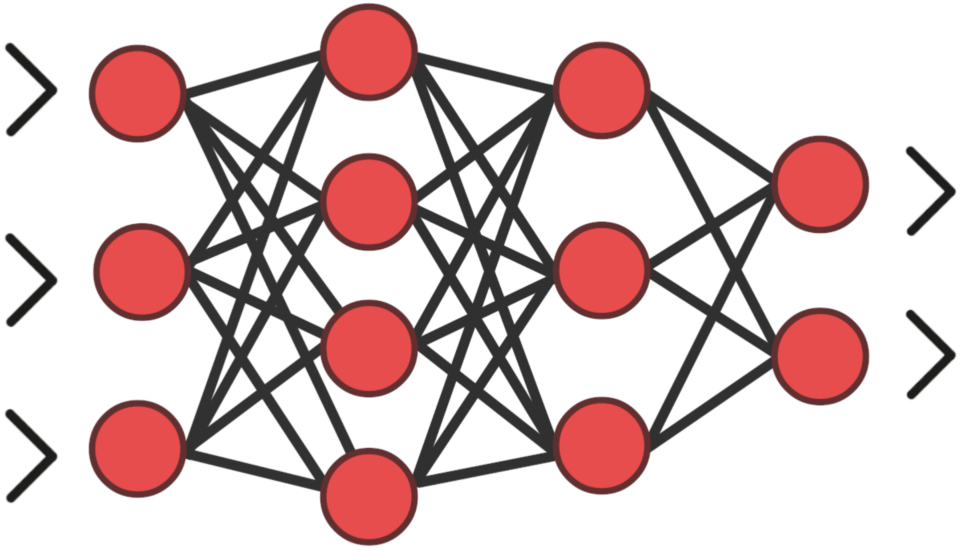
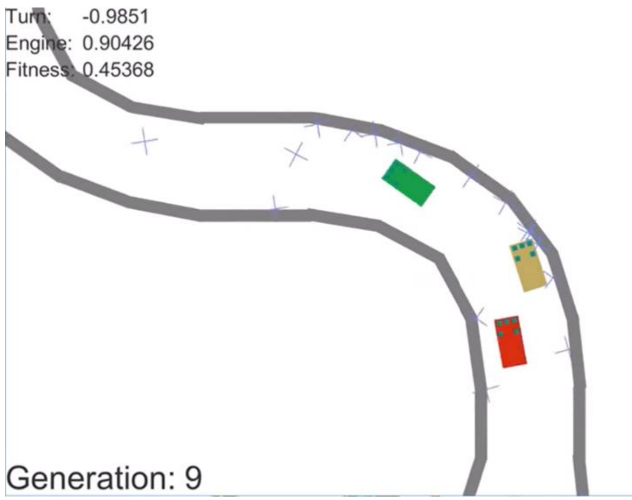
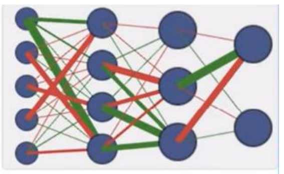
Deep Learning

From Mainstream Hype to training actual Video-Game Agents







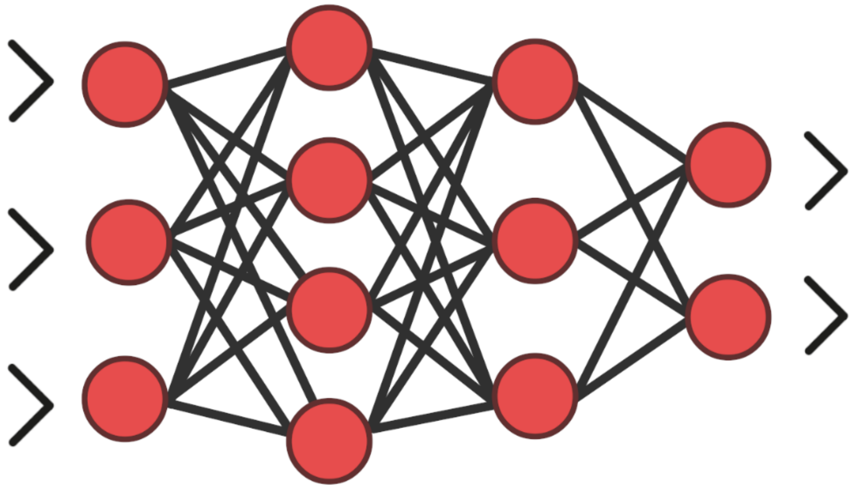
Samuel Arzt

Purpose of this talk

- Understand fundamental concepts of Deep Learning / Machine Learning
- Starting point to read more about and dive deeper into the field
- Second half: Focus on Learning Algorithms applied to Video Games
- Disclaimer: Sometimes not scientificly accurate and very simplified
  - Avoided complex formulars where possible

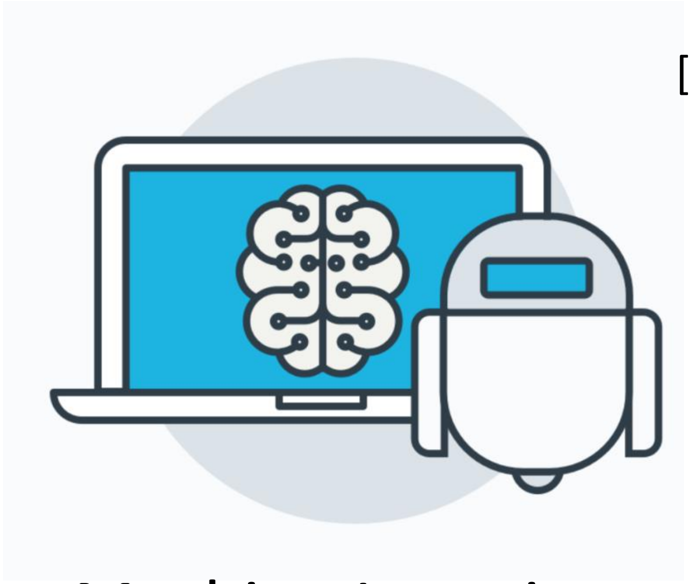
What is Deep Learning?

- Simplified:
  Combination of Deep Neural Networks and Machine Learning
  or
  Using Machine Learning Algorithms to train Deep Neural Networks



Deep Neural Networks

+



[1]

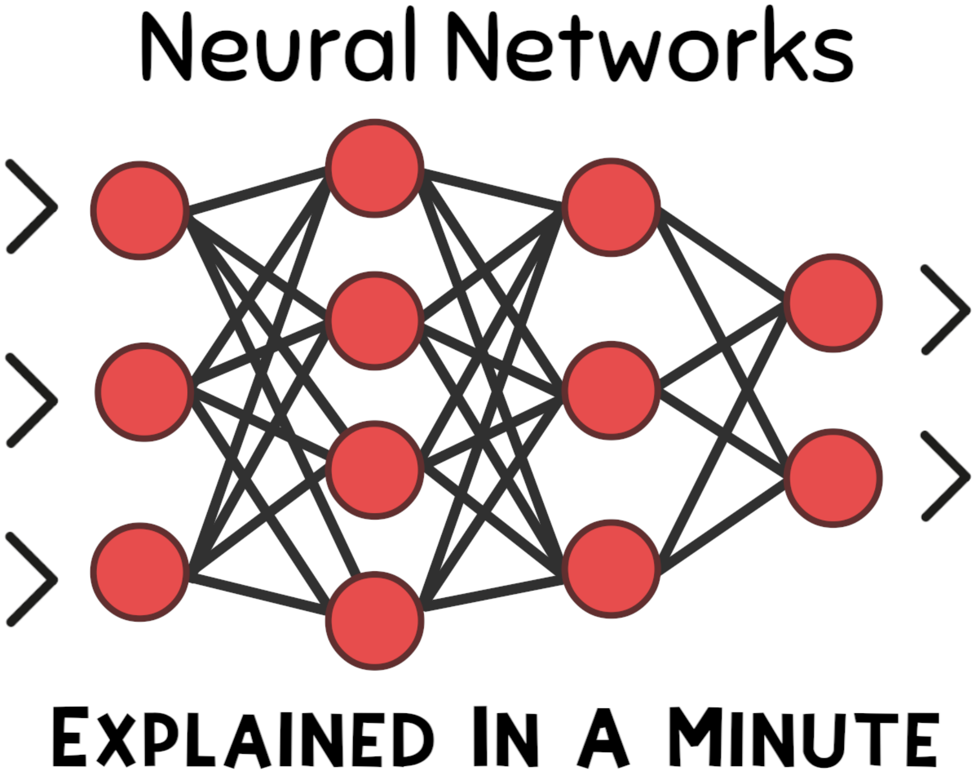
Machine Learning

[1]: Thumbnail Coursera Machine Learning: https://www.udacity.com/course/machine-learning-engineer-nanodegree--nd009t

What are Neural Networks?

https://youtu.be/rEDzUT3ymw4

[1]



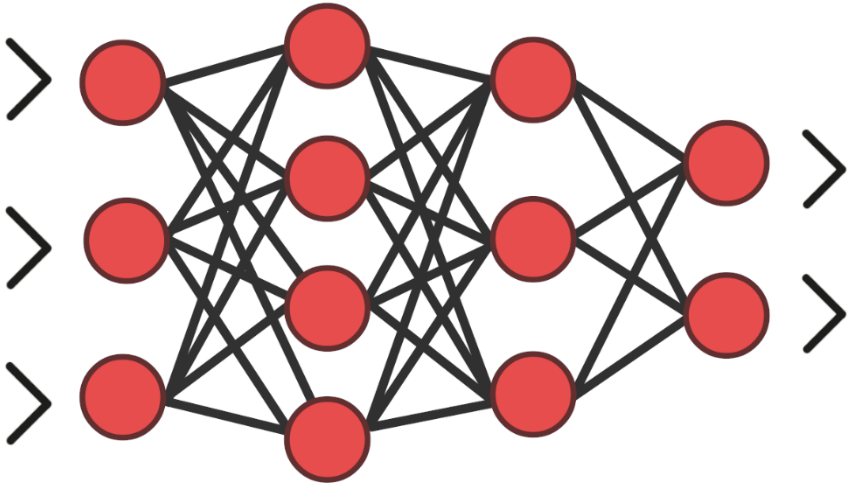
[1]: https://arztsamuel.github.io/en/projects/youtube/explained/explained.html

When does a NN become deep?

- Simple (and outdated) Definition:
  Any Network with more than one hidden layer

- Modern Understanding:
  Not well defined, „credit assignment of paths of sufficient length“ [1]

  „Problems of depth > 10 require *very deep* learners“ [1]

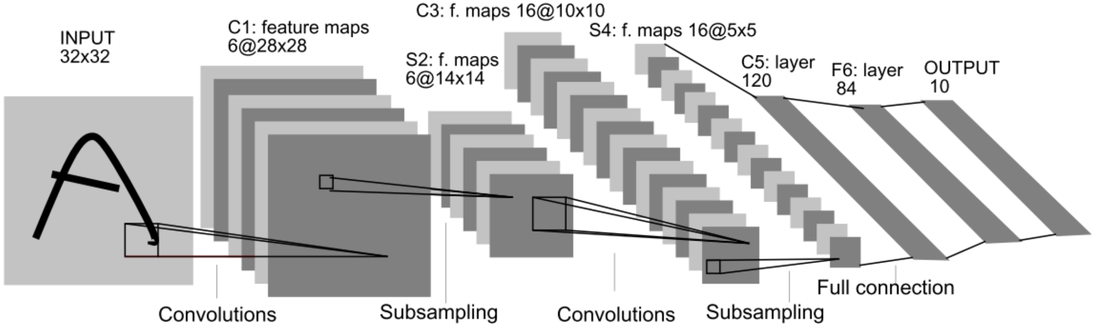


[1]: Jürgen Schmidhuber, 2014: „Deep Learning in Neural Networks: An Overview“

NN Architectures

- Feedforward Neural Networks
  - Acyclic Graphs, only connections to „upper“ layers
- Recurrent Neural Networks
  - Cycles and connections to previous layers allowed
- Convolutional Neural Networks
  - Weight sharing and spatial alignment of neurons, specialized for image input; Comparable to how image-filters work
  - Many new variants since 2014

[1]



[1]: LeCun et. al., 1998 „Gradient-based learning applied to document recognition.“

What is Machine Learning?

- Wikipedia: „Use statistical techniques to give computer systems the ability to *learn* (e.g., **progressively improve performance** on a specific task) from data, **without being explicitly programmed**." [1]

- „Algorithms that can learn from and make predictions on data by making data-driven predictions or decisions, through **building a model from sample inputs**" [1]

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

What is Machine Learning?

- Traditional Algorithms:
  Hard coded conditions and sequences of instructions.

- Machine Learning:
  Algorithm may start with random parameters and successively learns from input data.
  Goal: Improve performance (i.e. ability to solve task) over time.

Three Main Categories of ML

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised Learning

- Learning from „labeled“ data
  - E.g.: Large dataset of already labeled cat & dog images
    -> train classifier for cat / dog images

- Feed input to network -> depending on output, tweak weights to be „less wrong“ (minimize loss)
  - Stochastic gradient descent (backpropagation)

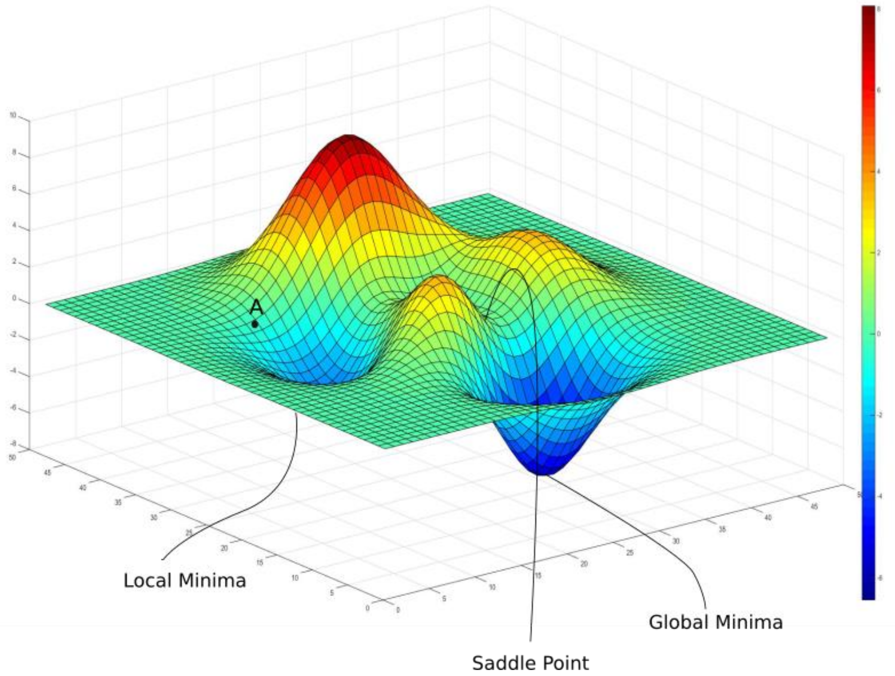
- In combination with ConvNets, most widely employed ML technique

Gradient Descent

- Gradient = direction of steepest descent for multivariate functions
  - Calculated using partial derivative of loss function

- Adjust parameters in direction of Gradient

- Stochastic GD
  - Use batch of data (random or simply order in training set) instead of entire data-set to compute gradient



[1]

[1]: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

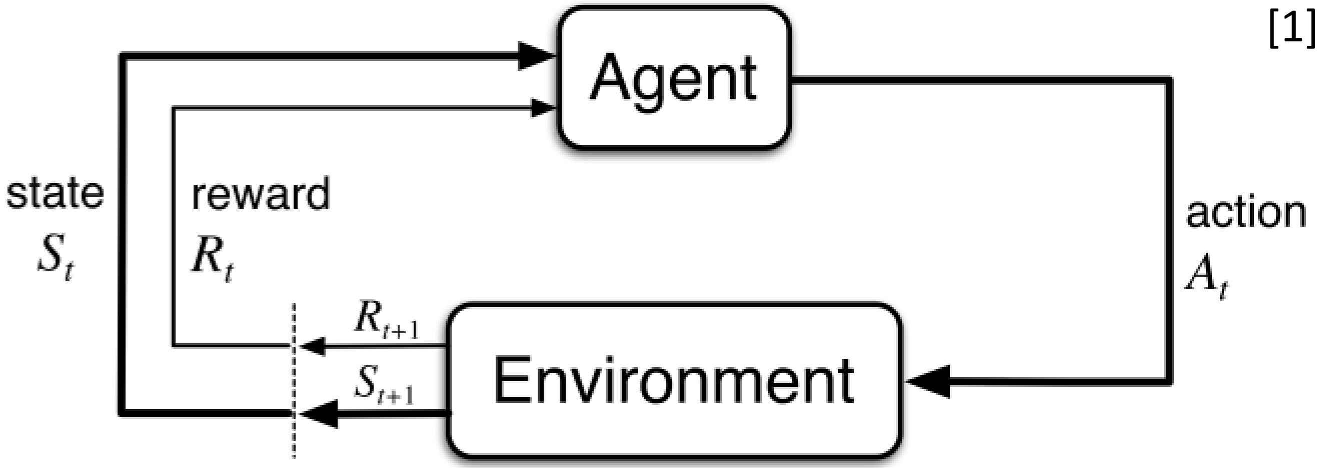
Reinforcement Learning

- „Agent“ learns through interacting with Environment
  - Feedback („Reward Signal“) from Environment
  - Goal: Learn a „Policy“, which maximizes accumulated reward
  - No labels, but trial and error

- Standard RL Setting:
  - State, Action, Reward
  - Epoch / Episode
  - Continuous / Sparse / Delayed Rewards

- Most famous: Q-Learning

[1]



The agent–environment interaction in a Markov decision process.

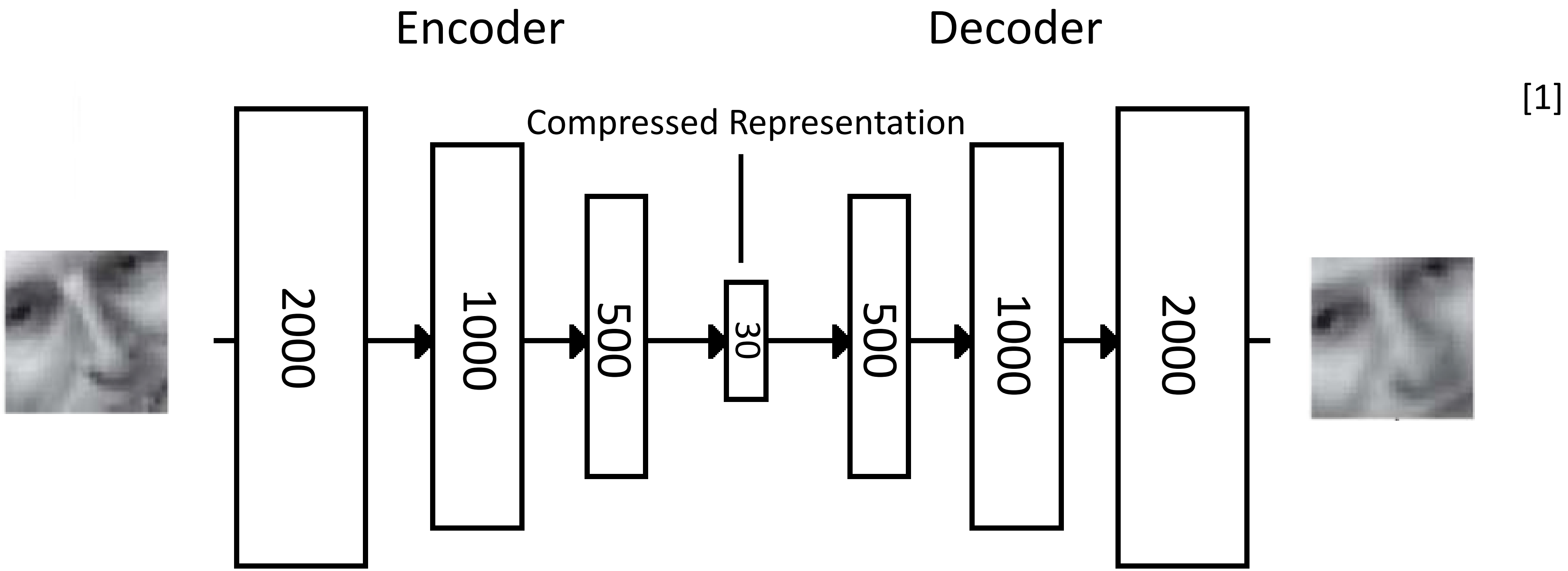
[1]: Sutton and Barto, 2017: „Reinforcement Learning: An Introduction“

Unsupervised Learning

- Learn from completely unlabeled data
- Probably least understood / progress from all three subcategories
- Happens in Humans / Animals all the time („common sense“)
- Example:
  Humans / Animals don‘t need to be taught Newton‘s 1st Law to understand and anticipate a ball falling of the table when pushed (learned solely from observation; no feedback, no labels)
- LeCun: *UL is the cake, SL the icing, RL the cherry on top*. (in respect to data efficiency)
  Unfortunate situation: we know how to make the icing and cherry but not how to make the cake [1]

[1]: Yann LeCun, 2016: Talk: „Deep Learning And The Future of AI“

UL Example: Encoder Decoder Network



[1]: Hinton et. al., 2006 „Reducing the Dimensionality of Data with Neural Networks“

ML Categories: Summary

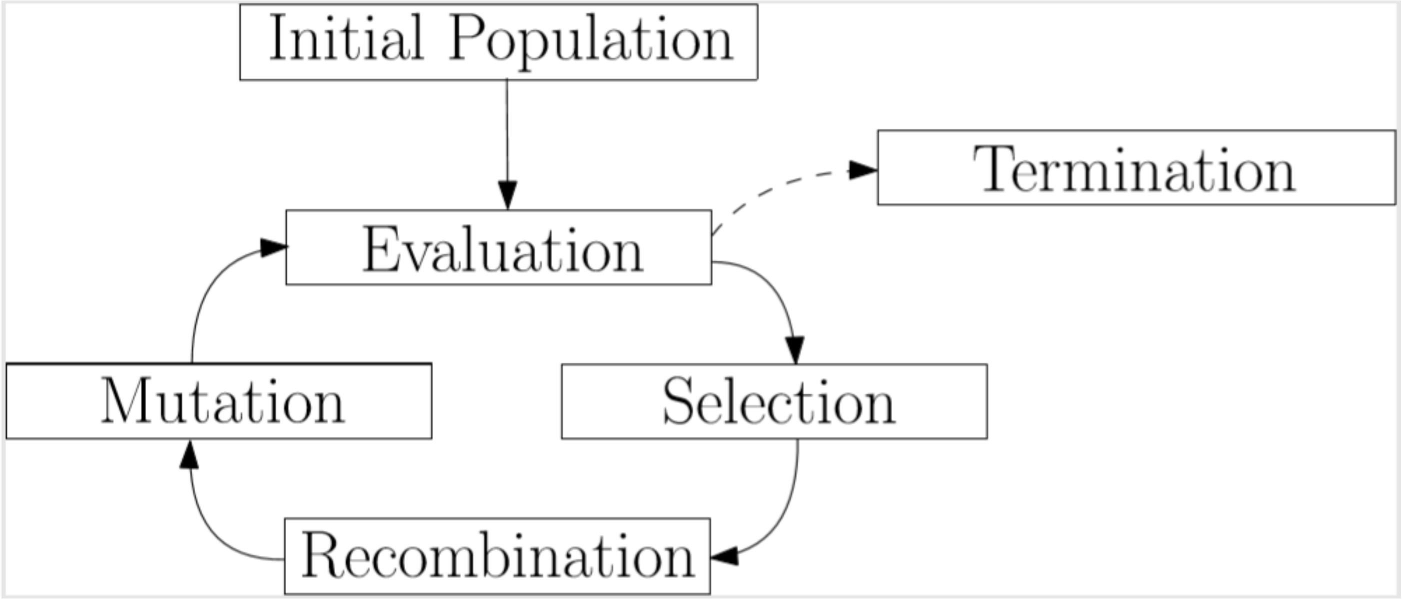
- Supervised Learning:
  Learn using labeled data; Most popular sub-category

- Reinforcement Learning
  Interact with environment and learn from feedback; Easy to integrate into typical game-setup

- Unsupervised Learning
  Learn completely without labels or feedback; Currently often applied for compression / clustering

Evolutionary Algorithms

- Typical Cycle:
  - Evaluate current Population
  - Select „best“
  - Recombine selected to form new Generation
  - Mutate to introduce new information



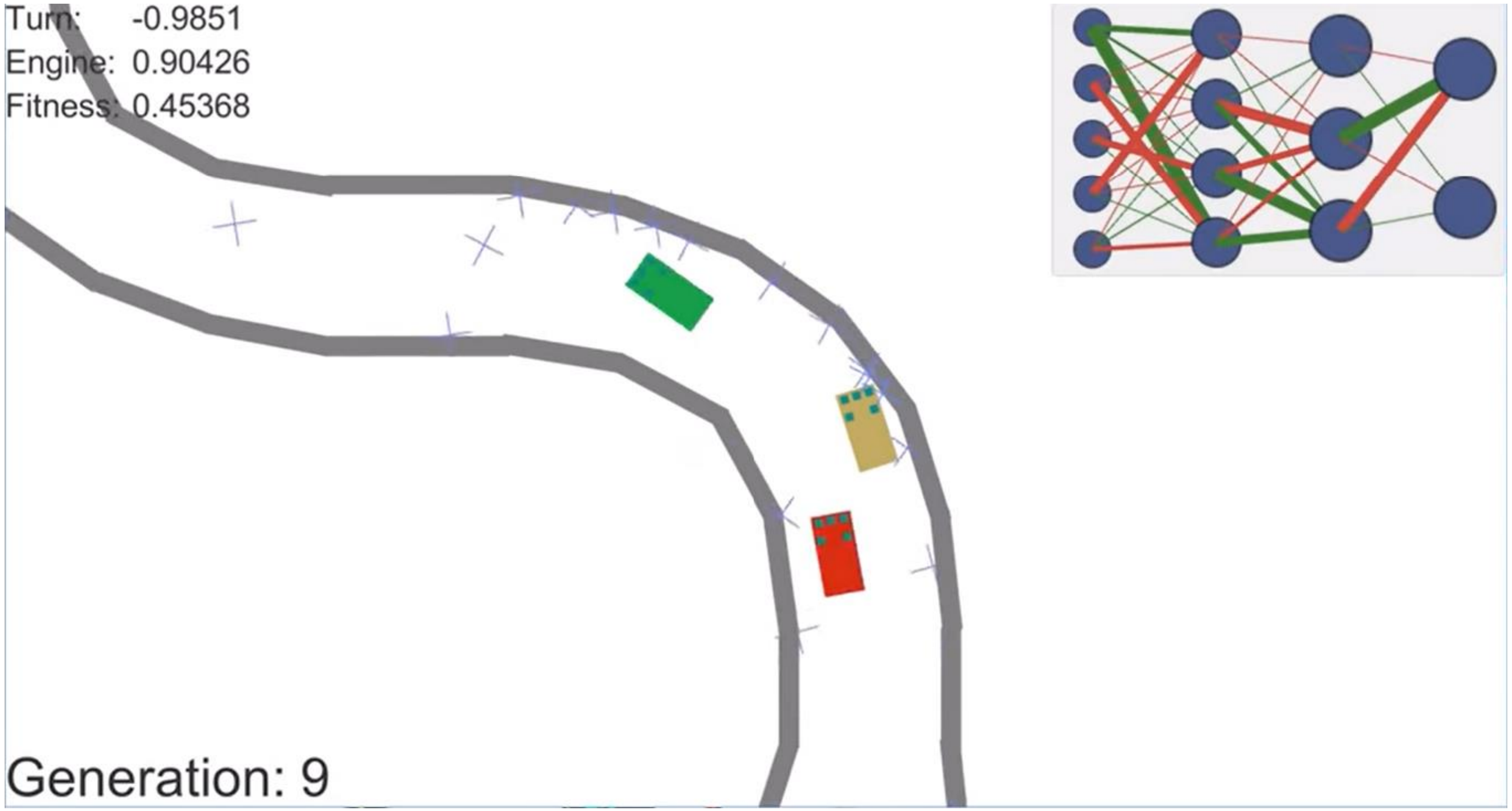
- Examples: Genetic Algorithm, Evolution Strategies

- Which subcategory of ML is this?

Neural Networks + Evolutionary algorithms

[1]

https://youtu.be/Aut32pR5PQA



[1]: https://arztsamuel.github.io/en/projects/unity/deepCars/deepCars.html

Q-Learning[1]

- Traditionally most well known RL algorithm

- Action-Value Function „Q^π“ calculates „value“ of taking a specific action ‚a‘ in state ‚s‘ under current policy ‚π‘:

$$Q^\pi(s_t, a_t) = r_t + \gamma Q^\pi(s_{t+1}, a_{t+1})$$

- Q-Function is used to determine the „quality“ of a policy.

[1]: Christopher Watkins, 1989 „Learning from delayed rewards.“

Q-Learning

- Q-Learning tries to approximate, i.e. „learn“, the optimal Q-Function ‚Q*‘:

$$Q^*(s_t, a_t) = r_t + \gamma \max_{a' \in \mathcal{A}} Q^*(s_{t+1}, a')$$

- Optimal policy can be derived from Q*:

$$\pi(s_t) = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s_t, a)$$

Q-Learning

- Standard Q-Learning update:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

[1]

- Use of ε-greedy policy for exploration
  - Choose optimal action with probability (1 - ε), otherwise random action

[1]: https://en.wikipedia.org/wiki/Q-learning

Deep Reinforcement Learning

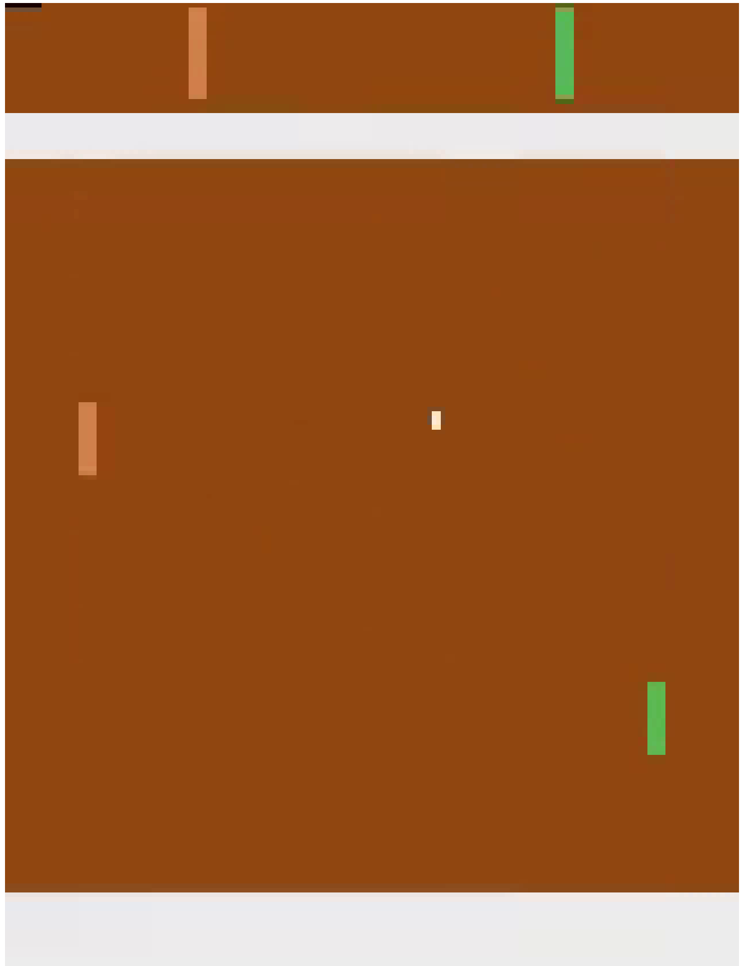
- Deep Q-Network [1]
  - Tabular Q-Function for large state-spaces unfeasible
    Idea: Use Neural Network to approximate Q-Function
- Alpha Go, Alpha Go Zero [2]
  - MCTS, Neural Network approximates Value function, Self-Play
- Newer Algorithms: PPO, A3C, ACKTR, Meta-Learning, …

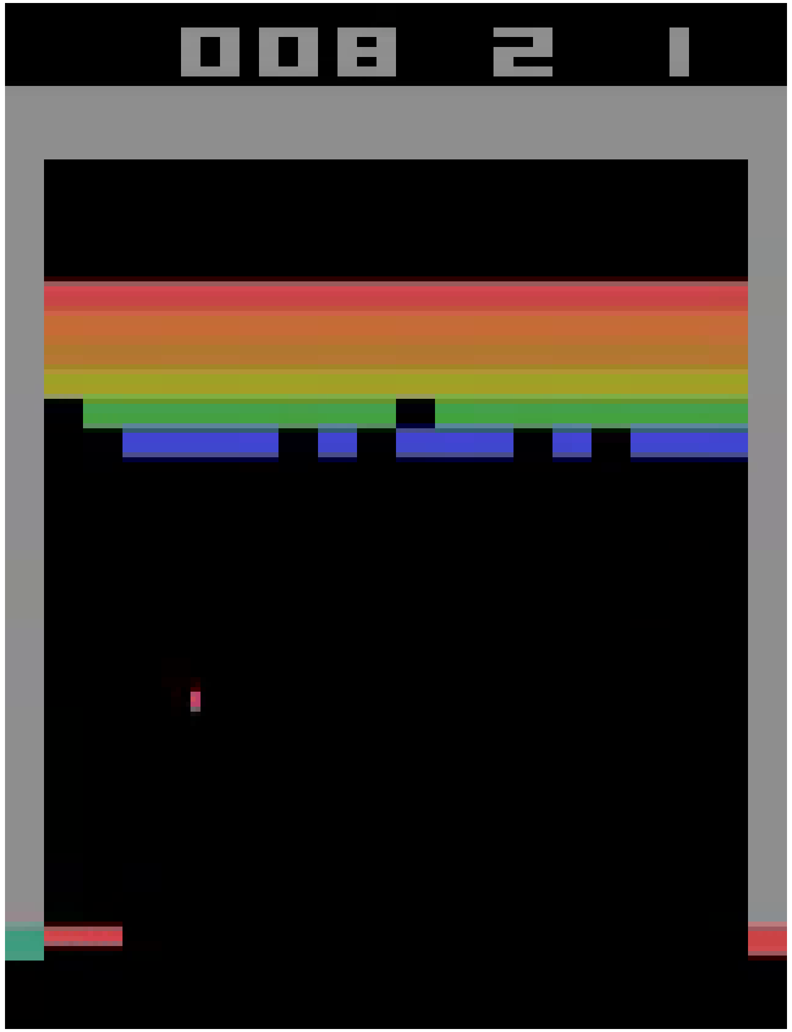
[1]: Mnih et. al., 2015 „Human-level control through deep reinforcement learning."
[2]: Silver et. al., 2017 „Mastering the game of go without human knowledge."

DQN

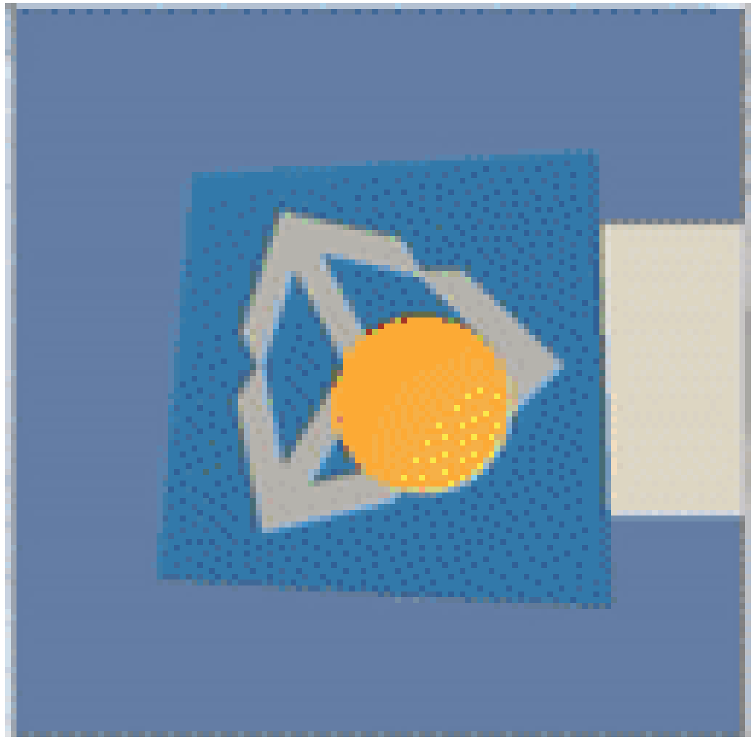
- Use Deep Neural Network to approximate Q*
- Update rule similar to traditional, but now used as loss for SGD
- Use ConvNet for end-to-end learning with pixel inputs
- Requires some extra tricks to work:
  - Separate target network for loss
  - Experience Replay (sampled from for SGD batches)
- Many improvements over the past years: Double DQN, Prioritized ER, Noisy Nets, Dueling DQN, … , Rainbow

DQN on Atari





DQN on more Complex visuals using ML-Agents [1]



[1]: https://unity3d.com/machine-learning

Good starting points

- Genetic Algorithms: Darrel Whitley,
  „A Genetic Algorithm Tutorial."
- RL in general: Sutton and Barto,
  „Reinforcement Learning: An Introduction" (new version in 2017/18)
- Unity ML-Agents:
  https://unity3d.com/machine-learning
- OpenAI Spinning Up in Deep RL (very new!):
  https://blog.openai.com/spinning-up-in-deep-rl/