

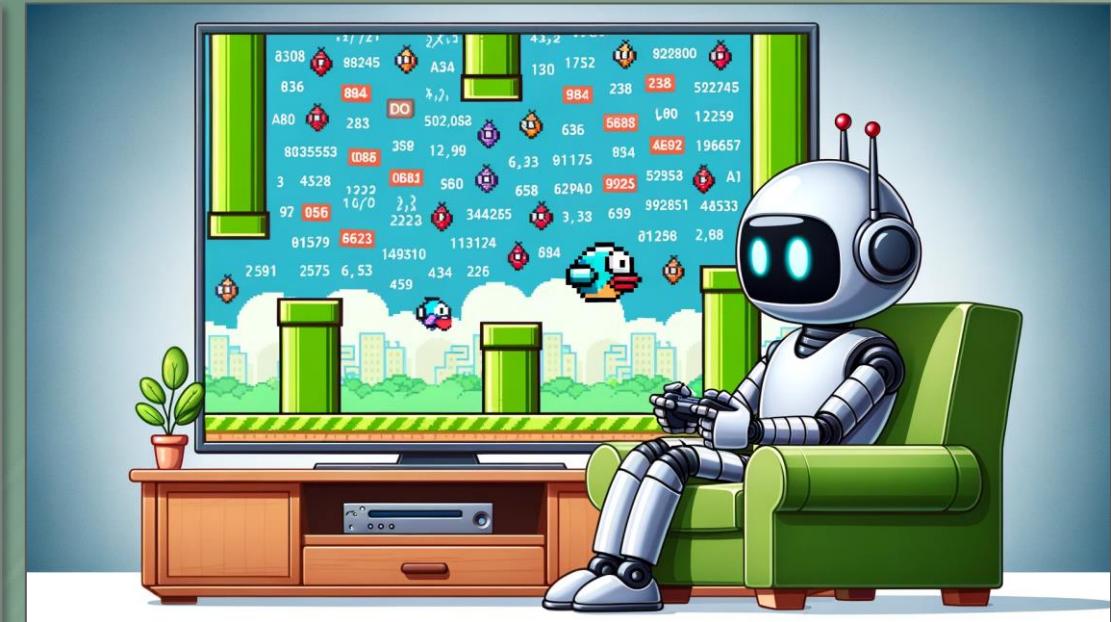


# Deep Q-Learning

Advanced Software-Engineering

Dr. Harald Stein, Prof. Dr.-Ing. Stefan Edlich

Dez 2023



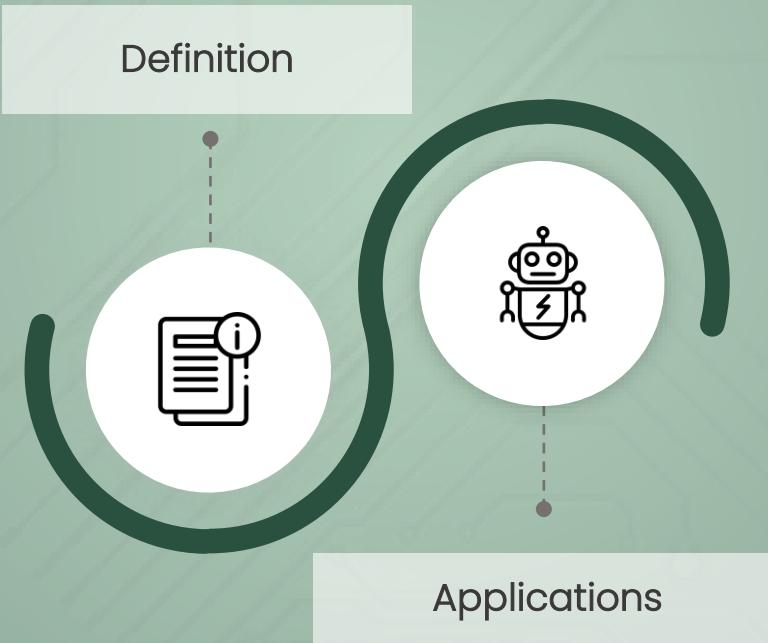
# Agenda

---

- **Deep Reinforcement Learning: Definition and applications**
- **Introduction to Neural Networks**
- **From Q-Learning to Deep Q-Learning**
- **Code example: Flappy Bird**



# Deep Reinforcement Learning: Definition and applications



# What is Deep Reinforcement Learning?

It combines neural networks with reinforcement learning. An agent learns to make decisions by interacting with environment, utilizing deep learning to interpret complex data.

## Key Components

- **Agent:** Entity that takes actions.
- **Environment:** The external setting where the agent operates.
- **State:** Current situation returned by the environment.
- **Action:** Move made by the agent.
- **Reward:** Feedback received after taking an action in a state.

## Benefits

- **Complex Problem Solving:**  
Handles high-dimensional input spaces, such as image data.
- **Generalization:**  
Can be applied to various tasks without task-specific engineering.
- **Continuous Learning:**  
Adapts to new information and changing environments.

# Applications of Deep Reinforcement Learning

## Gaming & Simulation

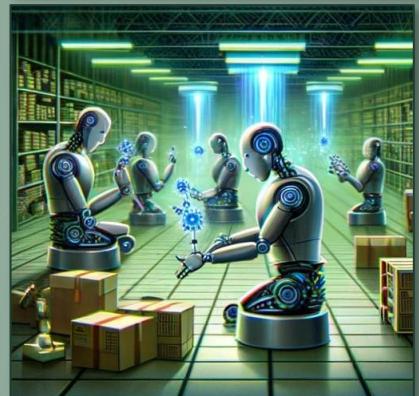
- AlphaGo: Surpassing human performance in board games like Go.
- Game Playing AI: Mastering video games like from Atari, Nintendo and even complex games like Dota 2 and StarCraft II.

## Robotics & Automation

- Robotic Manipulation: Teaching robots to perform intricate tasks like folding laundry or assembling items.
- Autonomous Navigation: Enabling vehicles, drones, robots, etc. to navigate in dynamic environments.

## Healthcare

- Personalized Treatment: Tailoring medical interventions based on patient data.
- Drug Discovery: Accelerating the process of drug creation and testing.



# Applications of Deep Reinforcement Learning

## Finance

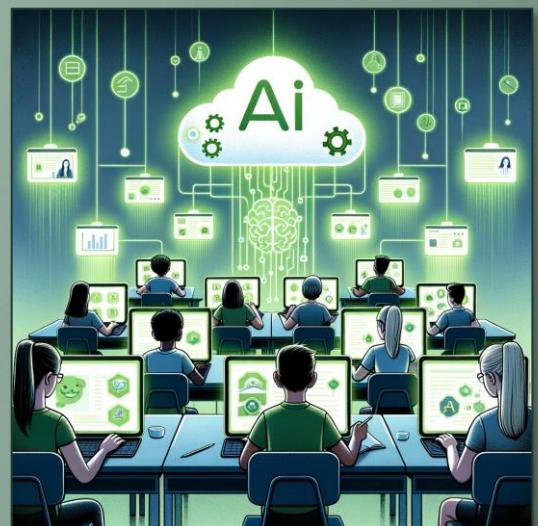
- Portfolio Management: Optimizing investment strategies.
- High-frequency Trading: Making rapid stock trading decisions.

## Energy

- Grid Management: Optimizing energy distribution in real-time.
- Energy Consumption: Predicting, optimizing energy usage in buildings.

## Recommendation Systems

- Personalized Content: Recommending movies, songs, or shopping items.
- Adaptive Learning: Personalizing online learning experiences for students.



# AlphaGo

... has revolutionized Game AI with Deep Learning in March 2016

## What is AlphaGo?

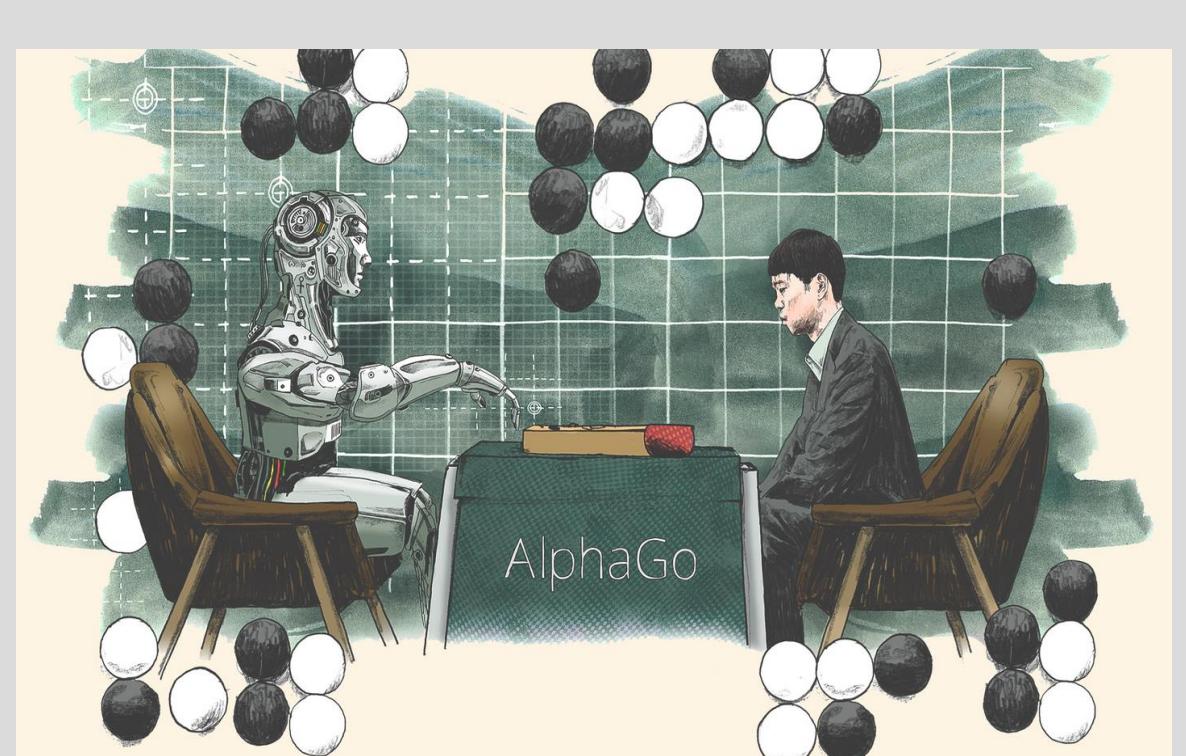
- Developed by DeepMind, a subsidiary of Google.
- First AI to defeat a world champion Go player, Lee Sedol, in 2016.

## Key Innovations

- Combination: Monte Carlo Tree Search (MCTS), deep neural networks.
- Utilized both policy networks (for move suggestions) and value networks (to evaluate board positions).

## Significance of the Victory

- Go is a highly complex game with more possible moves than atoms in the universe.
- Demonstrated the potential of AI to handle intricate tasks beyond games.



# The Road to Autonomous Driving

... is optimization approach for minimizing error in neural network. It calculates gradient of error function with respect to each weight by using chain rule.

## What is Autonomous Driving?

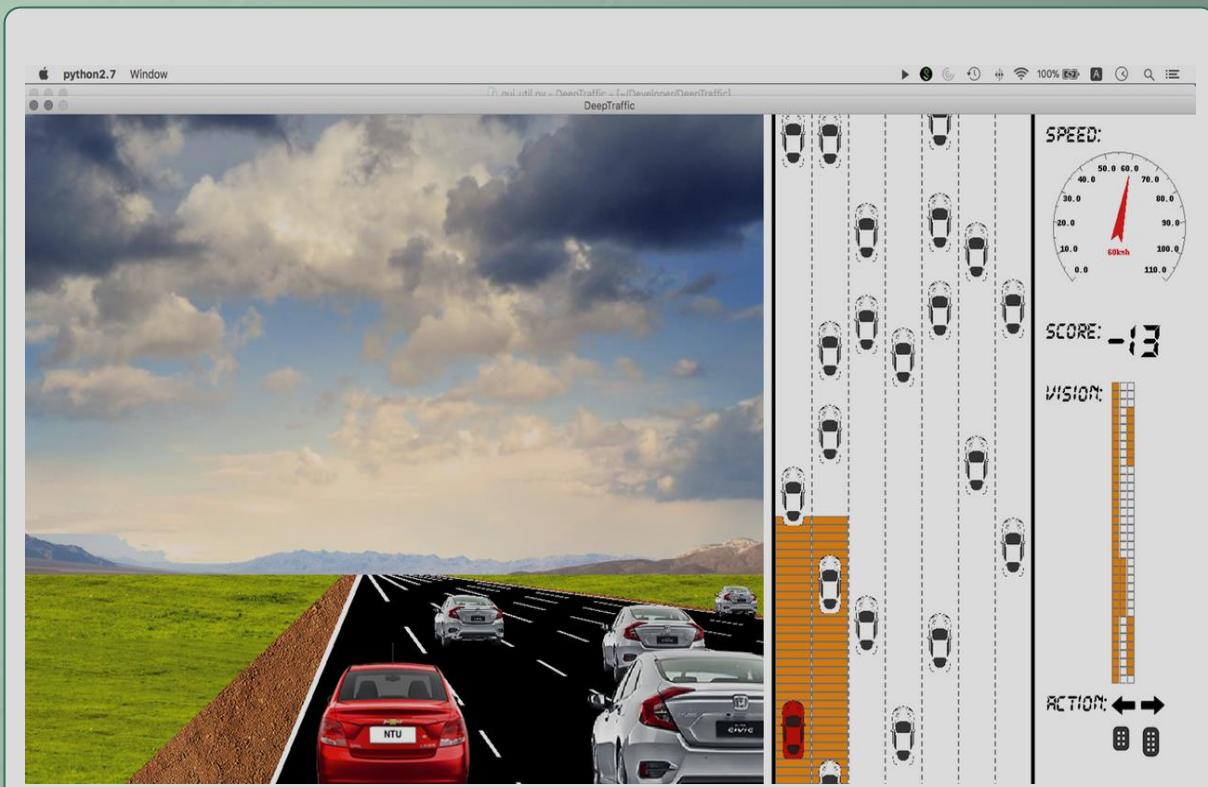
- Vehicles equipped with technology to navigate and control without human intervention.
- Levels of autonomy: 0 (no automation) to 5 (full automation).

## Benefits of Autonomous Vehicles

- Potential reduction in traffic accidents caused by human error.
- Improved fuel efficiency and traffic flow.
- Enhanced mobility for the elderly and physically challenged.

## Challenges Ahead

- Legal and regulatory hurdles.
- Ethical considerations: Decision-making in critical scenarios.
- Ensuring safety and reliability in diverse conditions.



# Reinforcement Learning in Gaming

## Atari & Nintendo

### Breakthrough with Atari

- DeepMind's Deep Q-Network (DQN): Combined deep neural networks with RL.
- Achieved human-level performance on games like Breakout, Pong, and Space Invaders.

### Scaling to Nintendo & Beyond

- Enhanced algorithms tackled more complex environments.
- Managed intricate games like Super Mario Bros using policy gradient methods.

### Key Innovations in RL for Gaming

- Experience Replay: Storing and reusing past experiences to break correlations.
- Epsilon-greedy exploration: Balancing exploration and exploitation.
- Model-based RL: Incorporating game models for better planning.

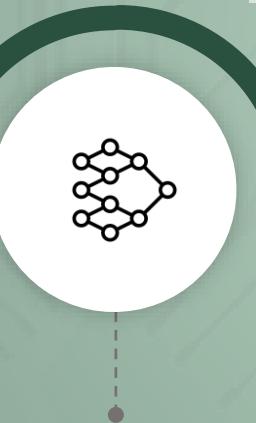


# Introduction to Neural Networks

Feed Forward  
Neural Networks



Backpropagation  
with Gradient  
descent



Example: Image  
classification



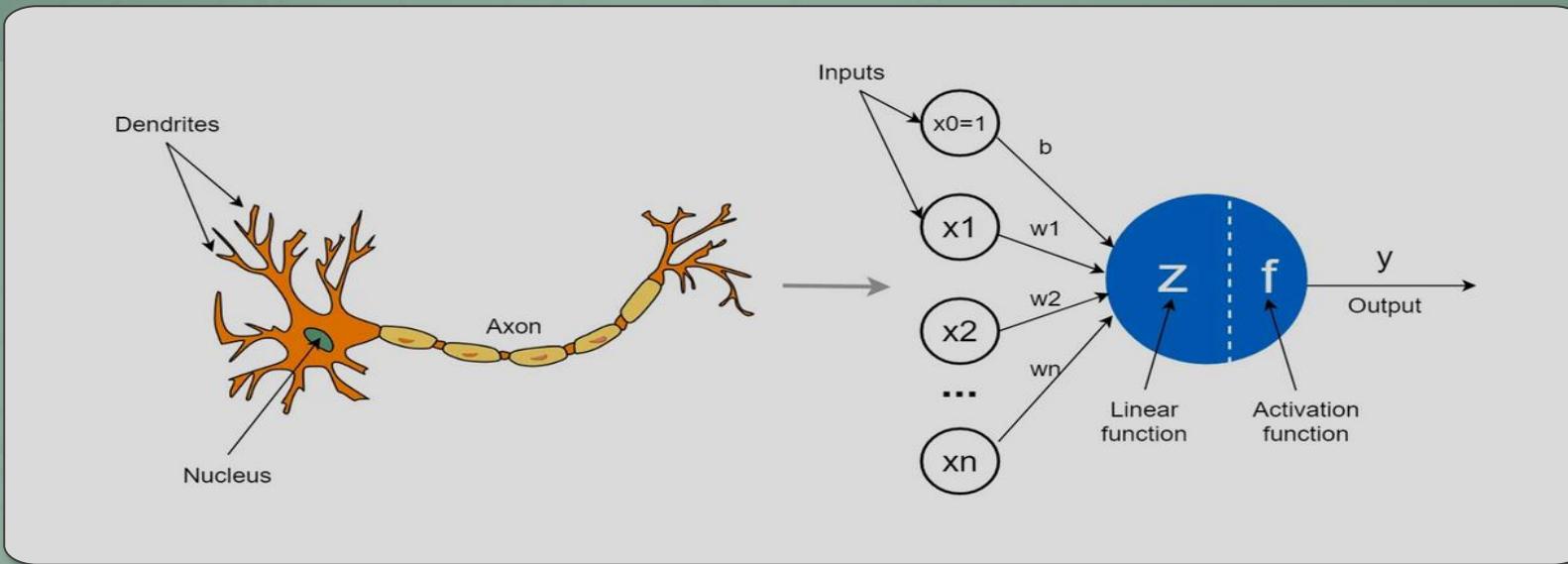
Activation functions

Convolutional  
Neural Nets (CNN)

Universal  
approximation  
theorem

# The neuron

... is basic processing unit in neural networks. It receives multiple inputs, multiplies each by weight, sums the products, passes result through activation function to produce output.



## Biological Neuron

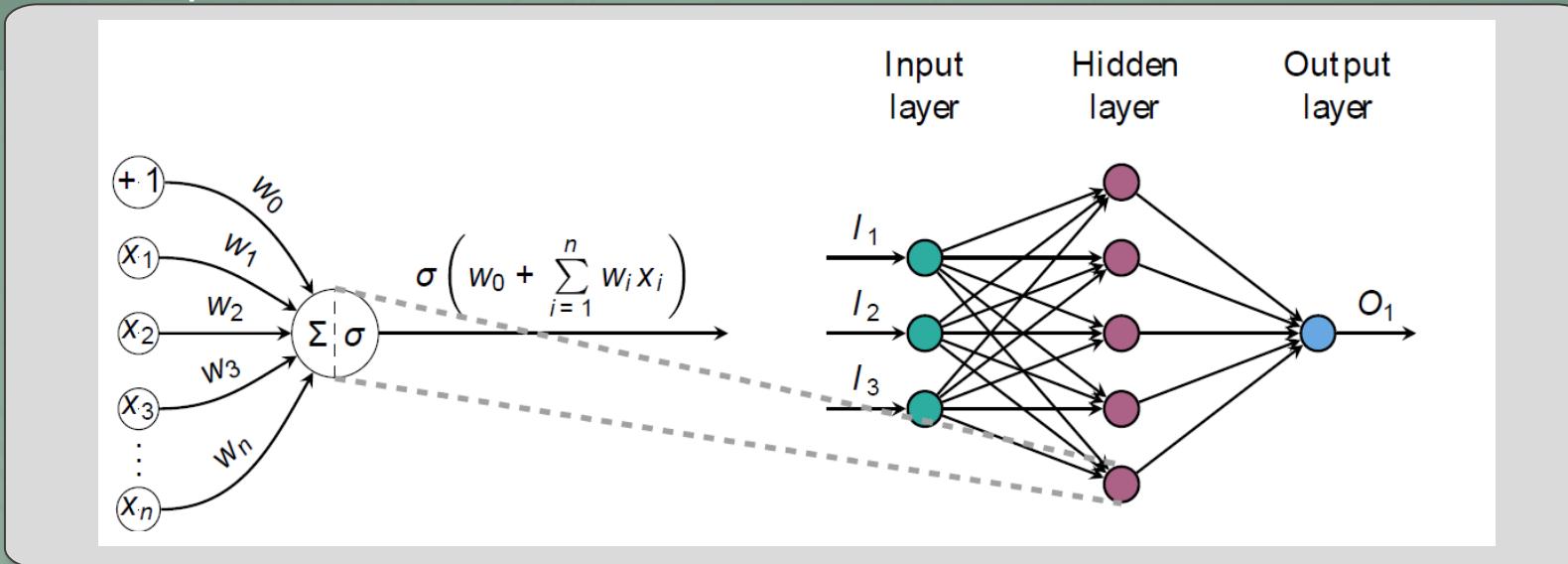
- **Dendrites:** Tree-like structures, receive signals from other neurons.
- **Axon:** Carries messages away from the neuron.
- **Nucleus:** Control center of the neuron.

## Artificial Neuron

- **Inputs:** Data fed into neuron, including bias ( $x_0=1$ ).
- **Weights ( $w_1, \dots, w_n$ ):** Determine importance of each input.
- **Linear Function ( $Z$ ):** Weighted sum of the inputs.
- **Activation Function ( $f$ ):** Transforms linear function output to generate final output ( $y$ ).
- **Output ( $y$ ):** Resulting value after input processing

# Anatomy of a Feed Forward Neural Network

This architecture learns complex relationships between input data and desired outputs, whether it be class probabilities or numeric values of interest.



**Artificial neuron computes output using weighted inputs and activation function**

- $+1$ : Bias input.
- $x_1, x_2, \dots, x_n$ : Input data points.
- $w_1, w_2, \dots, w_n$ : Associated weights for each input.
- $\Sigma$ : Represents the summation of weighted inputs.
- $\sigma$ : Activation function transforming summed input  $\rightarrow$  output.

**Flow of data is shown from input layer through hidden layer to output layer in one-layer fully-connected network (FCN)**

- **Input Layer:** Initial data inputs, exemplified by age, smoking status, and left ventricle ejection fraction.
- **Hidden Layer:** Intermediate layer where data is processed using weights and activation functions.
- **Output Layer:** Final processed data or prediction.

# Activation functions

... dictate output of a neuron, helping neural networks model diverse and intricate relationships in data.

## Purpose

Introduce non-linearity into network, enabling it to learn complex patterns.

## Types

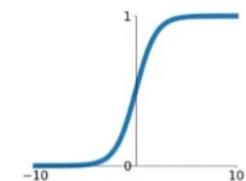
- **Sigmoid:**  
Maps input values 0 to 1. Suitable for binary classification.
- **ReLU (Rectified Linear Unit):** Maps negative values to 0 and retains positive values. Commonly used in deep networks.
- **Tanh:**  
Maps input values -1 to 1, providing a zero-centered output.
- **Softmax:**  
Converts vector → probability distribution. Ideal for multi-class classification.

## Choice

Activation function depends on problem type (regression, classification) and architecture of neural network.

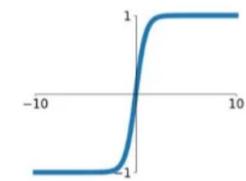
### Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



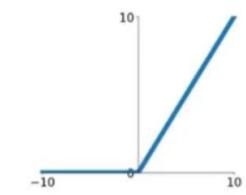
### tanh

$$\tanh(x)$$



### ReLU

$$\max(0, x)$$



LOGITS  
SCORES



SOFTMAX

PROBABILITIES

$$y \begin{bmatrix} 2.0 \rightarrow \\ 1.0 \rightarrow \\ 0.1 \rightarrow \end{bmatrix} S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \begin{bmatrix} \rightarrow p = 0.7 \\ \rightarrow p = 0.2 \\ \rightarrow p = 0.1 \end{bmatrix}$$

# Backpropagation (of errors)

... is optimization approach for minimizing error in neural network. It calculates gradient of error function with respect to each weight by using chain rule.

## Process

### 1. Feedforward Pass:

Input a dataset and compute the predicted output using current weights.

### 2. Compute Loss:

Calculate error, i.e. difference: predicted output  $\leftrightarrow$  actual target.

### 3. Backward Pass:

Propagate loss backward through network, calculating gradients for each weight.

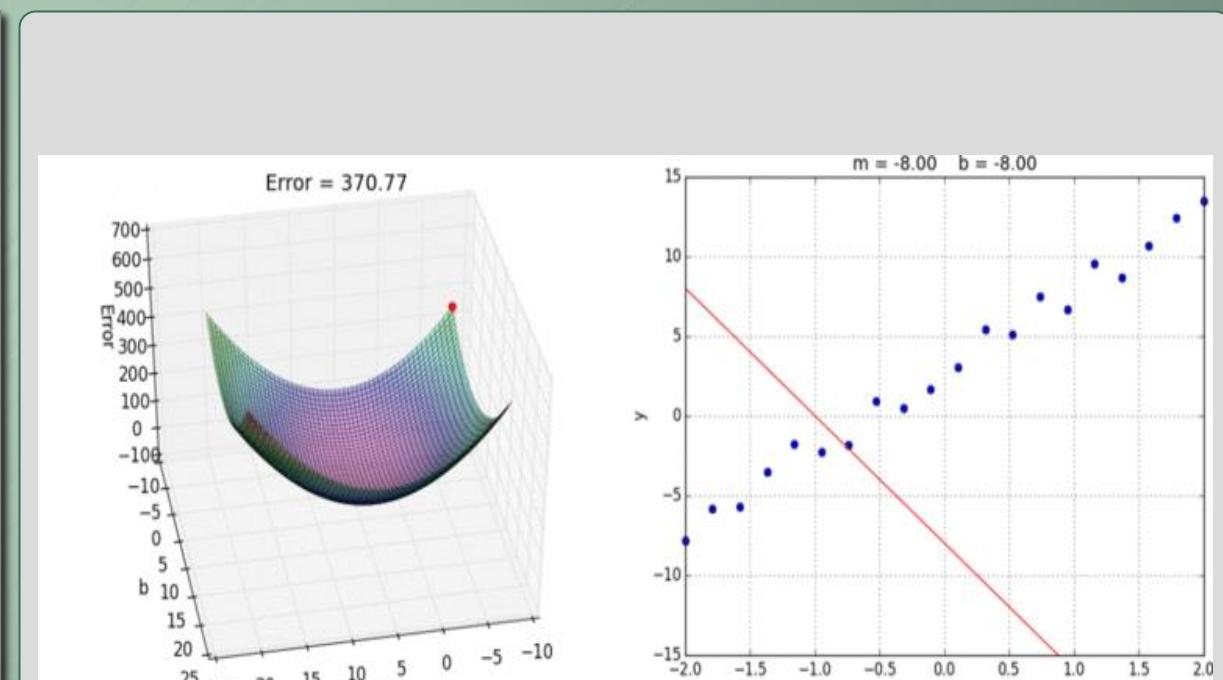
### 4. Weight Update (using Gradient Descent):

Once computed gradients are used to update weights of network.  
Aim: to adjust weights in direction that minimizes error.

### 5. Iterate 1-4

## Challenges

- Can get stuck in local minima  
(mitigated by variants like stochastic gradient descent).
- Sensitive to hyperparameters and initial weight values.



Animated

# Gradient Descent

... steers the model towards the best parameters by navigating the cost function landscape, aiming to find the lowest point (global minimum)

## Purpose

Algorithm to minimize the cost function, i.e. errors in fulfilling the task like false classification, guiding the model towards optimal parameters.

## Mechanism

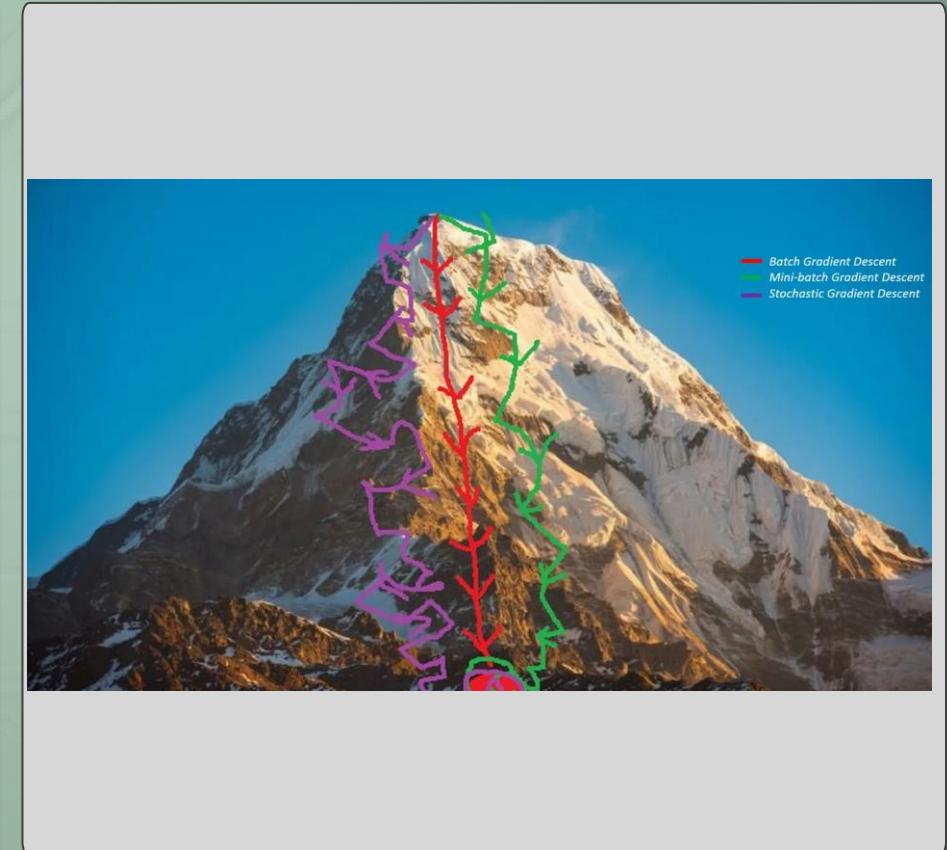
Iteratively adjusts model parameters in direction of steepest decrease in cost.

## Learning Rate

Determines step size during each iteration. A high rate might overshoot the minimum, while a low rate might be too slow.

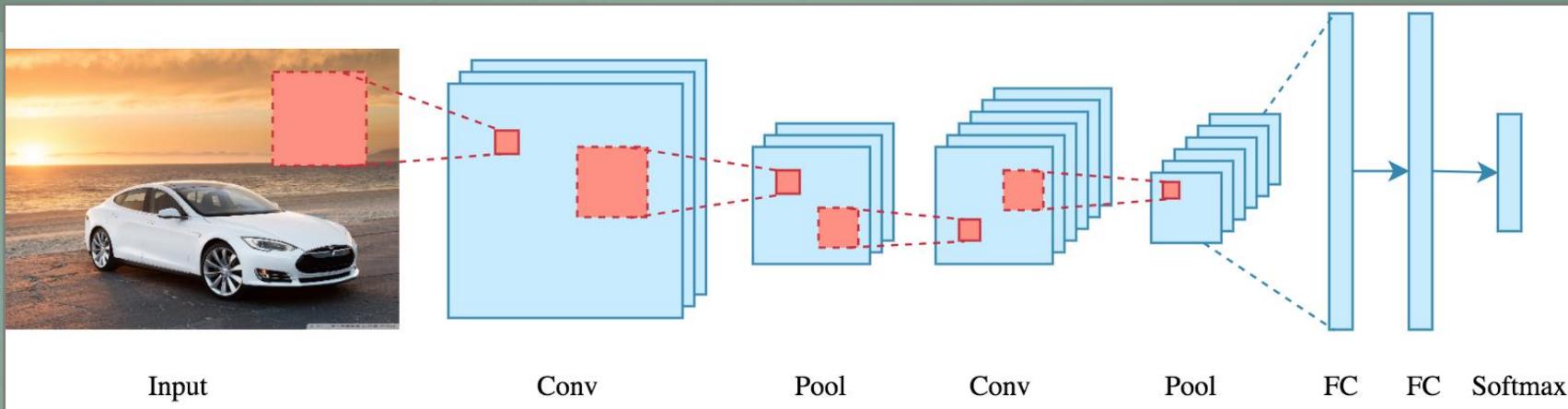
## Types

- **Batch Gradient Descent:** Uses entire training dataset to compute gradient.
- **Stochastic Gradient Descent (SGD):**  
Uses single training example per step. Faster but more erratic.
- **Mini-Batch Gradient Descent:**  
Compromise between Batch and SGD. Uses subset of training data.



# Convolutional Neural Networks (CNNs)

... process spatial data layer by layer, extracting hierarchical features that enable precise and complex recognition tasks.



## Inspired by

visual processing in human brain. Particularly tailored for spatial data

## Applications

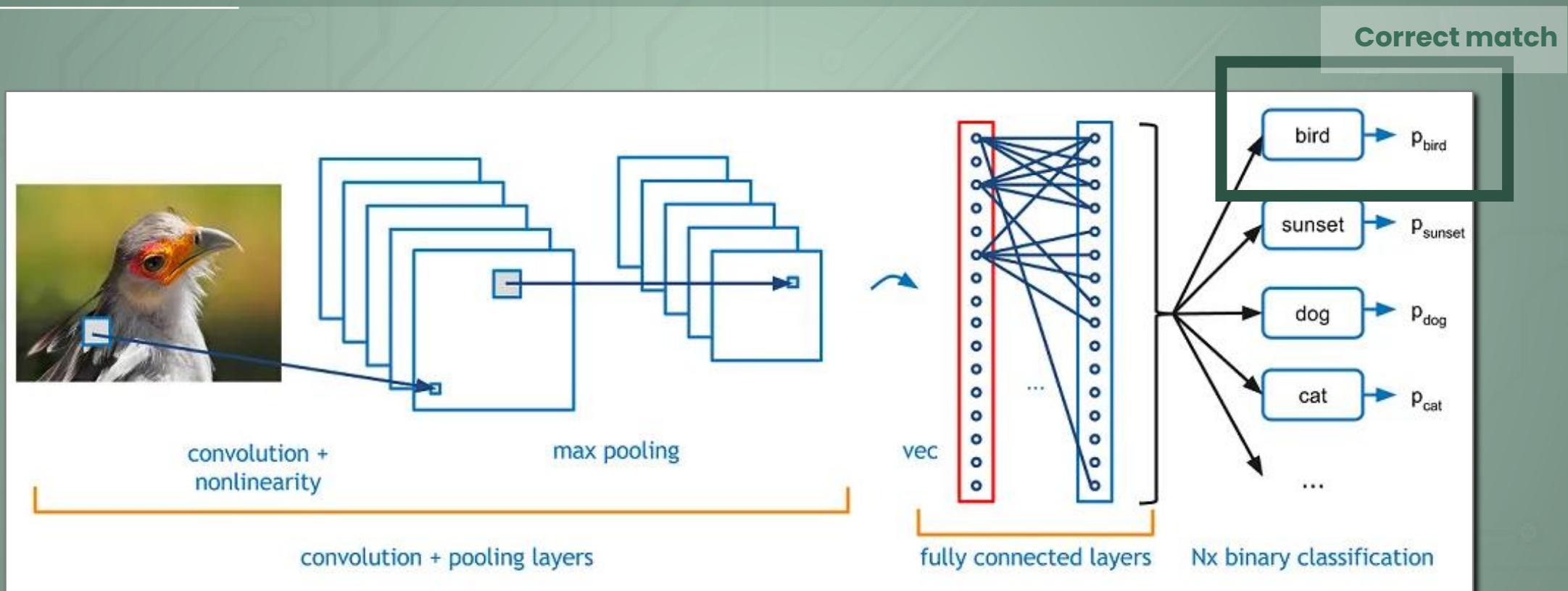
- **Images:** Classification (e.g. face recognition), segmentation (e.g. object detection for autonomous driving), medical image analysis
- **Style Transfer:** Applying artistic styles from one image to another.

## Main Components

- **Convolutional Layer:** Uses filters to scan input data, extract features. Captures spatial hierarchies.
- **Pooling Layer:** Reduces spatial dimensions (downsamples) without losing important feature information.
- **Fully Connected Layer:** Processes features extracted by convolutional and pooling layers, typically culminating in output classifications.

# Image classification with CNN

If the Neural net is correctly trained it classifies the image to the string “bird” with highest probability.



# Universal approximation theorem

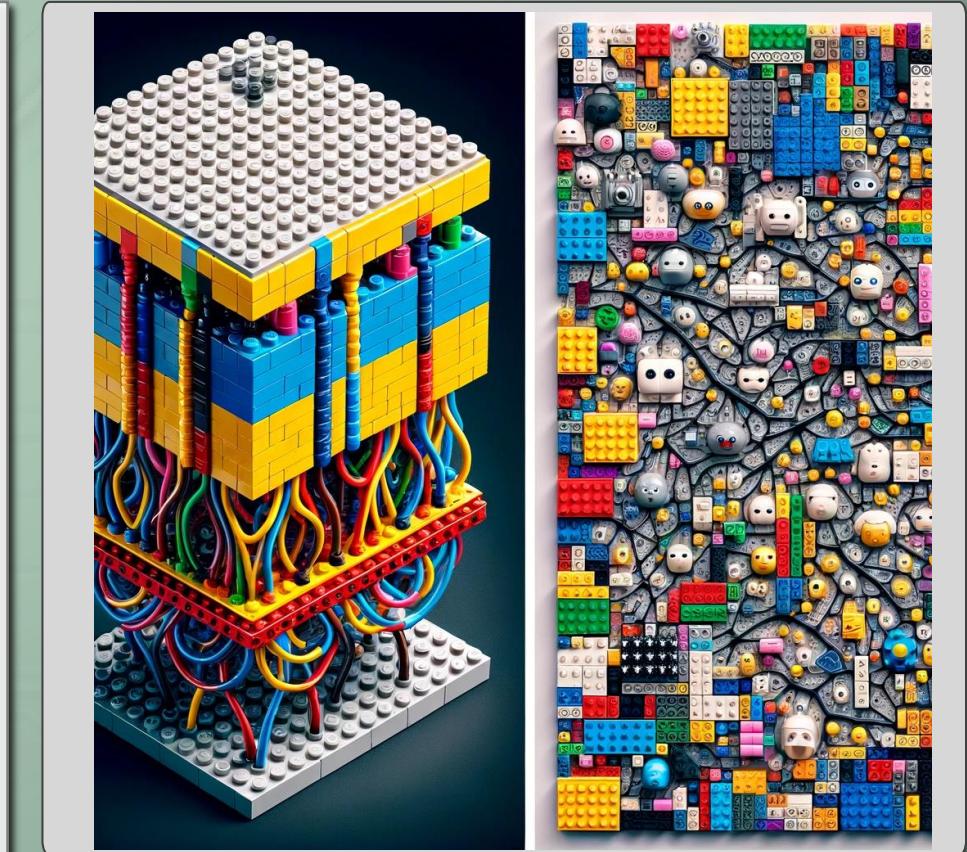
... states that feed-forward network with single hidden layer (or more) containing finite number of neurons can approximate any continuous function, given suitable activation function.

## Implication to Q-Learning

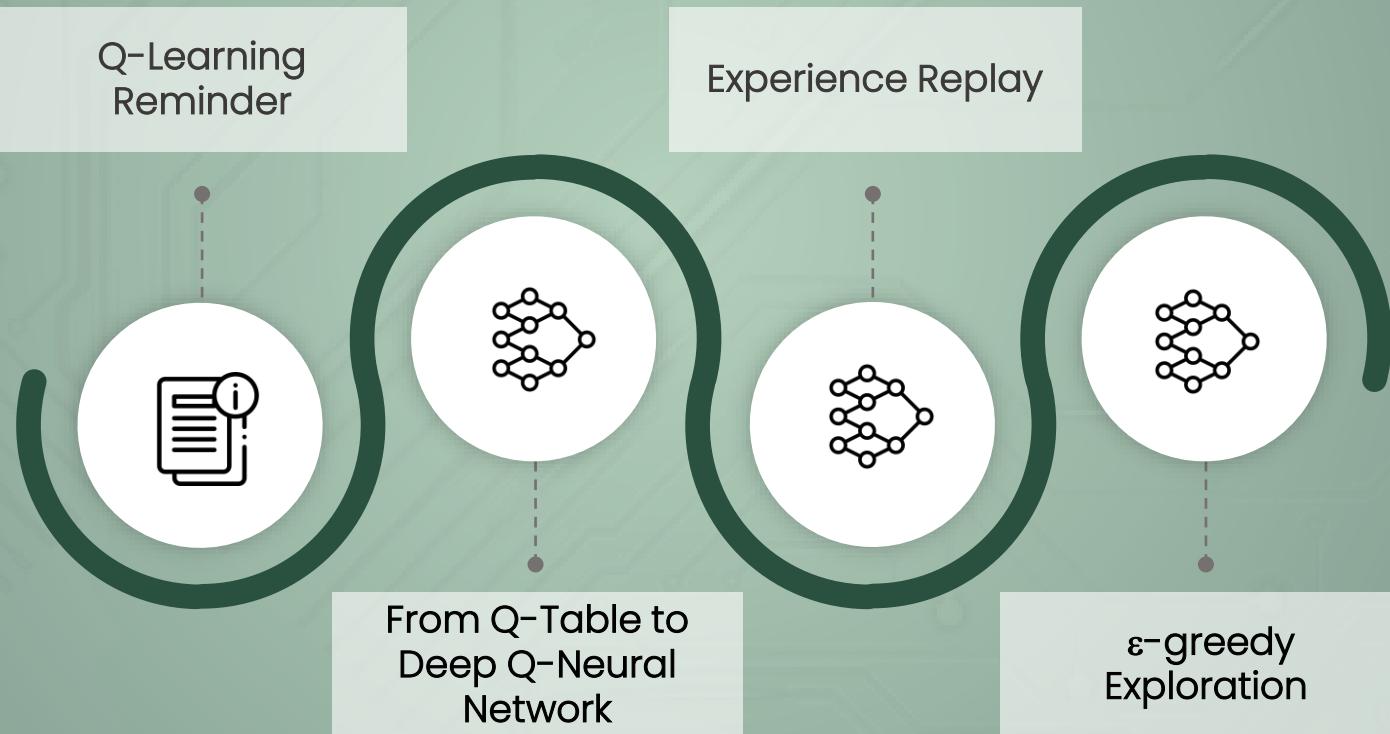
- Neural networks can be used to approximate any function, i.e. also Q-function in Q-learning.
- Given sufficient data and training, a neural network can estimate the expected reward for different actions in various states.

## Figure on the right side

- Depicts approximation of arbitrary dataset by Neural Network
- **Left:** Feed-forward neural network with its layers and connecting nodes.
- **Right:** LEGO bricks are assembled to symbolize a random dataset, emphasizing the diversity and intricacy of data.



# From Q-Learning to Deep Q-Learning



# Reinforcement Learning process

... is a Markov decision process that provides mathematical framework for modeling decision-making when outcomes are partly random, partly under decision-maker's control

- 5-tuple of  $(S, A, T, R, \gamma)$  with
  - $S$  the state space
  - $A$  the action space
  - $T$  the transition function
  - $R$  the reward function
  - $\gamma$  the discount factor

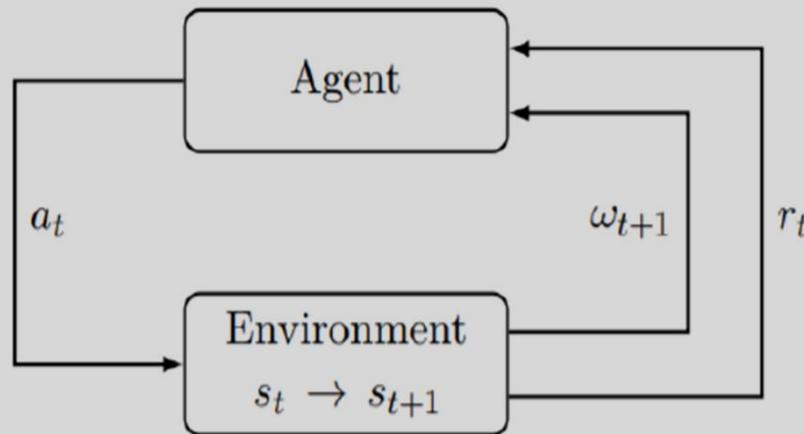


Figure 1: The agent-environment interaction [1]

- Theoretical Framework of Reinforcement Learning
- The agent wants to maximize his rewards

## Markov Property:

implies that our agent needs only current state to decide what action to take without the history of all states and actions they took before.

# Finding optimal policy

... by iterating between policy and value

## Policy Iteration

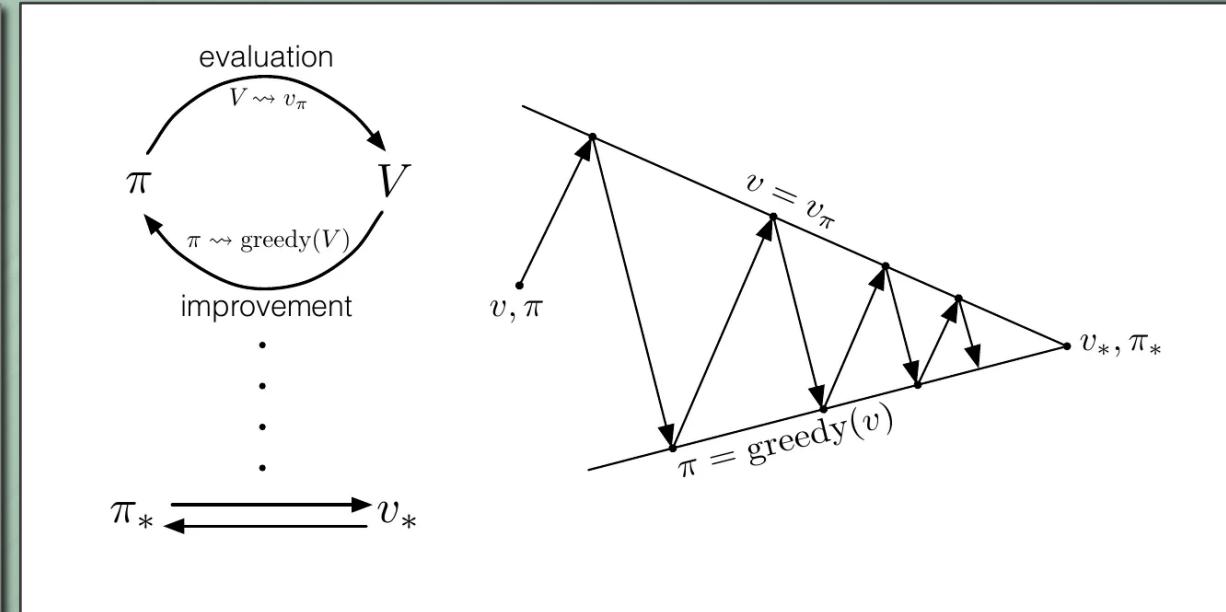
- Evaluation: Circle showing  $V \approx \pi$  (value function approximates policy).
- Improvement: Policy is refined using  $\pi \approx \text{greedy}(V)$

## Convergence

- Zig-zag arrows: Iterative steps from initial  $(v, \pi)$  to optimal  $(v^*, \pi^*)$

## Optimal Policy and Value

- Process concludes when reaching optimal policy  $\pi^*$  and corresponding value function  $v^*$ , represented at far right of image.



# Q-Function

... expected return of taking action  $a$  in state  $s$ . Captures immediate rewards and potential future gains. Q-Values: updated iteratively based on agent's experiences for deriving optimal policy

$$\text{New } Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \operatorname{argmax}_{a'} Q(s', a') - Q(s, a)]$$

- █ New Q Value for that state and the action
- █ Learning Rate
- █ Reward for taking that action at that state
- █ Current Q Values
- █ Maximum expected future reward given the new state ( $s'$ ) and all possible actions at that new state choosing the best available action regardless of policy function
- █ Discount Rate

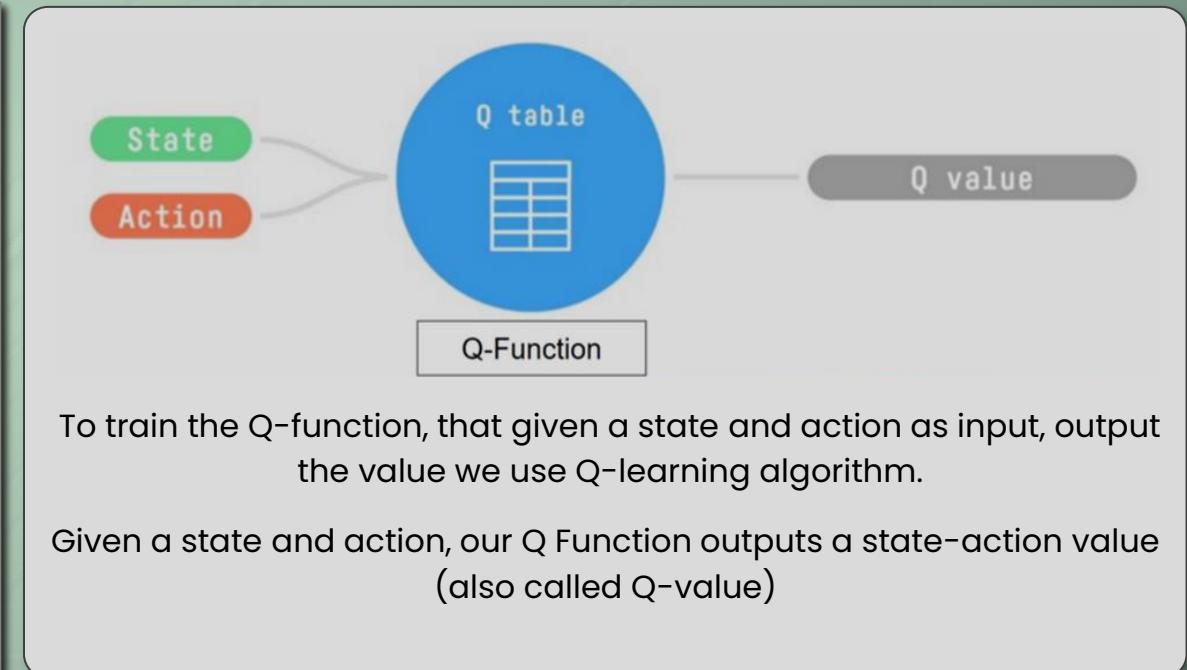
# Implementing Q-Function with Q-Table

## Implementing the Q-Function

- The Q-Table essentially implements the Q-Function.
- The agent consults the Q-Table to decide which action to take in a given state, usually choosing the action with the highest Q-Value.
- Over time, with enough exploration and consistent updates, the Q-Table converges to the optimal Q-Function,  $Q^*(s, a)$ .

## Be aware of exploration-exploitation dilemma

- Ensure adequate exploration of the state-action space to find the optimal policy
- i.e. do not always use Q-Table for determining next step, but to it randomly in small share of cases  $\epsilon$



# From Q-Table to Deep Q-Neural Network

From Tabular to Neural Representation

## Limitations of Q-Table

- suitable for environments with small number of states and actions
- Scalability: Infeasible for environments with large number of states.
- Does not generalize to unseen states.

## Deep Q-Network (DQN): The Evolution

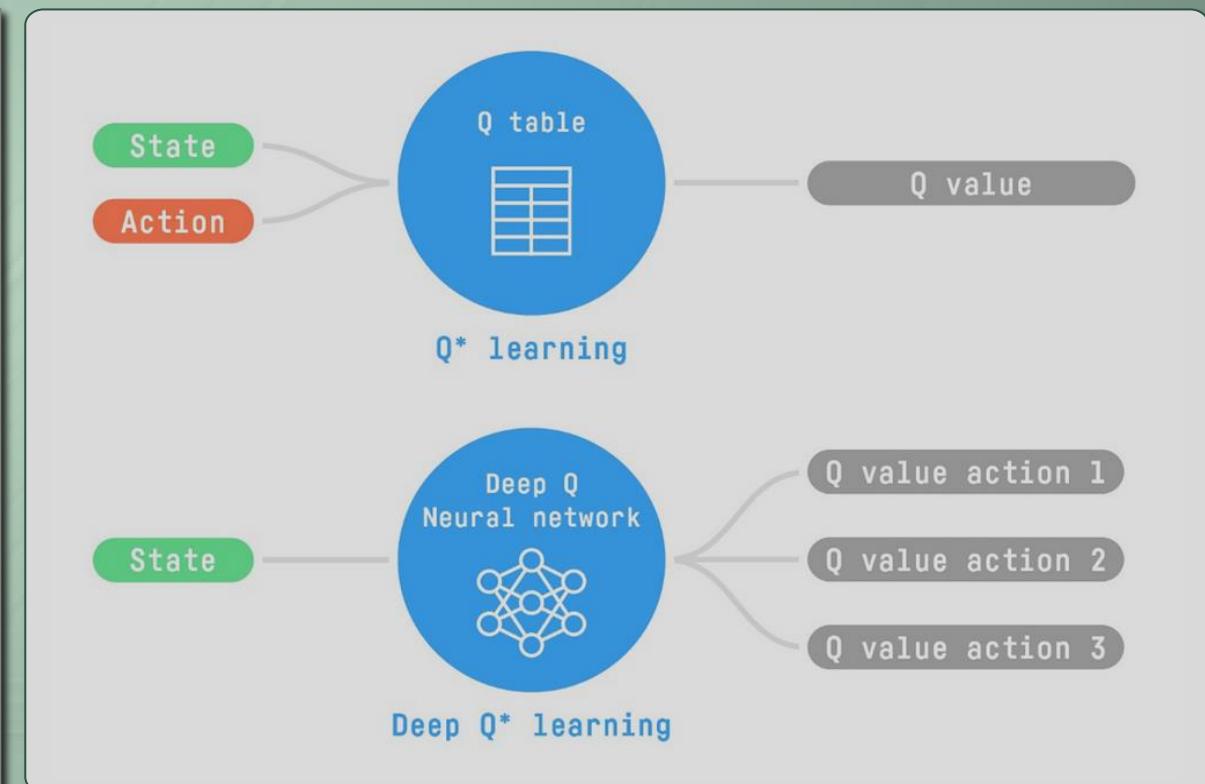
- Neural network approximation of the Q-function.
- Allows for generalization over similar states.
- Can handle continuous state spaces.

## Advantages of DQN

- Scalability: Can work with large and complex environments.
- Flexibility: Adapts to changing environments.

## Challenges

- Stability during training.
- Balancing exploration and exploitation.



# Stabilizing Deep Q-Learning with Experience Replay

## Enhancing Deep Q-learning Efficiency

### What is Experience Replay?

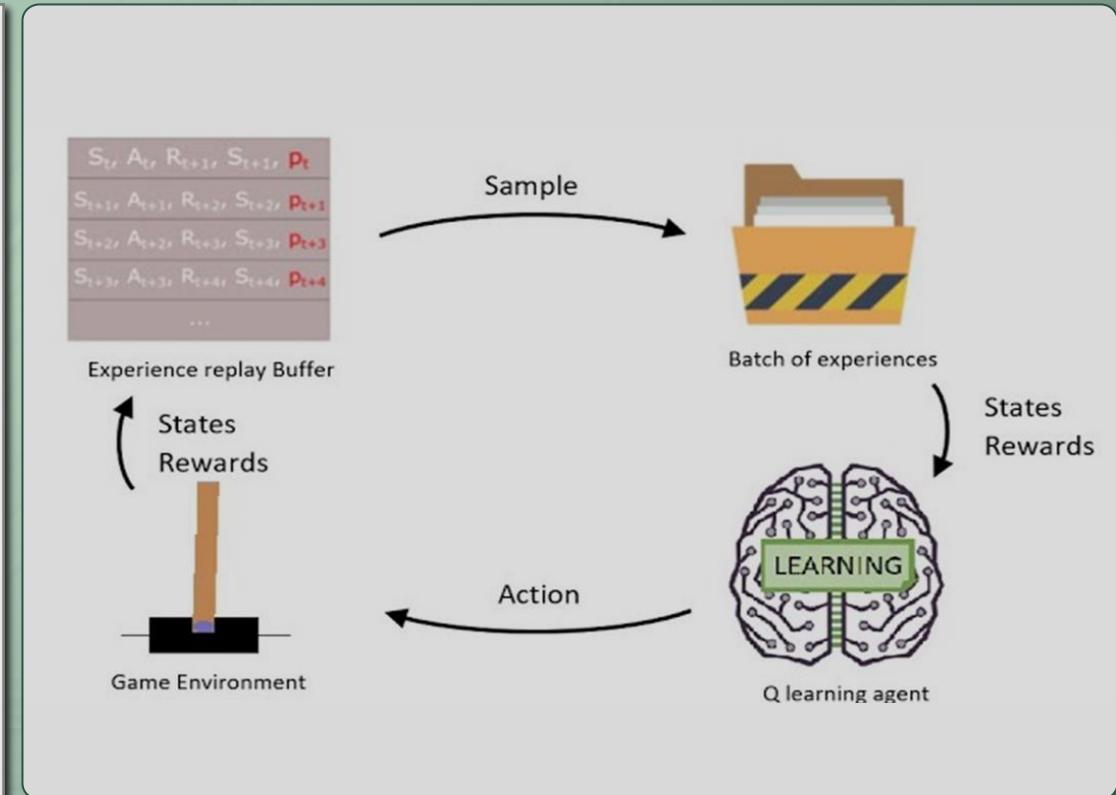
- A mechanism to store and reuse past experiences.
- Creates a "replay memory" of state, action, reward, next state.

### Why Experience Replay?

- Breaks harmful temporal correlations:  
Learning on consecutive samples can lead to inefficient learning.
- Increases sample efficiency:  
Each experience can be reused multiple times.
- Stabilizes training by reducing variance.

### Working of Experience Replay:

- Collect experiences during episodes and store them in a buffer.
- Randomly sample from the buffer to train the network.
- Older experiences get replaced by newer ones.



# $\epsilon$ -greedy Exploration in Deep Q-learning

Balancing exploration and exploitation in Deep Q-Networks. With probability  $\epsilon$  random action is taken for exploration. Otherwise best known action is taken.

## Why is Exploration Important?

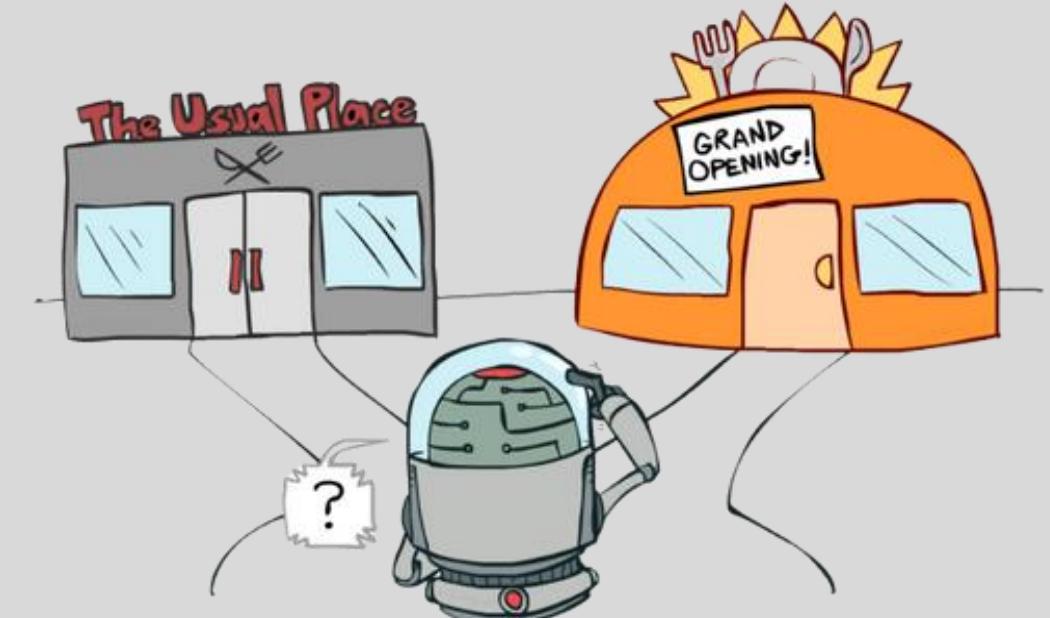
- Ensures global optimality: Prevents getting stuck in local optima.
- Gathers more information about the environment: Crucial for environments with uncertain rewards or dynamics.

## Decaying Epsilon

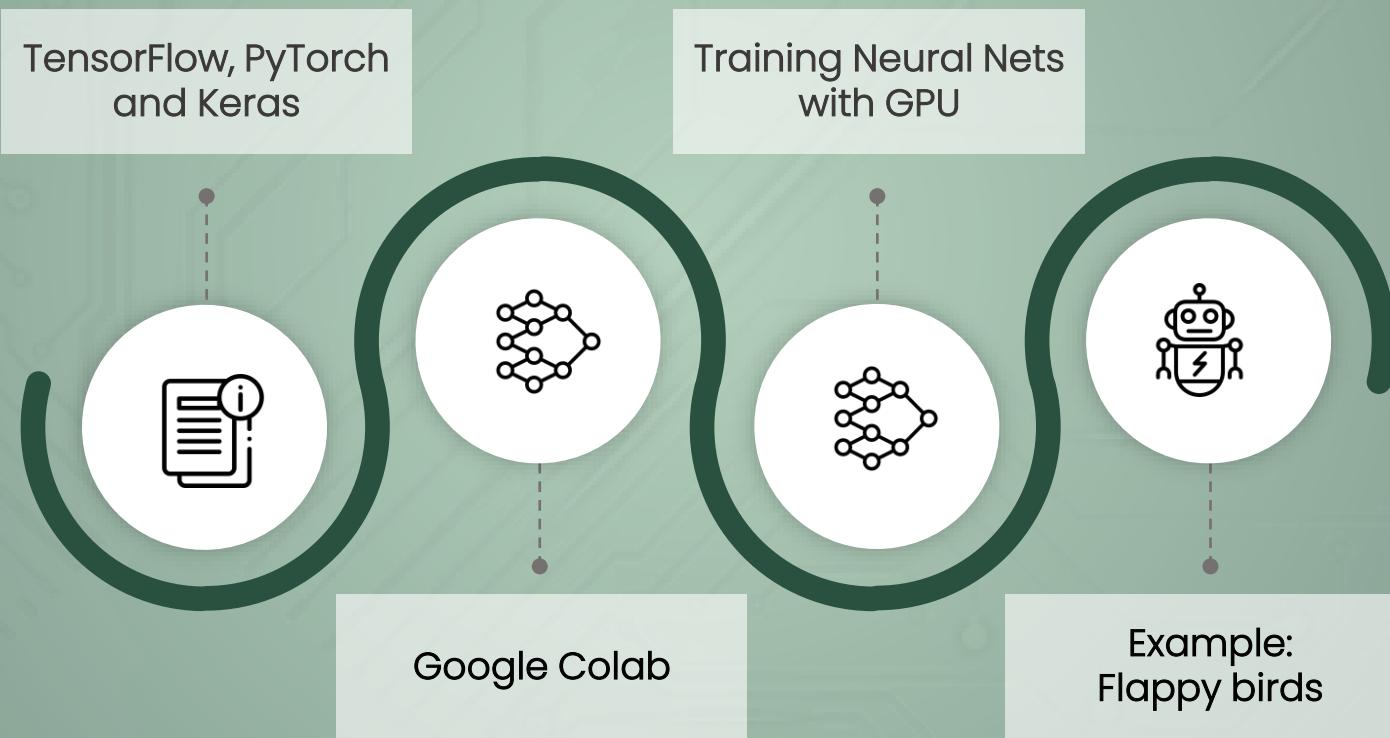
- Start with high exploration (high  $\epsilon$ ) and reduce over time.
- Allows agent to explore widely initially and refine its policy as it learns.

## Challenges & Considerations

- Setting the right initial value of  $\epsilon$ .
- **Choosing the decay rate:**  
Too fast can miss out on exploration; too slow can delay convergence.



# Use Case: autonomous agent for video game



# TensorFlow, PyTorch and Keras

... are popular open-source libraries used for machine learning and deep learning.  
Keras is API built on top of TensorFlow, PyTorch and others

1 TensorFlow		2 PyTorch	3 Keras
DEVELOPED BY	Google Brain team	Facebook's AI Research lab	François Chollet
EASE OF USE	Steeper learning curve	Easier to learn	Extremely user-friendly
FLEXIBILITY / CUSTOMIZATION	Very flexible	Very flexible	Limited flexibility
PERFORMANCE / SPEED	Excellent performance	Excellent performance	Performance is dependent on the backend used
USE CASES	Suitable for both production and research, especially in large-scale systems	Preferred for research, rapid prototyping, and academic purposes	Best for quick development of standard neural networks, less suited for research

# Google Colab

... is a cloud-based platform that provides free Jupyter notebook environment, requiring no setup and running entirely in the cloud. Offers GPU support for fast computation

## Features

- **Zero Configuration:** Start coding without any installations.
- **Interactive Visualizations:** Utilize libraries like Matplotlib, Seaborn, etc.

## Applications

- Machine Learning & Data Analysis.
- Educational Purposes & Tutorials.
- Prototyping and quick experiments.

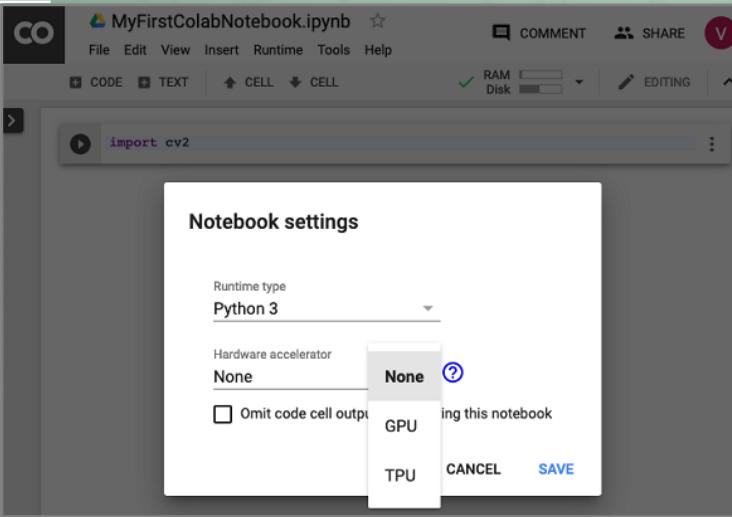
## Limitations

- Limited session durations.
- Memory restrictions with GPU/TPU usage.

The screenshot shows the Google Colab interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' options. On the right side of the bar are 'Share', 'Sign In', and other account-related buttons. Below the bar, there are tabs for '+ Code', '+ Text', and 'Copy to Drive'. To the right of these tabs are 'Connect', 'Editing', and other interface controls. The main content area has a title 'Welcome To Colaboratory' with a 'CO' logo. Below it is a section titled 'What is Colaboratory?' which contains a brief description and a bulleted list: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. A note below the list says, 'Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!'. There's also a section titled 'Getting started' with a note about the interactive environment being a 'Colab notebook'. A code cell example is shown, containing Python code to calculate seconds in a day: [ ] seconds\_in\_a\_day = 24 \* 60 \* 60 seconds\_in\_a\_day 86400. A note at the bottom explains how to execute code in a cell by selecting it and pressing 'Command/Ctrl+Enter'.

# Training Neural Nets with GPU

... on Google Colab for significantly faster processing compared to CPU for parallelizable tasks.  
Ideal for deep learning and large-scale data processing



## What is Google Colab?

- Free cloud service provided by Google.
- Allows users to write and execute Python in browser.
- No setup required, with access to powerful computing resources like GPU (Graphic Processing Unit), TPU (Tensor Processing Unit)

## Limitations

- Limited to 12 hours of free usage.
- Afterwards price is around 1€ per hour
- GPU availability subject to demand and usage limits.

# Example for autonomous agent: Flappy birds

... popular mobile game where player controls bird, attempting to fly between columns of green pipes without hitting them.

## Objective of AI Training

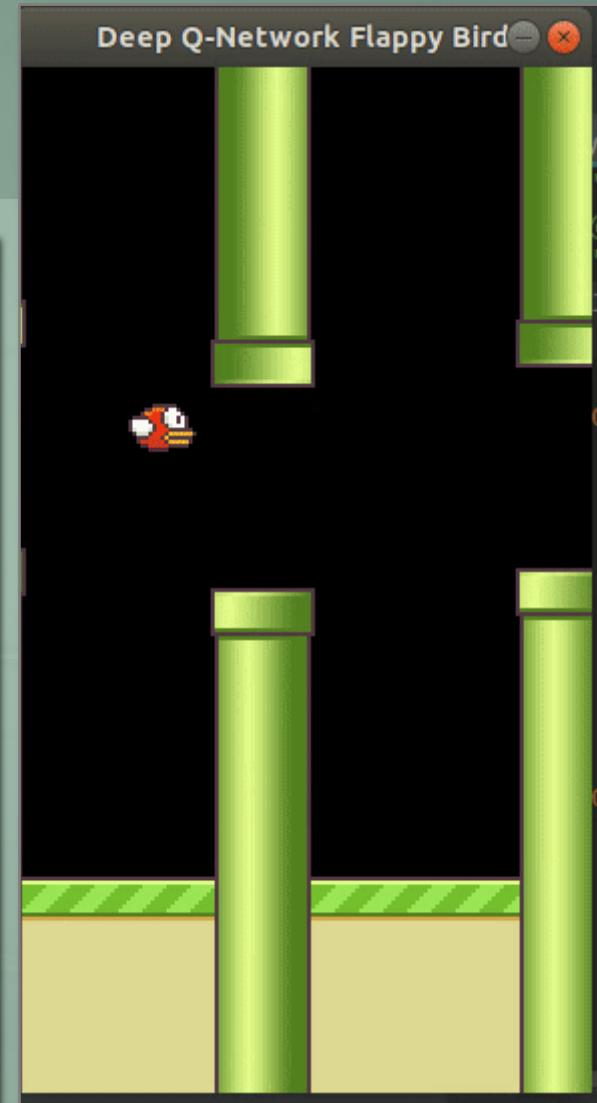
- Develop AI agent that learns to play Flappy Bird autonomously.
- Utilize machine learning techniques like Deep Q-Learning to master the game mechanics.

## Training Process

- AI agent receives game state as input (distance from pipes, bird's height, etc.).
- Agent makes decisions (to flap or not) to maximize game score.
- Rewards are given for survival time and passing through pipes.

## Challenges & Considerations

- Balancing
  - exploration (trying new actions) and
  - exploitation (using known successful actions).
- Adjusting learning rate and model parameters for optimal performance.



# Challenges of training Flappy Bird

Training AI in game like Flappy Bird presents microcosm of larger challenges in RL, like adapting to dynamic environments and balancing different aspects of learning process.

## Challenges

- **Dynamic Environment:**

Adapting to changing game scenarios with variable pipe heights and gaps.

- **Reward Strategy:**

Designing effective reward-penalty system for optimal AI learning.

- **Algorithm Selection:**

Choosing and tuning the right machine learning algorithm (like Deep Q-Learning or more complex algorithm) and size of NN for the game's requirements.

- **Exploration vs. Exploitation:**

Balancing new strategy trials with using known successful tactics.

- **Overfitting Risks:**

Ensuring AI generalizes well and doesn't overfit to specific game patterns.

```
Iteration: 9973/10000, Action: 0, Loss: 0.005297533236443996, Epsilon 0.00038971, Reward: 0.1, Q-value: 0.3540782332420349
Iteration: 9974/10000, Action: 0, Loss: 0.005426669493317604, Epsilon 0.00037972, Reward: 0.1, Q-value: 0.25436174869537354
Iteration: 9975/10000, Action: 0, Loss: 0.004872589837759733, Epsilon 0.00036973, Reward: 0.1, Q-value: 0.22313547134399414
Iteration: 9976/10000, Action: 0, Loss: 0.00538425799459219, Epsilon 0.00035974, Reward: 0.1, Q-value: 0.18547561764717102
Iteration: 9977/10000, Action: 0, Loss: 0.004924535285681486, Epsilon 0.00034975, Reward: 0.1, Q-value: 0.15512171387672424
Iteration: 9978/10000, Action: 0, Loss: 0.005377267487347126, Epsilon 0.00033976000000000004, Reward: 0.1, Q-value: 0.13742676377296448
Iteration: 9979/10000, Action: 0, Loss: 0.0057626753114163876, Epsilon 0.00032977, Reward: 0.1, Q-value: 0.10282588005065918
Iteration: 9980/10000, Action: 1, Loss: 0.005359238479286432, Epsilon 0.00031978, Reward: 0.1, Q-value: 0.06482703238725662
Iteration: 9981/10000, Action: 0, Loss: 0.006176586262881756, Epsilon 0.00030979, Reward: 0.1, Q-value: 0.0494028776884079
Iteration: 9982/10000, Action: 0, Loss: 0.005284531507641077, Epsilon 0.0002998, Reward: 0.1, Q-value: 0.048927877098321915
Iteration: 9983/10000, Action: 0, Loss: 0.005190863274037838, Epsilon 0.00028981, Reward: 0.1, Q-value: 0.0434945784509182
Iteration: 9984/10000, Action: 1, Loss: 0.00590750016272068, Epsilon 0.00027982, Reward: 0.1, Q-value: 0.03898671269416809
Iteration: 9985/10000, Action: 1, Loss: 0.006253475788980722, Epsilon 0.0002698300000000004, Reward: 0.1, Q-value: 0.03432696312665939
Iteration: 9986/10000, Action: 0, Loss: 0.00552640529349463, Epsilon 0.00025984, Reward: 0.1, Q-value: 0.036225613206624985
Iteration: 9987/10000, Action: 0, Loss: 0.005409938748925924, Epsilon 0.00024985, Reward: 0.1, Q-value: 0.018590280786156654
Iteration: 9988/10000, Action: 0, Loss: 0.0071717253886163235, Epsilon 0.00023986, Reward: 0.1, Q-value: -0.004058002959936857
Iteration: 9989/10000, Action: 1, Loss: 0.005234466399997473, Epsilon 0.0002298700000000002, Reward: 0.1, Q-value: -0.037806667387485504
Iteration: 9990/10000, Action: 1, Loss: 0.005826396867632866, Epsilon 0.0002198800000000002, Reward: 0.1, Q-value: -0.06961045414209366
Iteration: 9991/10000, Action: 0, Loss: 0.00531484792008996, Epsilon 0.00020989, Reward: 0.1, Q-value: -0.09065603464841843
Iteration: 9992/10000, Action: 0, Loss: 0.0051507112005353, Epsilon 0.0001999, Reward: 0.1, Q-value: -0.1181730255484581
Iteration: 9993/10000, Action: 0, Loss: 0.00575040141120553, Epsilon 0.00018991, Reward: 0.1, Q-value: -0.14101427793502808
Iteration: 9994/10000, Action: 0, Loss: 0.005488832015544176, Epsilon 0.0001799200000000003, Reward: 0.1, Q-value: -0.16633589565753937
Iteration: 9995/10000, Action: 0, Loss: 0.0052717626094818115, Epsilon 0.00016993, Reward: 0.1, Q-value: -0.1930693781375885
Iteration: 9996/10000, Action: 0, Loss: 0.005276841577142477, Epsilon 0.0001599400000000002, Reward: 0.1, Q-value: -0.21482783555984497
Iteration: 9997/10000, Action: 0, Loss: 0.005185238551348448, Epsilon 0.00014995, Reward: 0.1, Q-value: -0.2422558069229126
Iteration: 9998/10000, Action: 0, Loss: 0.005216964986175299, Epsilon 0.00013996, Reward: 0.1, Q-value: -0.2662467658519745
Iteration: 9999/10000, Action: 0, Loss: 0.005879316478967667, Epsilon 0.00012997, Reward: 0.1, Q-value: -0.2931475043296814
Iteration: 10000/10000, Action: 0, Loss: 0.005111475940793753, Epsilon 0.00011998, Reward: 0.1, Q-value: -0.30926597118377686
Iteration: 10001/10000, Action: 0, Loss: 0.005014549009501934, Epsilon 0.00010999000000000001, Reward: 0.1, Q-value: -0.3316221237182617
```

In this example: 10,000 epochs, however 2 Mio. epochs necessary

# Run trained Flappy Bird

Optimally trained AI agent allows bird to fly infinitely



Activities PyCharm Community Edition ▾

pytorch\_flappy\_

File Edit View Navigate Code Refactor Run Tools V

pytorch\_flappy\_bird test.py

Project 1: Project

pytorch\_flappy\_bird ~Py

assets

src

- deep\_q\_network.py
- flappy\_bird.py
- utils.py

tensorboard

trained\_models

.Rhistory

README.md

test.py

train.py

External Libraries

Scratches and Consoles

test.py

```
1 """  
2 @author: Vi  
3 """  
4 import ...  
5  
6 def get_arg_parser():  
7     parser = argparse.ArgumentParser()  
8     parser.add_argument('--model', type=str, required=True, help='Path to model file or '...  
9     args = parser.parse_args()  
10    return args  
11  
12 def test(...):  
13     if torch.cuda.is_available():  
14         torch.set_default_tensor_type('torch.FloatTensor')  
15     else:  
16         torch.set_default_tensor_type('torch.FloatTensor')  
17     if torch.cuda.is_available():  
18         mod = model.Model().cuda()  
19     else:  
20         mod = model.Model()  
21     game_st
```

Animated

33

# Links

i.e. sources for self-learning

	Title	Link
Neural Networks	Neural Networks Visualized	<a href="https://levelup.gitconnected.com/neural-networks-visualized-6cc657f9d7c5">https://levelup.gitconnected.com/neural-networks-visualized-6cc657f9d7c5</a>
	The Universal Approximation Theorem	<a href="https://www.deep-mind.org/2023/03/26/the-universal-approximation-theorem/">https://www.deep-mind.org/2023/03/26/the-universal-approximation-theorem/</a>
	The Concept of Artificial Neurons (Perceptrons) in Neural Networks	<a href="https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc">https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc</a>
	Understanding Feedforward Neural Networks	<a href="https://learnopencv.com/understanding-feedforward-neural-networks/">https://learnopencv.com/understanding-feedforward-neural-networks/</a>
	Artificial Intelligence Applications in Cardiovascular Magnetic Resonance Imaging	<a href="https://www.researchgate.net/publication/371616648_Artificial_Intelligence_Applications_in_Cardiovascular_Magnetic_Resonance_Imaging_Are_We_on_the_Path_to_Avoiding_the_Administration_of_Contrast_Media">https://www.researchgate.net/publication/371616648_Artificial_Intelligence_Applications_in_Cardiovascular_Magnetic_Resonance_Imaging_Are_We_on_the_Path_to_Avoiding_the_Administration_of_Contrast_Media</a>
	Understanding Backpropagation	<a href="https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0">https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0</a>
	How does an AI learn? Training Neural Networks with Backpropagation	<a href="https://medium.com/@rubentak/how-does-an-ai-learn-training-neural-networks-with-backpropagation-a8b89d8bf330">https://medium.com/@rubentak/how-does-an-ai-learn-training-neural-networks-with-backpropagation-a8b89d8bf330</a>
	Deep Learning Optimizer algorithms - Gradient Descent and RMSprop	<a href="https://www.linkedin.com/pulse/deep-learning-optimization-algorithms-gradient-descent-shashi-singh/">https://www.linkedin.com/pulse/deep-learning-optimization-algorithms-gradient-descent-shashi-singh/</a>
	D3QN Agent with Prioritized Experience Replay	<a href="https://pylessons.com/CartPole-PER">https://pylessons.com/CartPole-PER</a>
	Universal Approximation Theorem, Neural Nets & Lego Blocks	<a href="https://medium.com/analytics-vidhya/universal-approximation-theorem-neural-nets-lego-blocks-1f5a7d93542a">https://medium.com/analytics-vidhya/universal-approximation-theorem-neural-nets-lego-blocks-1f5a7d93542a</a>

# Links

i.e. sources for self-learning

	Title	Link
Tutorials Deep Q-Learning	A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python	<a href="https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/">https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/</a>
	The Hugging Face Deep Reinforcement Learning Course 🎓 (v2.0)	<a href="https://github.com/huggingface/deep-rl-class">https://github.com/huggingface/deep-rl-class</a>
	The Fundamentals of Reinforcement Learning and How to Apply It	<a href="https://cnvrg.io/reinforcement-learning/">https://cnvrg.io/reinforcement-learning/</a>
	An Introduction to Reinforcement Learning with OpenAI Gym, RLLib, and Google Colab	<a href="https://www.anyscale.com/blog/an-introduction-to-reinforcement-learning-with-openai-gym-rllib-and-google">https://www.anyscale.com/blog/an-introduction-to-reinforcement-learning-with-openai-gym-rllib-and-google</a>
	Techniques to Improve the Performance of a DQN Agent	<a href="https://towardsdatascience.com/techniques-to-improve-the-performance-of-a-dqn-agent-29da8a7a0a7e">https://towardsdatascience.com/techniques-to-improve-the-performance-of-a-dqn-agent-29da8a7a0a7e</a>
	Deep Q-Learning Tutorial: minDQN	<a href="https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc">https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc</a>
		<a href="https://www.researchgate.net/publication/329029610_Deep_Q-Learning_Explained">https://www.researchgate.net/publication/329029610_Deep_Q-Learning_Explained</a>
		<a href="https://slideplayer.com/slide/12397094/">https://slideplayer.com/slide/12397094/</a>
	Deep Q Learning	<a href="https://leonardoaraujosantos.gitbook.io/artificial-intelligence/artificial_intelligence/reinforcement_learning/deep_q_learning">https://leonardoaraujosantos.gitbook.io/artificial-intelligence/artificial_intelligence/reinforcement_learning/deep_q_learning</a>
	[PYTORCH] Deep Q-learning for playing Flappy Bird	<a href="https://github.com/uvipen/Flappy-bird-deep-Q-learning-pytorch/tree/master">https://github.com/uvipen/Flappy-bird-deep-Q-learning-pytorch/tree/master</a>

# Links

i.e. sources for self-learning

	Title	Link
Applications	Applications	<a href="https://neuravest.net/how-to-apply-deep-reinforcement-learning-to-trading/">https://neuravest.net/how-to-apply-deep-reinforcement-learning-to-trading/</a>
	Deep Reinforcement Learning and Its Applications	<a href="https://www.linkedin.com/pulse/deep-reinforcement-learning-its-applications-anand-pansare/">https://www.linkedin.com/pulse/deep-reinforcement-learning-its-applications-anand-pansare/</a>
	Deep Reinforcement Learning for Recommendation Systems / Xiangyu Zhao (City U, HK)	<a href="https://www.youtube.com/watch?v=spx6PoccI04">https://www.youtube.com/watch?v=spx6PoccI04</a>
	Google DeepMind - AlphaGo	<a href="https://www.deepmind.com/research/highlighted-research/alphago">https://www.deepmind.com/research/highlighted-research/alphago</a>
	How DeepMind's AlphaGo Became the World's Top Go Player	<a href="https://ai.plainenglish.io/how-deepminds-alphago-became-the-world-s-top-go-player-5b275e553d6a">https://ai.plainenglish.io/how-deepminds-alphago-became-the-world-s-top-go-player-5b275e553d6a</a>
	Flappy Bird RL	<a href="https://sarvagyavaish.github.io/FlappyBirdRL/">https://sarvagyavaish.github.io/FlappyBirdRL/</a>
	Deep Reinforcement Learning in Pac-man	<a href="https://www.youtube.com/watch?v=QilHGSYbjDQ">https://www.youtube.com/watch?v=QilHGSYbjDQ</a>
	Deep Reinforcement Learning for Robotic Manipulation	<a href="https://www.youtube.com/watch?v=ZhsEKT07V04">https://www.youtube.com/watch?v=ZhsEKT07V04</a>
	Deep Q-Learning for Atari Breakout	<a href="https://keras.io/examples/rl/deep_q_network_breakout/">https://keras.io/examples/rl/deep_q_network_breakout/</a>

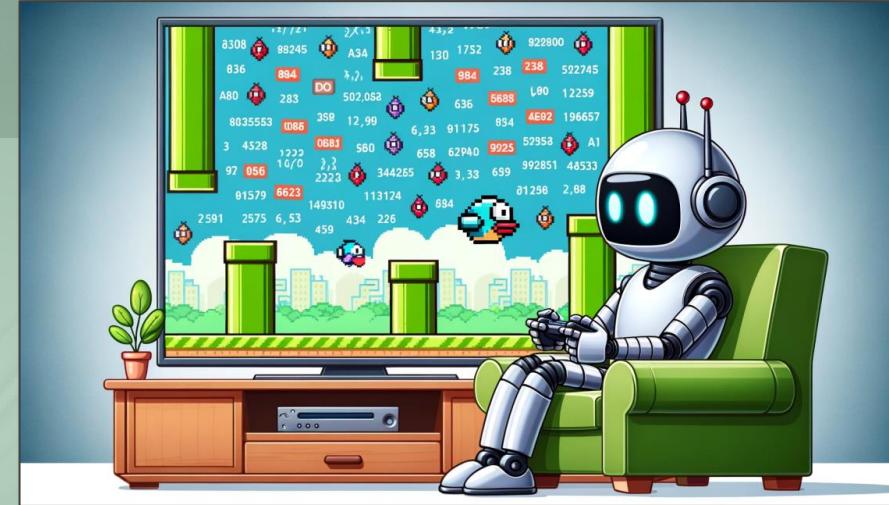
# Links

i.e. sources for self-learning

	Title	Link
Autonomous Driving	Deep Reinforcement Learning for Autonomous Driving: A Survey	<a href="https://arxiv.org/pdf/2002.00444.pdf">https://arxiv.org/pdf/2002.00444.pdf</a>
	Simulating self-driving cars using Reinforcement learning	<a href="https://medium.com/analytics-vidhya/simulating-self-driving-ai-race-using-unity-ml-2ac314f67980">https://medium.com/analytics-vidhya/simulating-self-driving-ai-race-using-unity-ml-2ac314f67980</a>
	Multi-Agent Deep Reinforcement Learning for Connected Autonomous Driving - Praveen Palanisamy	<a href="https://www.youtube.com/watch?v=a3tDo6Aof_k">https://www.youtube.com/watch?v=a3tDo6Aof_k</a>
	Reinforcement learning: Self-driving cars in the browser (DDPG)	<a href="https://www.youtube.com/watch?v=G-GpY7beuvw">https://www.youtube.com/watch?v=G-GpY7beuvw</a>
	Reinforcement-Learning-for-Self-Driving-Cars	<a href="https://songyanho.github.io/Reinforcement-Learning-for-Self-Driving-Cars">https://songyanho.github.io/Reinforcement-Learning-for-Self-Driving-Cars</a>
	Title	Link
Robotics	DeepMind's AI Athletes Play In The Real World!	<a href="https://www.youtube.com/watch?v=efw8xuex4uI">https://www.youtube.com/watch?v=efw8xuex4uI</a>
	Reinforcement Learning in Healthcare: A Survey	<a href="https://arxiv.org/pdf/1908.08796.pdf">https://arxiv.org/pdf/1908.08796.pdf</a>
	Using reinforcement learning to identify high-risk states and treatments in healthcare	<a href="https://www.microsoft.com/en-us/research/blog/using-reinforcement-learning-to-identify-high-risk-states-and-treatments-in-healthcare/">https://www.microsoft.com/en-us/research/blog/using-reinforcement-learning-to-identify-high-risk-states-and-treatments-in-healthcare/</a>
	Applications of Deep Reinforcement Learning for Drug Discovery	<a href="https://link.springer.com/chapter/10.1007/978-981-99-1620-7_11">https://link.springer.com/chapter/10.1007/978-981-99-1620-7_11</a>

# ChatGPT/Dall-E3 Prompts

Cartoon illustration of a robot sitting on a couch, focused on playing 'Flappy Bird' on a large TV. The bird, in the game, is surrounded by floating numbers and algorithms, symbolizing the AI's computations to ace the game.



High-quality render of a gaming desk against a gradient backdrop flowing from a bright green to a muted gray. The computer screen showcases the 'Go' game in mid-play, with stones positioned on the board. Beside the monitor, a modern AI module glowing in green and built with gray components actively interfaces with the game, hinting at its role in strategic gameplay analysis.



# ChatGPT/Dall-E3 Prompts

Illustration showcasing the Universal Approximation Theorem with LEGO bricks. On the left, a detailed LEGO construction represents a feed-forward neural network, complete with layers and connecting nodes. On the right side, a diverse assortment of LEGO pieces are arranged in patterns to depict an arbitrary dataset, showing various data points and their distribution.





# About me

## Dr. Harald Stein

- Data Scientist                   ~ 7 years experience
  - Algotrader                      ~ 4 years experience
  - Ph.D.                            in Economics, Game Theory
- 
- LinkedIn:                     <https://www.linkedin.com/in/harald-stein-phd-1648b51a>
  - ResearchGate:                 <https://www.researchgate.net/profile/Harald-Stein>

