Artificial Intelligence and Data Analytics for Engineers (AIDAE)

Lecture 10
July, 10th

Andrés Posada

Today's Lecturer

Anas Abdelrazeq

Marco Kemmerling

Vladimir Samsonov

Artificial Intelligence and Data Analytics for Engineers
Overview Lectures 1 – 4

1. **Introduction to Data Analytics and Artificial Intelligence in Engineering**: Organizational matters (e.g. exam, exercises, dates). Goals, Challenges, Obstacles, and Processes.

2. **Introduction into the primary programming language of the lecture, Python**: Syntax, libraries, IDEs etc. Why is Python the *lingua franca* of the Data Scientist?

3. **Data Preparation**: Cleansing and Transformation. How do real world data sets look like and why is cleaning and transformation an integral part of a Data Scientist's workflow?

4. **Data Integration**: Architectures, Challenges, and Approaches. How can you integrate various data sources into an overarching consolidating schema and why is this important?

Artificial Intelligence and Data Analytics for Engineers
Overview Lectures 5 – 8

✓ 5	**Data Representation**: Feature Extraction and Selection. How to pick relevant features for the task at hand. Manual vs automatic methods. What is the curse of dimensionality?
✓ 6	**Data-Driven Learning**: Supervised (Classification, Regression) methods and algorithms. What is an artificial neural net? What methods are there for evaluation of your model?
✓ 7	**Data-Driven Learning**: Unsupervised (Clustering) methods and algorithms. How can machines learn without labels? What methods are there for evaluation of your model?
✓ 8	**Environment-Driven Learning:** Reinforcement Learning

Artificial Intelligence and Data Analytics for Engineers
Overview Lectures 5 – 8

9	Data-Driven Learning: Artificial Neural Networks
10	**State-of-the-Art Methods**: Deep Neural Networks (e.g. GANs, CNNs, Restricted Boltzman Machines).

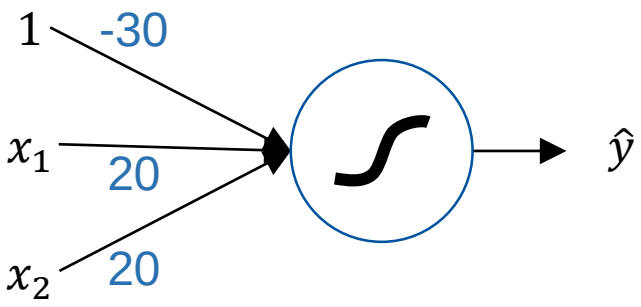
Recap Lecture 9

Single Neurons – The Linear Case

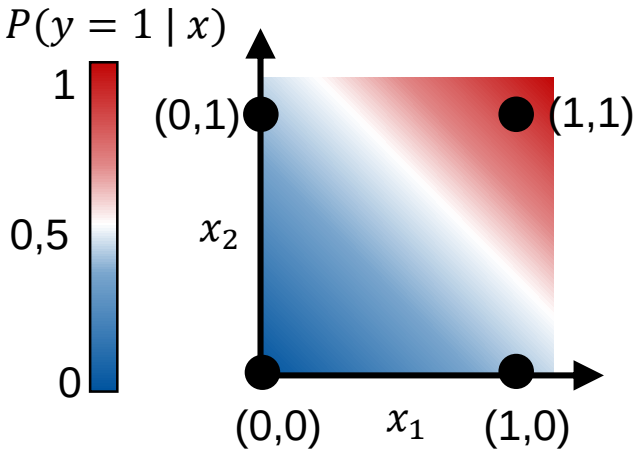
Linear separability
A single neuron is suitable for classification if the data can be separated linearly!

Simple Example: the AND-Function

- $x_1, x_2 \in \{0,1\}$
- $y = x_1\ AND\ x_2$



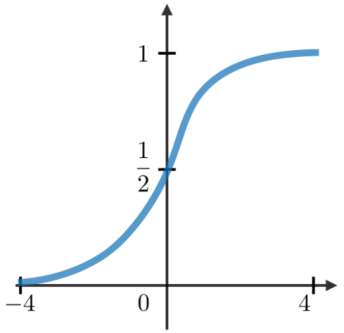
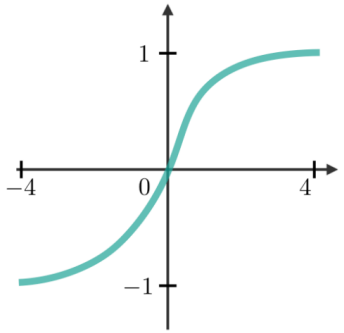
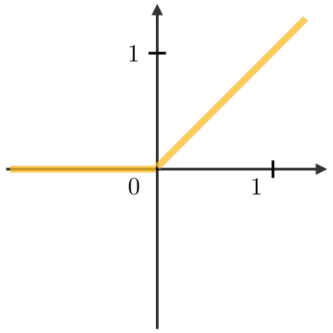
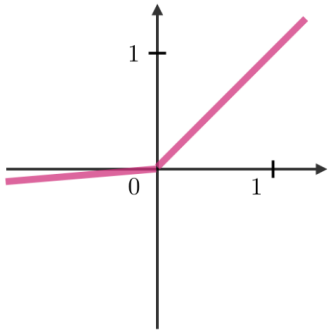
$f(x) = g(-30 + 20 \times x_1 + 20 \times x_2)$ with $g(z)$ Sigmoid-Function



x_1	x_2	$f(x)$
0	*0*	$g(-30) \approx 0$
0	*1*	$g(-10) \approx 0$
1	*0*	$g(-10) \approx 0$
1	*1*	$g(10) \approx 1$

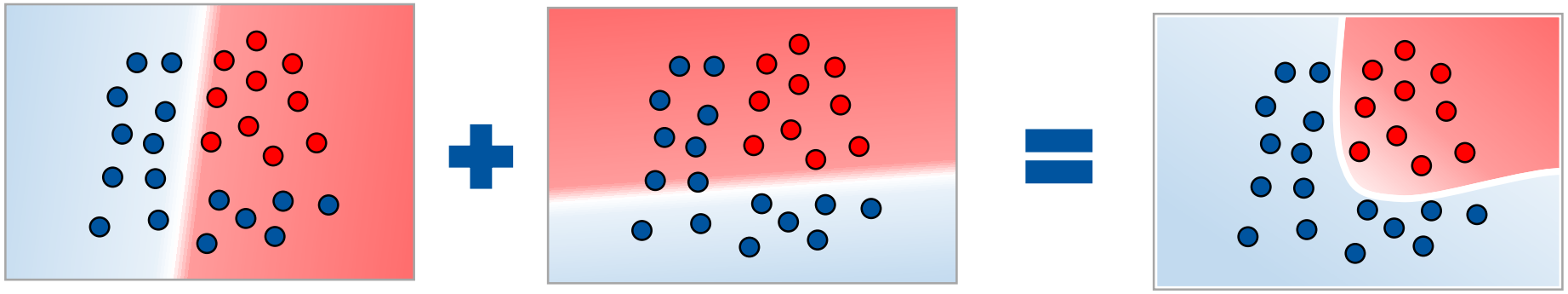
Activation Functions

- Various activation functions are possible. Important: they must be differentiable in order to be usable for learning (gradient method)
- Tangens Hyperbolicus (Tanh) behaves similar to the logistic function (Sigmoid)
- Leaky ReLU addresses the problem of vanishing gradients.
- Simple gradient calculation without numerical approximation.
- Not constantly differentiable in x=0 and unlimited.

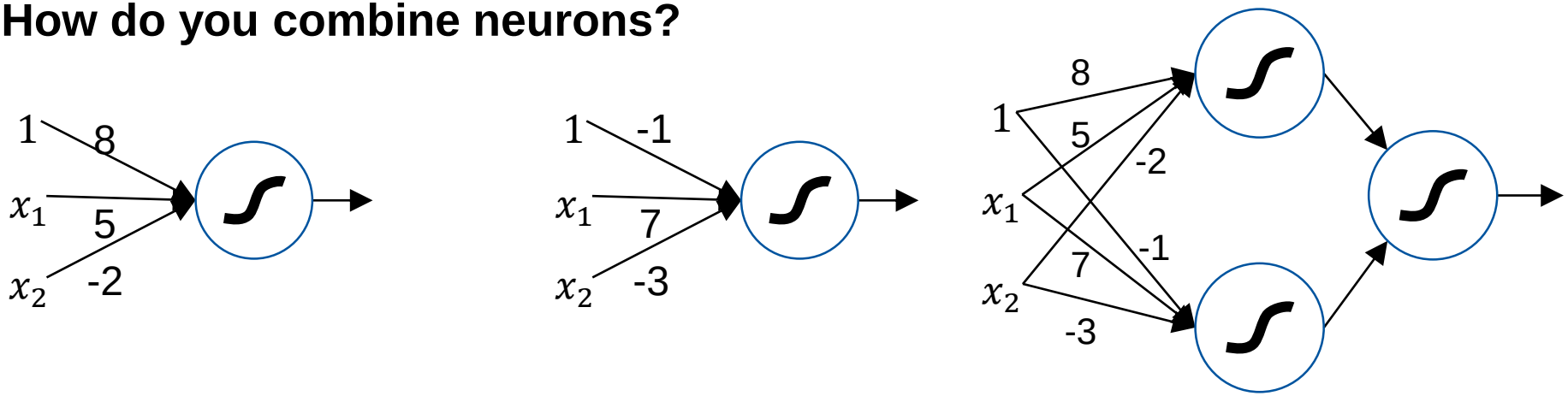
Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

The Non-Linear Case – Combination of Neurons

Illustration: Combination of regions



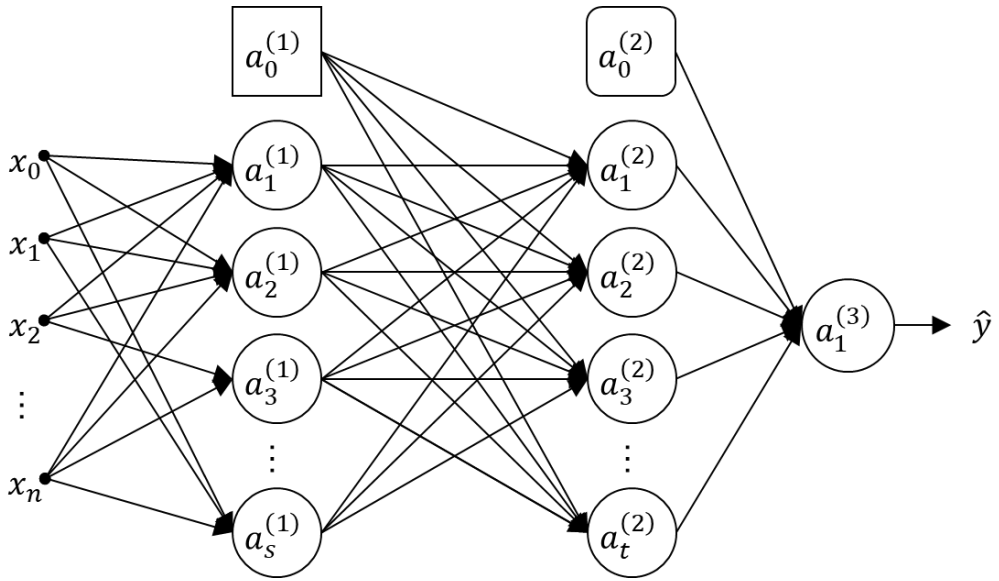
How do you combine neurons?



The Artificial Neural Net

Computation of activations

- $a_i^{(j)}$= activation of neuron i in layer j
- $W^{(j)}$ = weight matrix of layer j



Activations of hidden layer 1

- $a_1^{(1)} = g(w_{1,0}^{(1)} x_0 + w_{1,1}^{(1)} x_1 + \cdots + w_{1,n}^{(1)} x_n)$
- $a_2^{(1)} = g\left(w_{2,0}^{(1)} x_0 + w_{2,1}^{(1)} x_1 + \cdots + w_{2,n}^{(1)} x_n\right)$

…

- $a_s^{(1)} = g(w_{s,0}^{(1)} x_0 + w_{s,1}^{(1)} x_1 + \cdots + w_{s,n}^{(1)} x_n)$

Analog: Activations of hidden layer 2(here: example for $a_1^{(2)}$)

- $a_1^{(2)} = g(w_{1,0}^{(2)} a_0^{(1)} + w_{1,1}^{(2)} a_1^{(1)} + \cdots + w_{1,s}^{(2)} a_s^{(1)})$
- If the net has s_j neurons in layer j and s_{j+1}neurons in layer $j+1$ hat, then the weight matrix $W^{(j+1)}$ has dimensions $s_{j+1} \times (s_j + 1)$

Gradient Descent

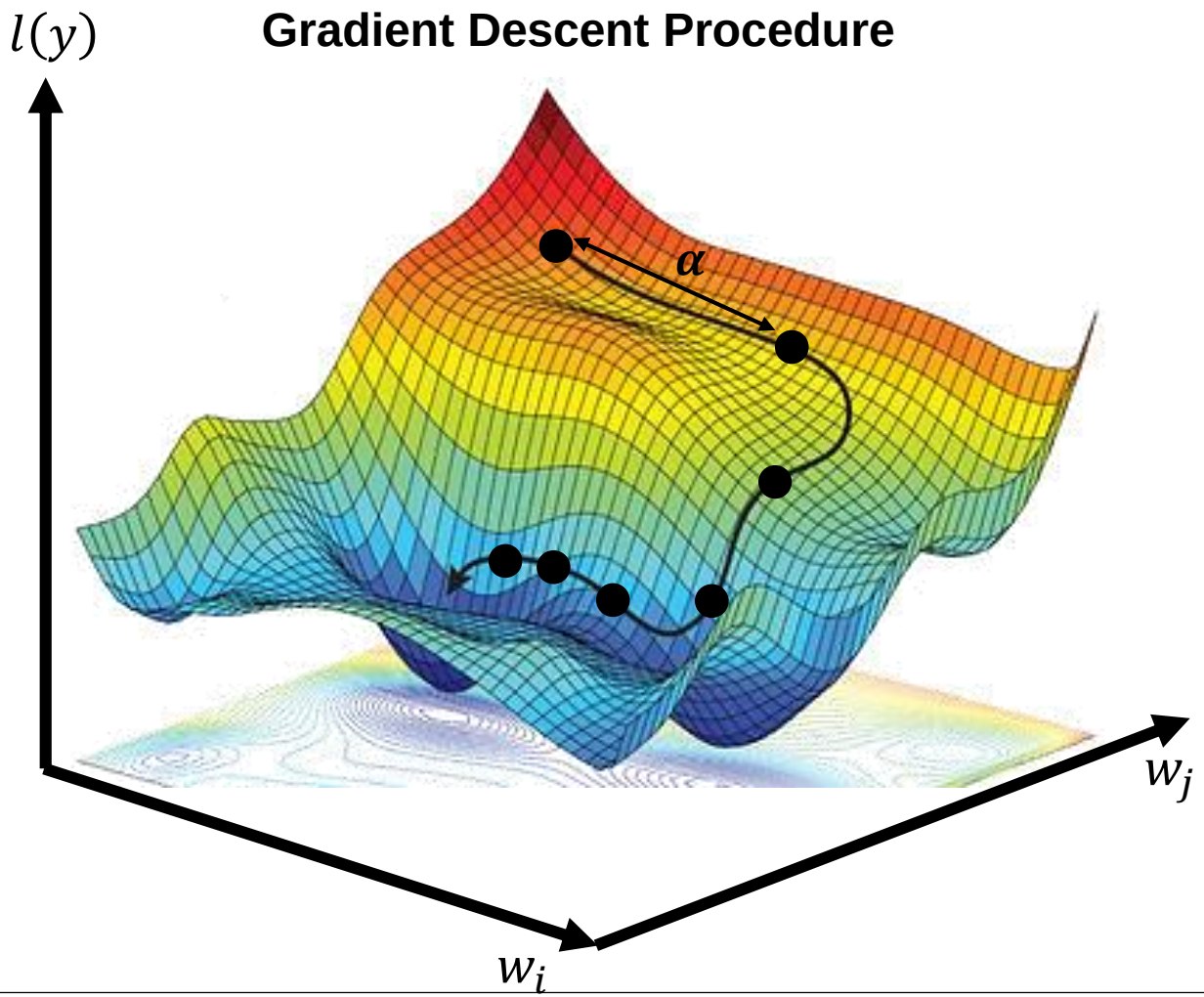
Minimization of Cost Function

Starting point Initialization of weights

Descent along the largest negative gradient

Weight adjustment by increment α

Reaching a local minimum



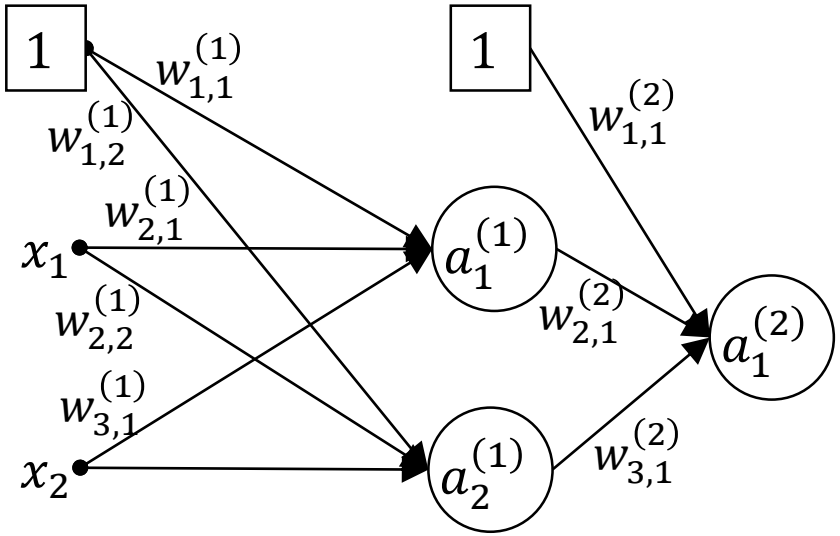
Gradient Descent

Gradient of the cost function

- Using the chain rule you can compute the gradient
- Example: Gradient w.r.t. weight $w_{1,1}^{(1)}$:

$$\frac{\partial L(f(x))}{\partial w_{1,1}^{(1)}} = \frac{\partial L(f(x))}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w_{1,1}^{(1)}}$$

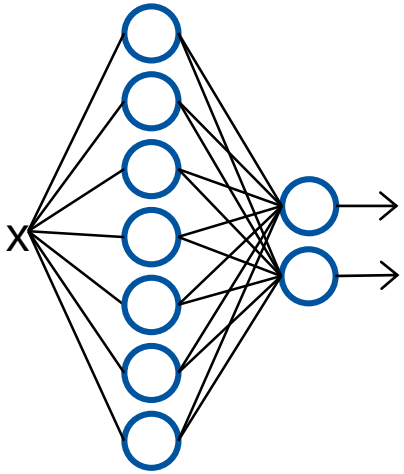
- The partial derivative $\frac{\partial L(f(x))}{\partial w_{1,1}^{(1)}}$ is composed of the product of above derivatives.



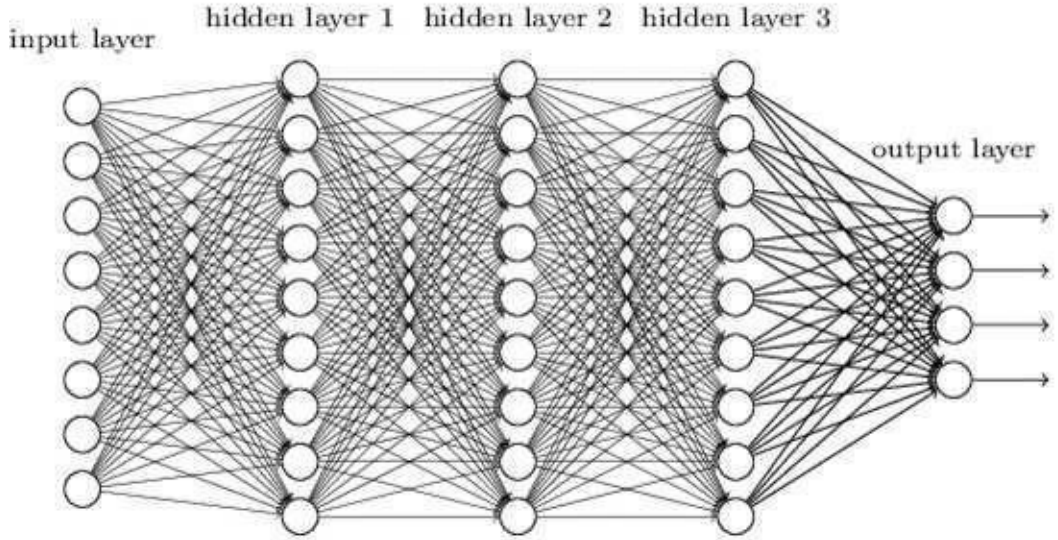
Please note: $g(z)$ is a sigmoid-function

Lecture 10: Deep Neural Networks state of the art

Outlook



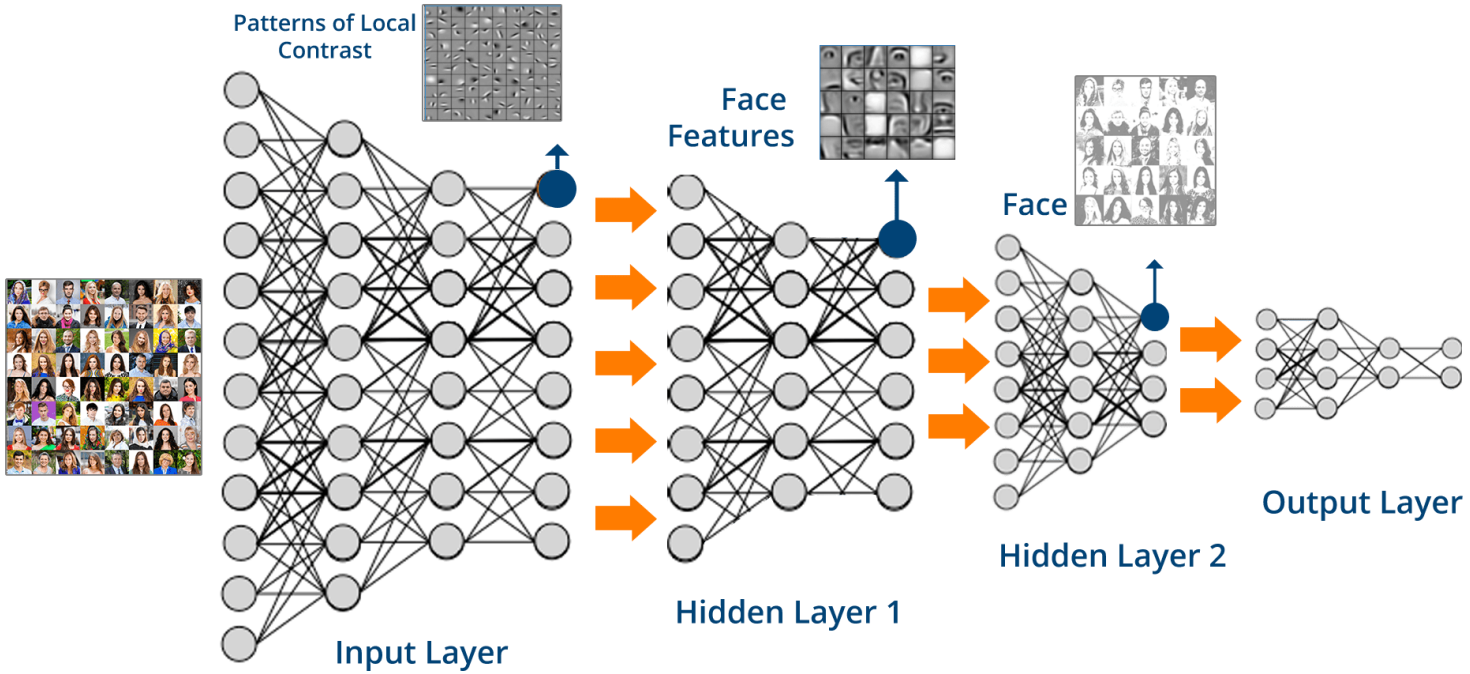
Today's topic



Deep Neural Networks

What is Deep Learning?

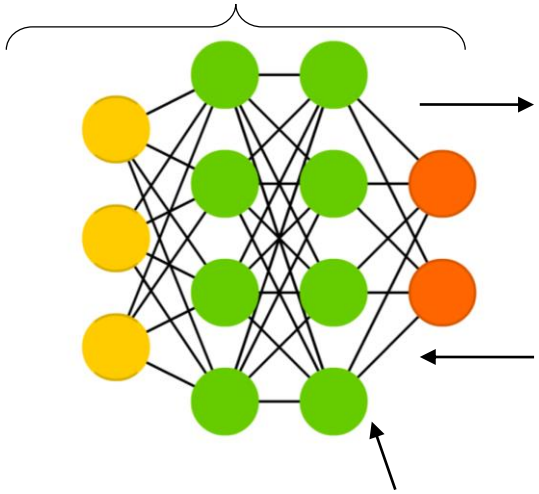
! Deep learning is a subset of machine learning algorithms, in which multi-layered neural networks are used to model the dependencies between input data and a desired output.



Source: Nvidia deep learning

Anatomy a deep neural network

Architecture:
Deep Neural networks are composed of multiple interconnected layers (groups of units). Its architecture refers to the way the units and layers are connected.



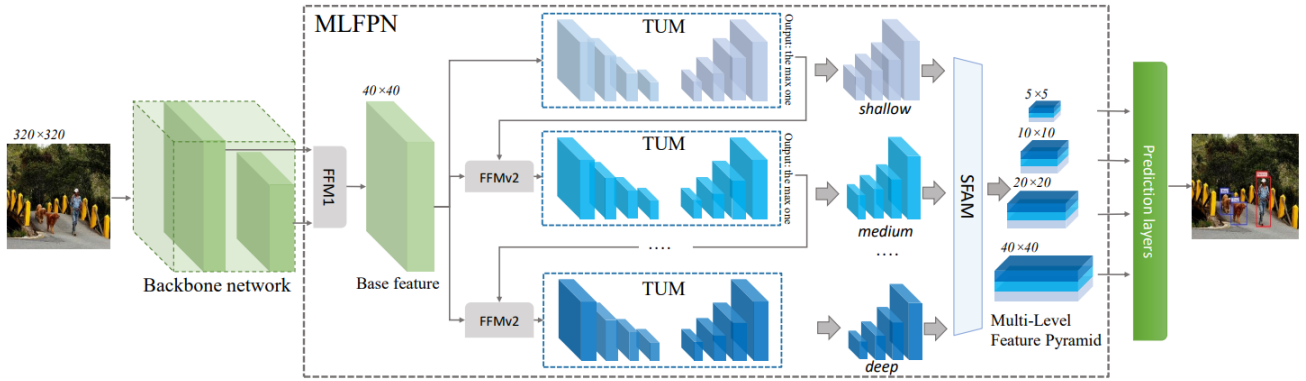
In the **Forward pass**, the input values are used and the operations of the units are computed systematically to obtain the output values of the network.

In the **Backward pass**, the obtained output and the desired output are used by the **Loss function** to obtain the errors/gradients and back propagate them through the network. This is latter used to modify the parameters/weights of the units.

Neurons/Units: basic computational units, it receives inputs and computes certain output based on internal parameters and a predefined function.

Anatomy a deep neural network

Architectures:
Bigger neural networks are composed of functional groups or modules. Said modules are composed of multiple layers connected for a specific purpose.

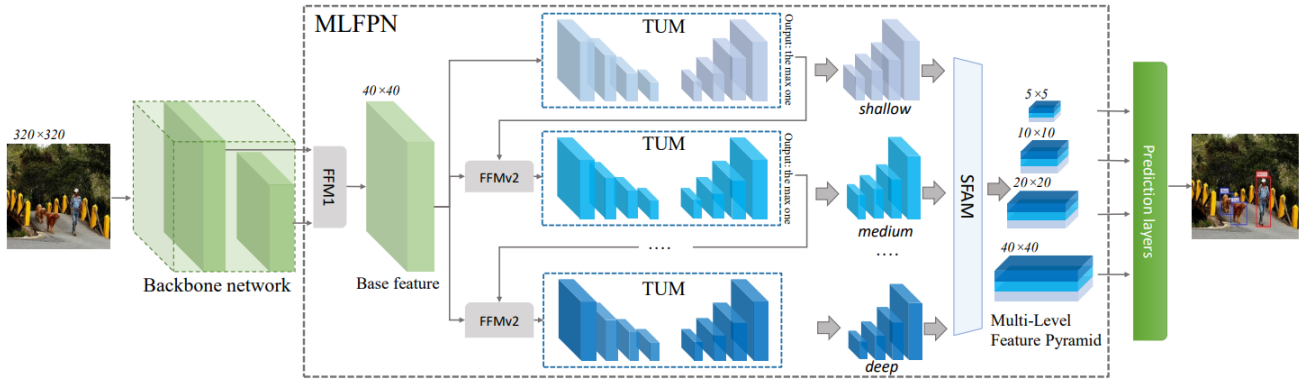


Source: M2Det

Backbone: feature extraction modules
FFM: feature fusion module
TUM: Thinned U-shape Modules
MLFPN: Multi-Level Feature Pyramid Network

Anatomy a deep neural network

Architectures:
Bigger neural networks are composed of functional groups or modules. Said modules are composed of multiple layers connected for a specific purpose.

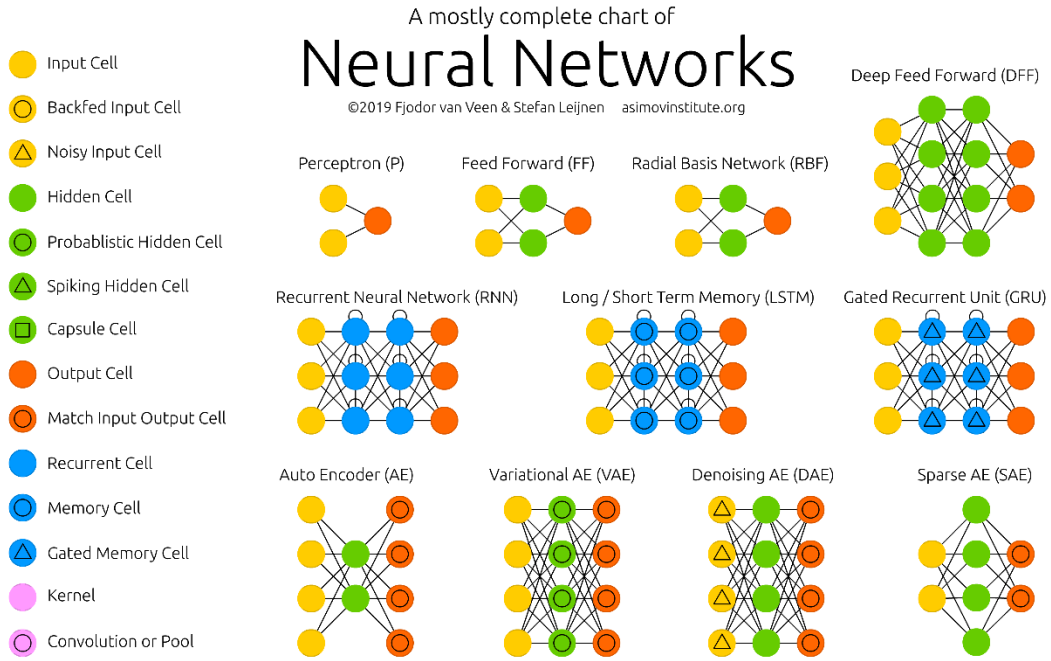


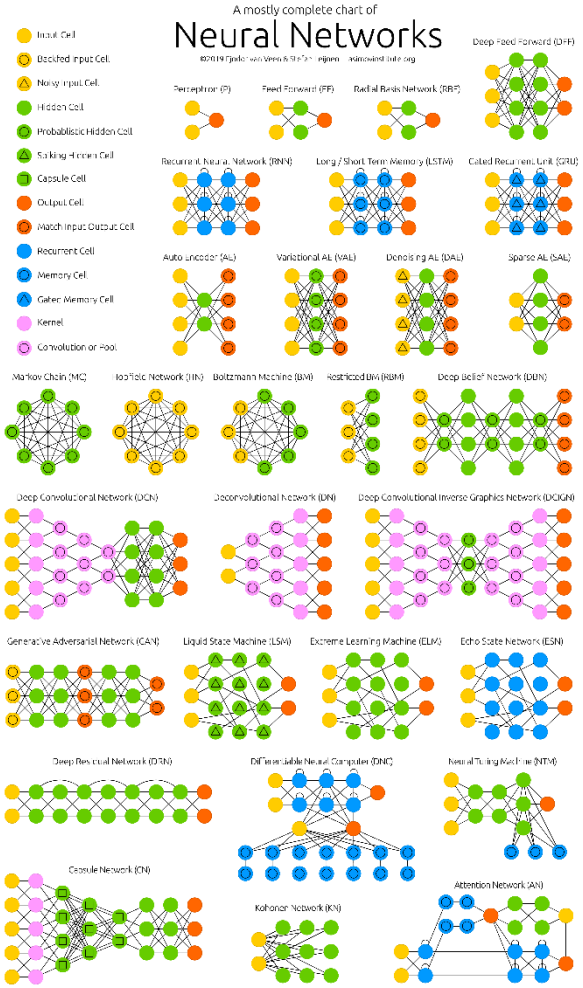
Source: M2Det

Backbone: feature extraction modules
FFM: feature fusion module
TUM: Thinned U-shape Modules
MLFPN: Multi-Level Feature Pyramid Network

Some types of Deep neural networks

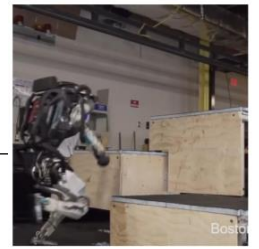
Depending on the type of units and architectures of the neural networks, they can be used for different tasks, such as object detection, segmentation, classification, and translation.





http://www.asimovinstitute.org/wp-content/uploads/2019/04/NeuralNetworkZo19High.png

Artificial Intelligence in Context of Human History

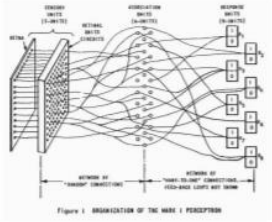


Robots on two legs.

Robots on four wheels.

Lee Sedol vs AlphaGo, 2016
GAN, ImageNet+AlexNet

Kasparov vs Deep Blue 1997
CNN, LSTM, RNN, Backprop



Frank Rosenblatt, Perceptron (1957,1962)
Early description of single layer and multi layer neural network

Perceptron

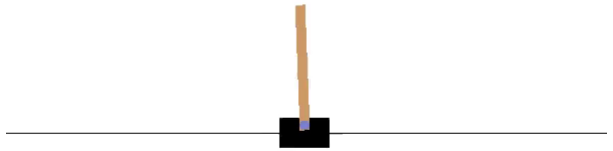
TensorFlow 2.0 and PyTorch 1.5.1

- Eager execution by default (imperative programming)
- Keras integration + promotion
- Cleanup (API, etc.)
- TensorFlow.js
- TensorFlow Lite
- TensorFlow Serving

- TorchScript (graph representation)
- Quantization
- PyTorch Mobile (experimental)
- TPU support

Deep Reinforcement Learning – Frameworks

1. Open AI Stable Baselines
   - ➢ Good for documentation
   - ➢ Easy to get started and use



2. TersorForce
   - ➢ 4 high level abstractions of an Environment, Runner, Agent, Model
   - ➢ Code is complicated

3. Dopamine (Google)
   - ➢ Relatively new entrant to the RL framework space
   - ➢ Emphasises configuration as code
   - ➢ Promotes plugability and reuse

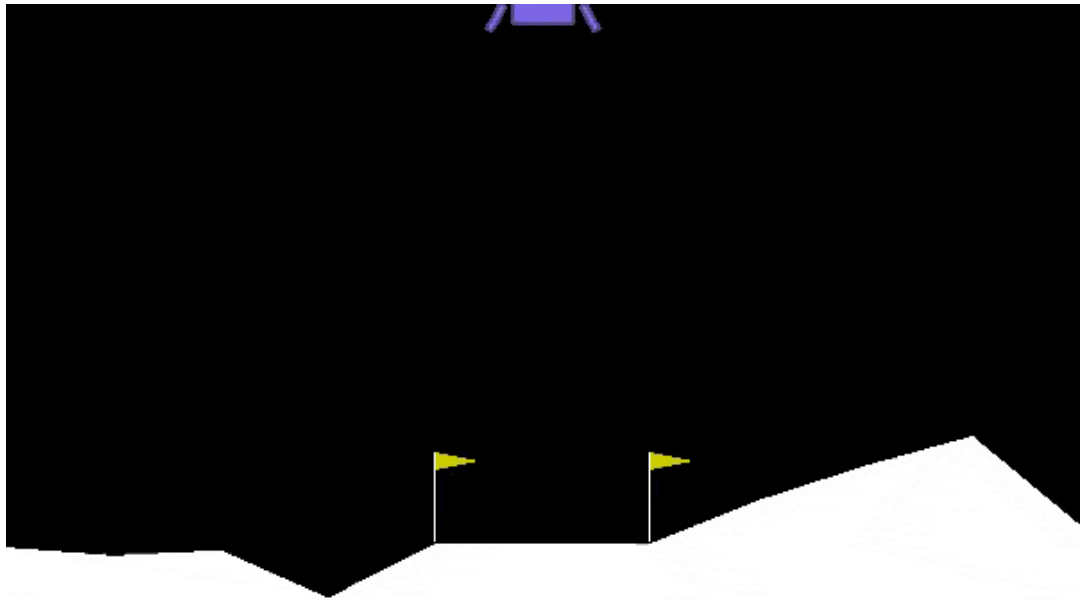
4. RLLib via ray-project
5. Keras RL
   - ➢ Quick prototyping

Frameworks

- Stable Baselines

- A2C, PPO, TRPO, DQN, ACKTR, ACER and DDPG
- Good documentation (and code commenting)
- Easy to get started and use

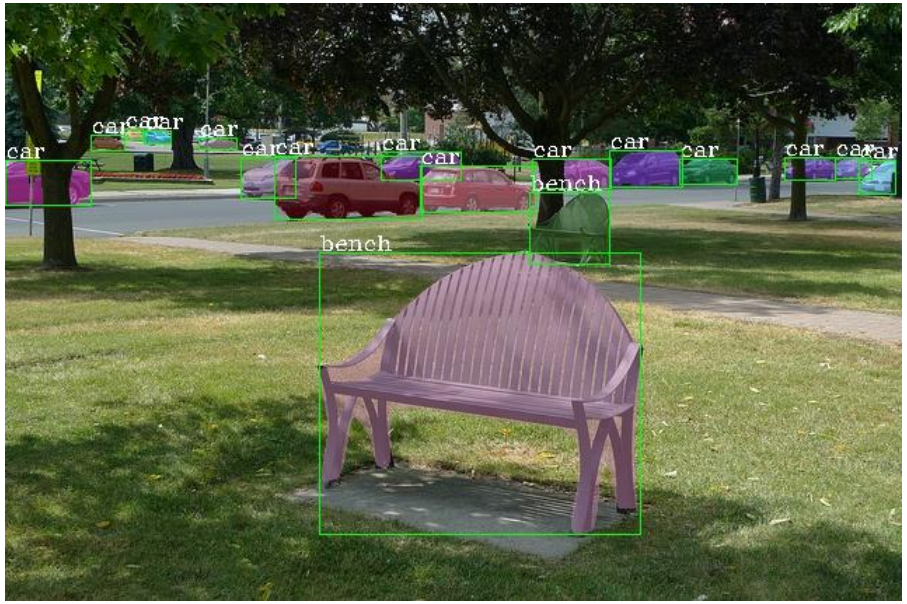
```python
import gym
from stable_baselines import DQN

# Create environment
env = gym.make('LunarLander-v2')

# Instantiate the agent
model = DQN('MlpPolicy', env, learning_rate=1e-3)
# Train the agent
model.learn(total_timesteps=int(2e5))


# Enjoy trained agent
obs = env.reset()
for i in range(1000):
    action, _states = model.predict(obs)
    obs, rewards, dones, info = env.step(action)
    env.render()
```



Frameworks

- MM Detection

- Modular
- Efficient
- State of the art (up to date)
- Easy to use

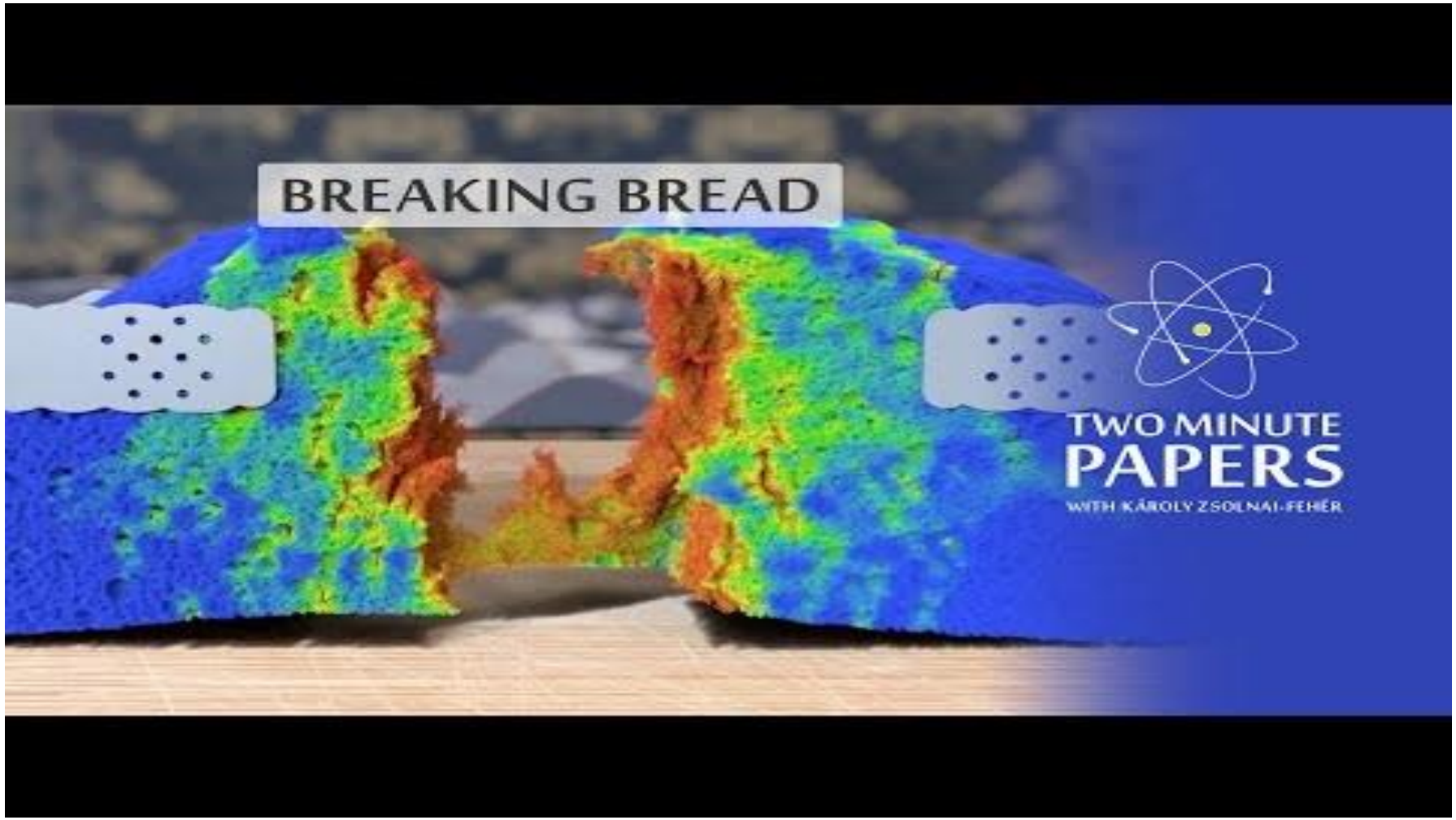


MMDetection

Breast cancer detection

Deep Fakes

DNN in simulation

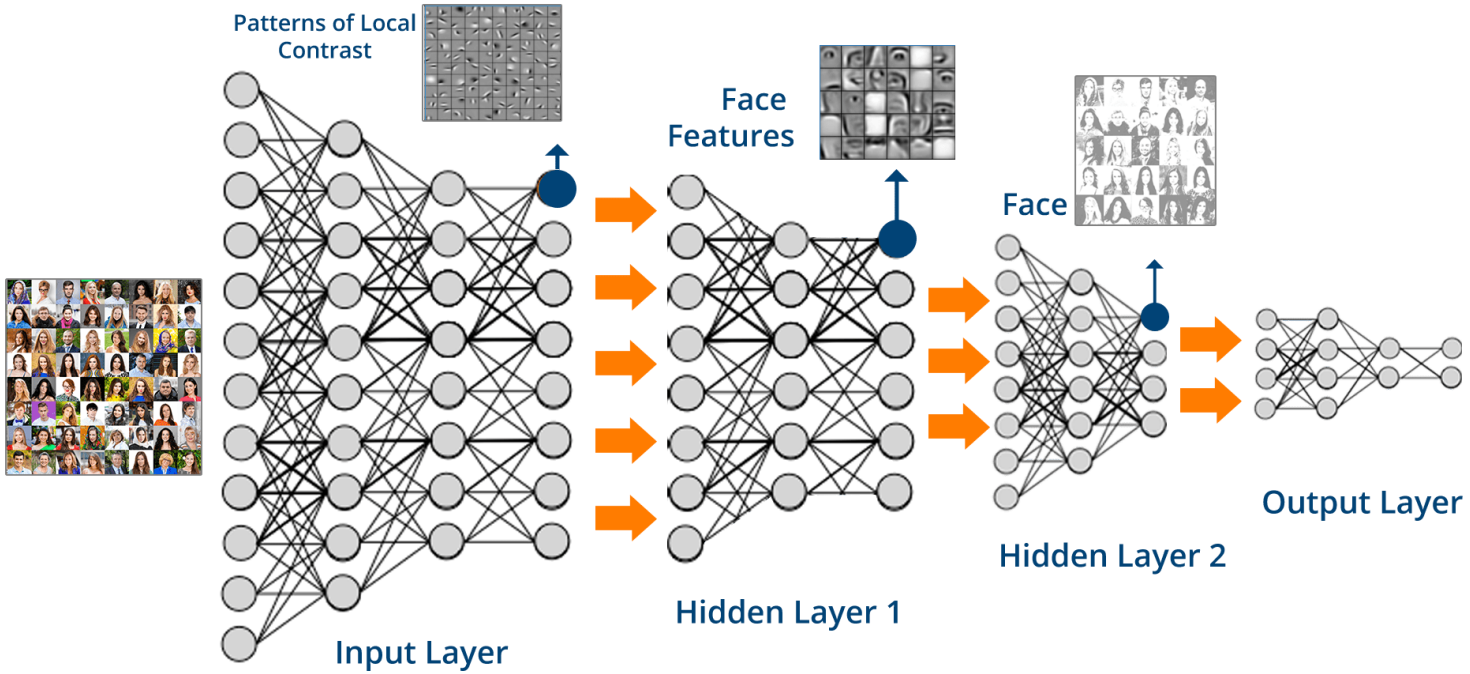


DRL

Deep Neural Networks in Computer vision

What are CNNs?

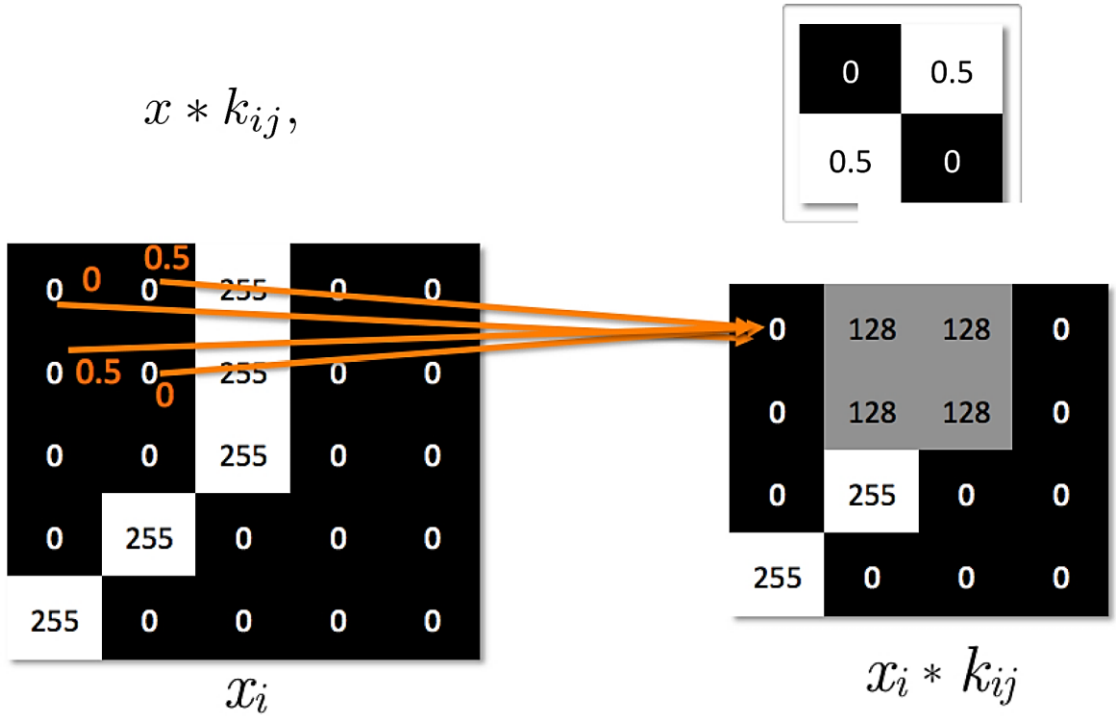
CNNs (convolutional neural networks): are NN that use convolutions as a base computational unit. These networks are used mainly for computer vision.



Source: Nvidia deep learning

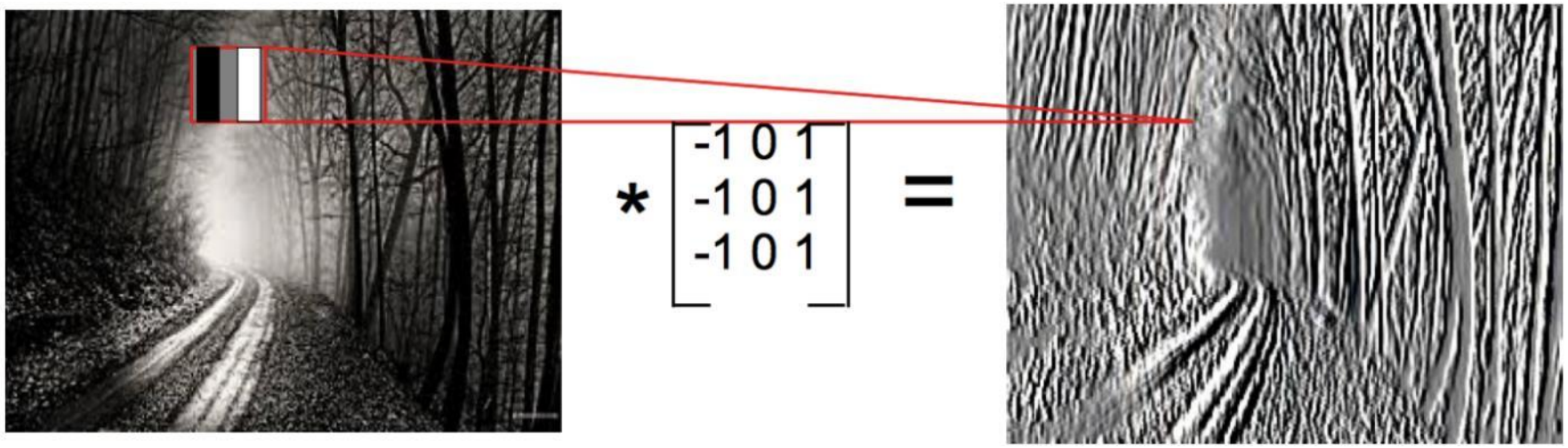
CNN units

Units: the basic computational units of the CNN are the (discrete) convolutions.



Example: http://scs.ryerson.ca/~aharley/vis/conv/

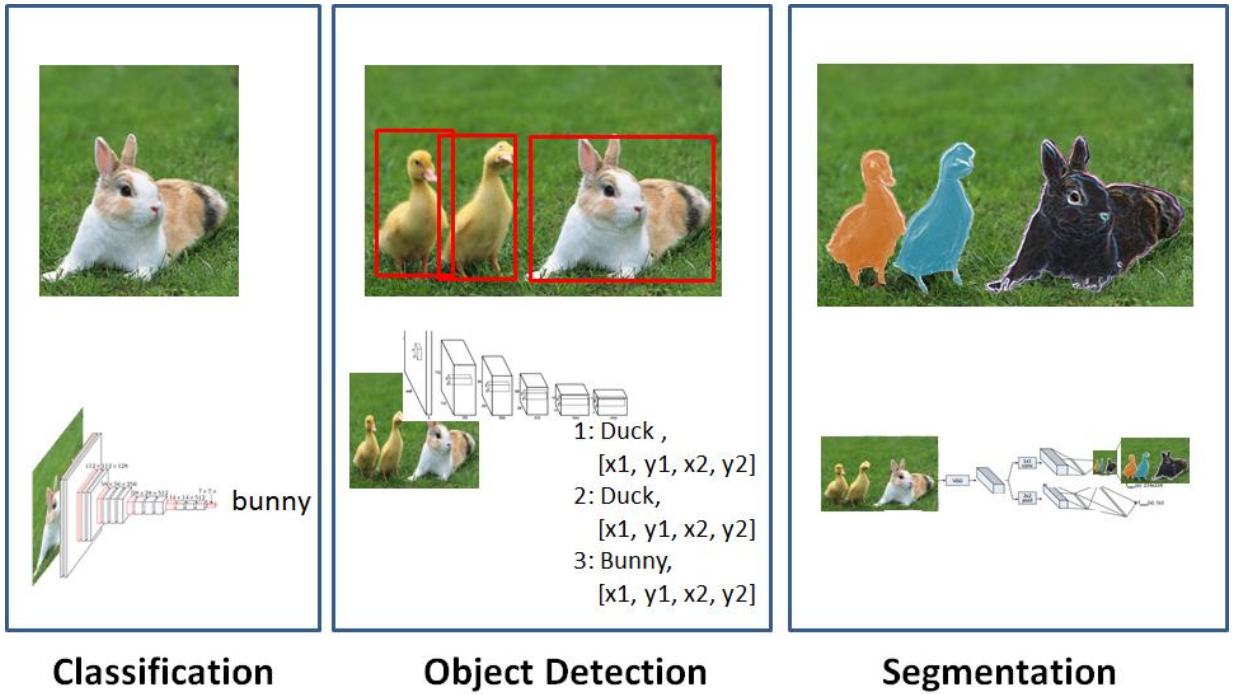
Edge extraction by convolution



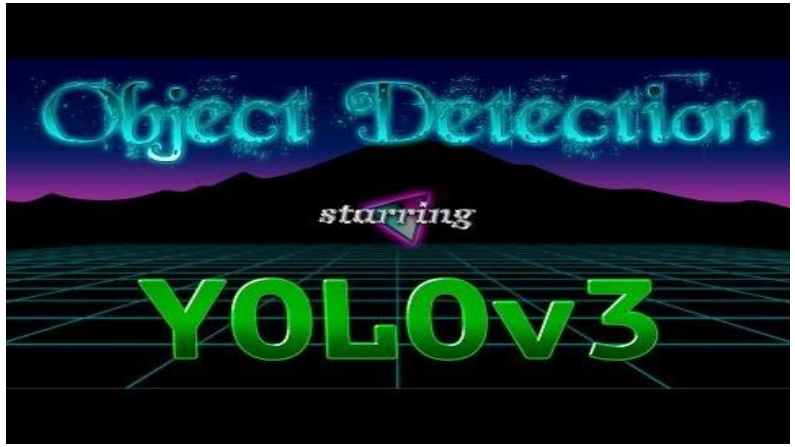
$\mathbf{x}$ $\mathbf{k}$ $\mathbf{x} * \mathbf{k}$

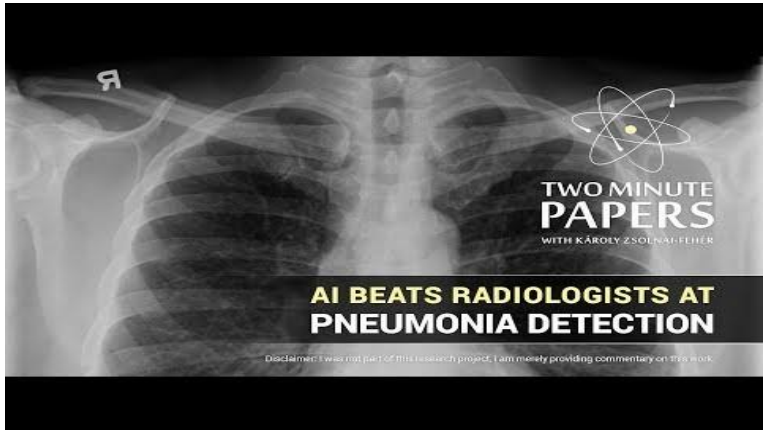
Problems

- CNNs are mainly used for:
  - Classification (cat? dog?)
  - Object detection (where is the cat?)
  - Segmentation (which pixels are the cat?)



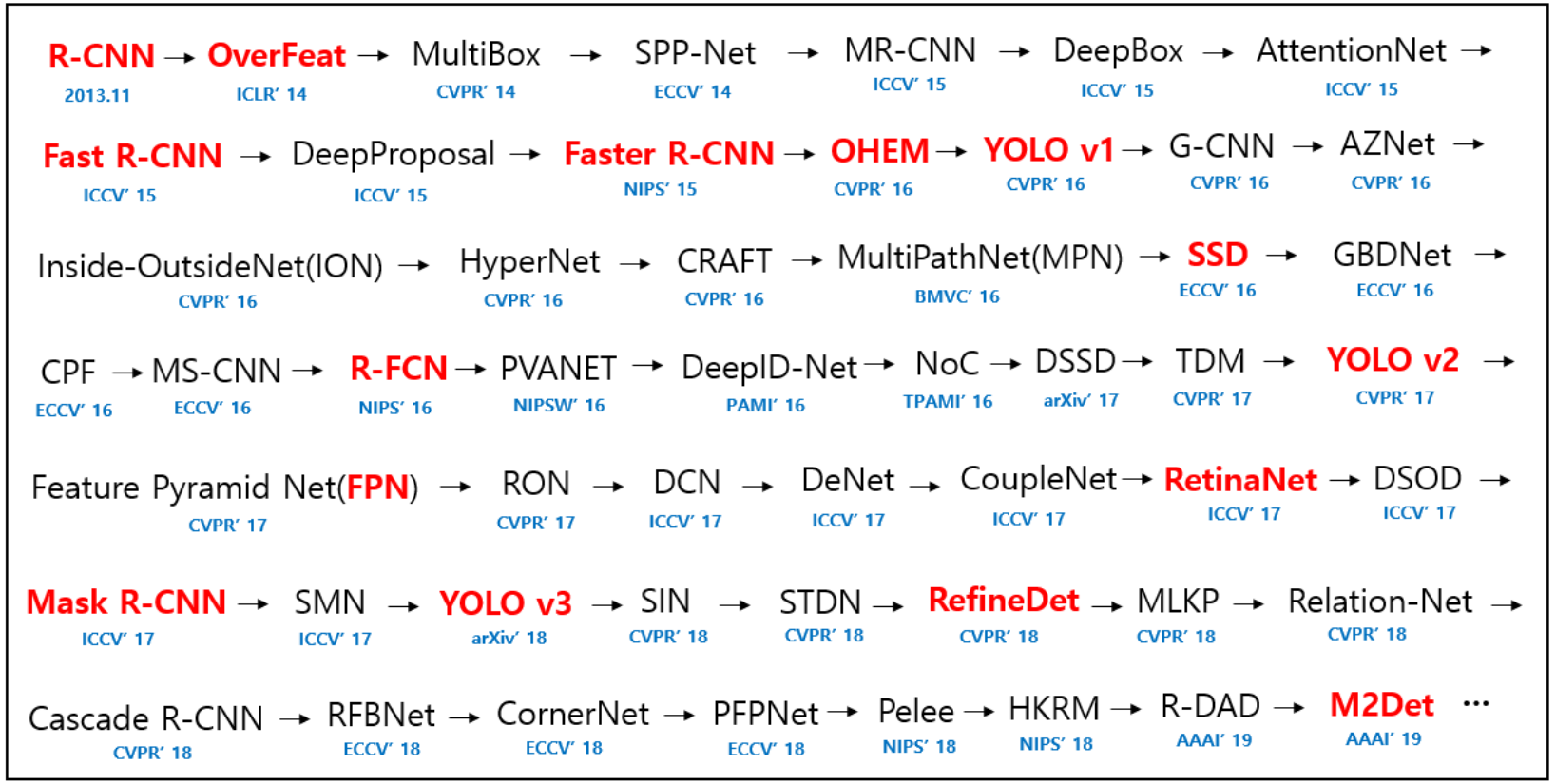
Examples


Object Detection
starring
YOLOv3


TWO MINUTE
PAPERS
AI BEATS RADIOLOGISTS AT
PNEUMONIA DETECTION

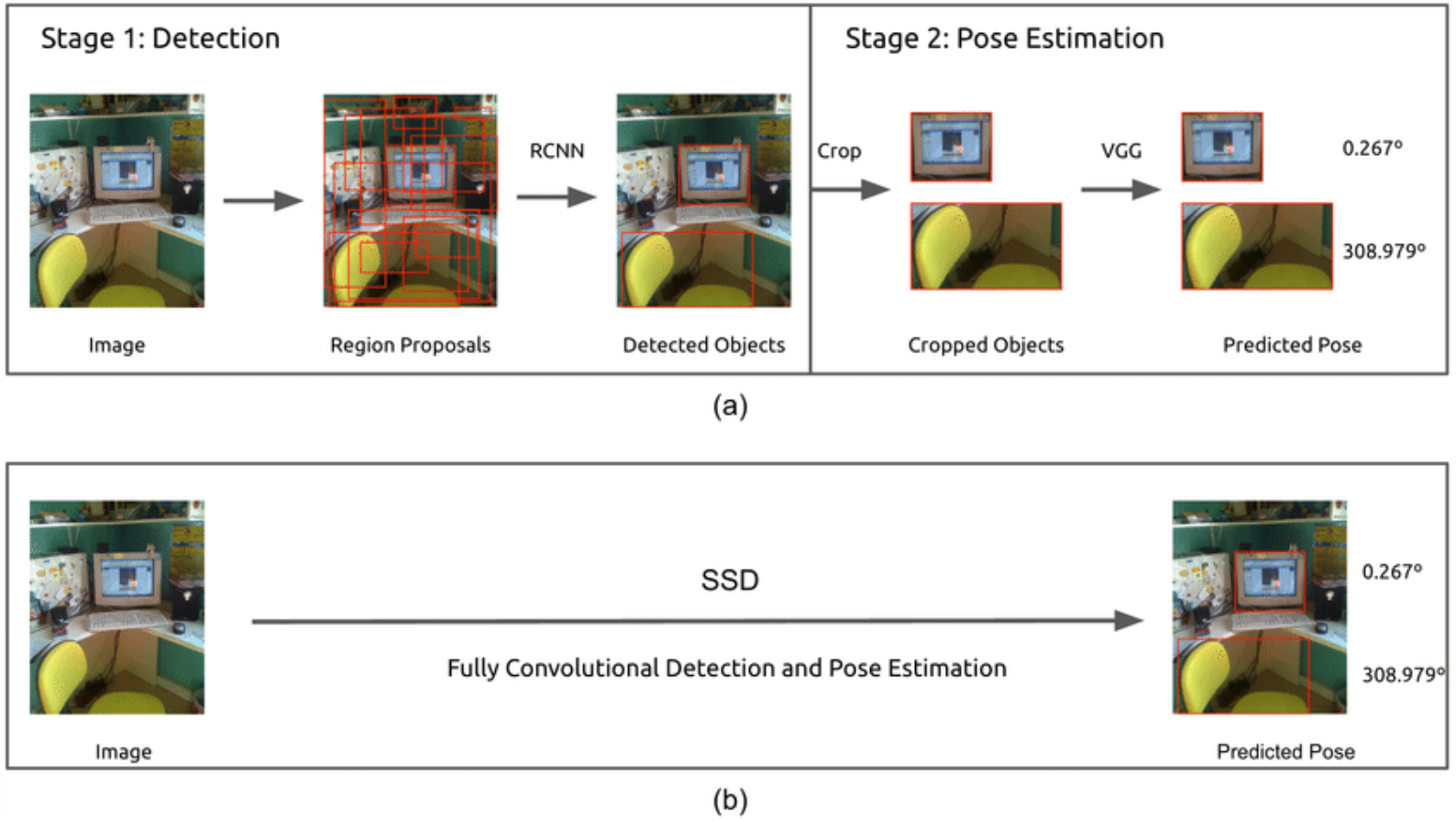
Object Detection

History



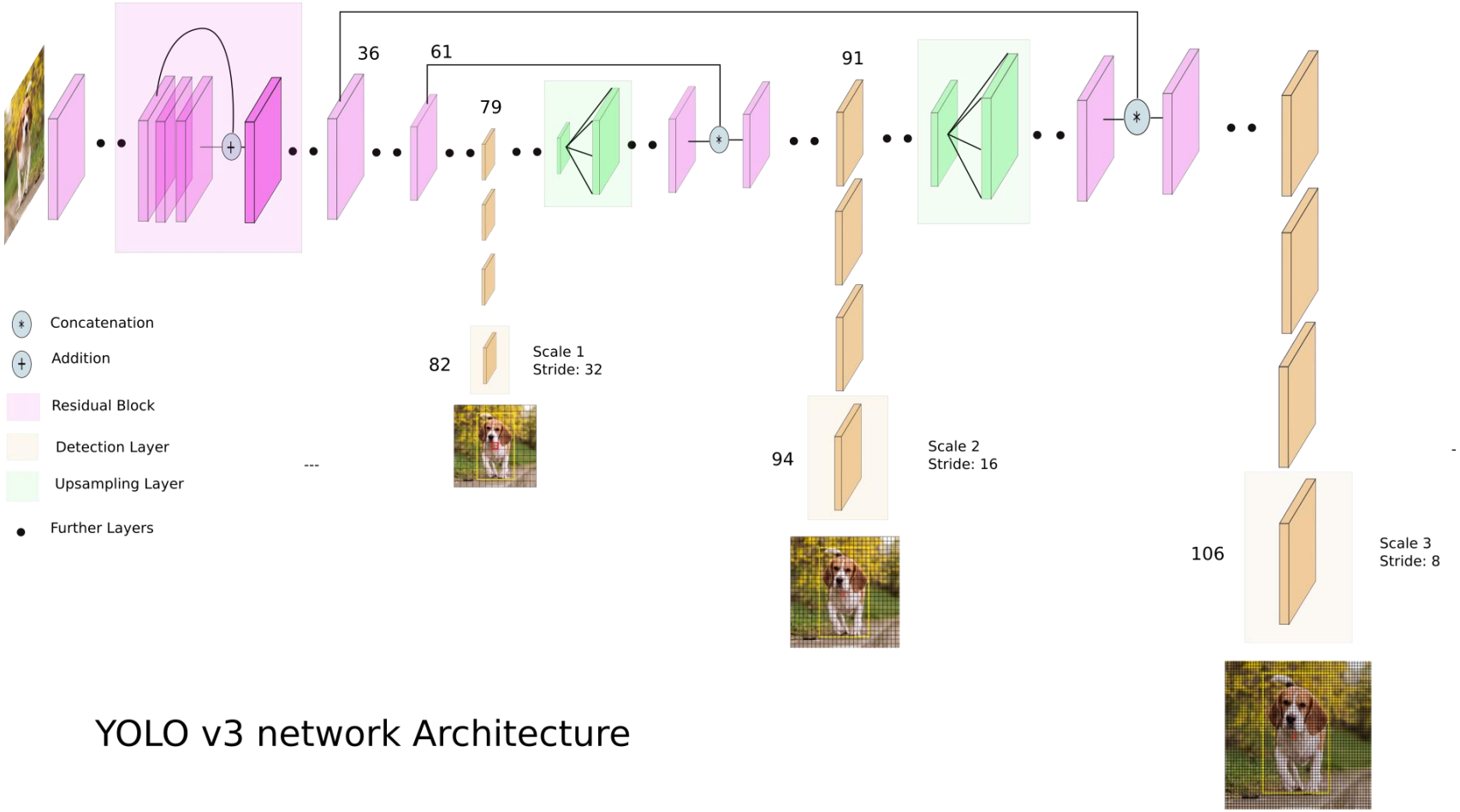
https://github.com/hoya012/deep_learning_object_detection#2019

Single-Stage vs. Two-Stage approach



Object Detection

YOLOv3



YOLO v3 network Architecture

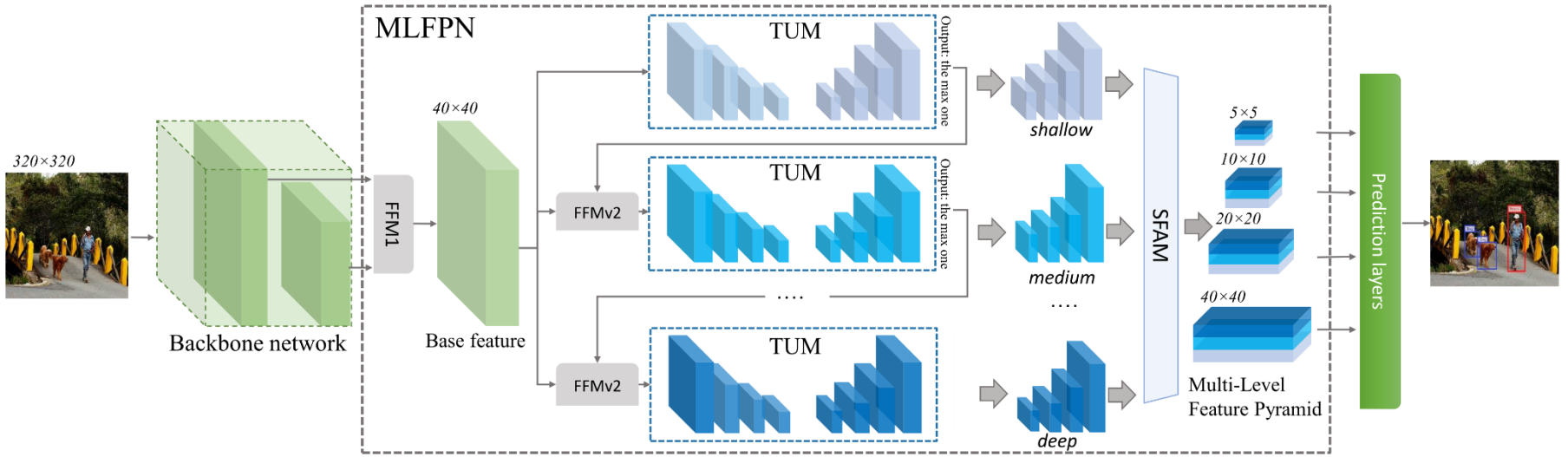
Object Detection

YOLOv3

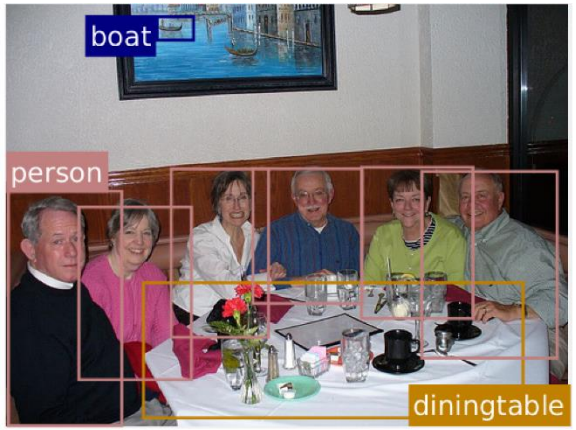
- Backbone net: extract features
- Use upsampling + skip connections to preserve detailed information for deeper layers
- predictions on 3 different grid sizes → finer grid for smaller objects
- predict several anchors for each grid cell
- For each anchor predict
  - Objectness score
  - Bounding box
  - Object class
- Apply non-maximum suppression on object predictions

M2Det

- Use Multi-level Feature Pyramid Network (MLFPN) to extract features from input image



Semantic vs. Instance segmentation



Object Detection

Semantic Segmentation

Instance Segmentation

Fully Convolutional Network

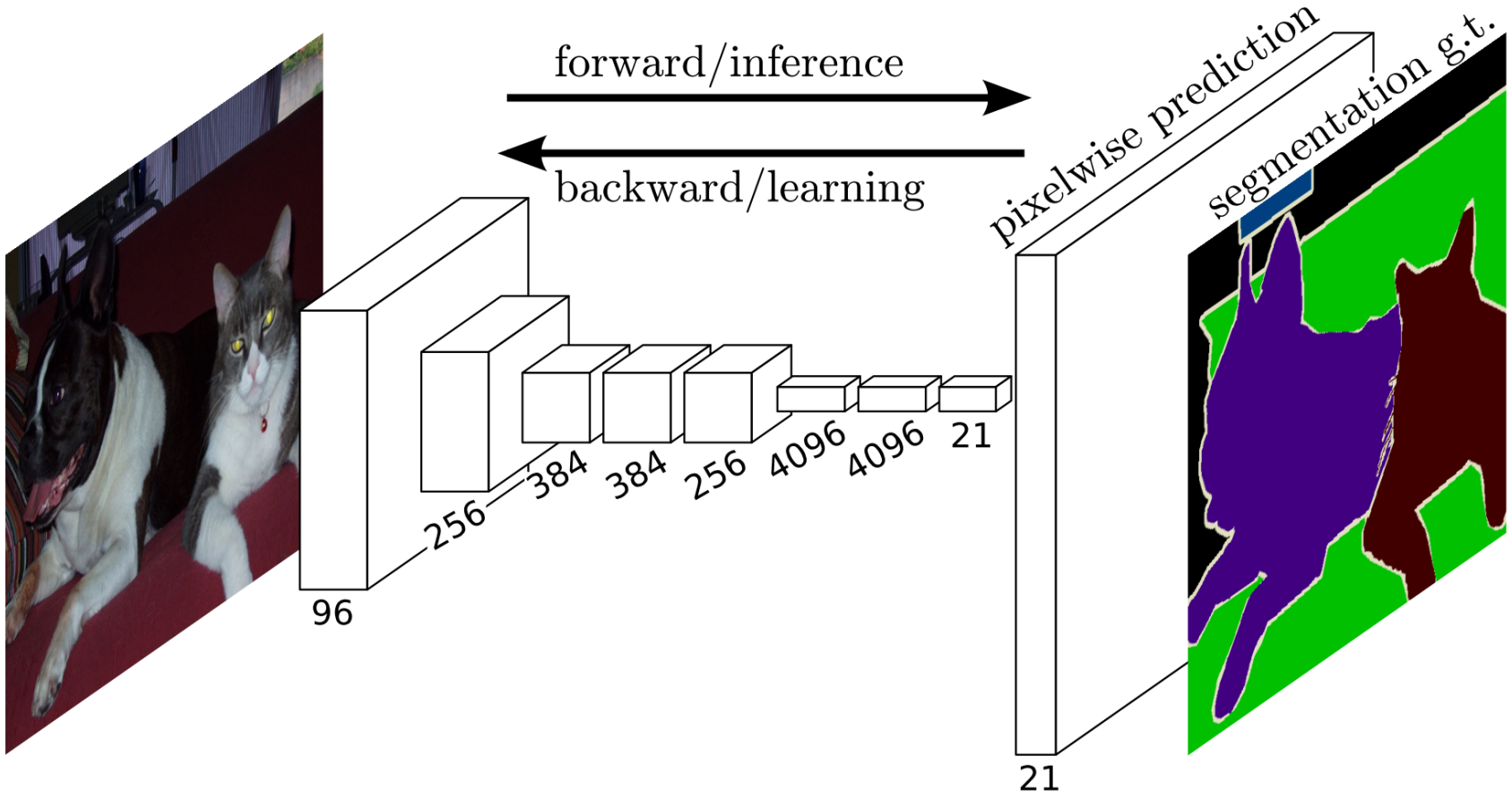
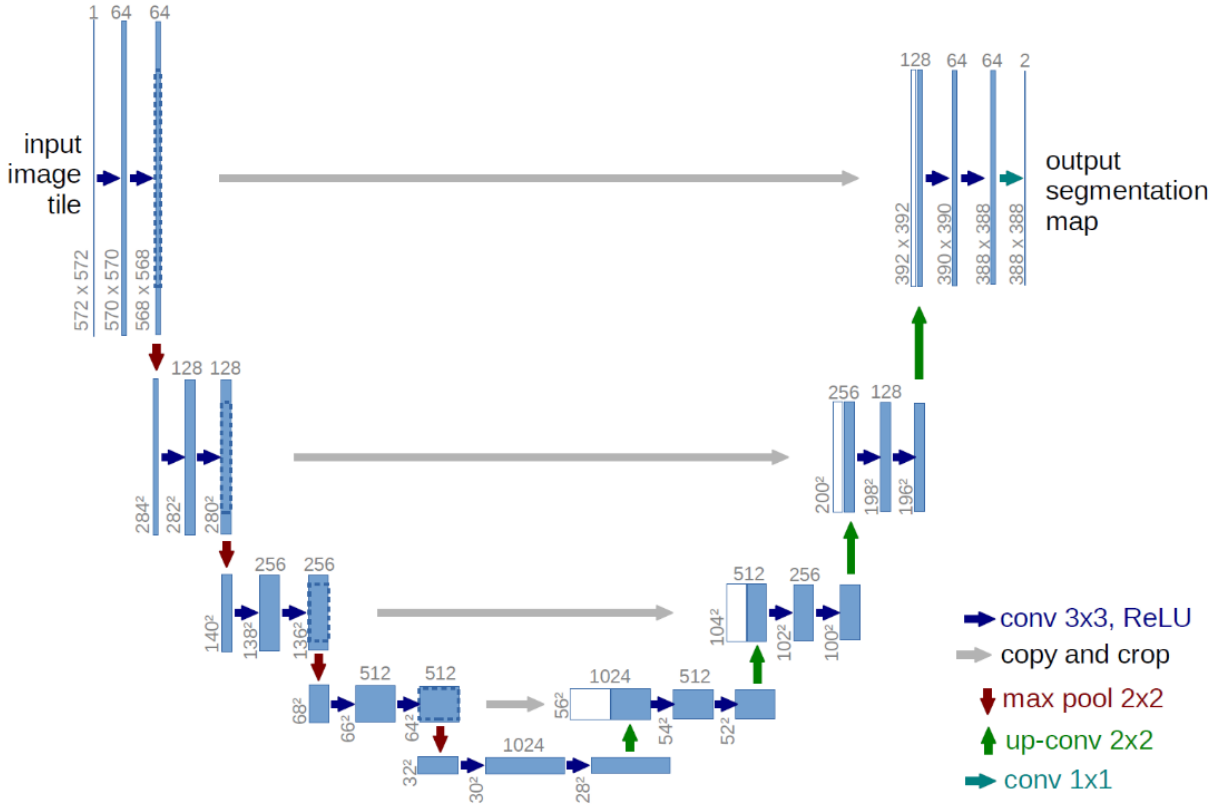


Image Segmentation

U-Net

- Bottleneck architecture
- contracting + expansive path
- Expanding to full resolution using upsampling
- Skip connections to preserve detailed information



Gated-SCNN: Gated Shape CNN

- two-stream CNN architecture
- explicitly wires shape information as a separate processing stream
- gating mechanism to connect the intermediate layers
- Fusion of information between streams at the very end through a fusion module

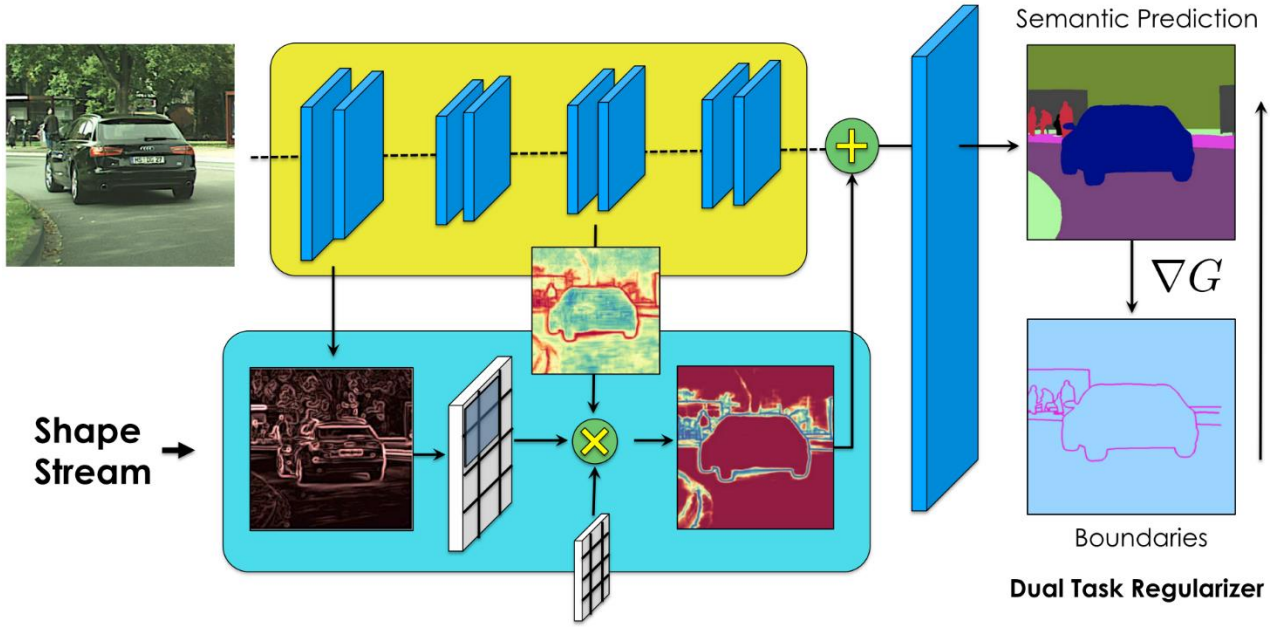


Image Segmentation

Gated-SCNN

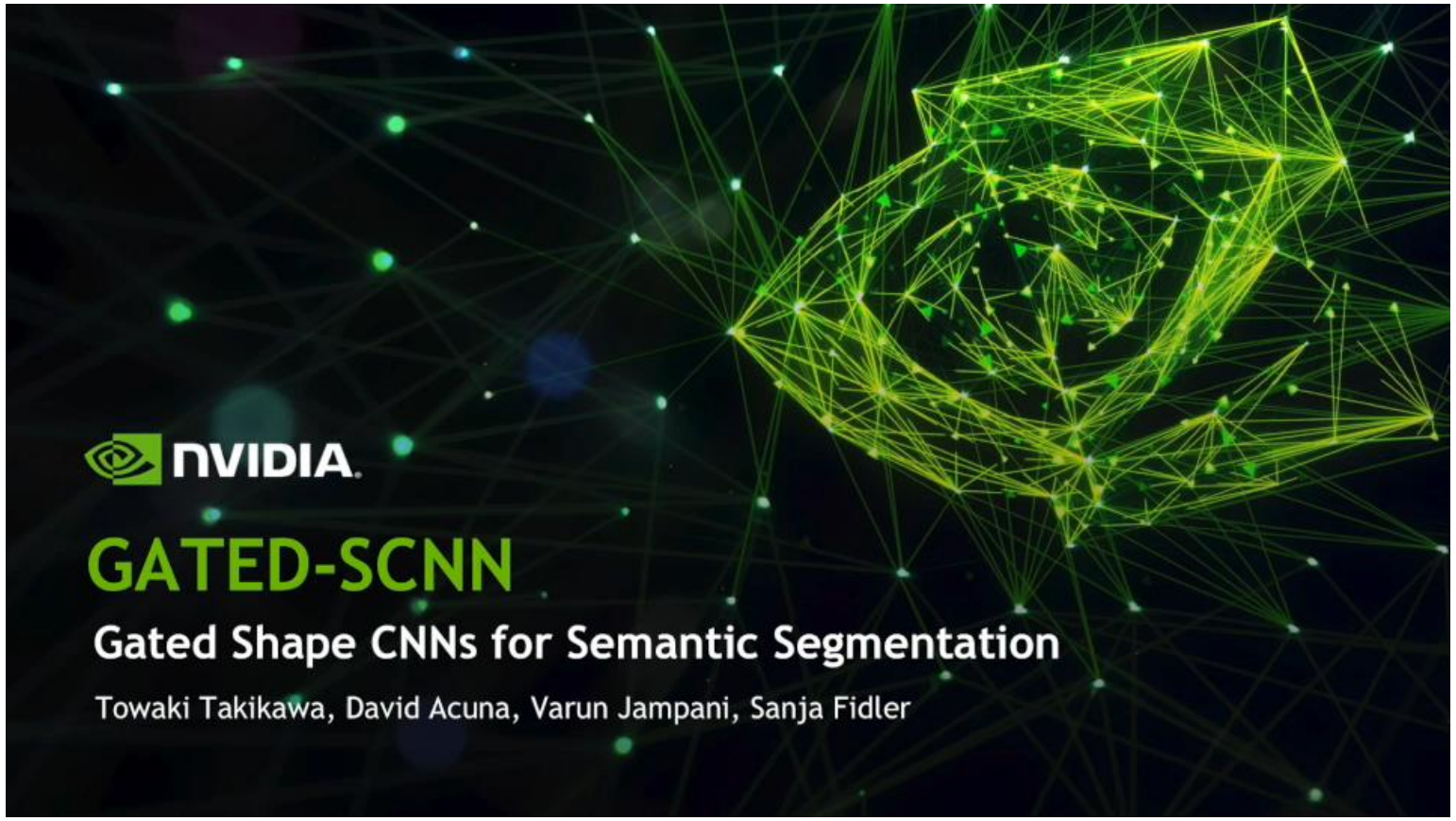
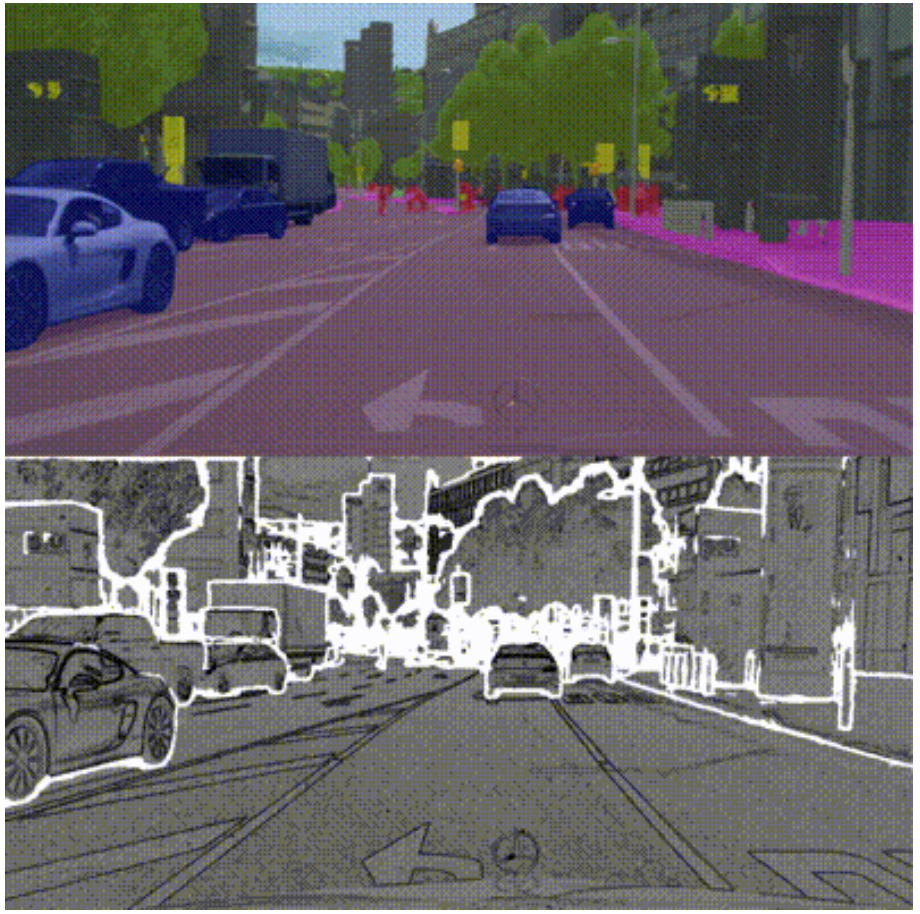


Image Segmentation

Gated-SCNN



Generative Adversarial Networks (GANs)

- Idea: Generator vs. Discriminator
  - Generator: trained to generate realistic images
  - Discriminator: trained to detect fake images

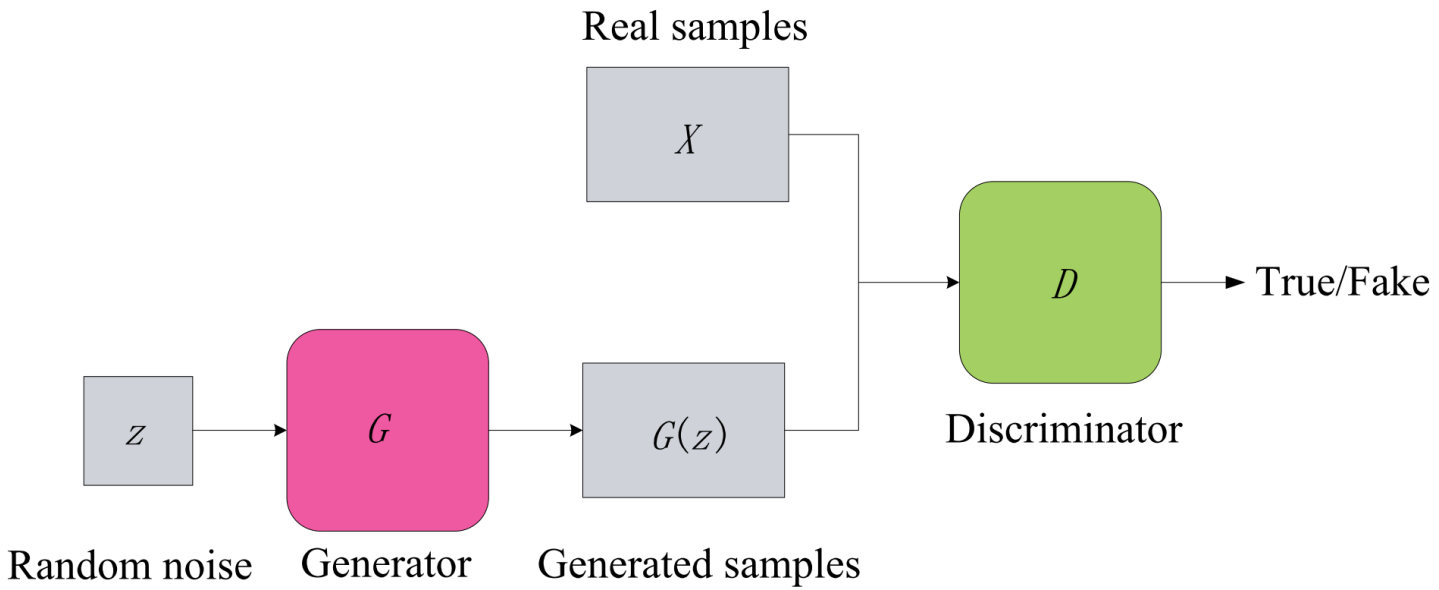


Image Super-Resolution: ProGAN

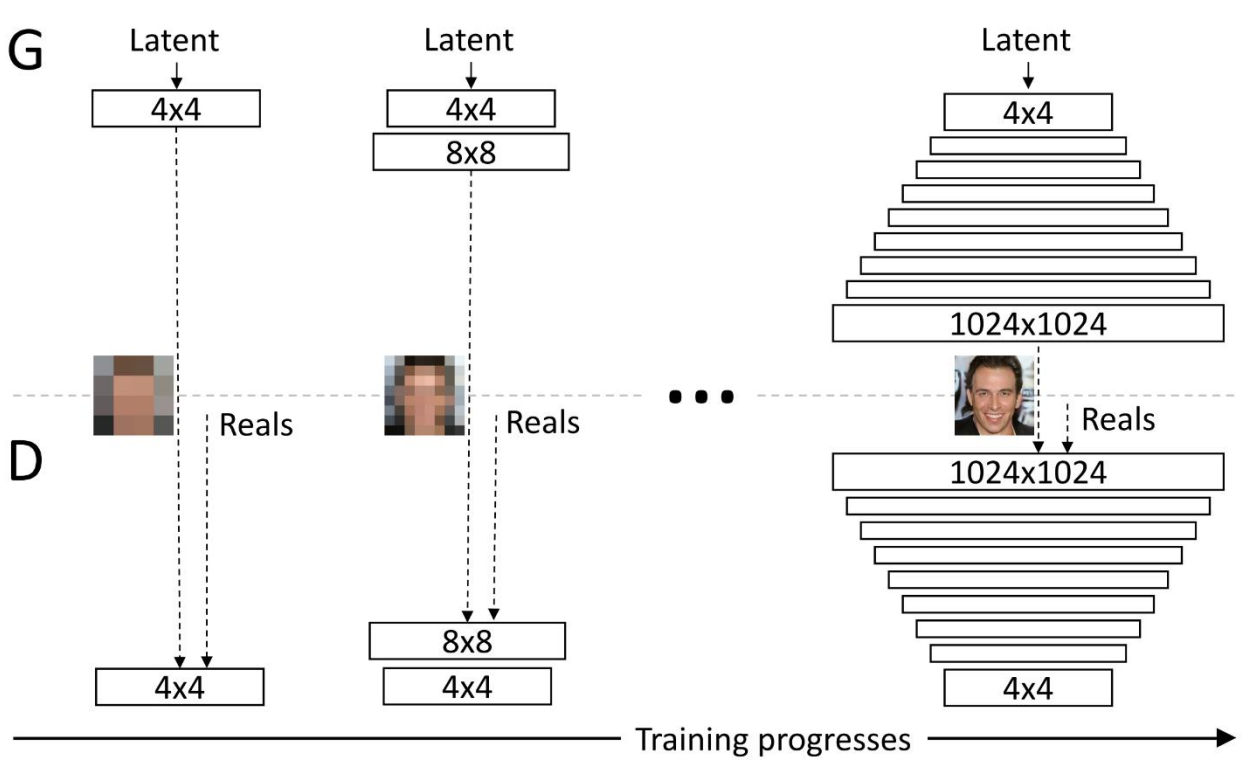


Image Generation

ProGAN

Autonomous Vehicles and AI

Autonomous Vehicles and AI-Assisted Driving I

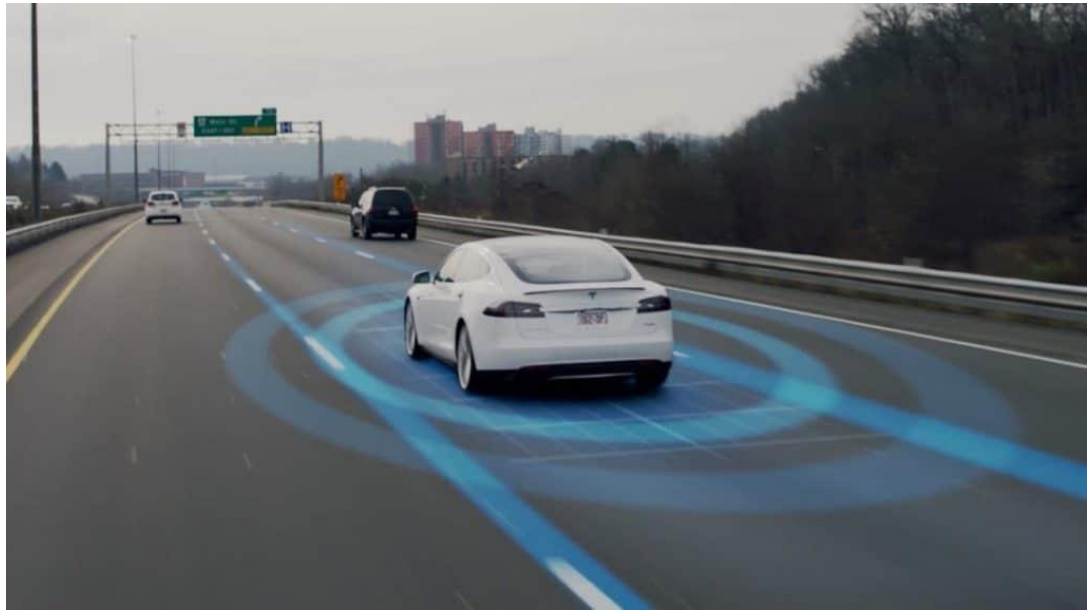
- Waymo
  - On-road: 20 million miles
  - Simulation: 10 billion miles
  - Testing & Validation: 20,000 classes of structured tests
  - Initiated testing without a safety driver

www.waymo.com

Autonomous Vehicles and AI-Assisted Driving II

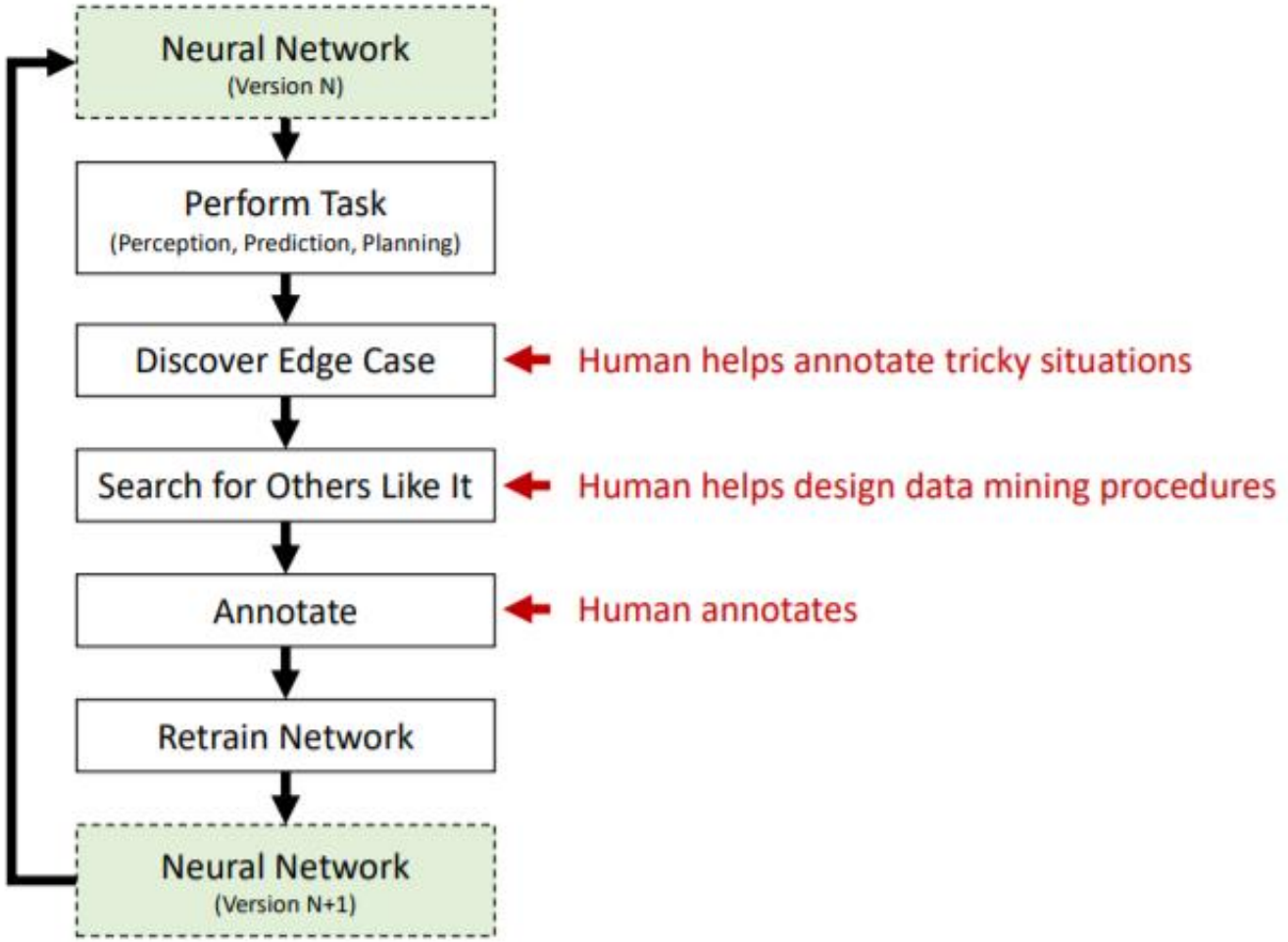
- Tesla Autopilot

  - The combination of various driver assistance systems is also called autopilot by Tesla .
  - The autopilot difference lies in the hardware installed in the Tesla vehicle, the installed software version, and the range of functions that this enables.
  - From October 2014, all Tesla vehicles were equipped with autopilot hardware.

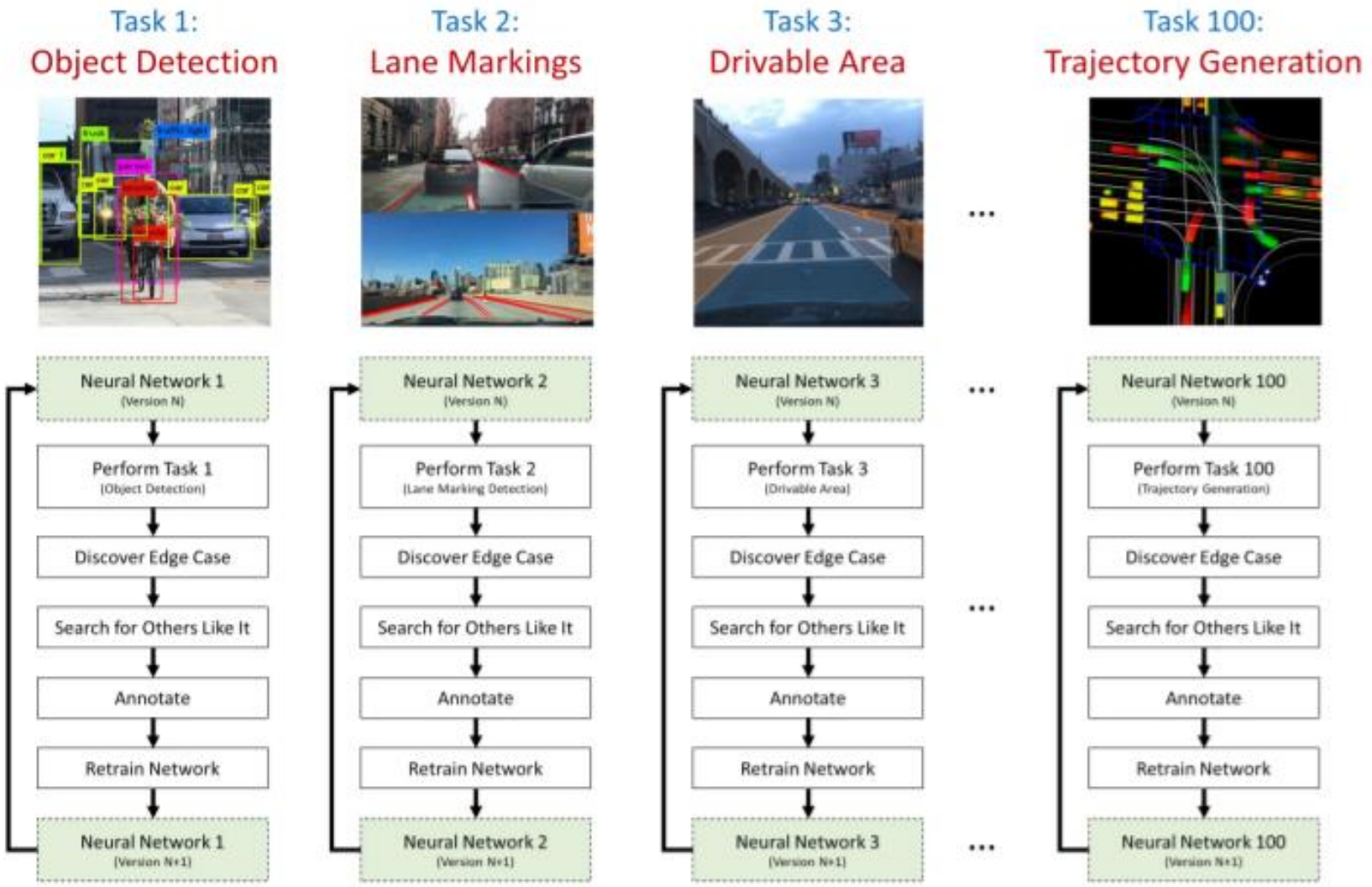


https://teslawissen.ch/tesla-autopilot-unterschied-ap1ap2ap2-5/

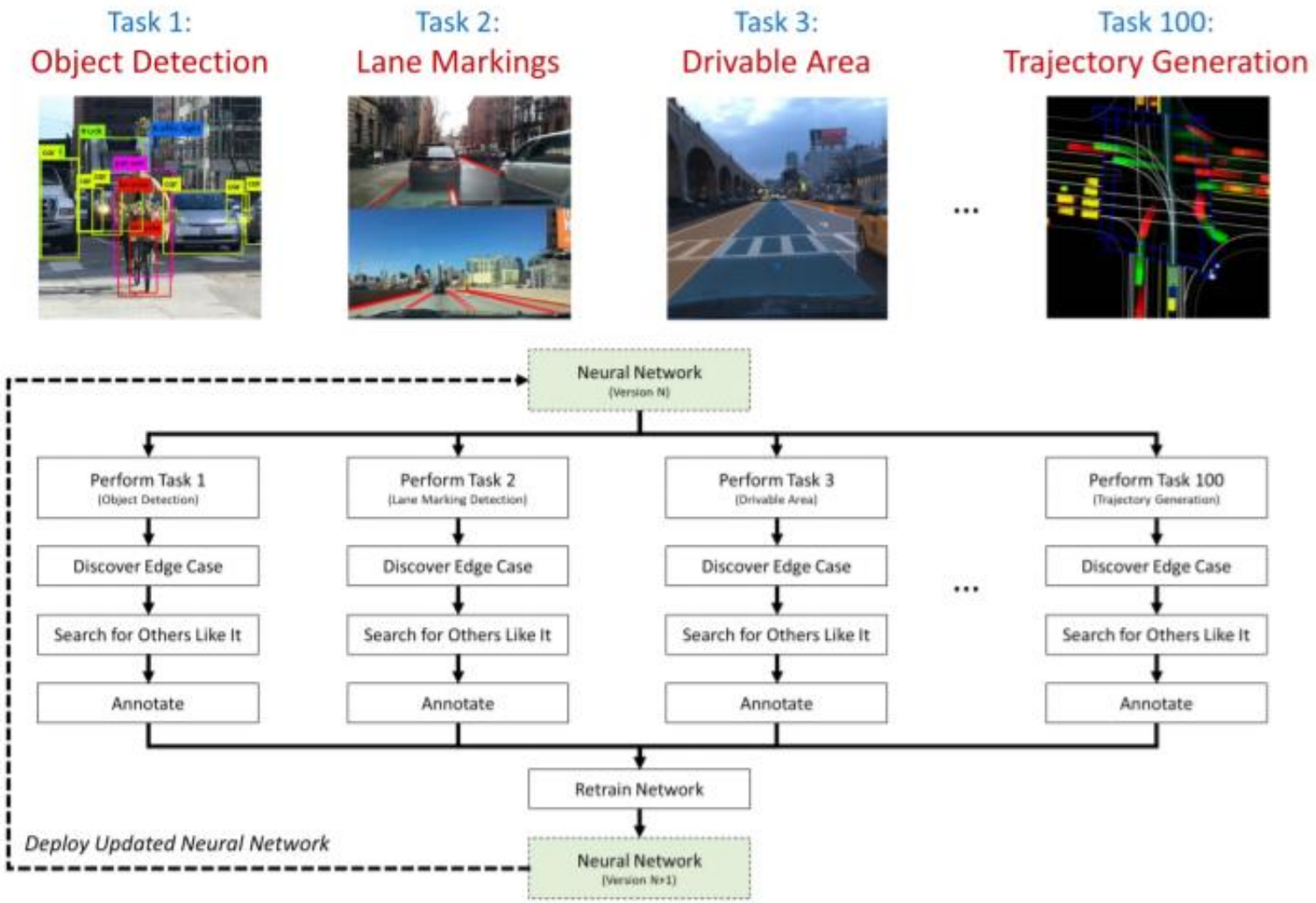
Autonomous Vehicles and AI-Assisted Driving - Active Learning Pipeline



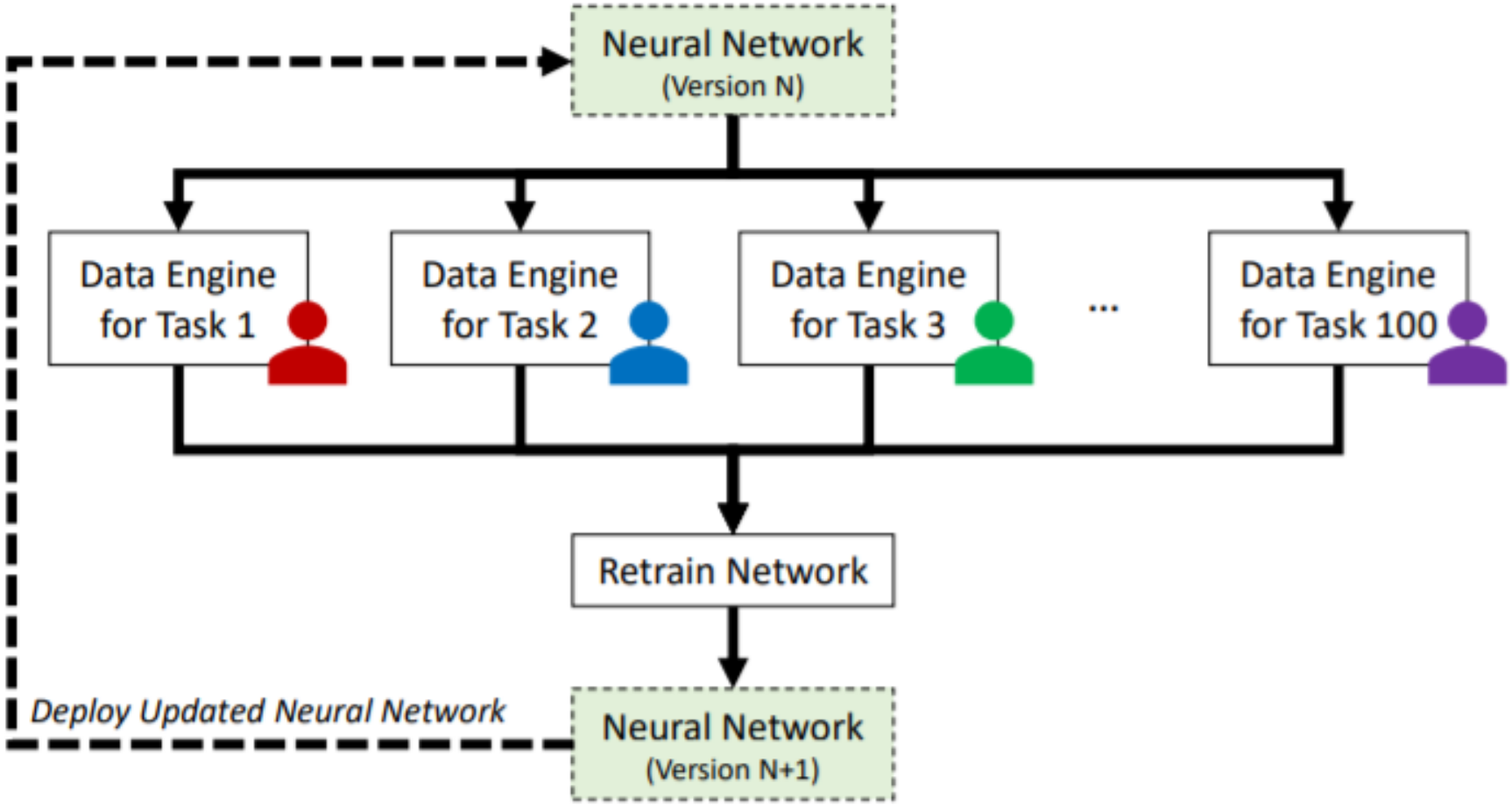
Autonomous Vehicles and AI-Assisted Driving - Single Task Learning



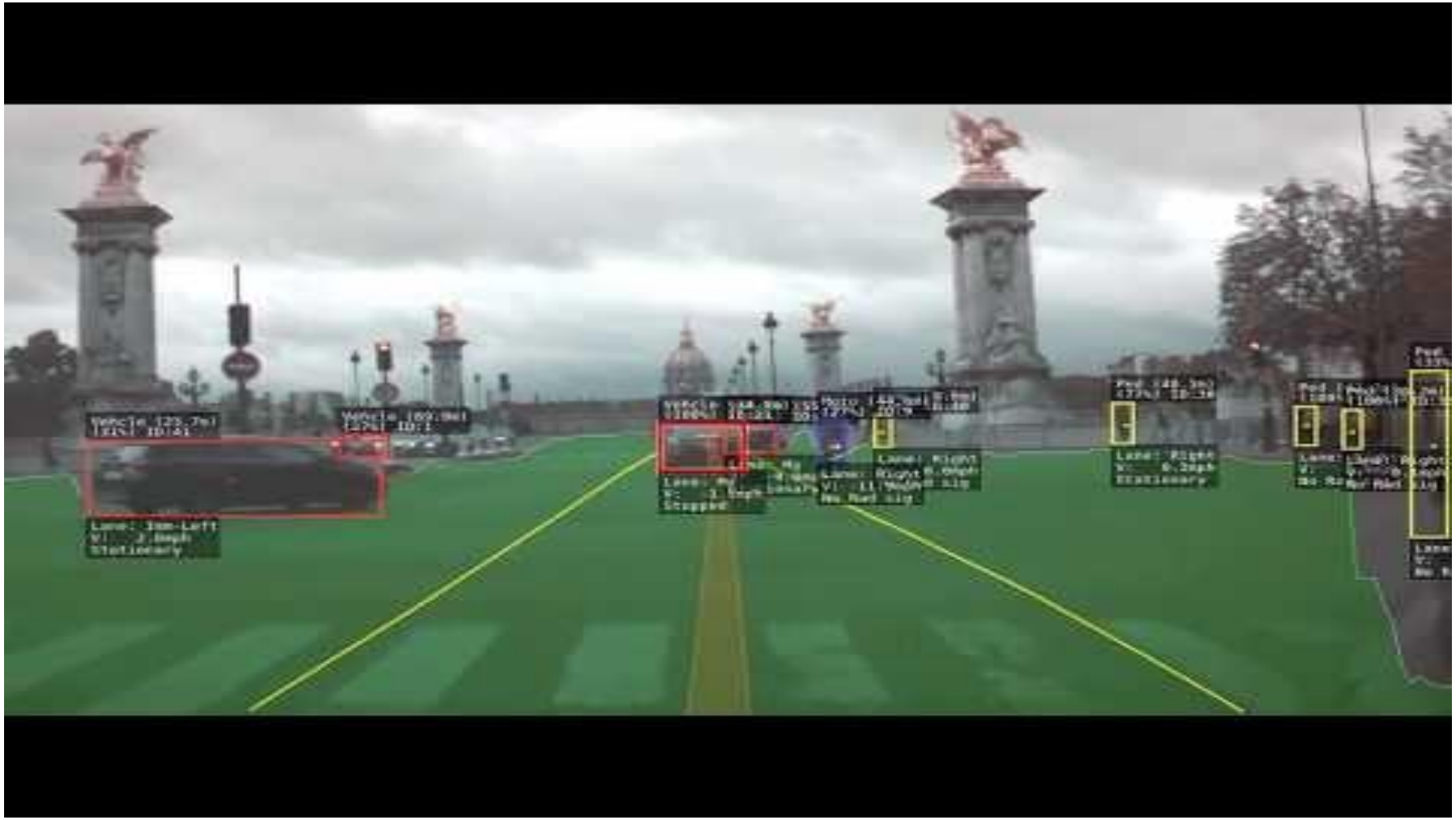
Autonomous Vehicles and AI-Assisted Driving - Multi Task Learning



Autonomous Vehicles and AI-Assisted Driving - Collaborative Deep Learning



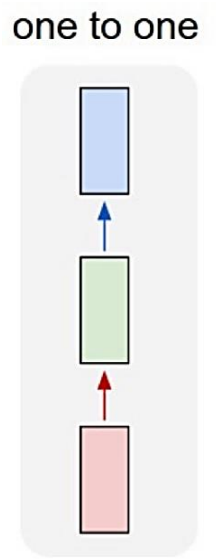
Autonomous Vehicles and AI-Assisted Driving



Natural Language Processing

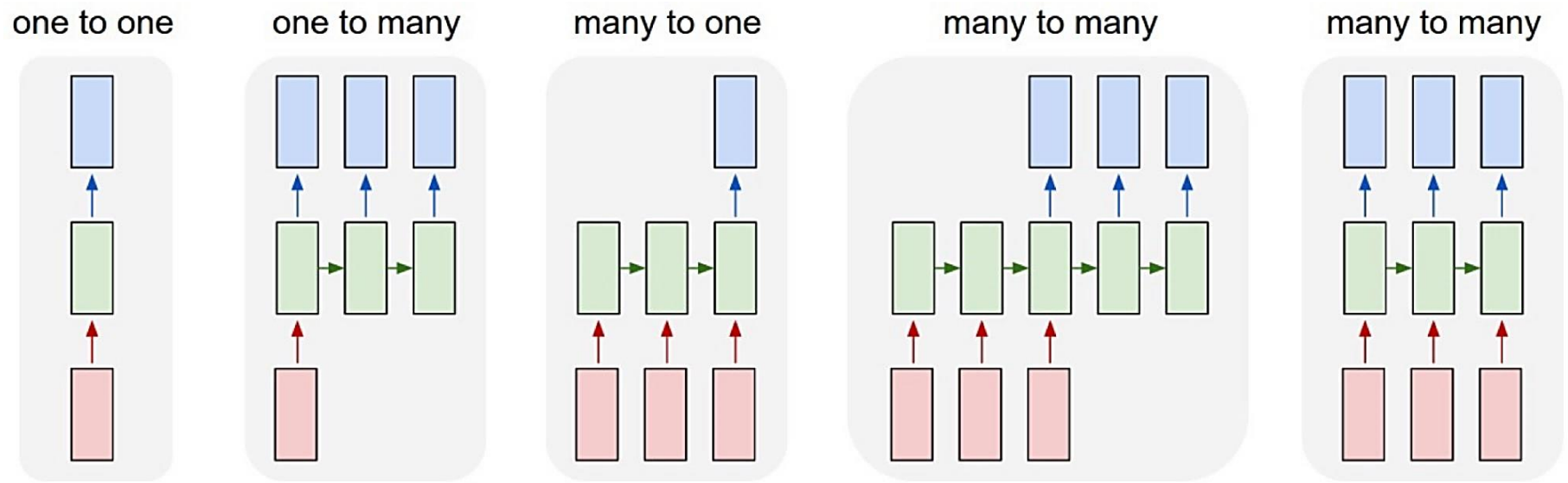
Sequence-to-Sequence Models

- Feedforward networks calculate the output on the basis of an input
- RNNs calculate a sequence of outputs $\{o_0, \dots, o_n\}$ based on a sequence of inputs $\{x_0, \dots, x_n\}$



Sequence-to-Sequence Models

- Feedforward networks calculate the output on the basis of an input
- RNNs calculate a sequence of outputs $\{o_0, \dots, o_n\}$ based on a sequence of inputs $\{x_0, \dots, x_n\}$



base

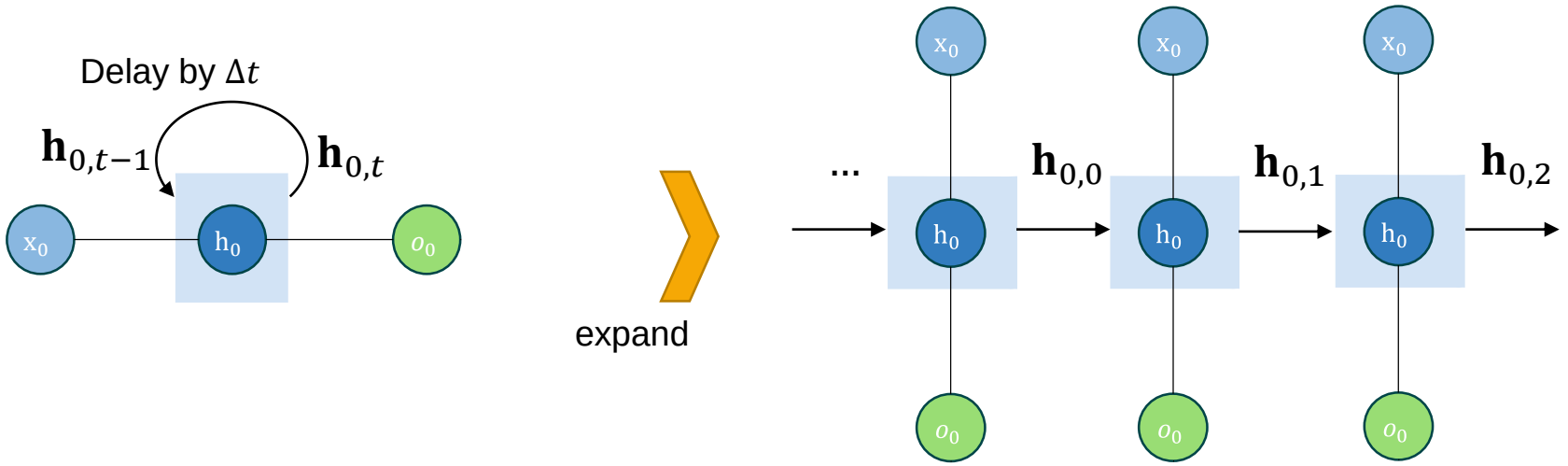
- Use the output of a hidden layer as input for the following time step

$$\mathbf{h}_t = \boldsymbol{\theta}\phi(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x\mathbf{x}_t + \mathbf{b}_h$$

$$\mathbf{o}_t = \boldsymbol{\theta}_o\phi(\mathbf{h}_t) + \mathbf{b}_o$$

$\mathbf{h}_t$ Pre-activation of the hidden layer

ϕ Non-linear activation function



- Vanishinig Gradient Problem [1]
  - Gradient for RNNs disappears due to update based on previous hidden states

[1] https://en.wikipedia.org/wiki/Vanishing_gradient_problem

Long Short Term Memories (LSTM)

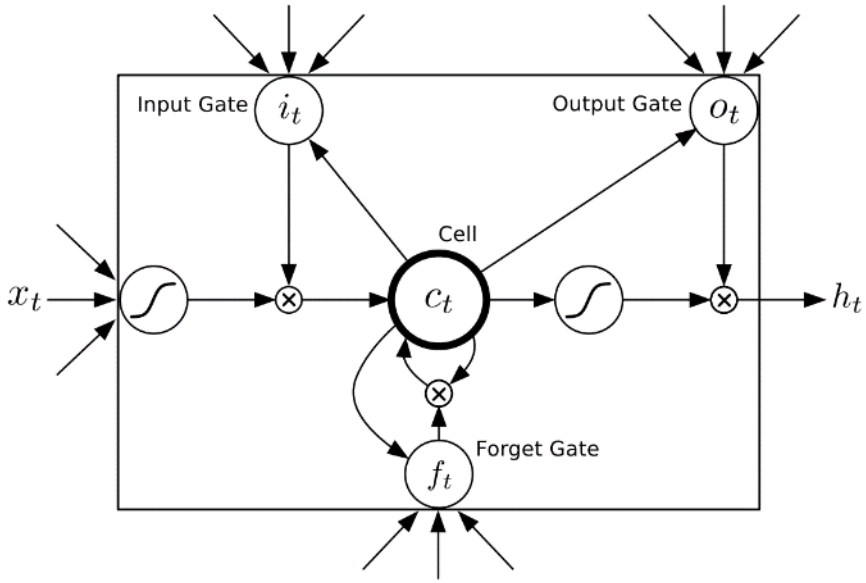
- LSTM solves the Vanishing Gradient problem by a **Memorycell c_t**

Scaling through sigmoid activation:

- input gate c_t Scaled input for cell (write)

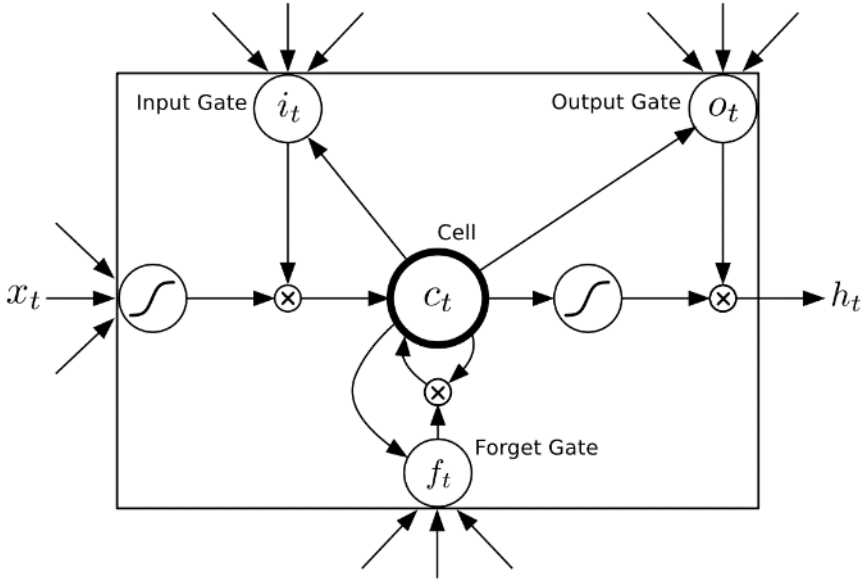
- output gate o_t Scales output of the cell (read)

- Forget Gate f_t Scales previous cell value (reset)

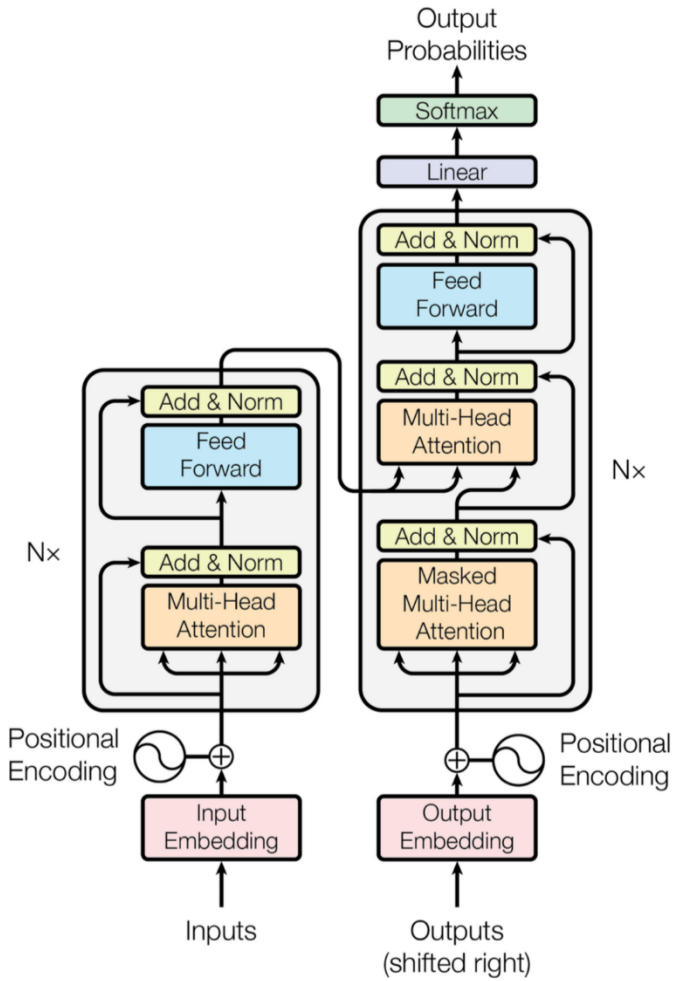


Long Short Term Memories (LSTM)

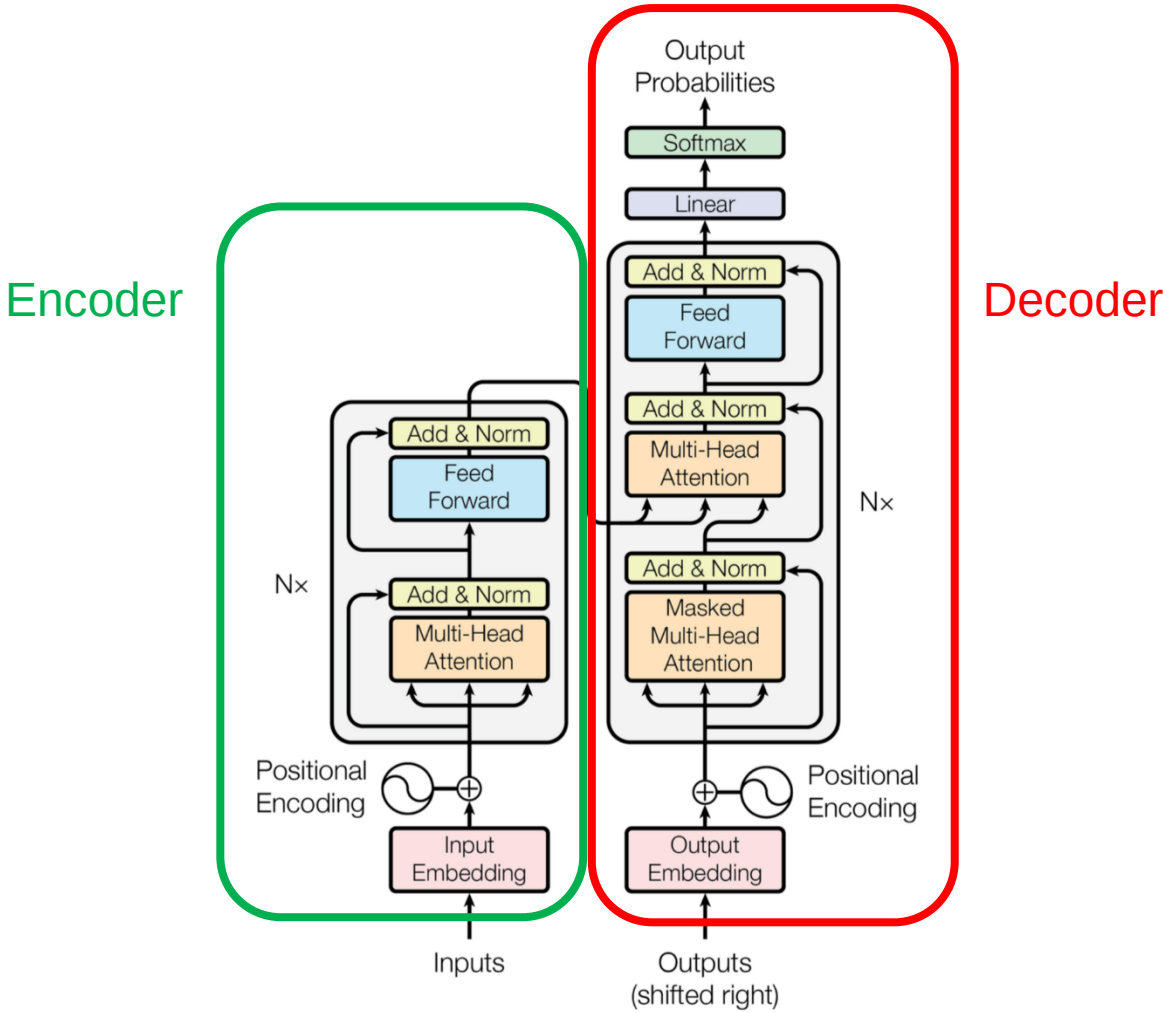
$$\mathbf{i}_t = sigm(\boldsymbol{\theta}_{xi}\mathbf{x}_t + \boldsymbol{\theta}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{f}_t = sigm(\boldsymbol{\theta}_{xf}\mathbf{x}_t + \boldsymbol{\theta}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{o}_t = sigm(\boldsymbol{\theta}_{xo}\mathbf{x}_t + \boldsymbol{\theta}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\mathbf{g}_t = tanh(\boldsymbol{\theta}_{xg}\mathbf{x}_t + \boldsymbol{\theta}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$



Transformer

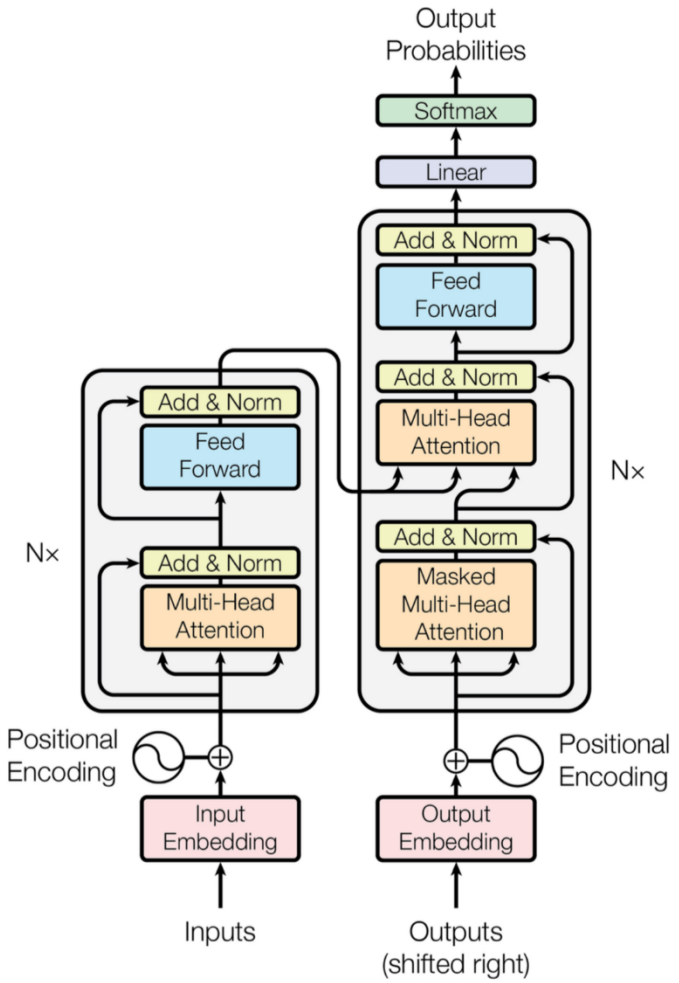


Transformer



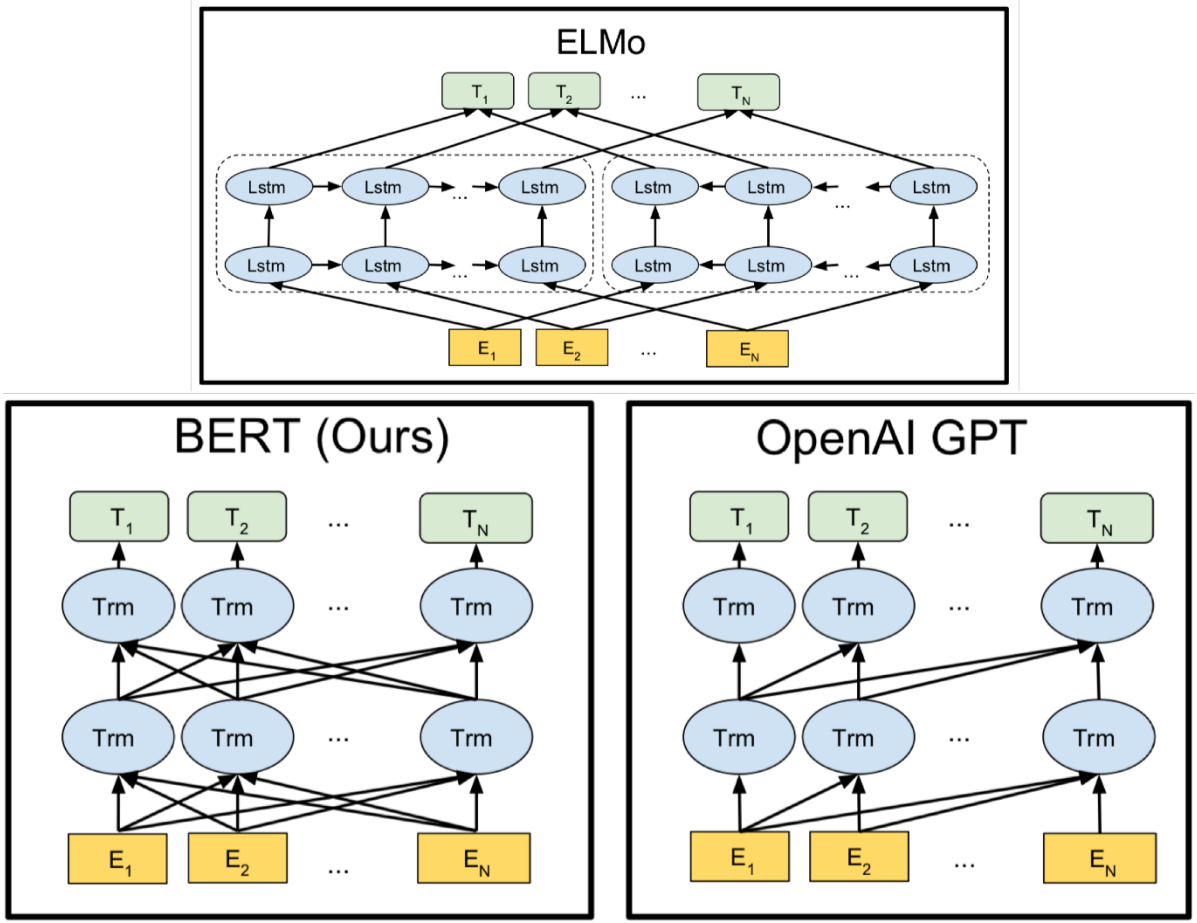
Transformer

- Key ideas:
  - Use byte pair encodings to process unknown words
  - Process tokens of input sequence in parallel
  - Enhance token embeddings by positional encoding to include specific position of token in sequence
- Further variations of attention
  - Self-Attention
  - Multi-head attention
  - Masked attention

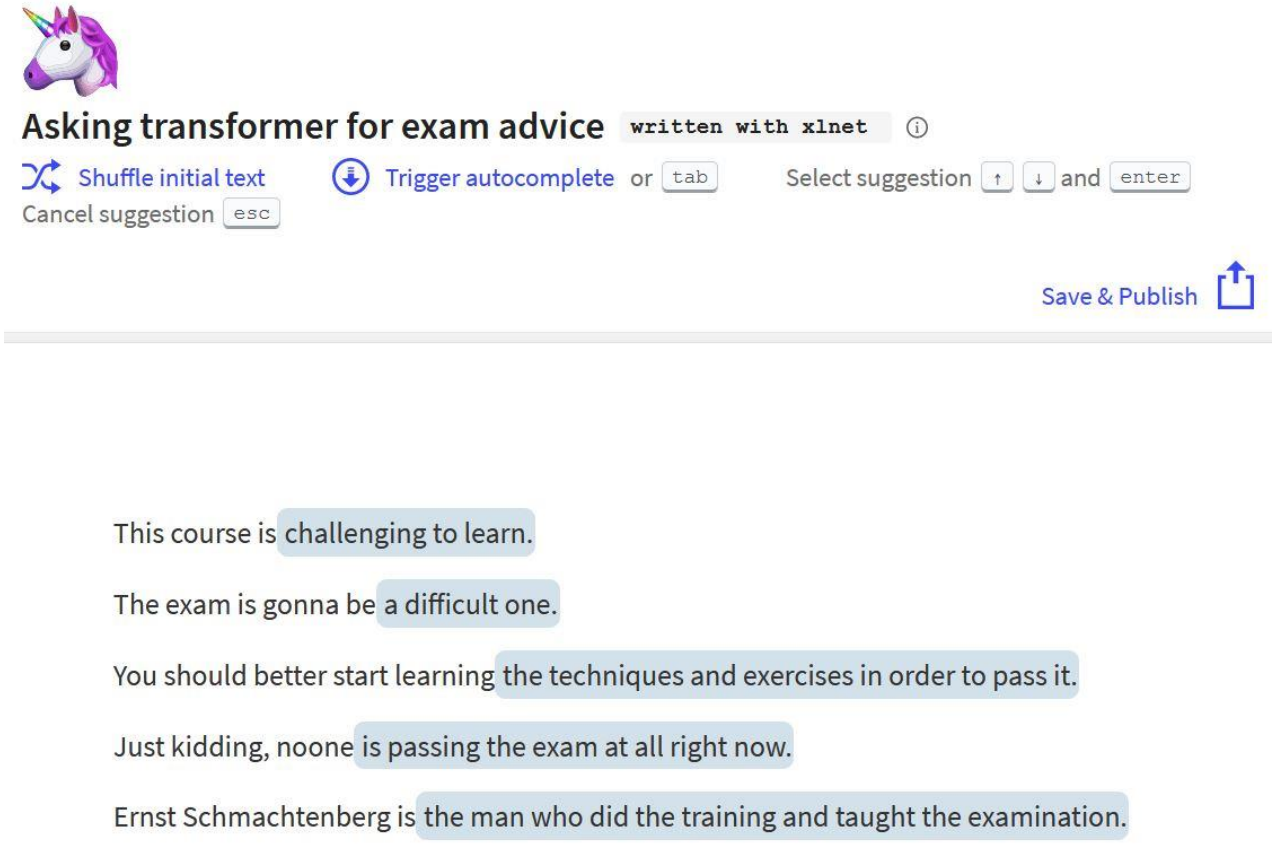


State-of-the-Art models

- GPT-2 (OpenAI)
- Megatron (NVIDIA)
- XLNET (Google)
- ALBERT (Google)



Write with transformer



https://transformer.huggingface.co/

Natural Language Processing

An apple a day keeps the doctor away...

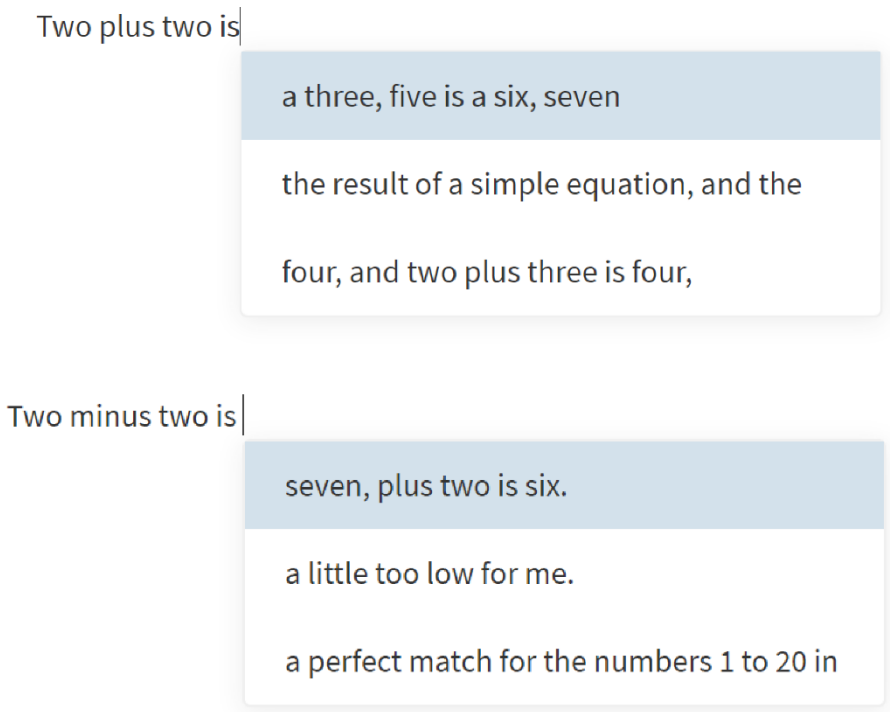
But is this healthy?

The only thing you need to know is... It's not a good idea. What is a "bad" apple?

Well, I'm sure you've heard it said that the healthiest apple is one that has only one type of apple. This is incorrect . The reason that it is not healthy to eat only one type of apple is that not all types of apples are created equal. There are different types of apples that have different nutrition and different nutritional qualities. If you are trying to lose weight, it 's probably best to have a variety of different types of apples, rather than only the same apple with the same nutrition . That ' s where the apple a day comes in .

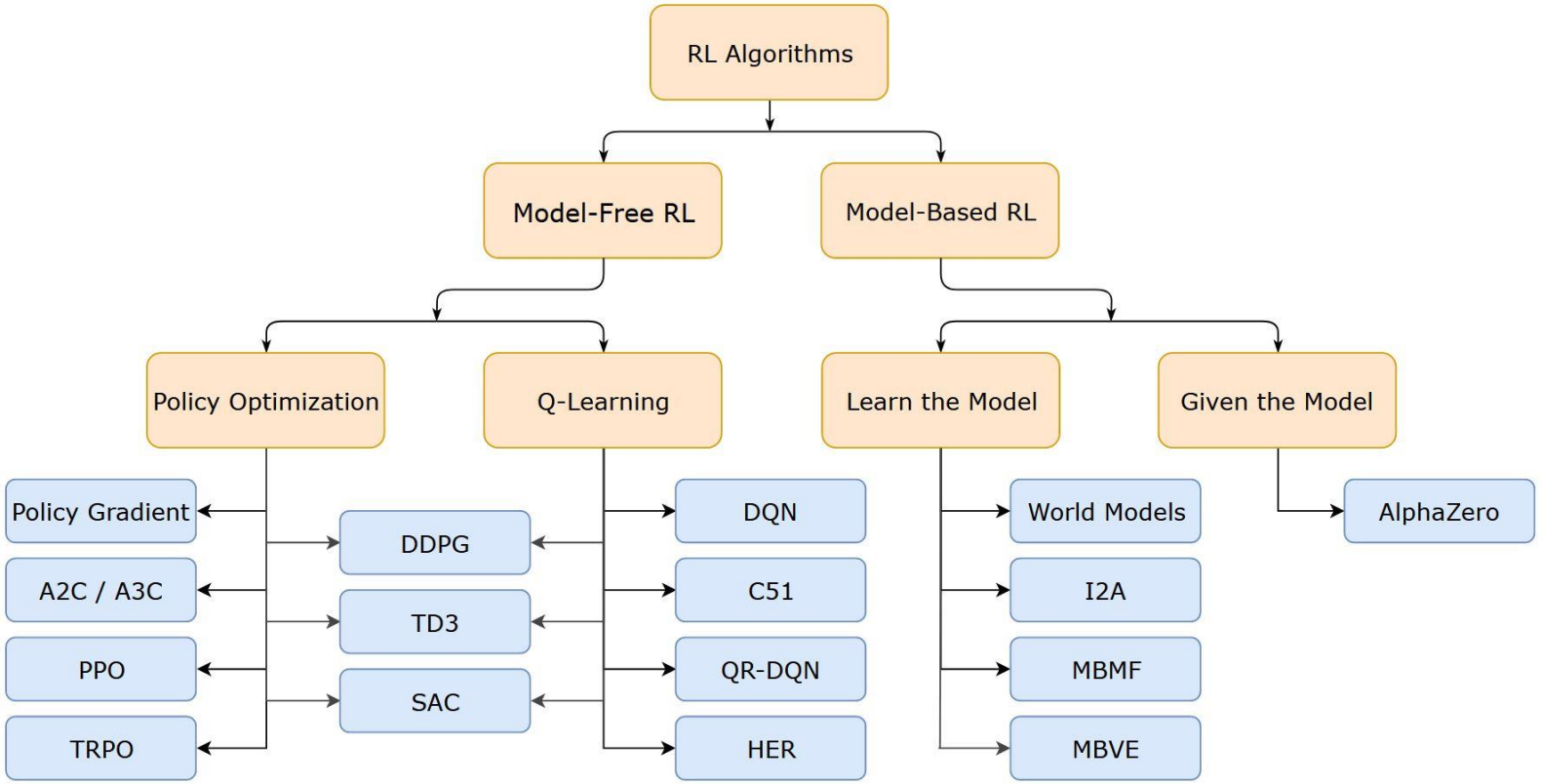
Remaining problems

- They do not understand language.
- Far from it (for now).



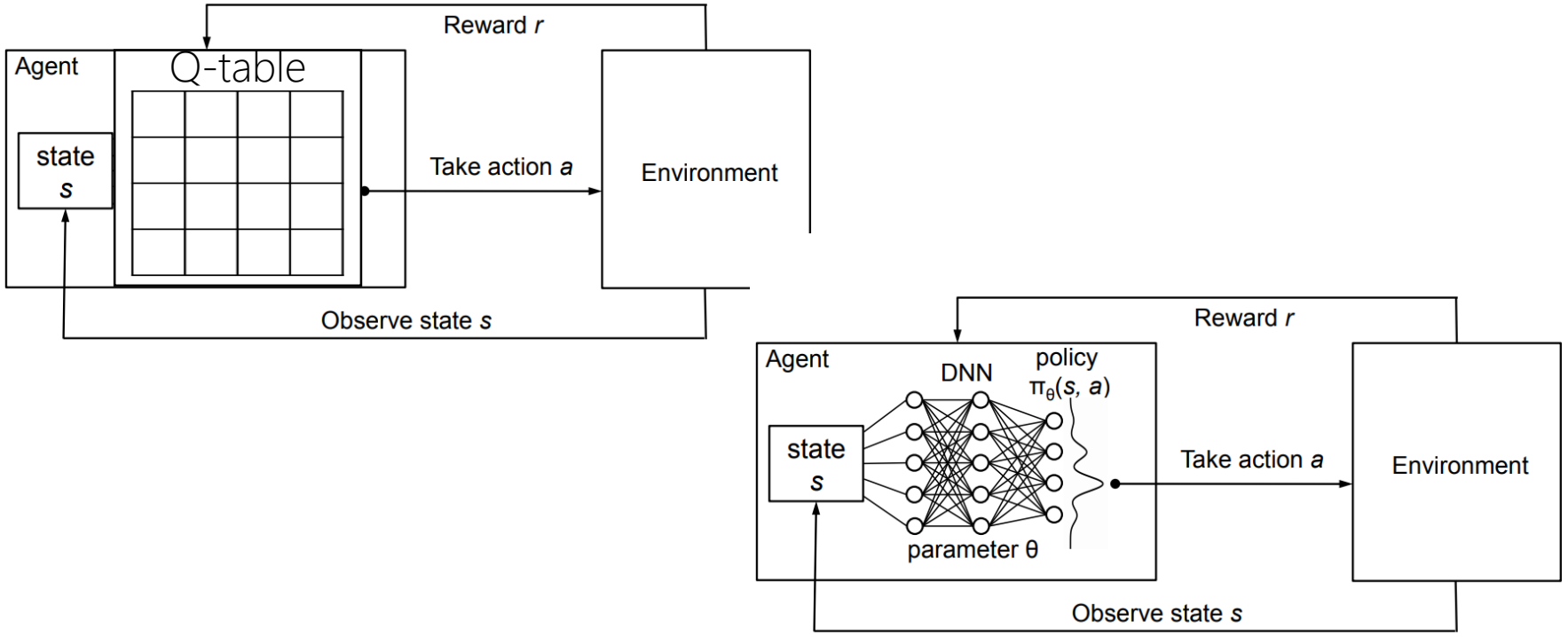
Deep Reinforcement Learning

Algorithms

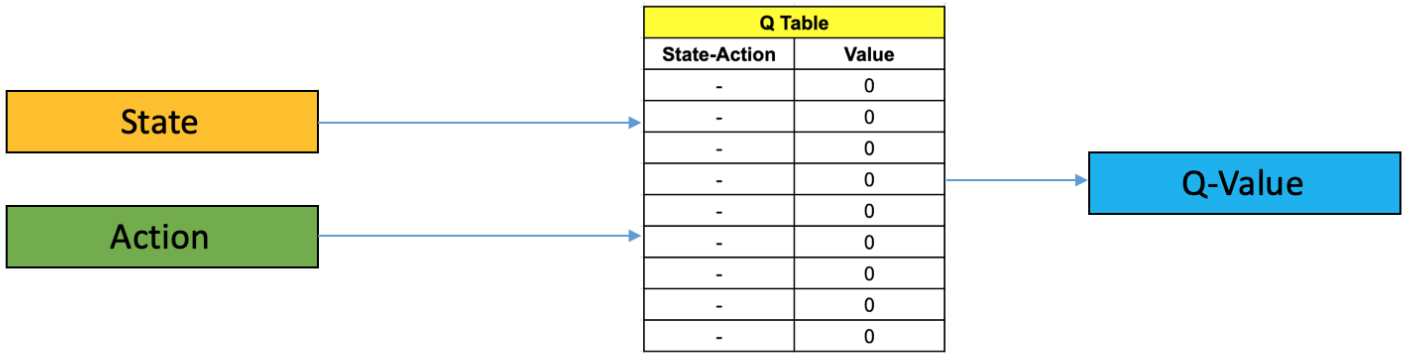

RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

What is DRL?

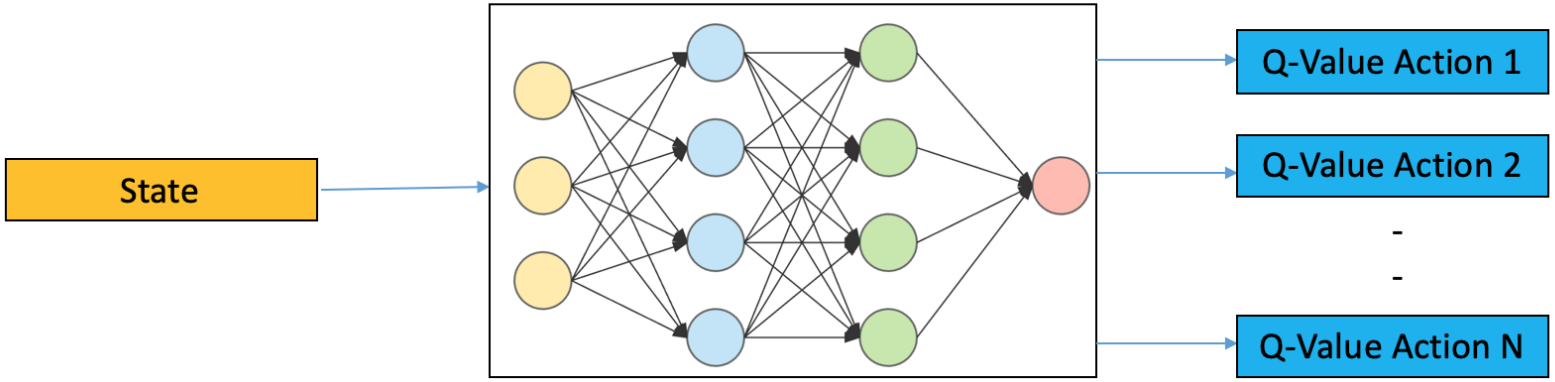
In Deep Reinforcement Learning (DRL), RL and DL are used to create efficient algorithms by approximating part of the algorithms decision making steps with deep neural networks.



DRL



Q Learning



Deep Q Learning

DRL Q-learning approximation

Algorithm:

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, ... till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s'

else:

$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) \leftarrow (1 - \alpha) Q_k(s, a) + \alpha \left[\text{target}\right]$$

$$s \leftarrow s'$$

DRL Q-learning approximation

Algorithm:

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, ... till convergence

Sample action a, get next state s'

If s' is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s'

Chasing a nonstationary target!

else:

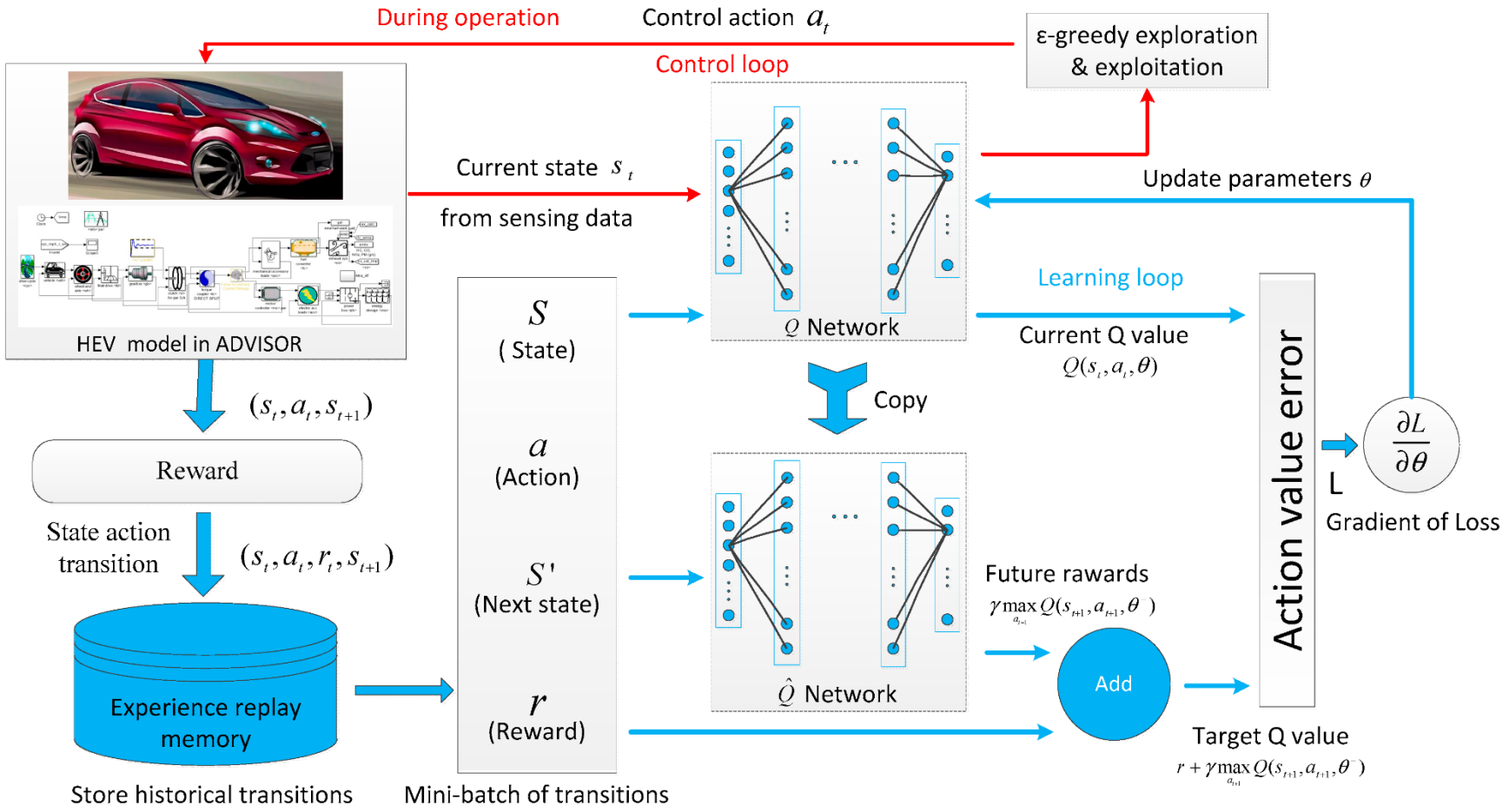
$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \mathbb{E}_{s' \sim P(s'|s,a)} \left[(Q_\theta(s, a) - \text{target}(s'))^2 \right] \Big|_{\theta=\theta_k}$$

$$s \leftarrow s'$$

Updates are correlated within a trajectory!

Example of DRL

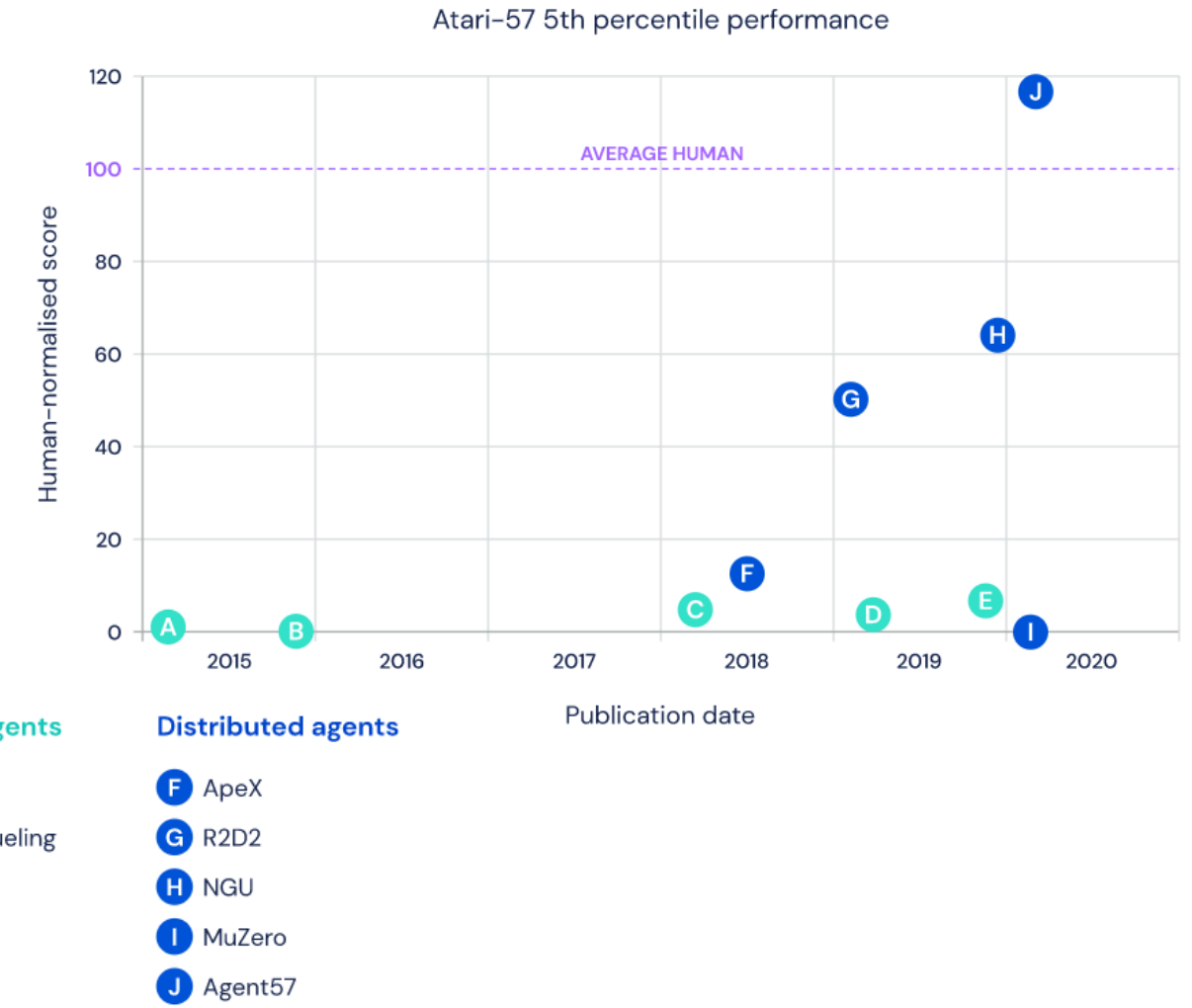


Source: Energy Management Strategy for a Hybrid Electric Vehicle Based on Deep Reinforcement Learning

Agent 57

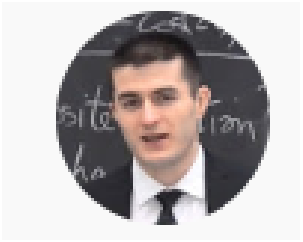
Agent 57 uses DQN agents with modifications:

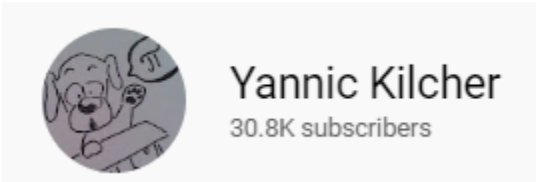
- Short term memory modules
- Base algorithm improvements
- Exploration
- Episodic memory
- Meta controllers



Further Reading Material

Check the lastest AI papers with the Arxiv sanity website http://www.arxiv-sanity.com/





Check out the YouTube podcast from Lex Fridman and Yannic Kilcher

Thank you for your attention!

Lecture Team AIDAE
aidae@ima-ifu.rwth-aachen.de