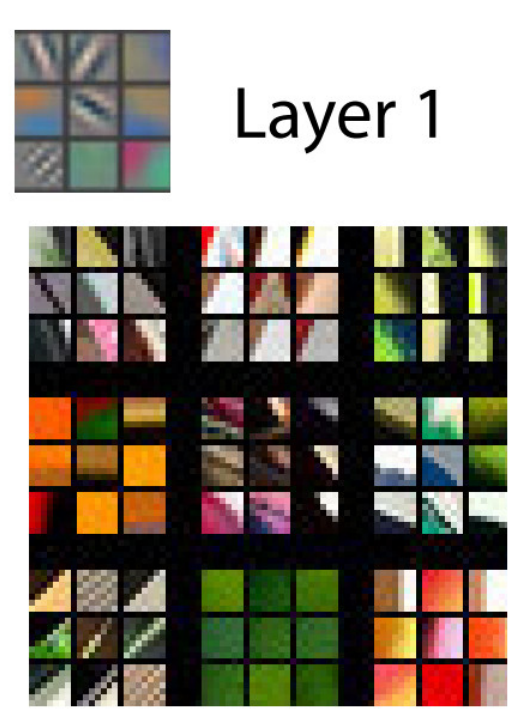


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

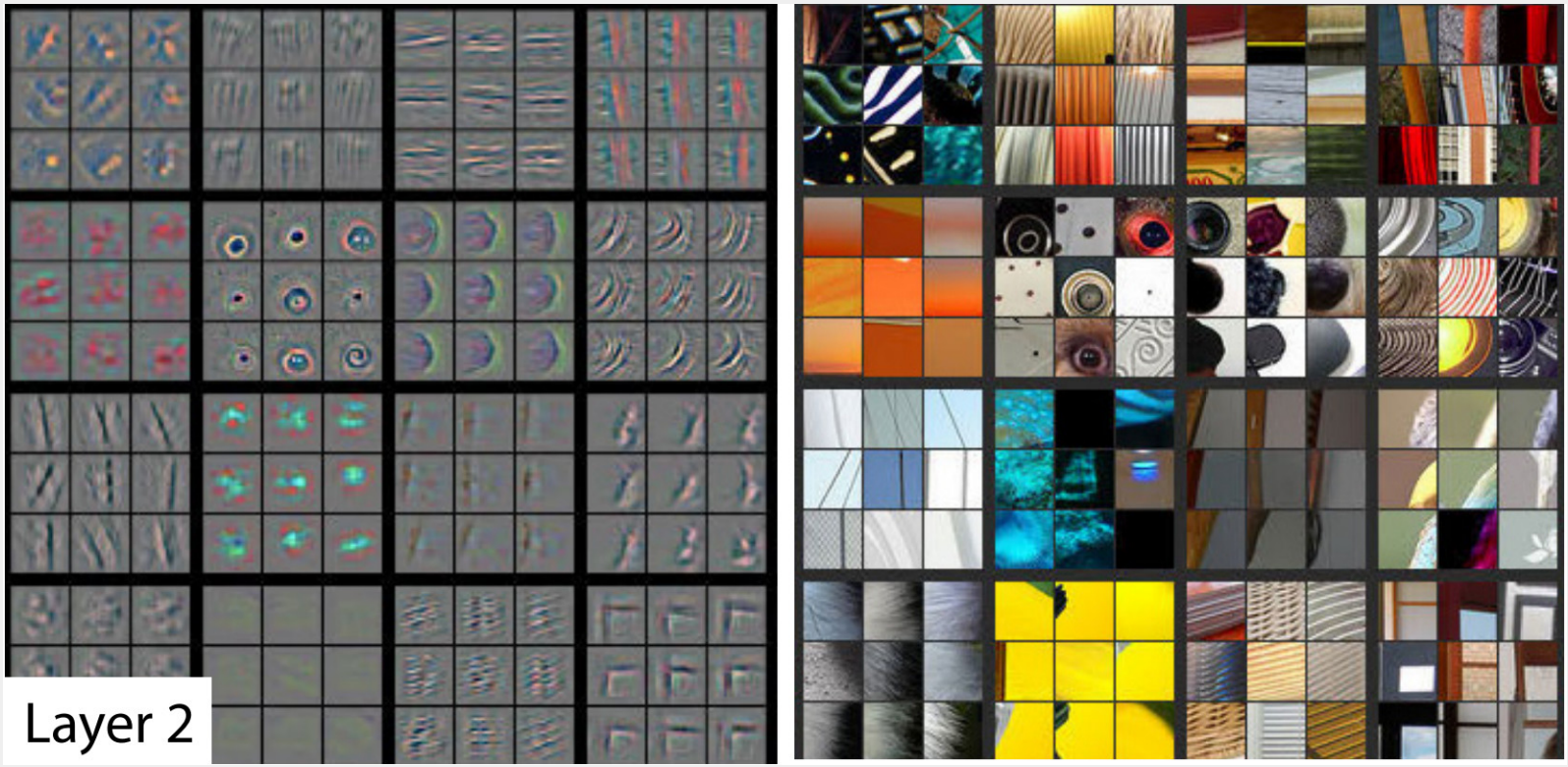


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

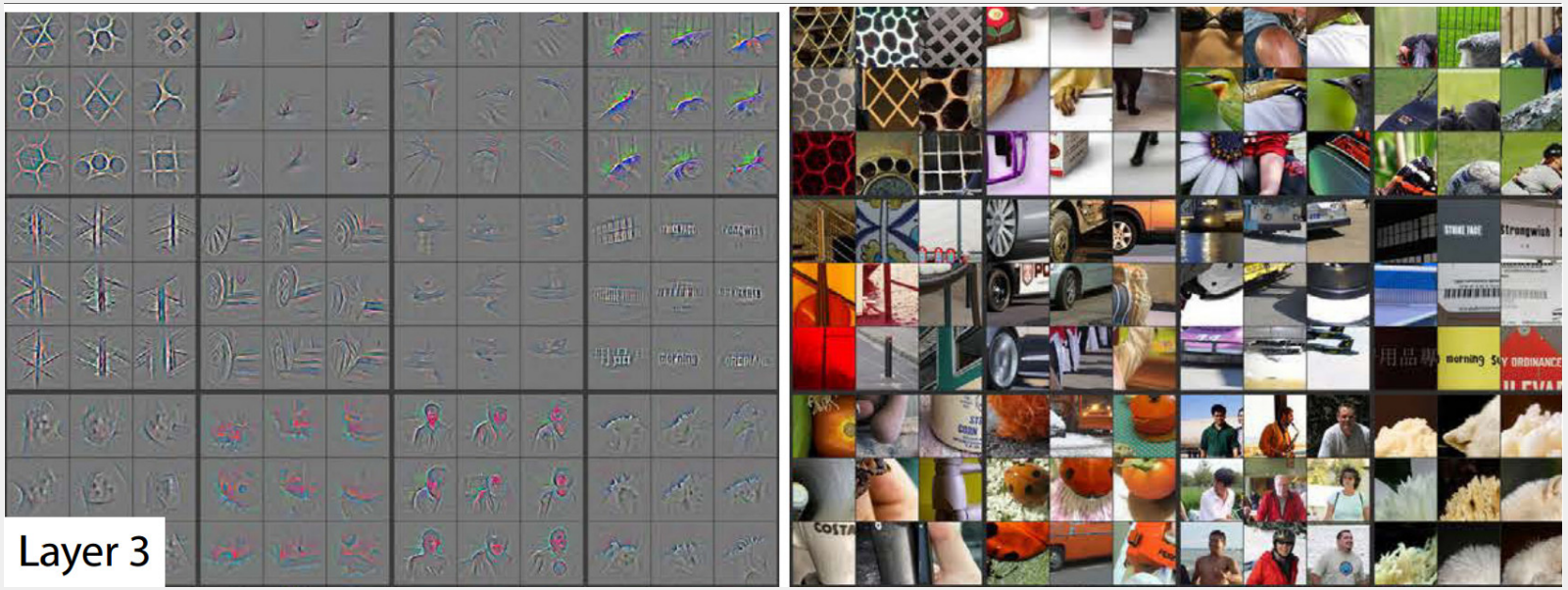


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

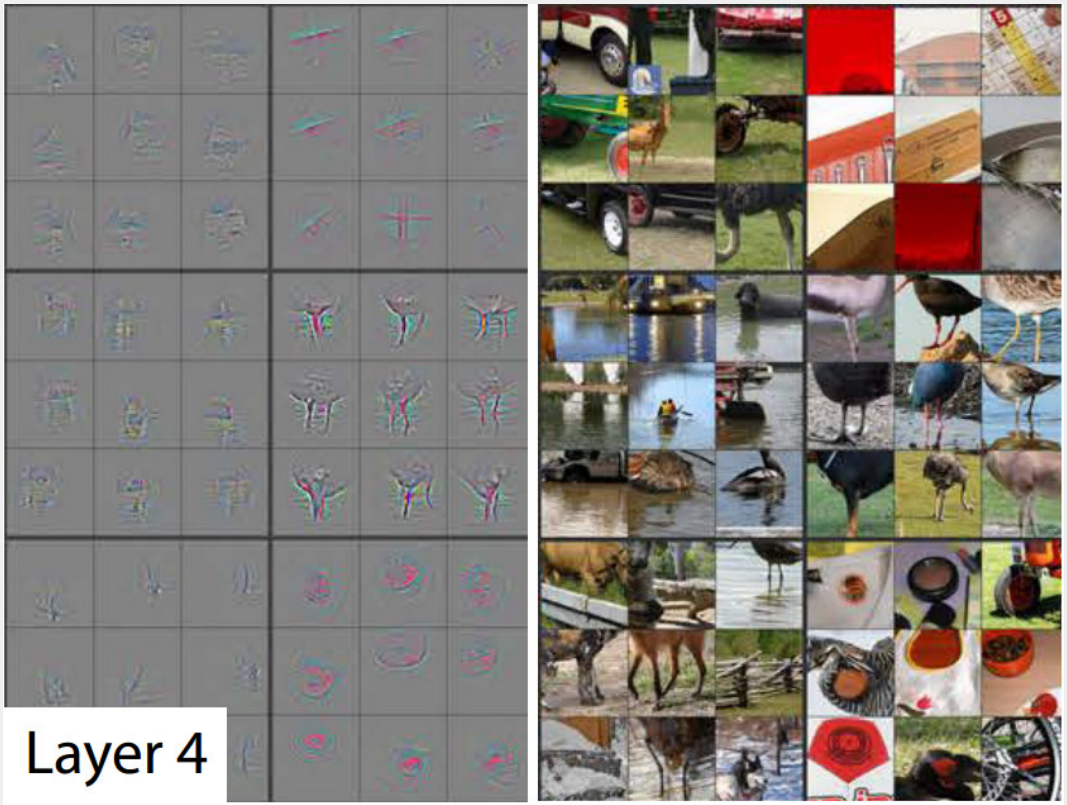


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

What do the filters recognize?

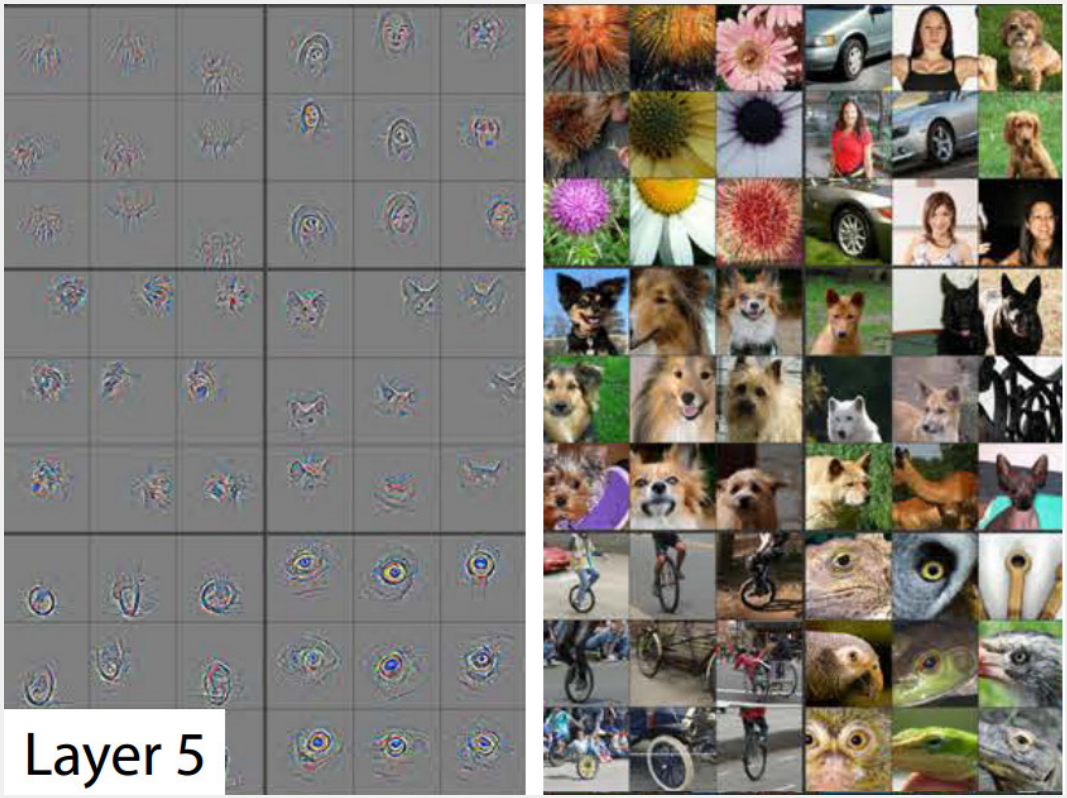


Image from `http://www.matthewzeiler.com/pubs/arxive2013/eccv2014.pdf`

VGG network (2014)

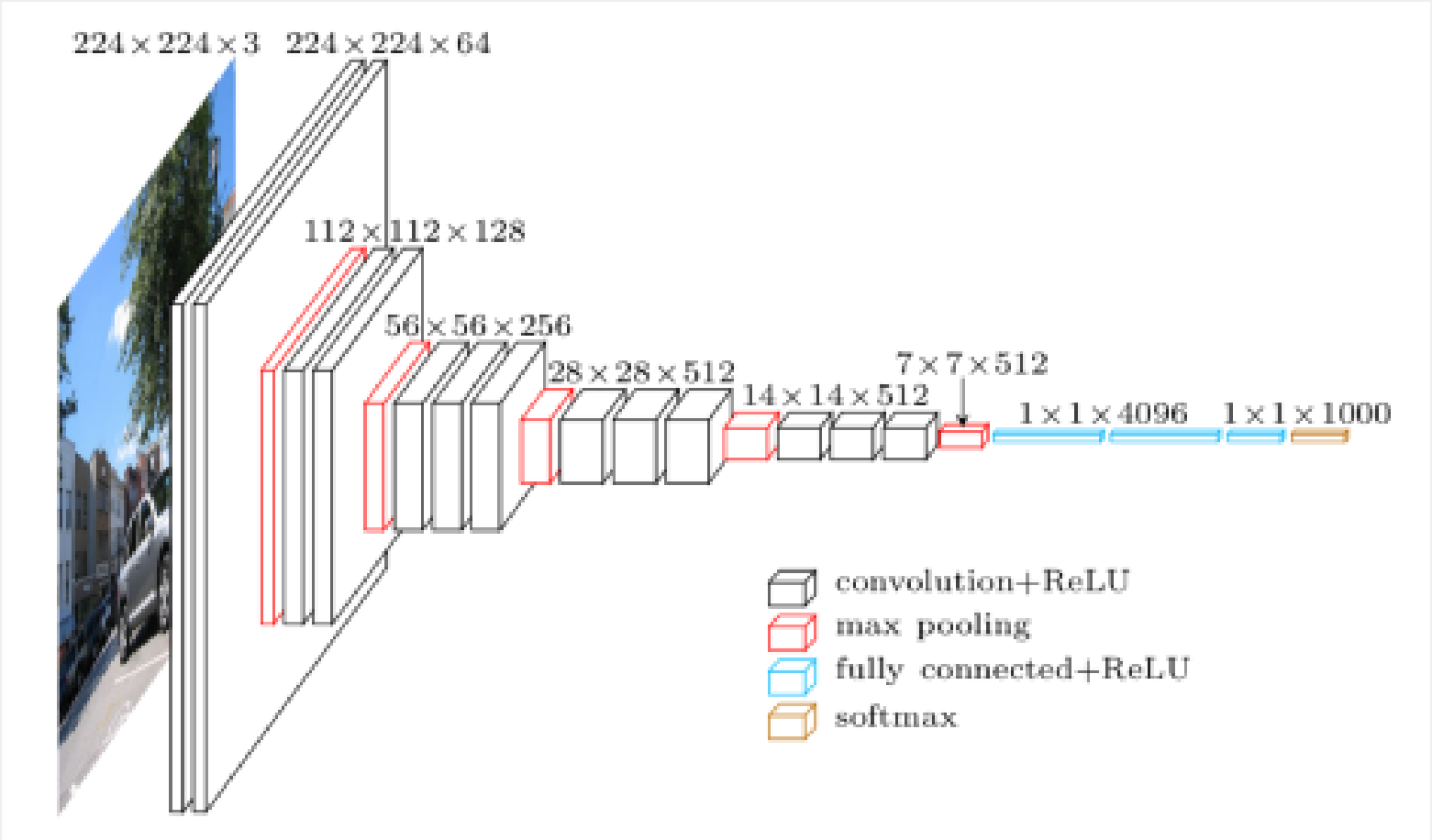
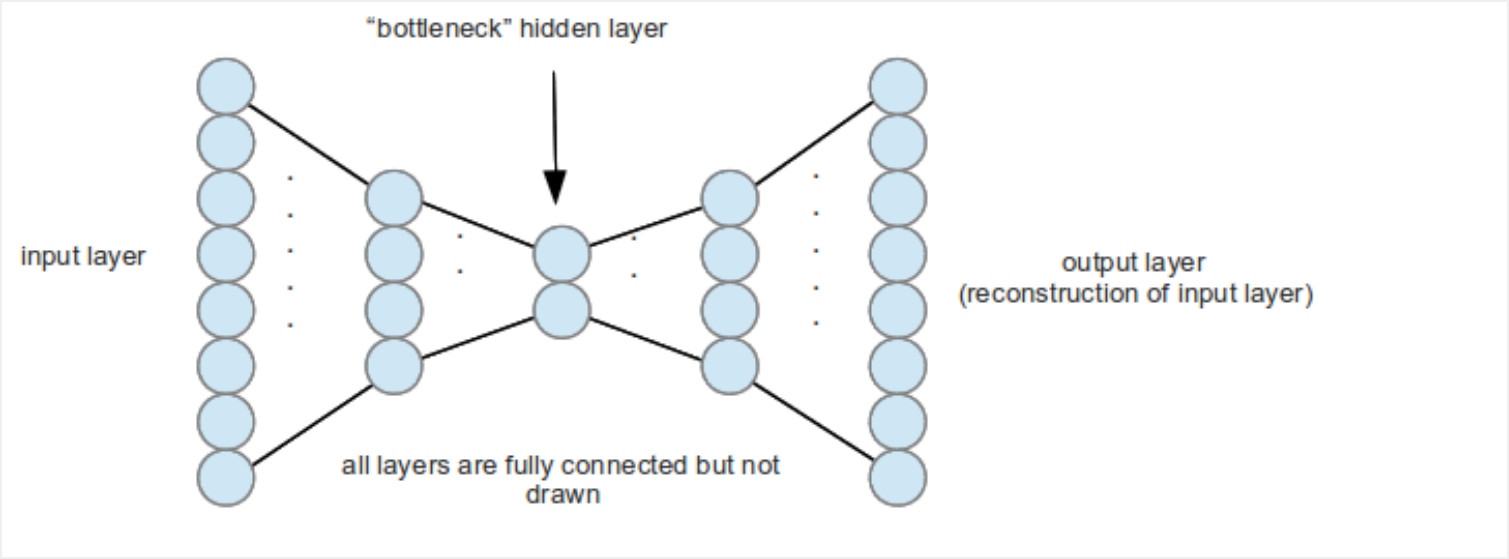


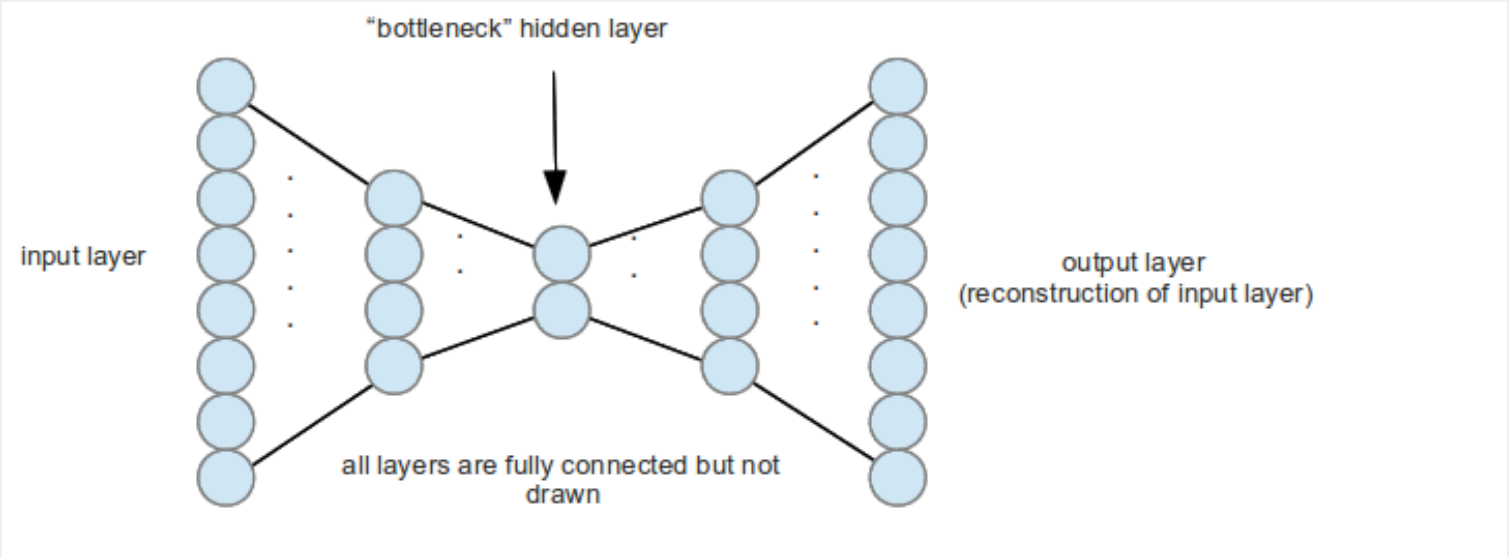
Image from `https://www.cs.toronto.edu/~frossard/post/vgg16/`

Autoencoder: data encoding



Hinton, Salakhutdinov (2006)

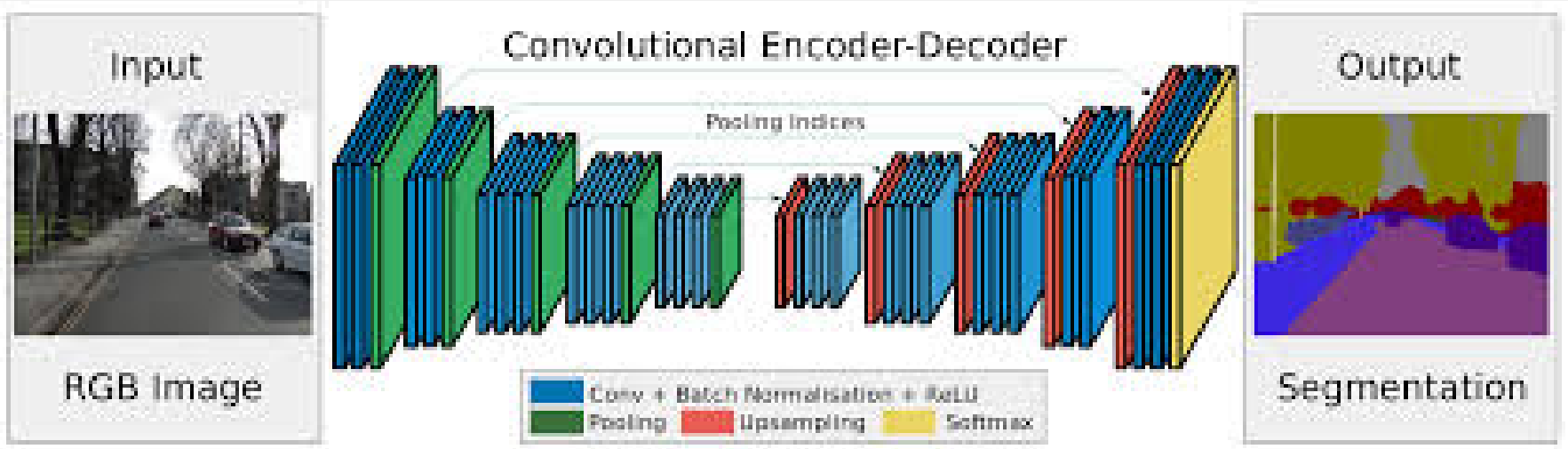
Autoencoder: data encoding



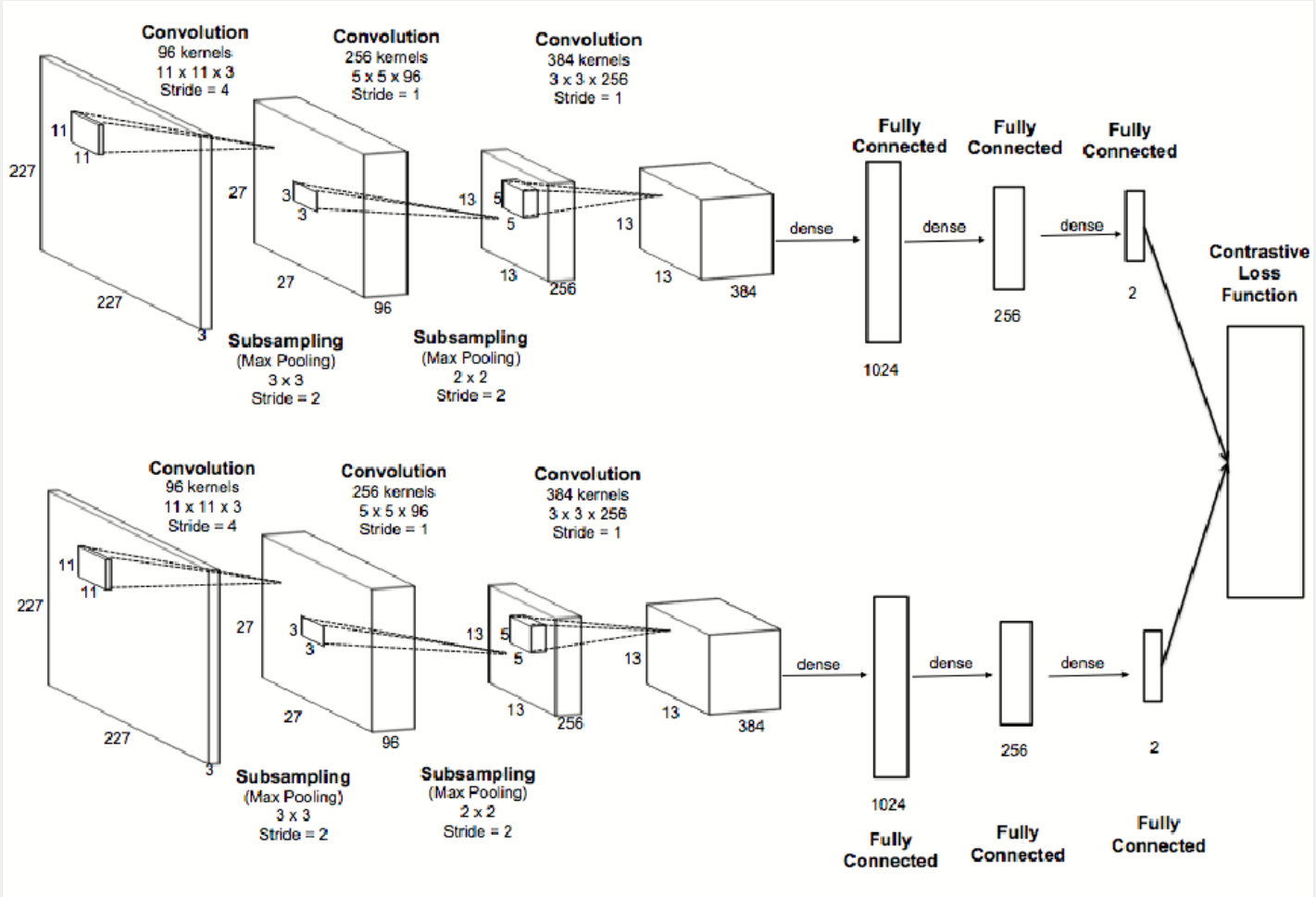
Hinton, Salakhutdinov (2006)

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.

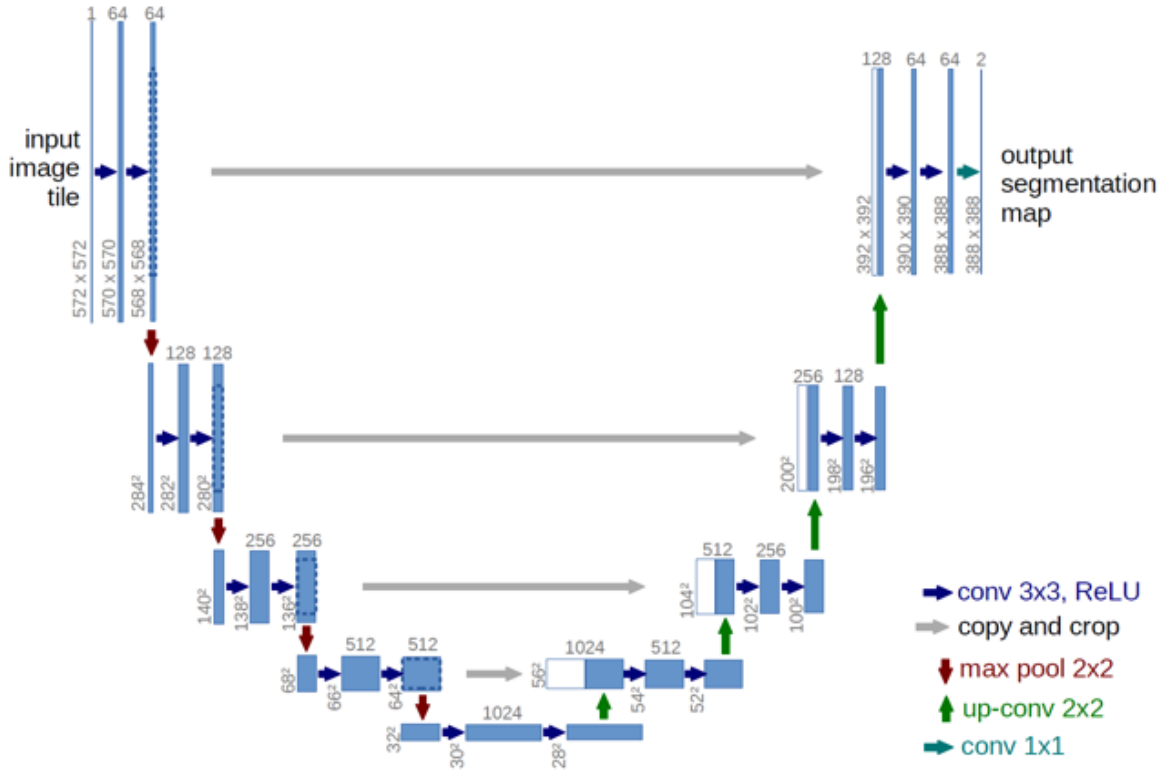
CNN + Autoencoder network: Semantic Segmentation



Siamese Networks



Unet: Semantic Segmentation



Recurrent neural network

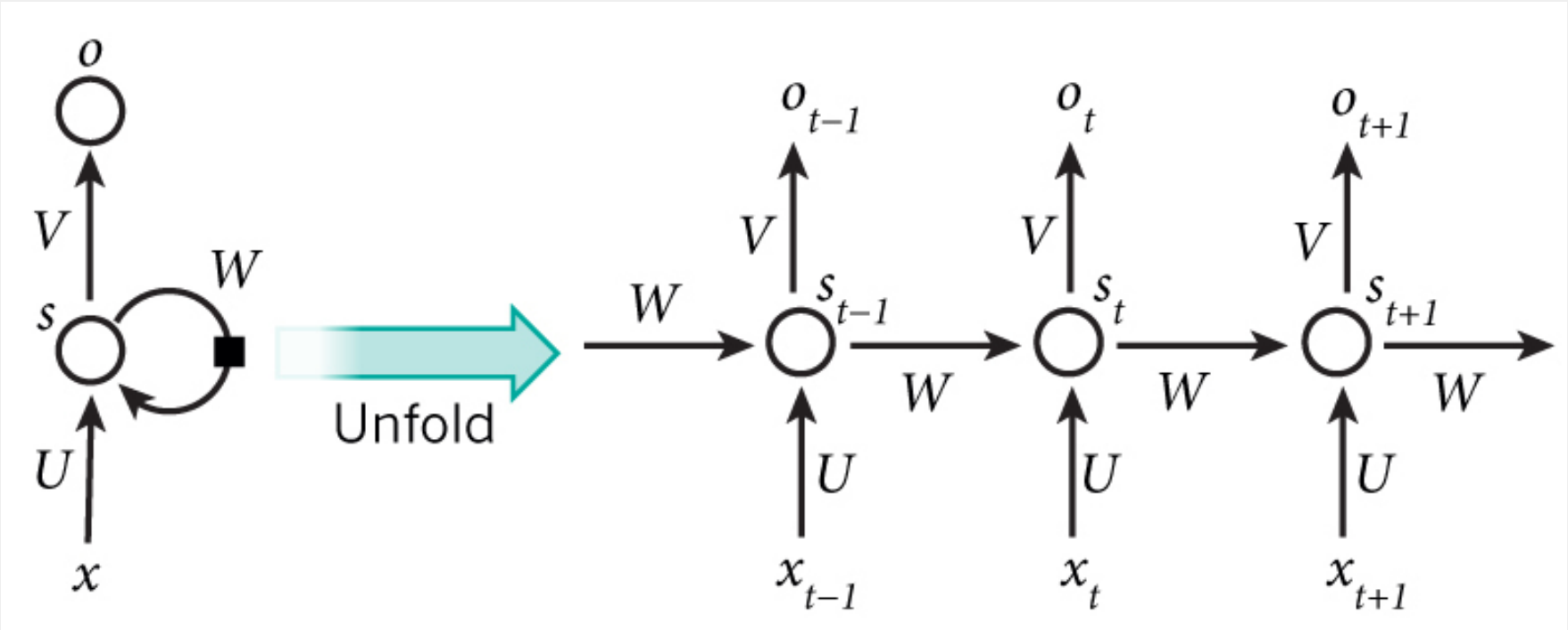


Image from `http://www.wildml.com`

Sequence to class network: text classifier

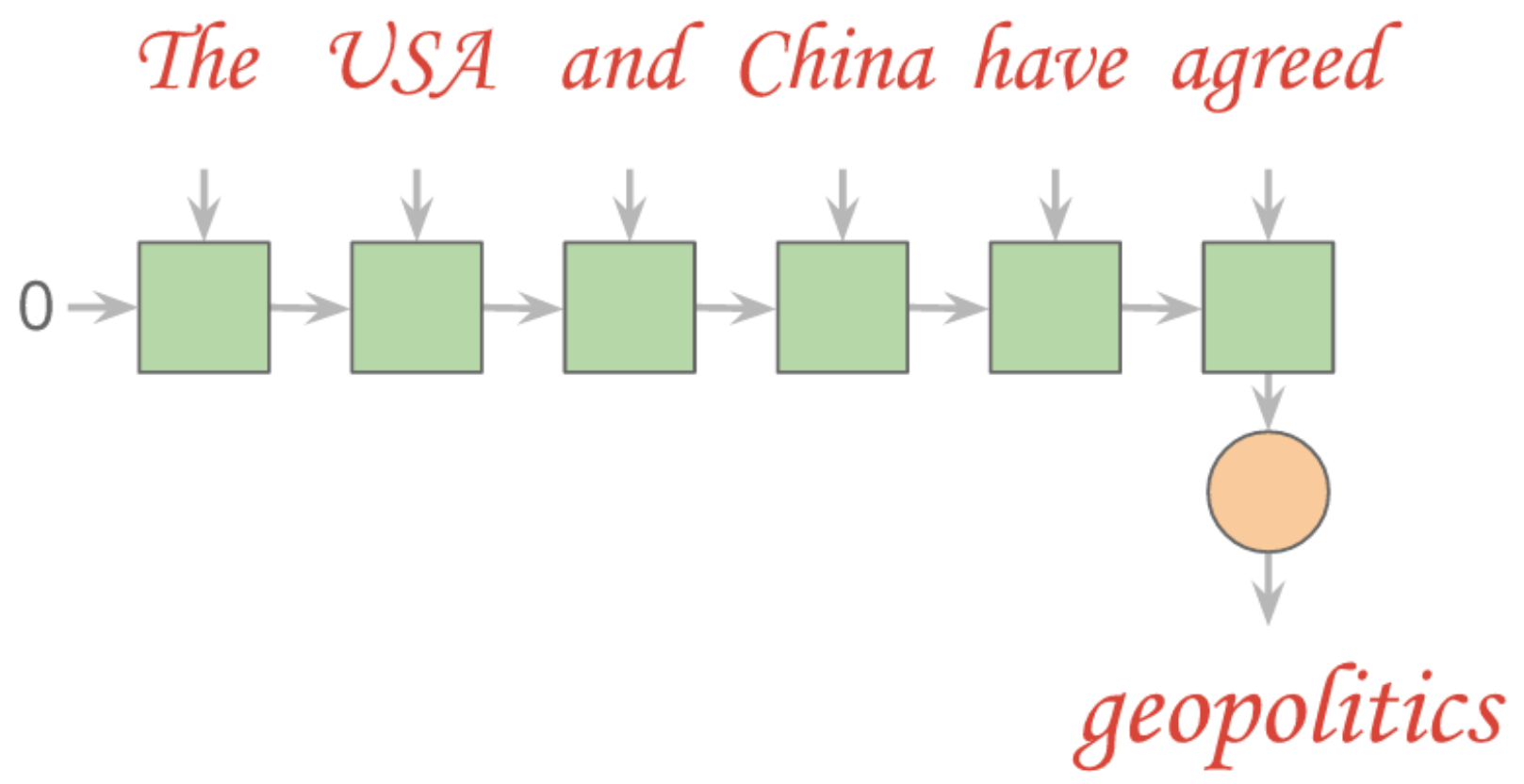


Image from Martin Gorner

Sequence to sequence network: Neural Machine Translation

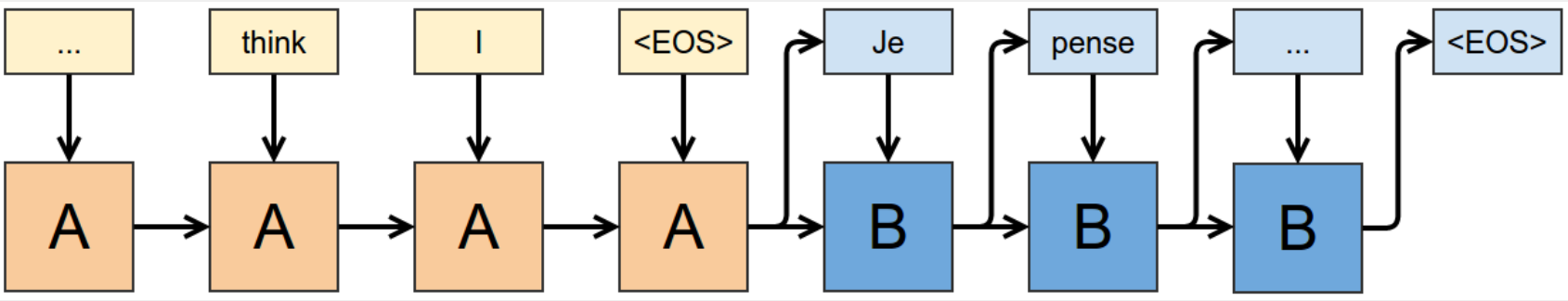


Image from `https://colah.github.io`

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

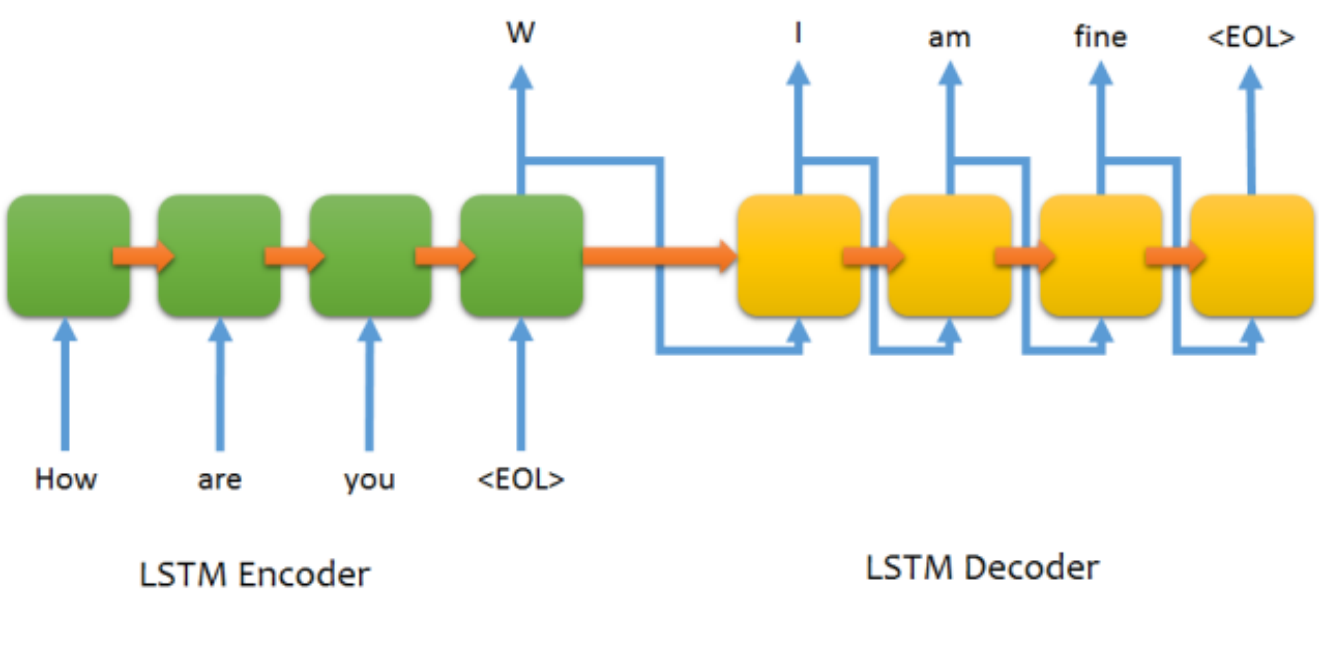


Image from `https://github.com/farizrahman4u/seq2seq`

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

Image to sequence: automatic captioning

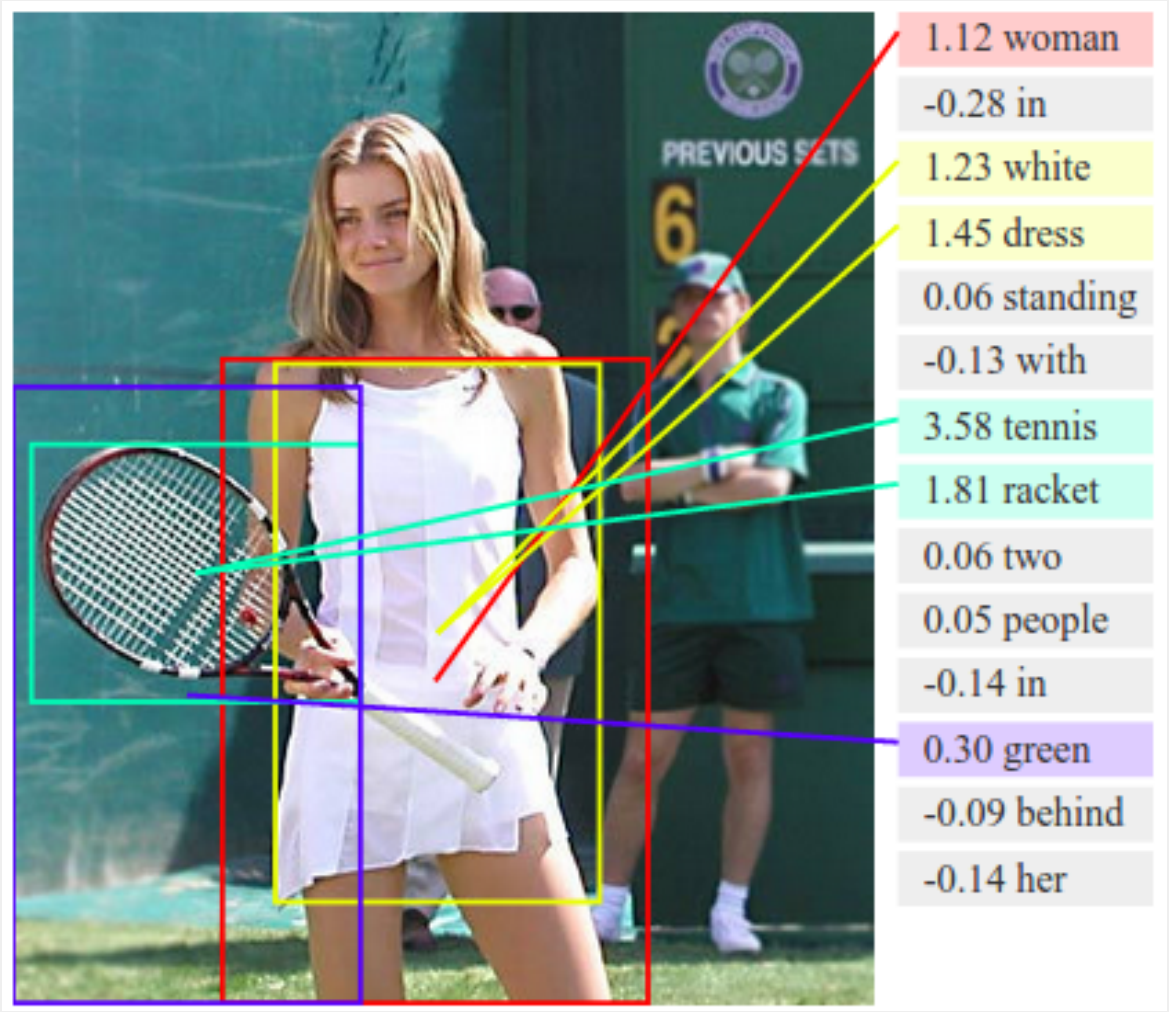


Image from `https://quantumfrontiers.com`

Image to sequence: automatic captioning

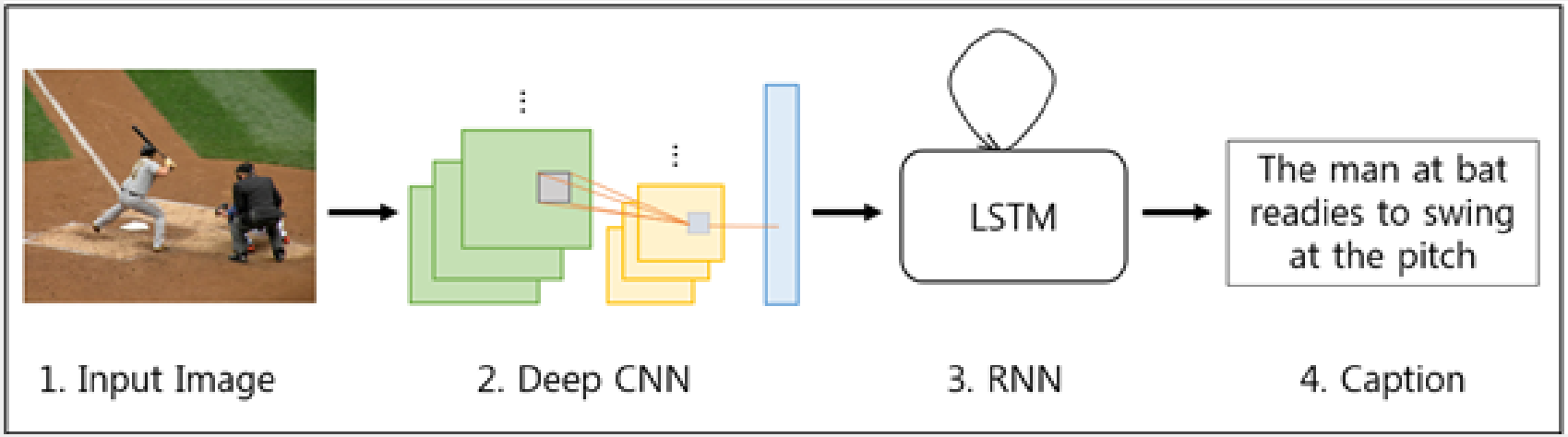


Image from `http://brain.kaist.ac.kr/`

Generative adversarial network

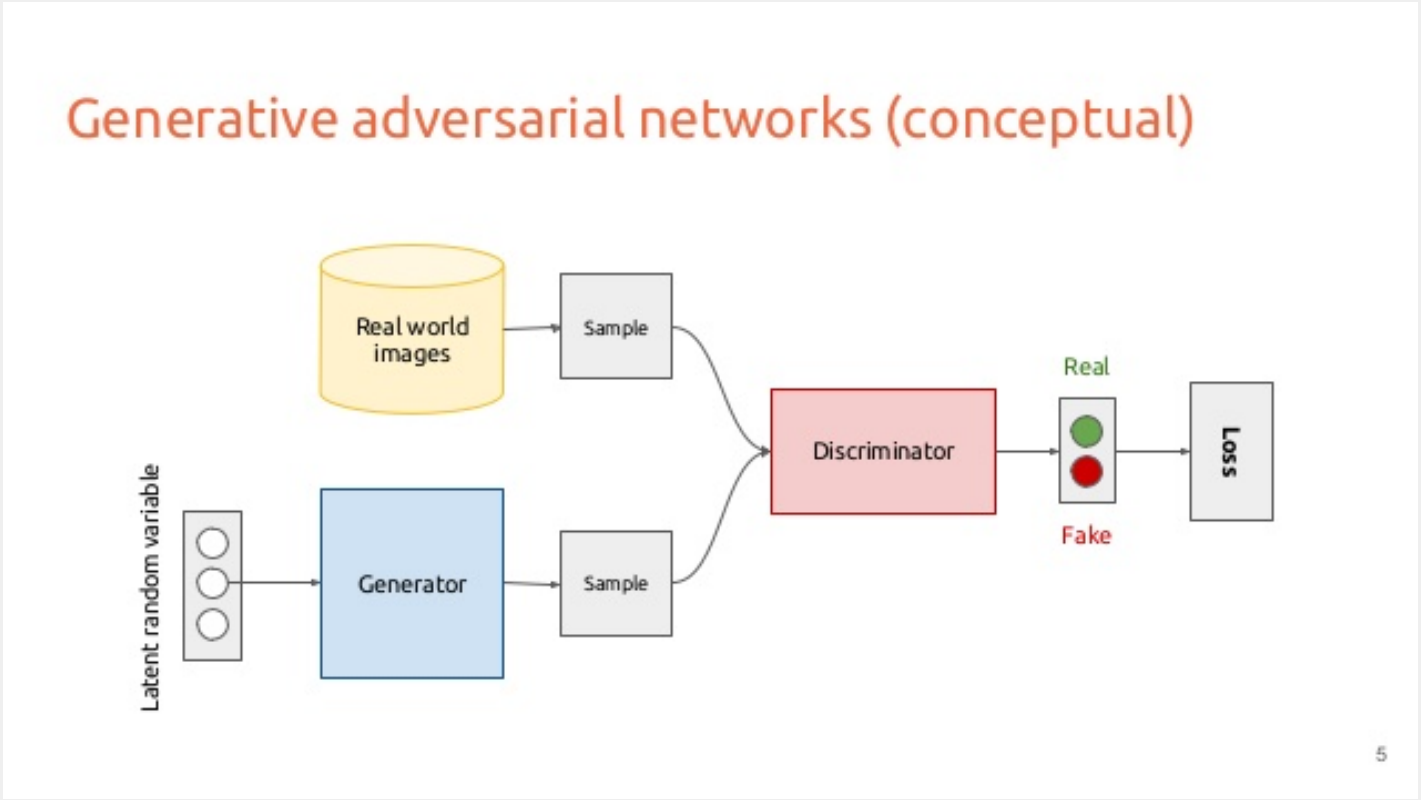


Image from `http://wiki.tum.de/`

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

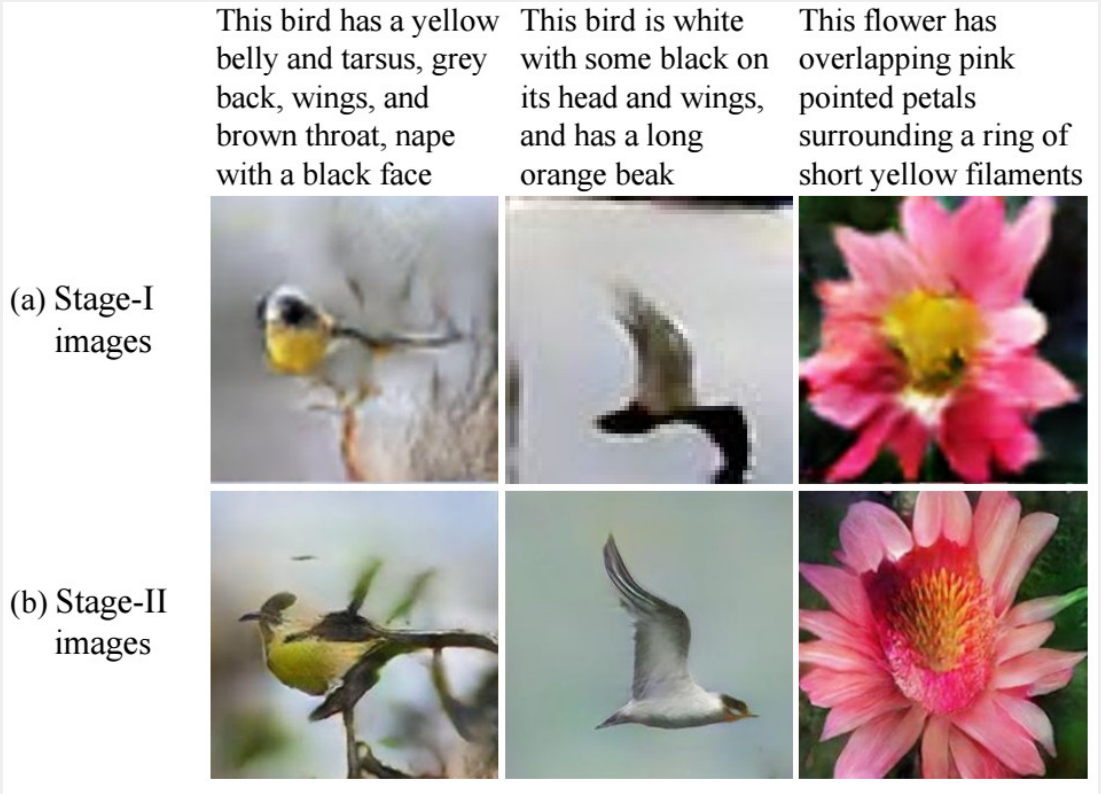
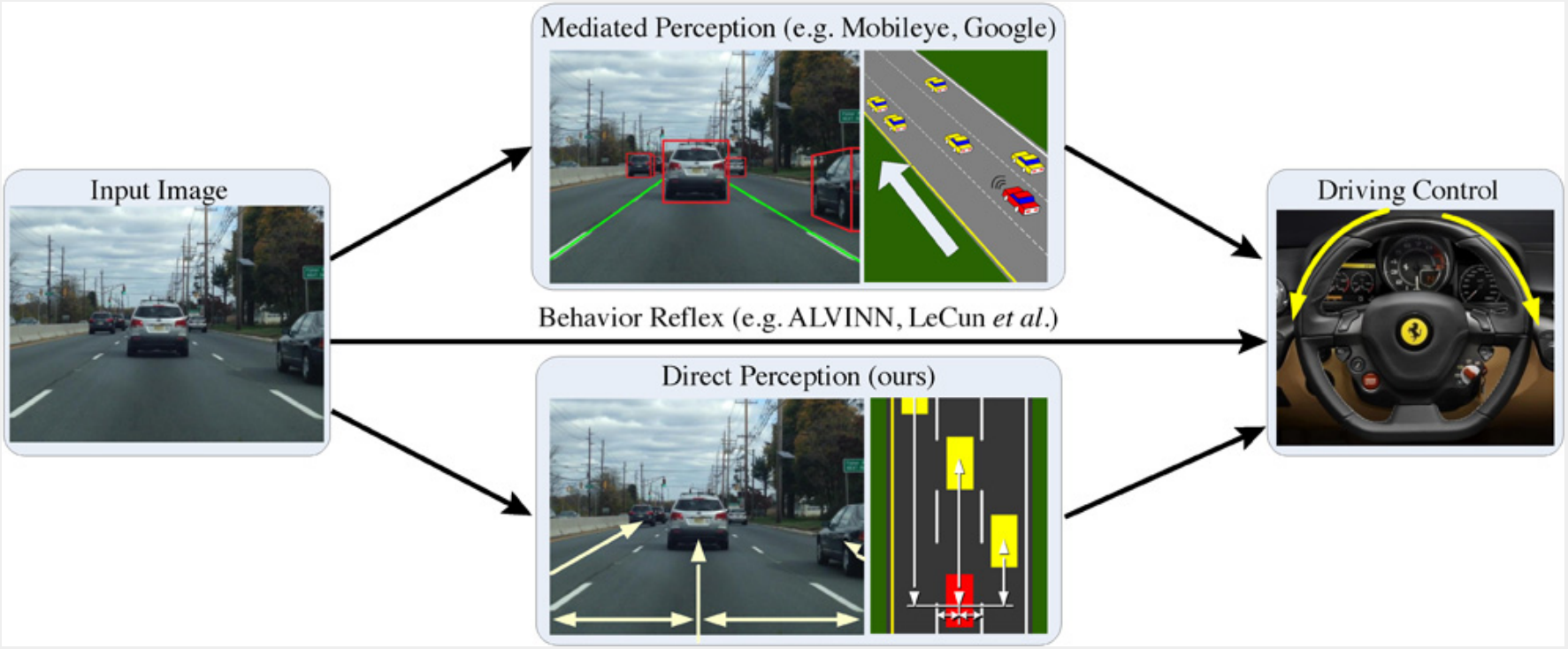


Image from `https://arxiv.org/pdf/1612.03242.pdf`



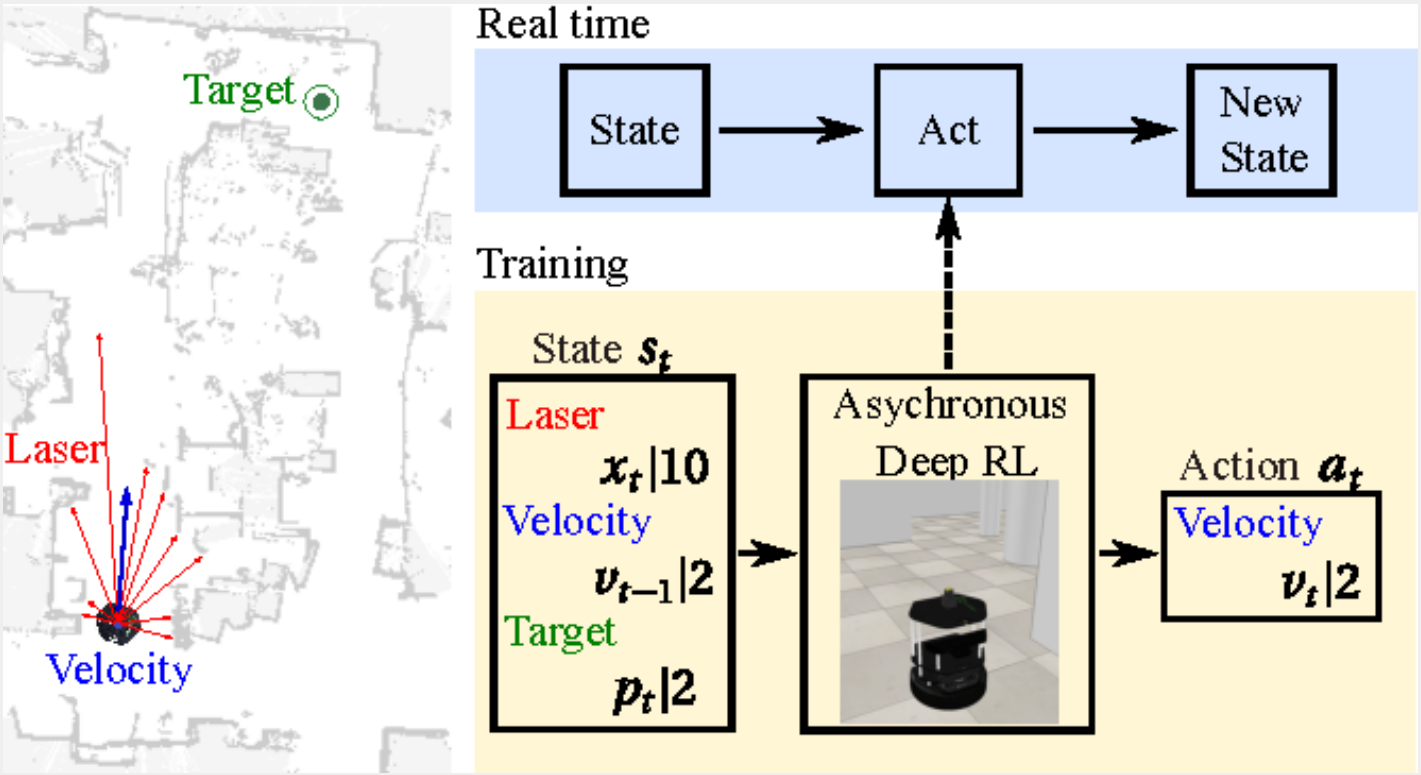
https://arxiv.org/pdf/1801.06734.pdf

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-learnir
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