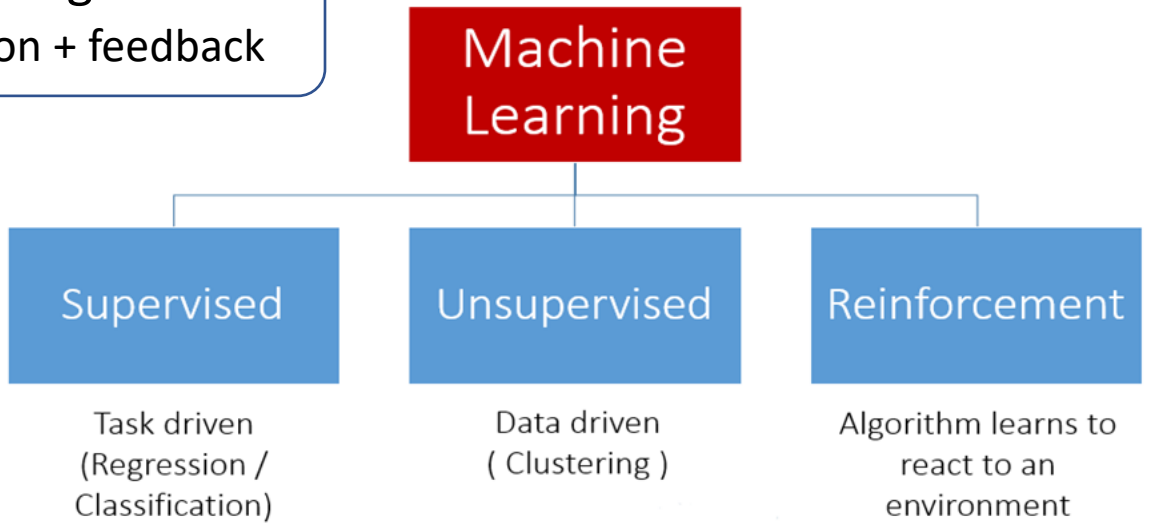
Welcoming the Era of Deep Neuroevolution

Original article by Uber Data:
https://eng.uber.com/deep-neuroevolution/

Presented paper – Genetic algorithms for deep learning:
https://arxiv.org/abs/1712.06567

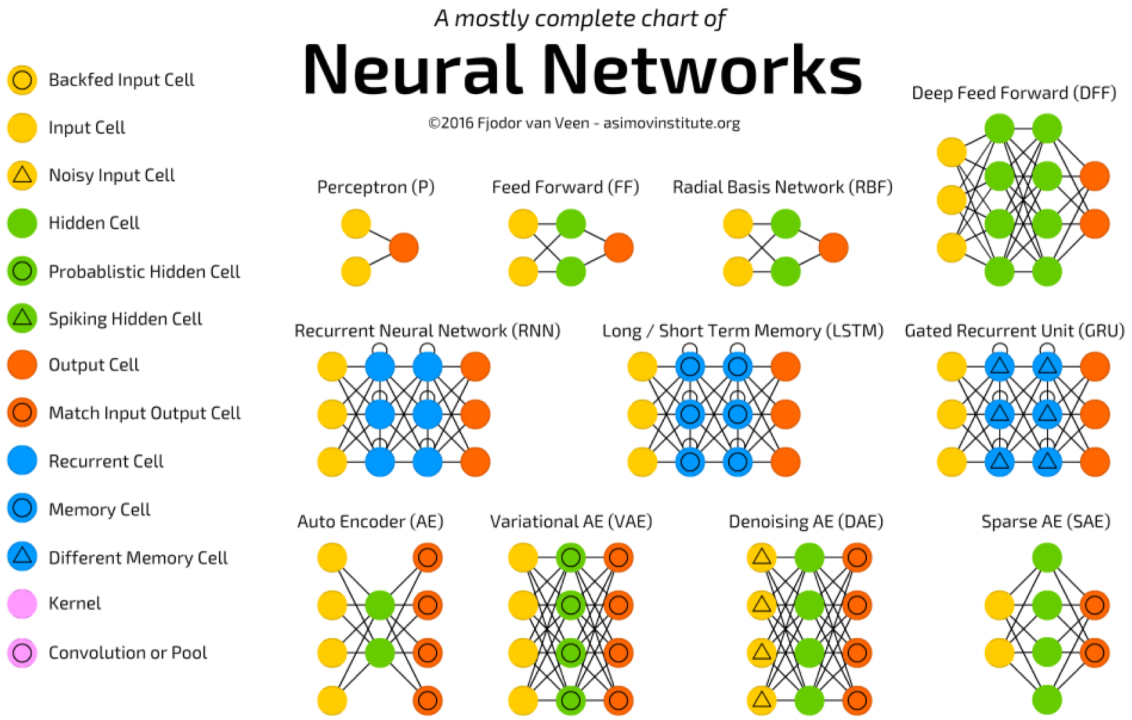
Types of machine learning

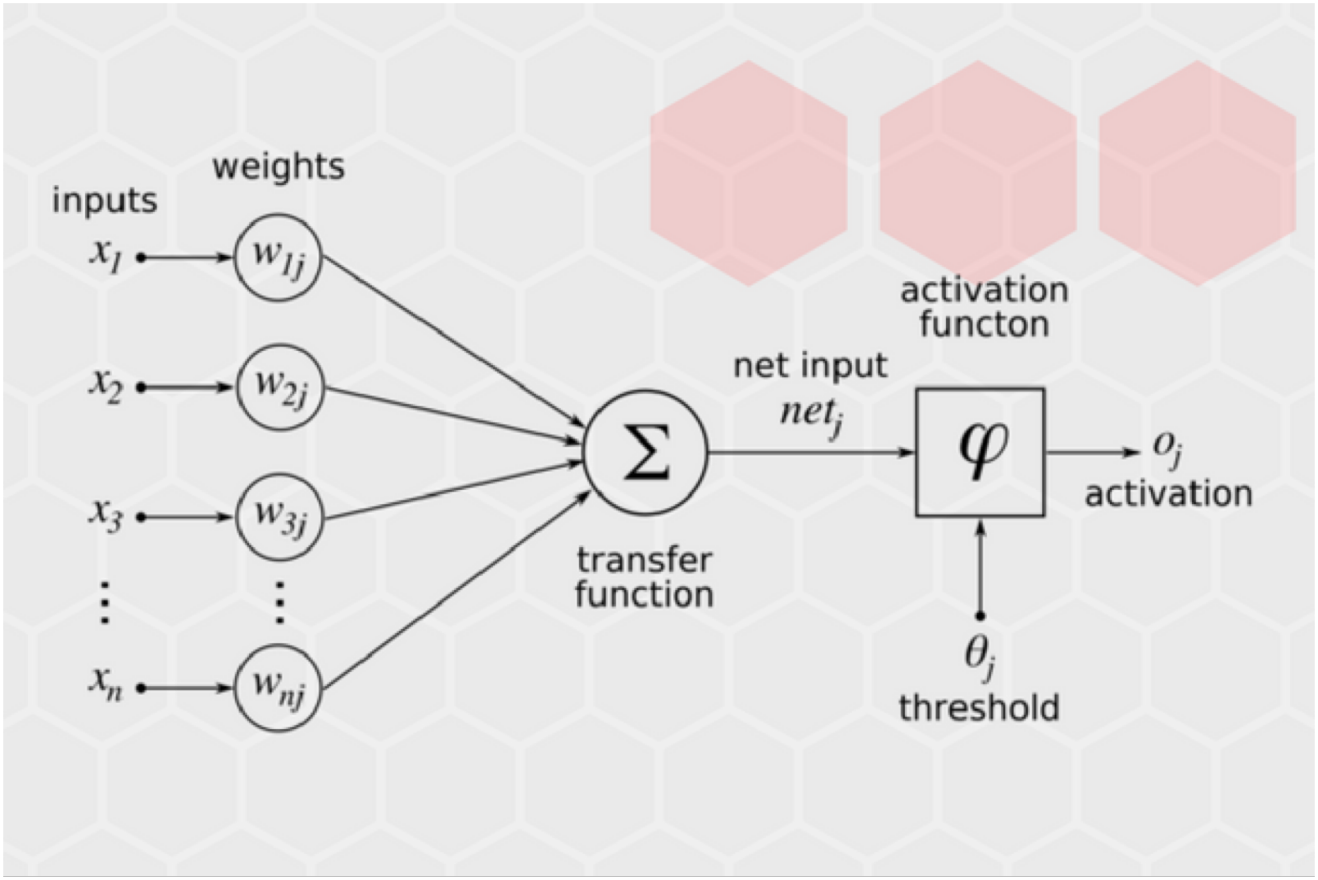
- **Supervised** learning
  - we know (input, output) data
- **Unsupervised** learning
  - we only know the (input) data
- **Reinforcement** learning
  - we know observation + feedback

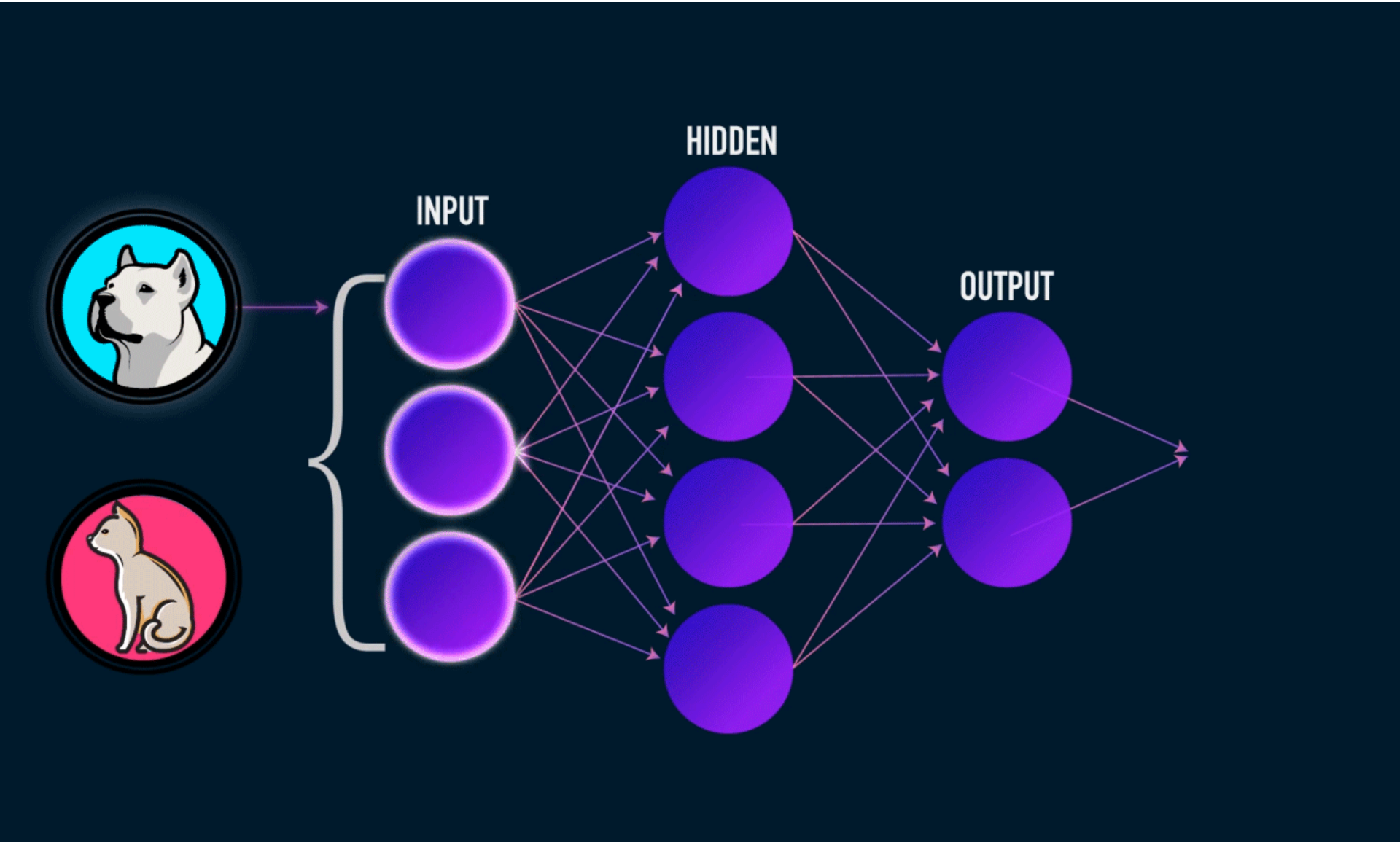


But what is a Neural Network (NN)?

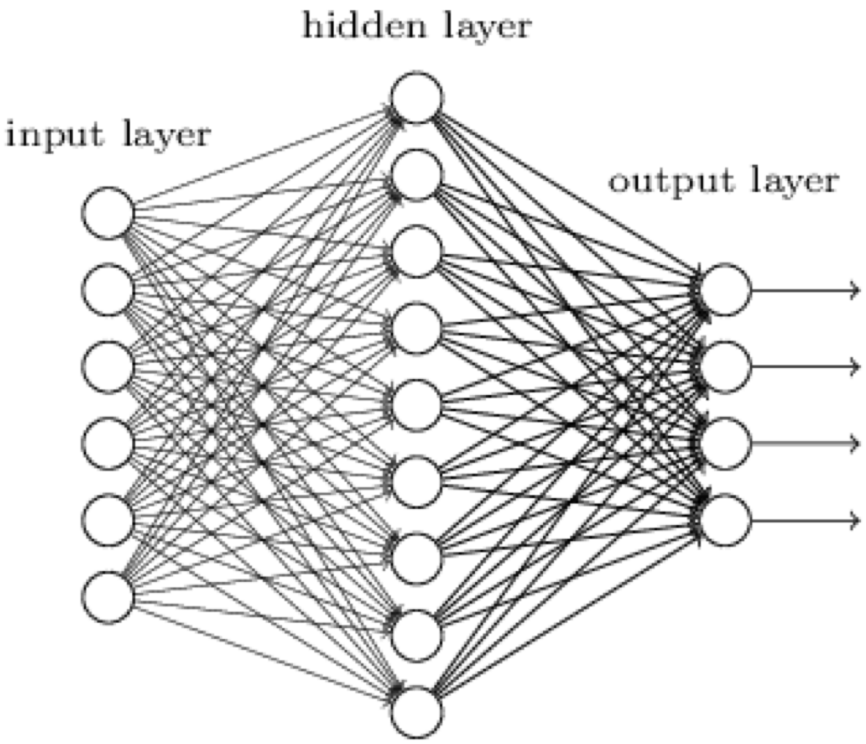
- Universal approximator. Inspired by human brain.
- It can do awesome things!




inputs
weights
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
transfer
function
net input
net_j
activation
functon
φ
o_j
activation
θ_j
threshold


HIDDEN
INPUT
OUTPUT

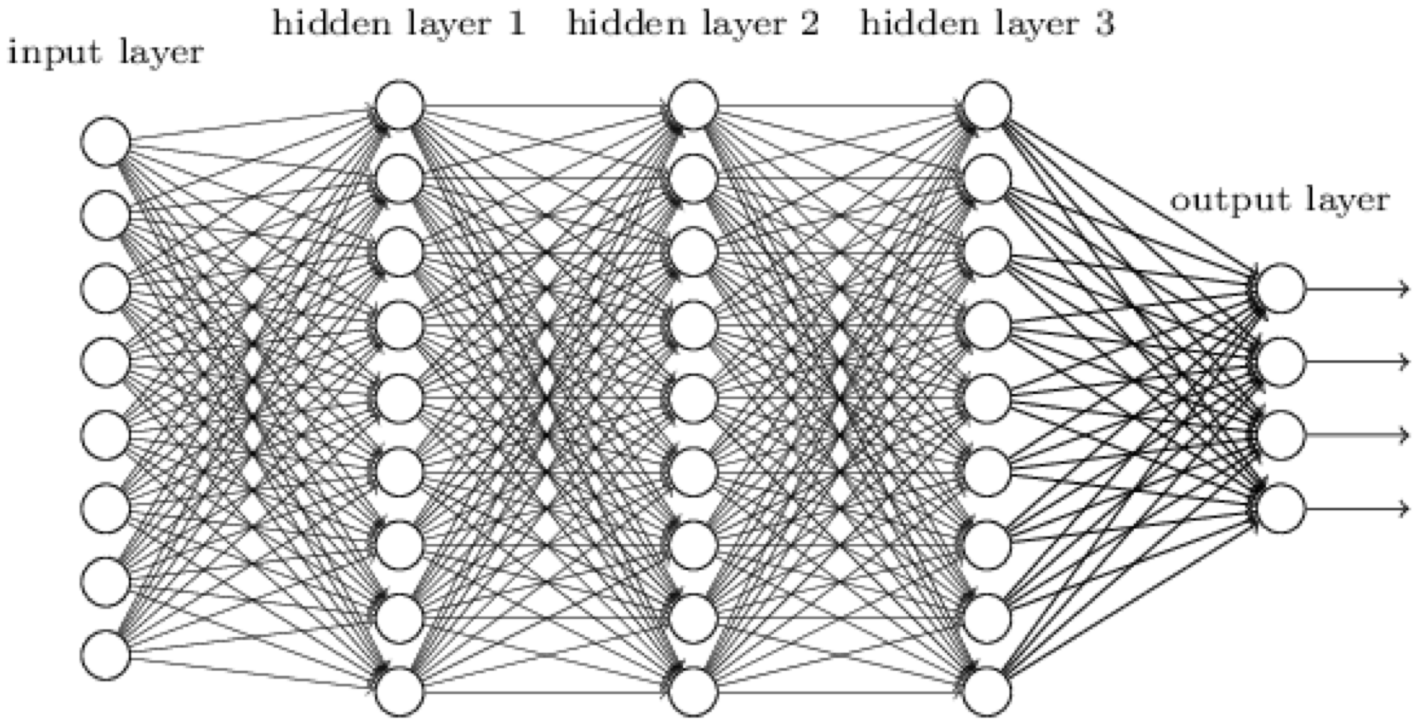
Is this a **deep** Neural Network?



well, it does not look very deep


WE NEED TO GO
DEEPER

Is this a deep Neural Network?



.. nobody really knows

How do Neural Networks learn?

Neural network is basically a function:

$$NN(x^1, x^2, x^3 \dots x^n) = (y^1, y^2, y^3 \dots y^m)$$

The inputs and outputs are given, so what do we adjust?

$$weights!$$

How do we know how much should the weights to be adjusted?

$$We\ calculate\ how\ „wrong“\ the\ NN\ is.$$

This is usually called a loss function: on all data we sum the error.

$$e.g.\ MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

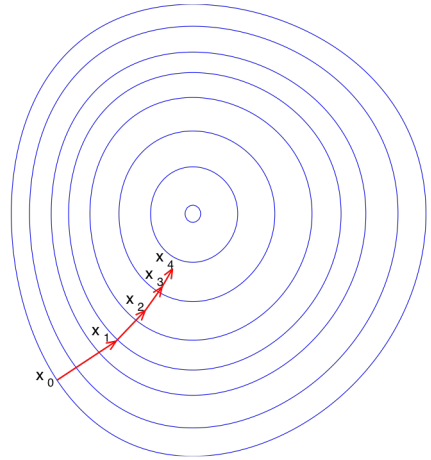
And we try to **minimise** this loss function **by adjusting weights**.

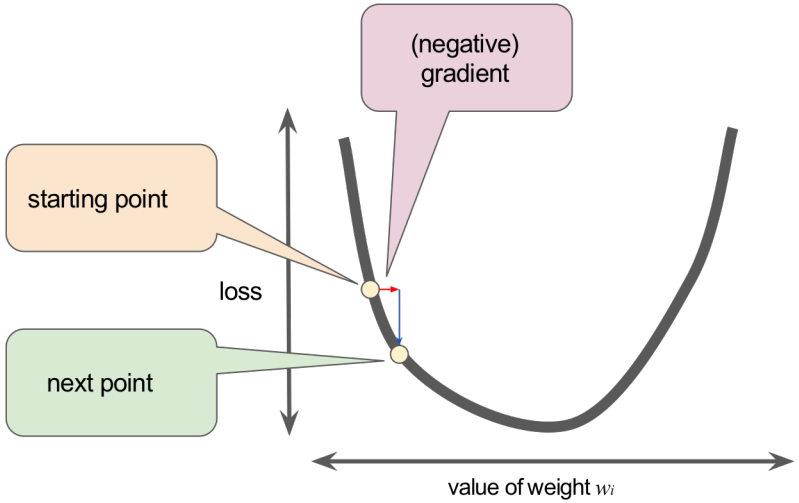
How do we minimise the loss function?

- we get the **gradient**! – ”vector” that shows in which way we adjust the weights to minimize a loss function

Gradient Descend (GD)

- iterative algorithm for finding a minimum of a function

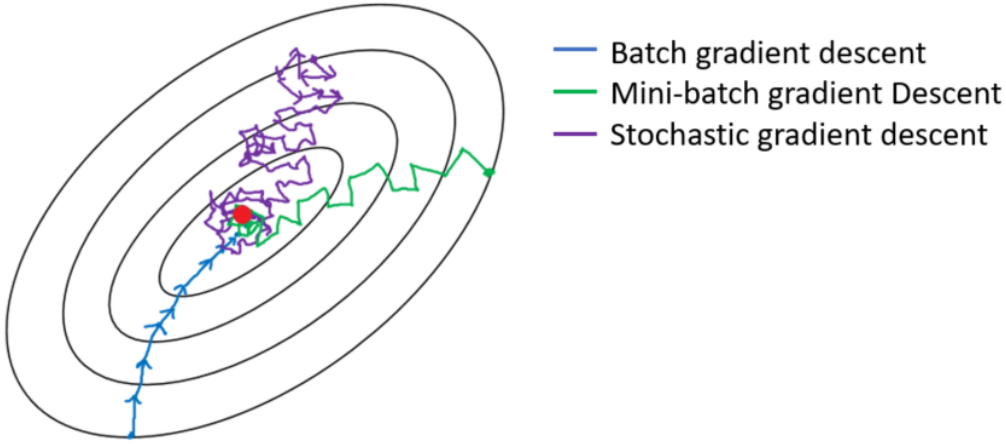




How do we minimise the loss function?

Stochastic Gradient Descend (SGD)

- most used for deep learning
- uses only a small data sample to calculate the gradient
  - less precision
  - much faster to compute (remember: we do this iteratively)

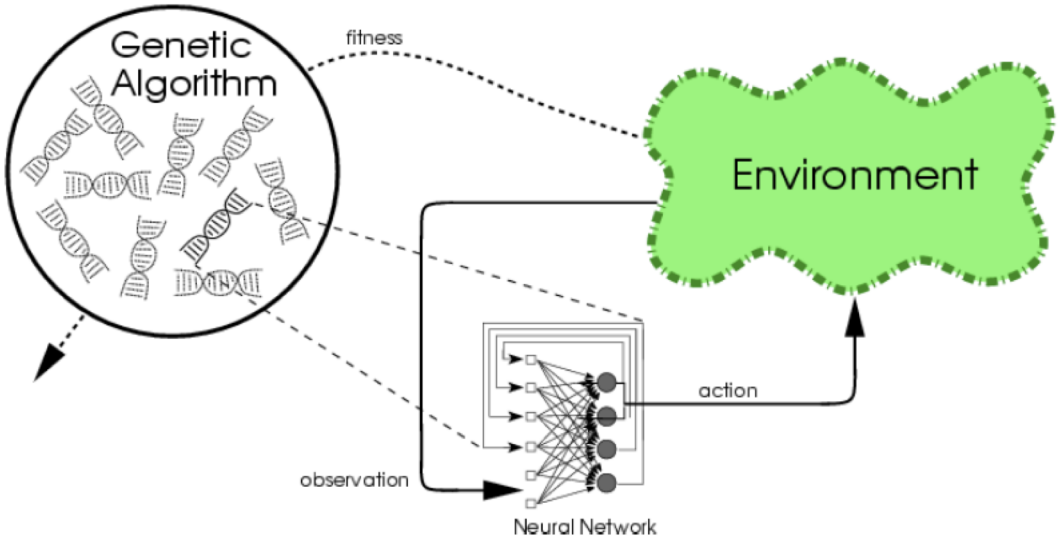


Neural Networks are...

- **AWESOME**!
- Finding the gradient involves some calculus ☹
  - kidding, that's not an excuse of a data scientist ☺

Neuroevolution

- **training NNs through evolution!**
- example of a **gradient-less training method**
- **simple** genetic algorithm can sometimes outperform modern RL algorithms
- can be faster due to good parallelization

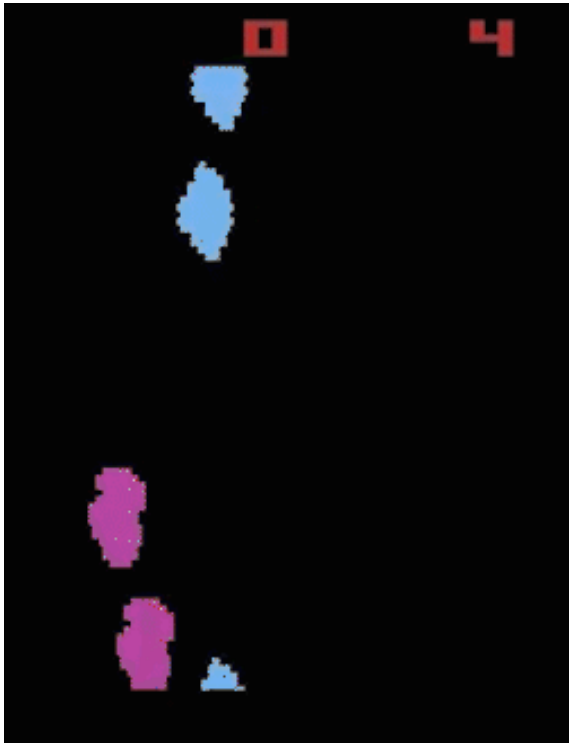


Genetic algorithms as a competitive alternative for training deep neural networks

Paper by Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman Kenneth O. Stanley and Jeff Clune from Uber AI labs

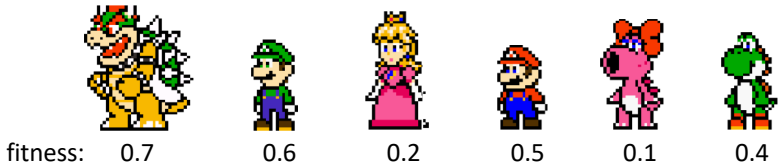
https://arxiv.org/abs/1712.06567

- DNNs trained to play **Atari games from pixels**
  - benchmark
- compares well to traditional RL algorithms
- **novel approach to store the genomes**
- largest neural network ever evolved with a traditional evolutionary algorithm

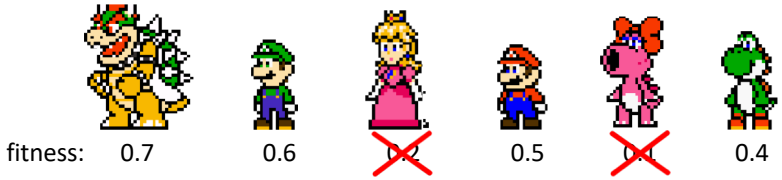


Principle of the Genetic Algorithm:

1. evaluate each individual to get the fitness score:

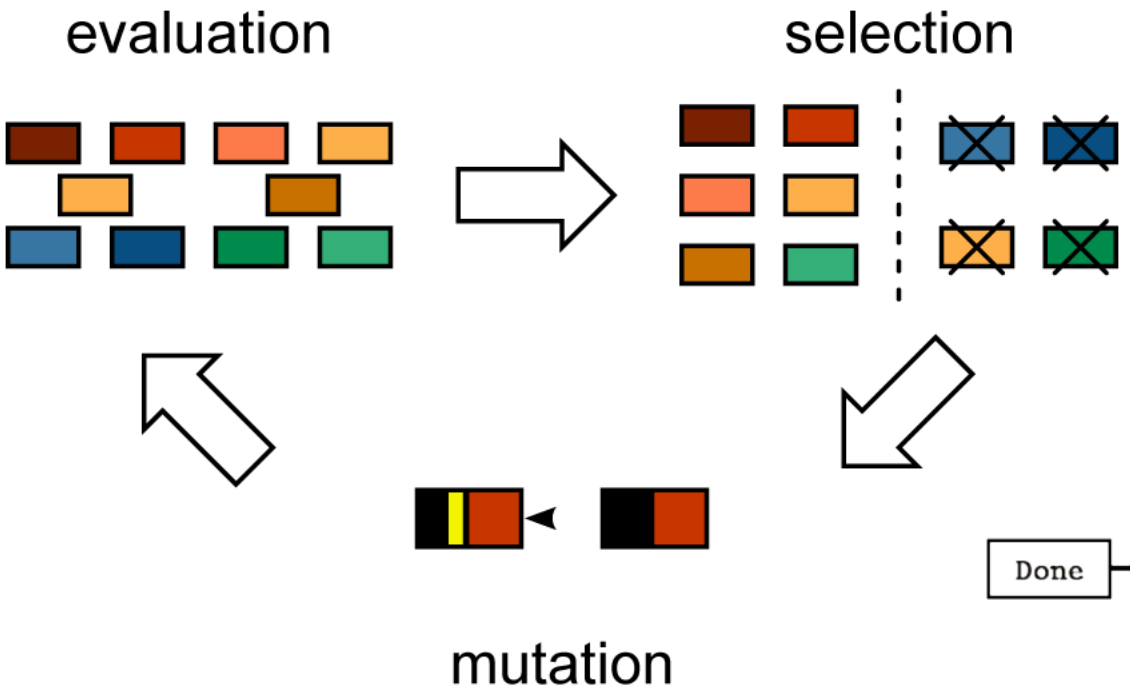


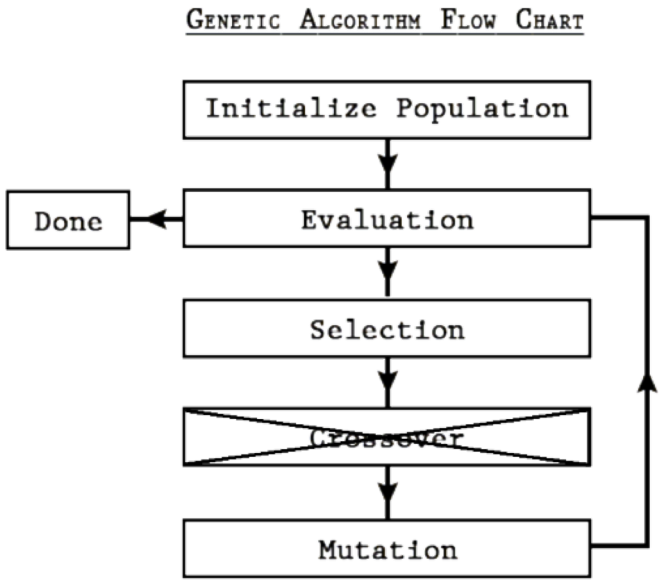
2. remove the low performing ones



3. produce offsprings with random mutations

No crossover involved!





New approach to store the  !

- traditional approach - store each individual θ as s vector of weights
  - takes up a lot of memory
  - does not scale well (especially in networks with large populations)

- **new approach** – store the initialisation seed and mutation seeds
  - seeds $s_1, s_2 \ldots s_n$ are fed into a deterministic mutation function
  - critical innovation for an efficient implementation of a distributed deep GA

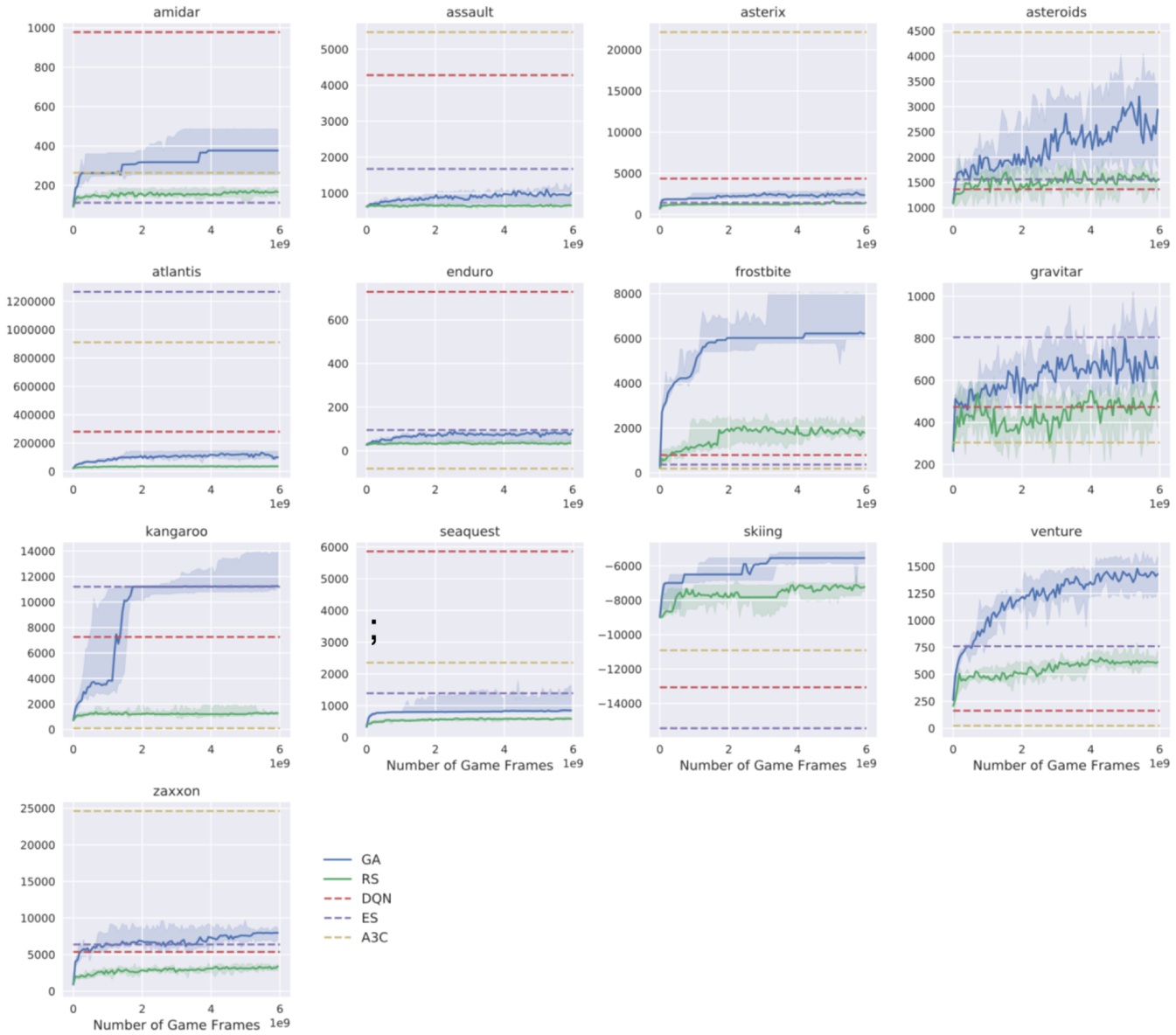
$$\theta = \psi(initialisation\ seed, \{s_1, s_2 \ldots s_n\})$$

- individual $\boldsymbol{\theta}$ **can be easily reconstructed** while being **efficiently stored**
  - size **scales linearly** with the number of generations,
  - **independent from the size of network!!!** ← this is awesome

Benchmarking

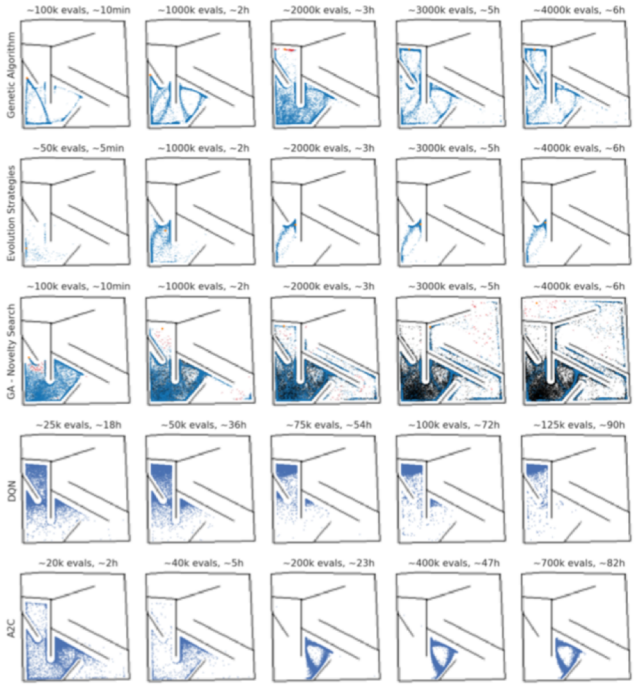
GA always outperforms random search

	DQN	ES	A3C	RS	GA	GA
Frames	200M	1B	1B	1B	1B	6B
Time	~7-10d	~ 1h	~ 4d	~ 1h or 4h	~ 1h or 4h	~ 6h or 24h
Forward Passes	450M	250M	250M	250M	250M	1.5B
Backward Passes	400M	0	250M	0	0	0
Operations	1.25B U	250M U	1B U	250M U	250M U	1.5B U
amidar	**978**	112	264	143	263	377
assault	4,280	1,674	**5,475**	649	714	814
asterix	4,359	1,440	**22,140**	1,197	1,850	2,255
asteroids	1,365	1,562	**4,475**	1,307	1,661	2,700
atlantis	279,987	**1,267,410**	911,091	26,371	76,273	129,167
enduro	**729**	95	-82	36	60	80
frostbite	797	370	191	1,164	**4,536**	**6,220**
gravitar	473	**805**	304	431	476	764
kangaroo	7,259	**11,200**	94	1,099	3,790	**11,254**
seaquest	**5,861**	1,390	2,355	503	798	850
skiing	-13,062	-15,443	-10,911	-7,679	†**-6,502**	†**-5,541**
venture	163	760	23	488	**969**	†**1,422**
zaxxon	5,363	6,380	**24,622**	2,538	6,180	7,864


amidar
assault
asterix
asteroids
atlantis
enduro
frostbite
gravitar
kangaroo
seaquest
skiing
venture
zaxxon
1e9
Number of Game Frames
GA
RS
DQN
ES
A3C

Using old algorithms developed in the neuroevolution - Novelty search

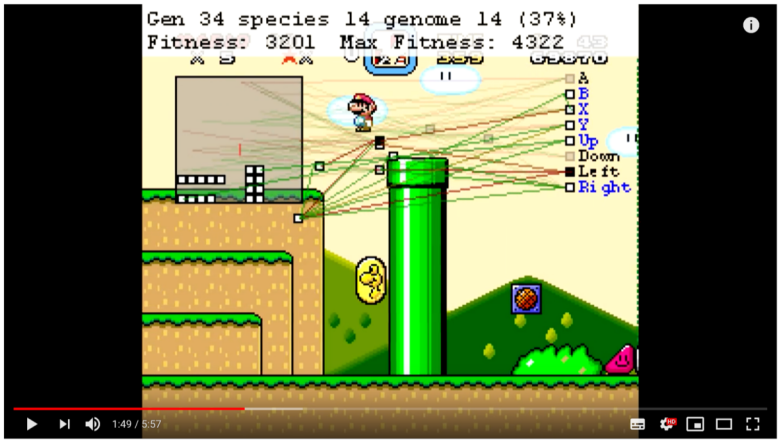
- example of using existing algorithms with DNNs
- rewarding agent for performing new behaviours – instead of fitness
  - novelty = average distance to kNN (sorted by distance) in the population/archieve
- works surprisingly well with GA
  - it can often outperform algorithms that utilize the reward signal



Conclusions

- **old algorithms** combined with **modern amounts of computing** can work surprisingly well
- neuroevolution searches differently than SGD, and thus offers an interesting alternative approach
- state-of-the-art compact **encoding technique**
  - allow for effective parallelization

Other examples of GA with NN



Marl/O - Machine Learning for Video Games

7 232 969 zhlédnutí

128 TIS. 2,3 TIS. SDÍLET ULOŽIT

SethBling
Publikováno 13. 6. 2015

POČET ODBĚRŮ: 1,9 MIL.