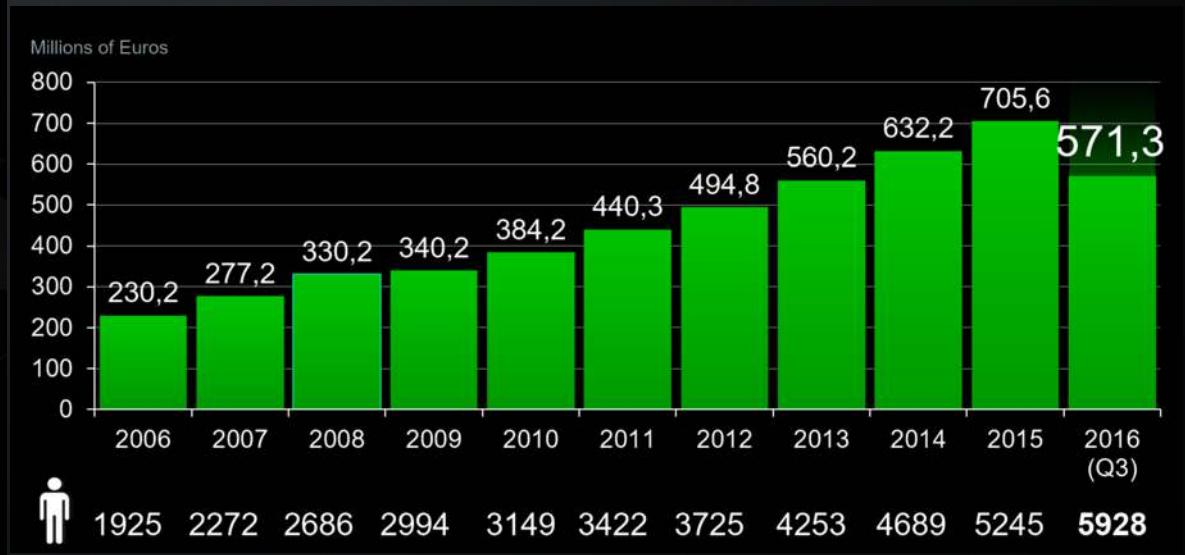
Blockchain meets Machine Learning: How to develop your own intelligent smart contracts system.

Data Reply: Thomas Schmiedel [t.schmiedel@reply.de]

Short Intro: [Data] Reply

- Hi, my name is Thomas Schmiedel
- I‘m data scientist at Data Reply in Munich, our specialty is Big Data, Data Science / Machine Learning and Data Engineering
- Data Reply is part of the Reply Network with over 6000 employees world-wide
- Strong focus on innovative IT-technologies

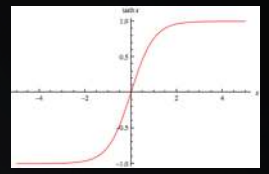


Motivation

Why Blockchain **and** Machine Learning?

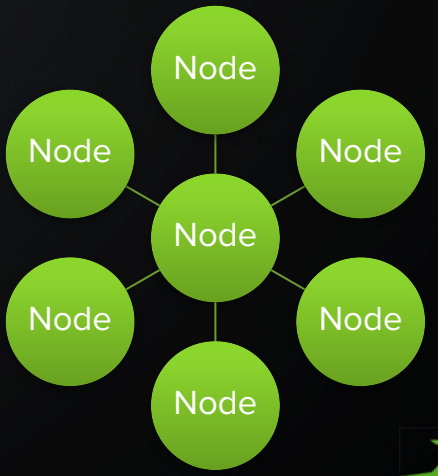
- Well, there's obviously some hype around each of those technologies, why not combine them and create even more hype? :-)
- The truth is: Every Blockchain network consists of decentralized participants, constantly exchanging messages between each other and keeping a record of history they all fully agree upon
- Up to now, the data inside that record is more or less unstructured, just a pile of bits and bytes
- Let's take it to the next level and build a colorful ecosystem on top of it, so people can collaborate more efficiently and find interesting things in seconds

Agenda



What are we going to do now?

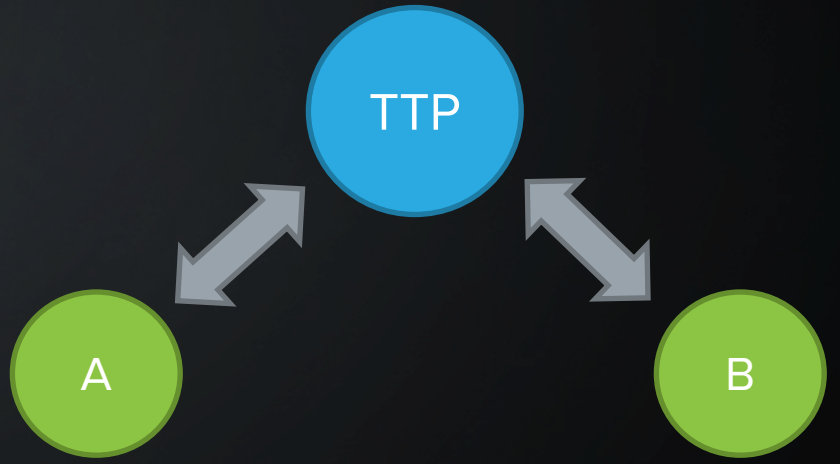
- Well, at least some theory about Blockchains and Machine Learning is necessary, we'll do that first
- Then, we'll build an architecture for our intelligent Blockchain client
- Afterwards, we'll write a smart contract and deploy it to the network
- The client software must also be written
- Testing it, of course!



Whom to Trust?

A short story about trust:

- Let's suppose you want to buy a car from a stranger, is there trust involved?
- Of course, he could show up with some "friends" and drive away with car and money
- Any idea what to do about that?
- Trusted Third Party (TTP):
  - Takes car from A and won't give it back
  - Receives money from B
  - Gives A the money and B the car



- Is there trust involved? Unfortunately yes, TTP could also run away
- You can't eliminate trust by forwarding responsibility to someone else (TTP, Server)

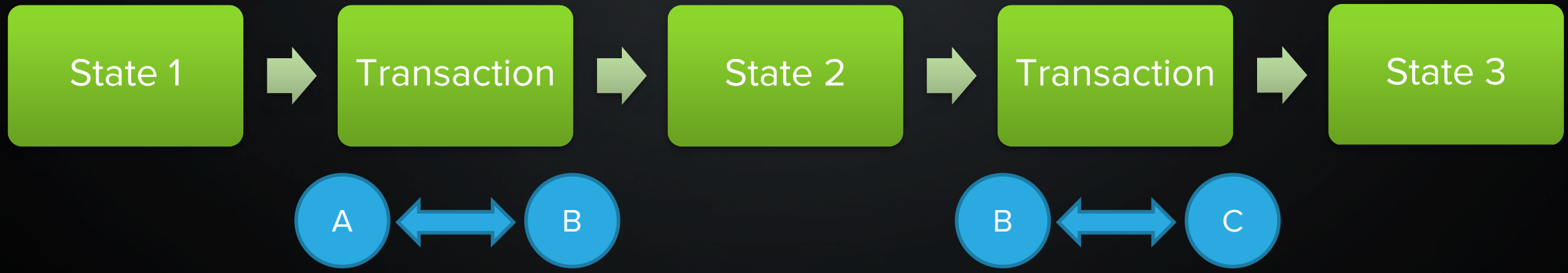
Atomic Transactions

Solution:

- Atomic transactions: either car and money change ownership simultaneously or nothing happens

But how?

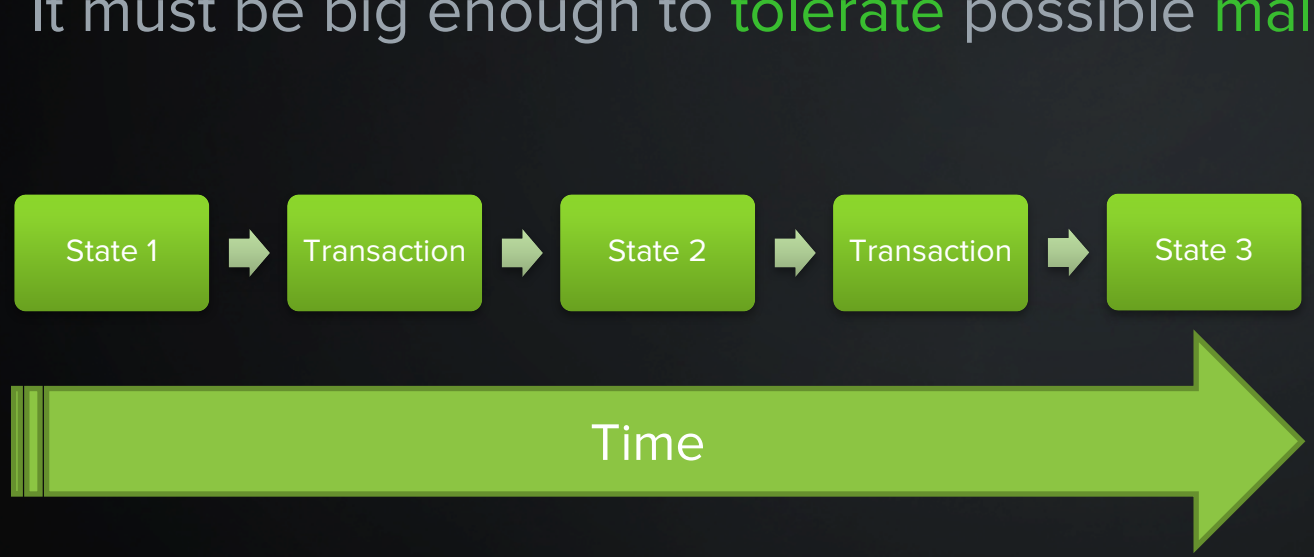
- Let‘s say we have some universal storage of ownership that everybody has access to
- And this storage can only be transformed according to certain rules
- Our rule: car and money can only change ownership in a single step
- ➔ Our universal storage of ownership can only be altered by following the rule

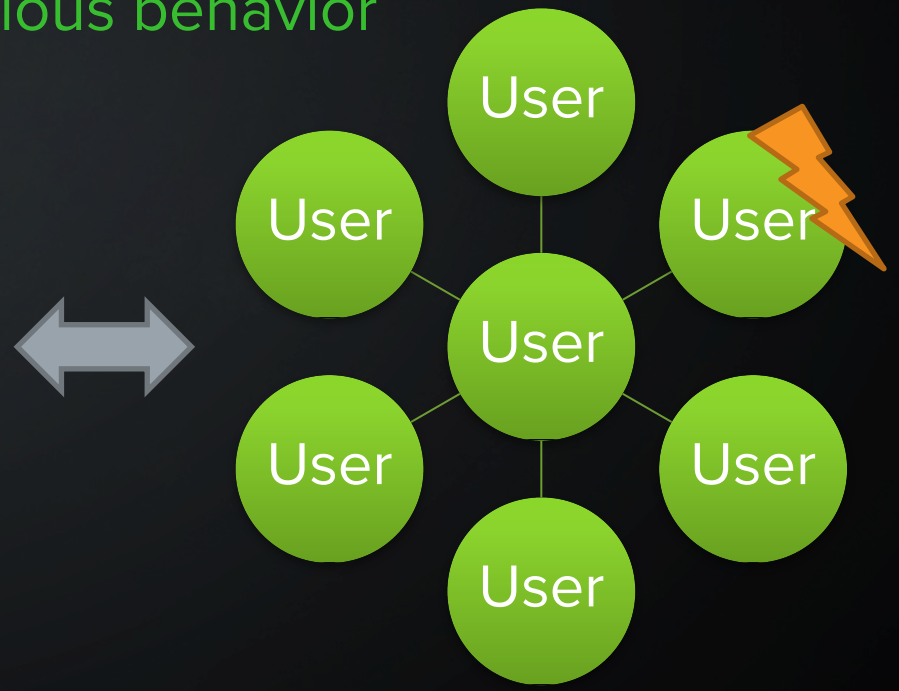


Core Feature: Decentralization

How to enforce following the rules?

- We need a whole network keeping the same record and verifying if changes to the record follow the rules
- It must be big enough to tolerate possible malicious behavior





Closing the Circle

Now we introduce some different names:

- Universal storage of ownerships… let‘s call it Blockchain ✓
- The transaction rule… let‘s call it smart contract ✓
- The participants in our network… let‘s call them nodes or peers ✓

Therefore:

- Each node has an identical copy of the Blockchain and only allows transformations of it according to a smart contract
- But: nodes must always be in consensus (sync) about current state of reality

- ➔ The complete record is transformed step by step from one consistent state to the next consistent state

How to keep nodes in sync? ➔ Consensus Algorithm

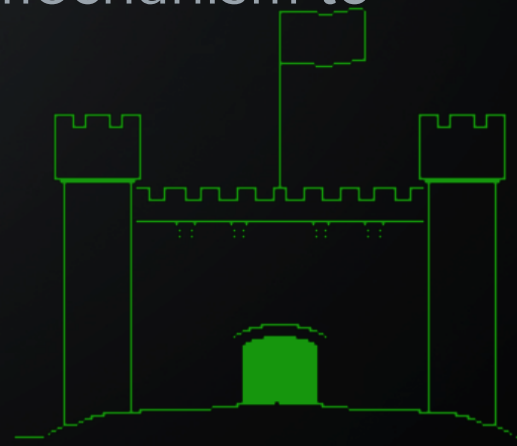
Keeping Consensus among Nodes

Byzantine Fault Tolerance:

- Can the network cope with malicious nodes sending manipulated messages?
- How many malicious nodes are needed to cause inconsistencies within the network state? (some nodes assuming a different current reality than the rest)

The Generals Problem:

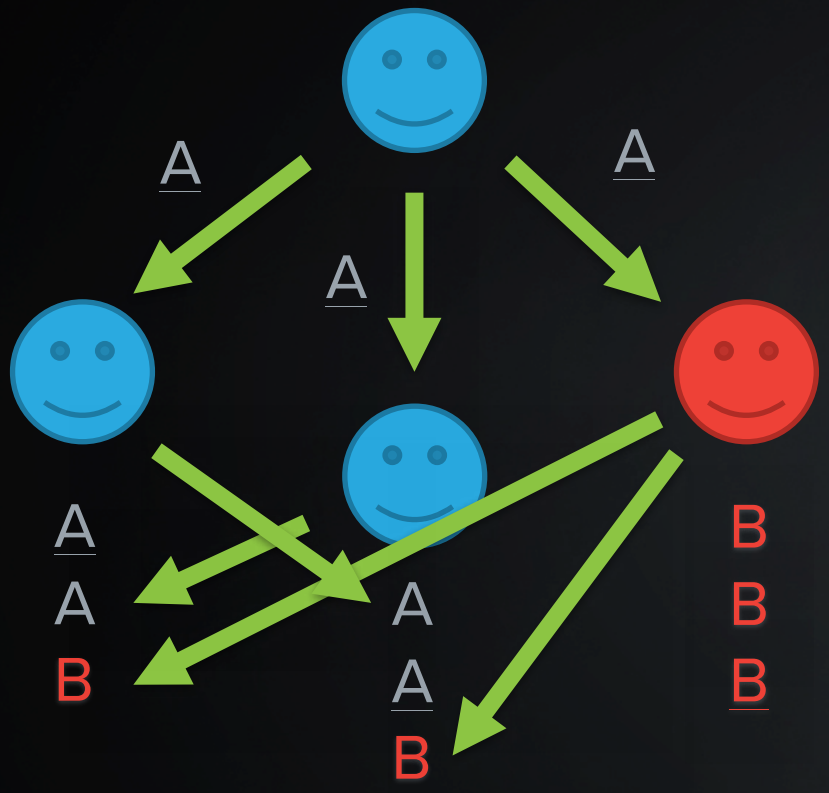
- N Generals want to attack a fortress, but they can only win if they attack together
- Among the generals there is a traitor (or more), trying to make the attack fail
- Suppose one of them gives a command, can you think of a mechanism to bring all remaining generals to consensus?

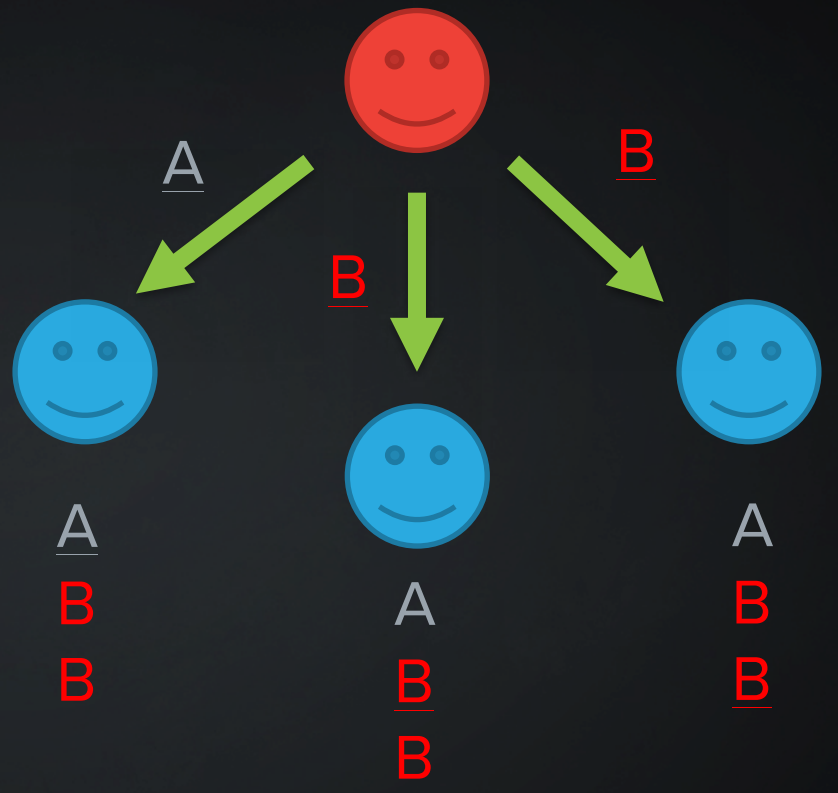


- Please try with pen and paper
- 5min

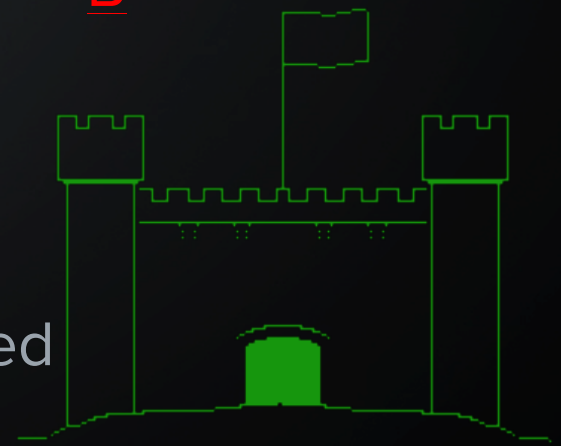
Byzantine Fault Tolerance

See: Miguel Castro, Barbara Liskov: Practical Byzantine Fault Tolerance (1999)



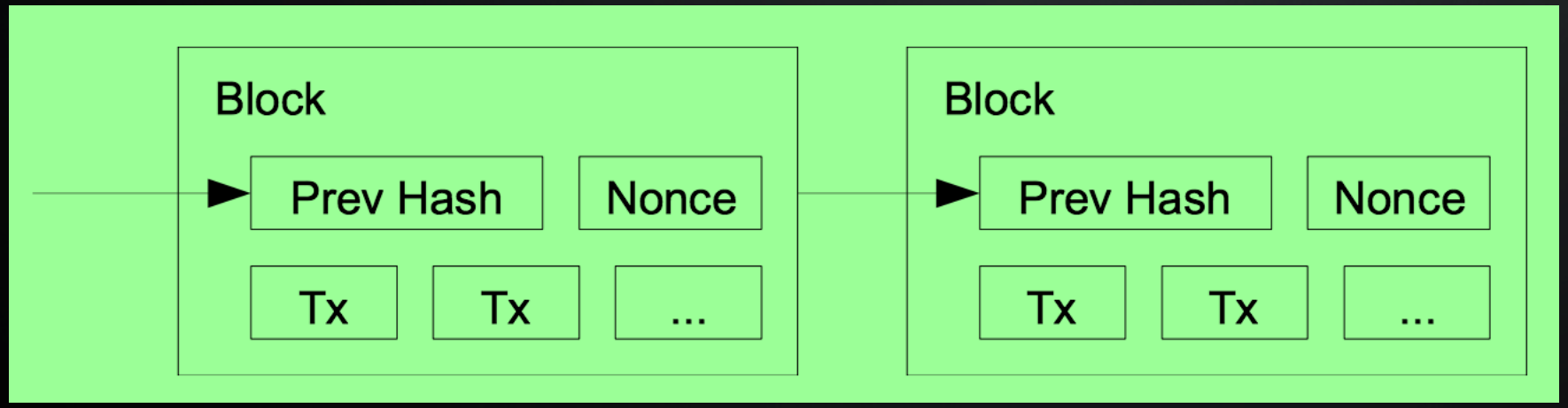


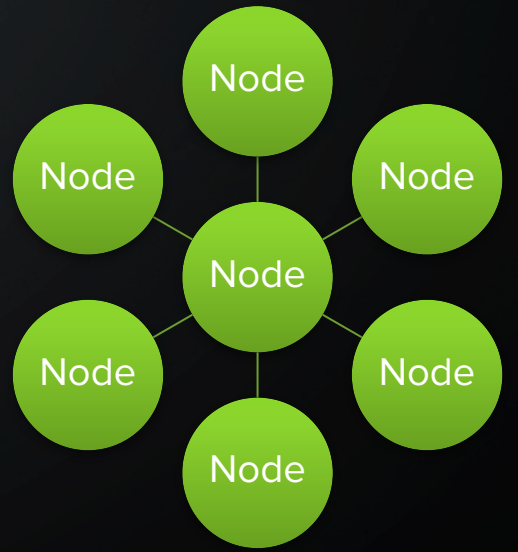
- One general sends a request to all others
- They again ask all others about what they received
- Each general does a majority vote on the responses he received
- To tolerate *f* traitors, we need $n > 3f$ generals



Proof Of Work

- Different approach
- Some nodes (=miners) are computing sha256 hashes until the leading N bits of the resulting hash-sum are 0
- This ensures they do difficult computational work
- If a node finds a such a hash ➔ new block
- The hash includes a hash of the block before and transactions that have occurred since the last block
- This forms a chain of blocks with payload (e.g. transactions)

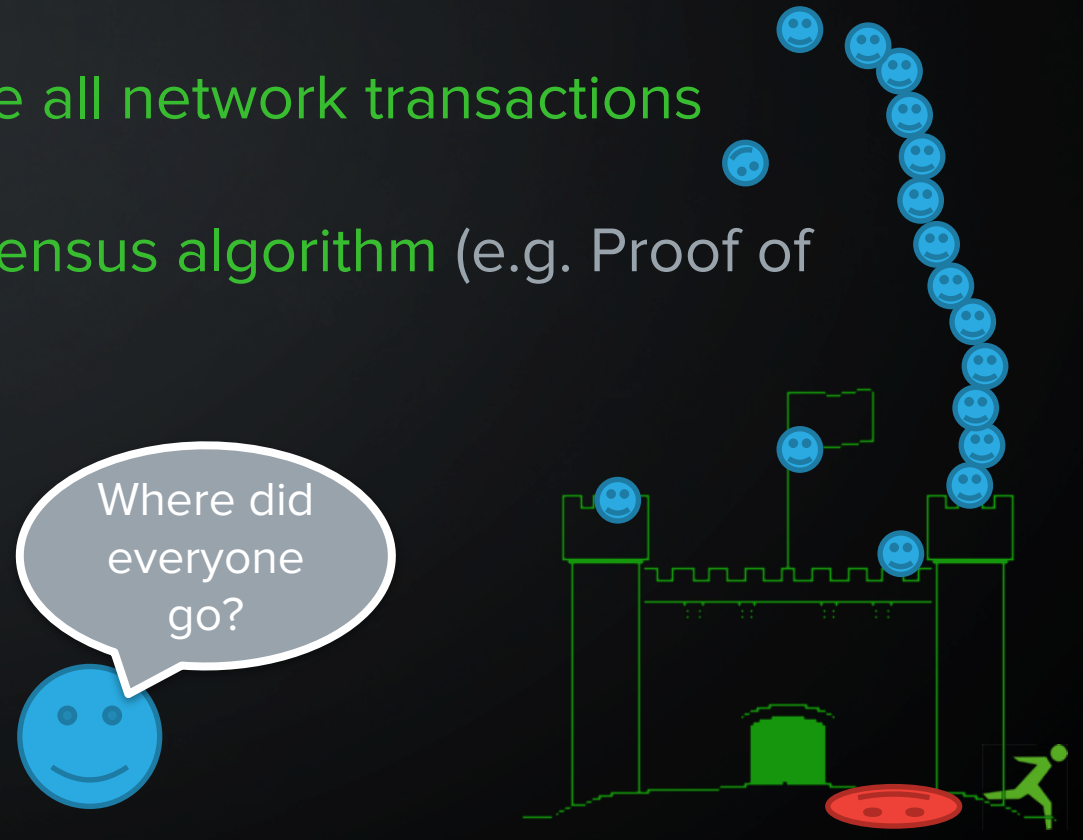




Bitcoin: A Peer-to-Peer Electronic Cash System Satoshi Nakamoto

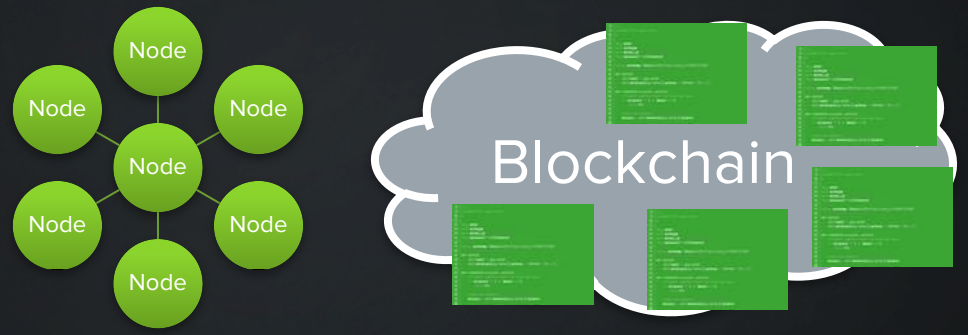
Blockchain: Recap

- Network of nodes / peers
- They all maintain a copy of the Blockchain
- The Blockchain consists of blocks which store all network transactions
- Consensus between nodes is kept by a consensus algorithm (e.g. Proof of Work, Proof Of Stake, PBFT, ...)



Smart Contracts

- The original idea (as published within the Bitcoin paper) included only basic transactions within blocks
- However, recently, Blockchain networks have appeared that support storing programs in the blockchain



- Such programs are called smart contracts, as they define rules to alter the state of the blockchain (e.g. variables stored in on-chain-memory)
- Each node within the network verifies correctness of program execution, hence the Blockchain is transformed from one consistent state to the next

Chains with Smart Contract Support

Hyperledger:

- Suited for enterprise Blockchain solutions
- Open-source project started by Linux-Foundation

HYPERLEDGER

Ethereum:

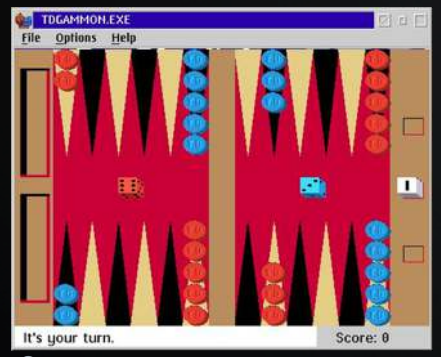
- Blockchain implementation with smart contract support
- Easy tools for development of smart contracts and testing
- Good way to start and learn smart contract development

ethereum

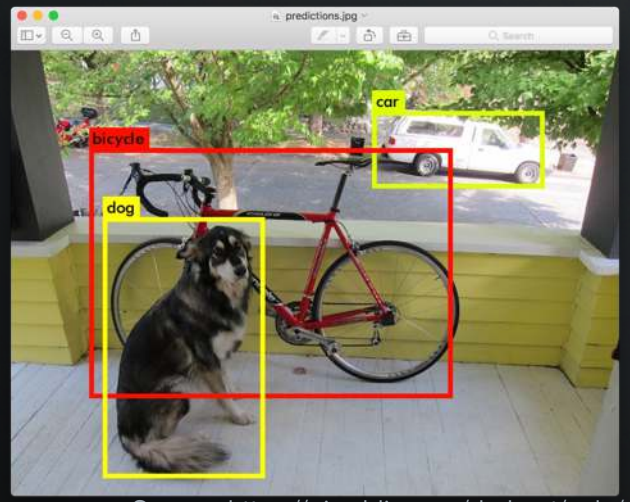
Intro: Machine Learning

- Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed. (Arthur Samuel, 1959).

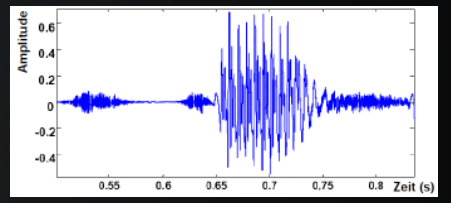
- Fantastic possibilities: image recognition, text mining, time series prediction, anomaly detection, image de-noising, language translation, speech recognition, Backgammon, autonomous driving, and a lot more

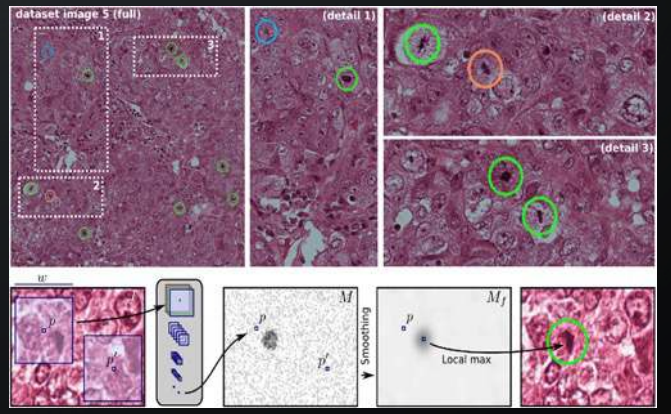


Source: https://en.wikipedia.org/wiki/TD-Gammon



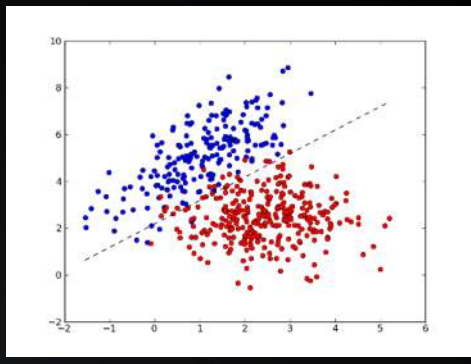
Source: https://pjreddie.com/darknet/yolo/





Source: http://people.idsia.ch/~juergen/deeplearningwinsMICCAIgrandchallenge.html

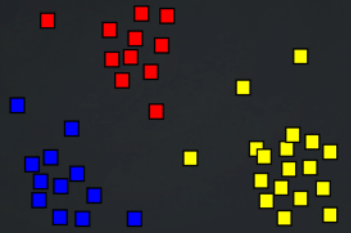
Overview: Machine Learning

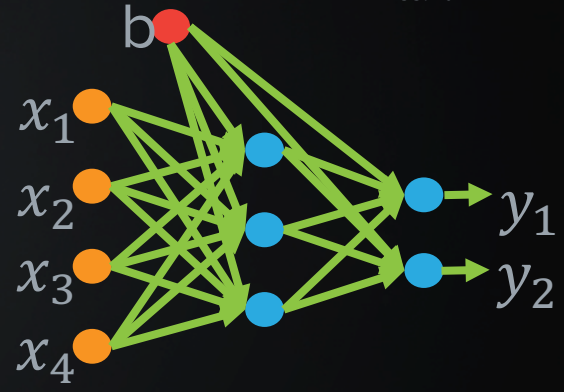


Source: http://mlpy.sourceforge.net/docs/3.5/lin_class.html

Supervised Learning:

- Classification / Regression
- For each input vector a label or target value is given
- Algorithms: Multilayer Perceptron, Support Vector Machine, Random Forest, Adaboost, and many more





Unsupervised Learning:

- No labels / target given
- Purpose is to find clusters within data
- Algorithms: K-Means, DBSCAN, OPTICS, Expectation Maximization, Kohonen-Maps, and many more

Reinforcement Learning:

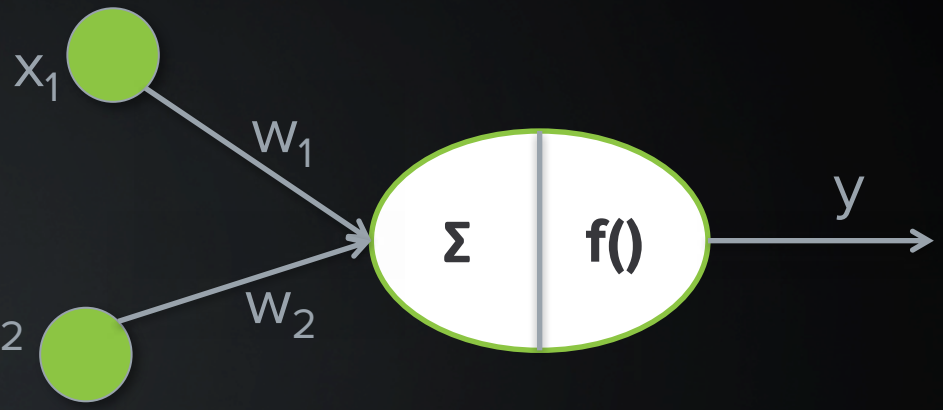
- No exact label given, algorithm learns using positive and negative reward

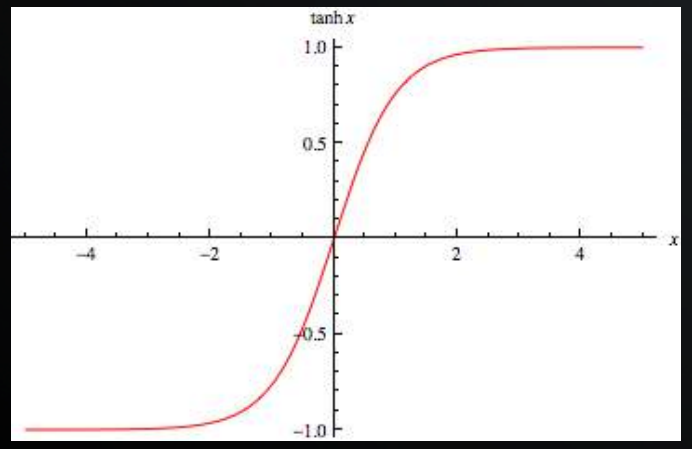
Artificial Neural Networks (ANN)

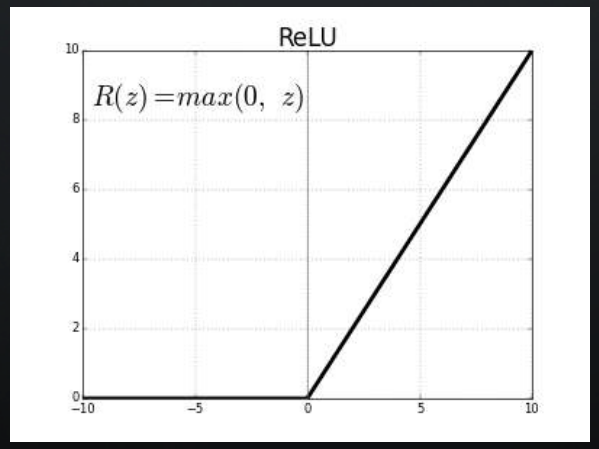
- Biologically inspired family of algorithms to solve learning tasks
- We'll focus on supervised learning here

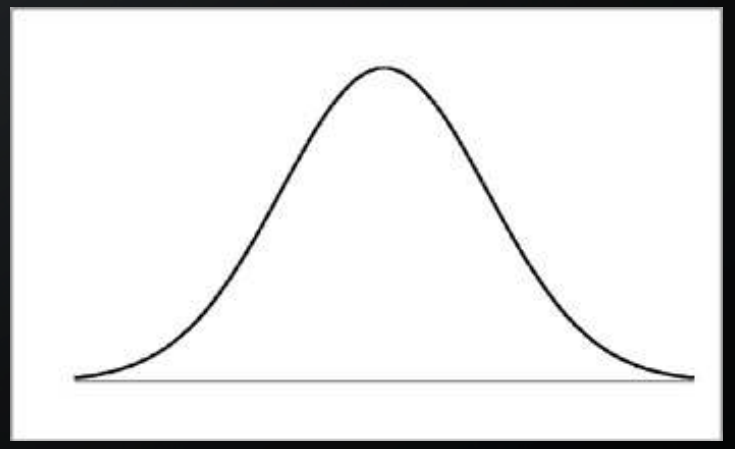
Neuron:

- Basic building-block of an ANN
- Maps several inputs to one output
- Each neuron has weights associated to its inputs
- $y = f(x_1 w_1 + x_2 w_2 + \cdots + x_n w_n + b)$
- $f(z)$ could be: sigmoid: $\tanh(z)$, relu: $\max(0, z)$, linear, gaussian, etc.



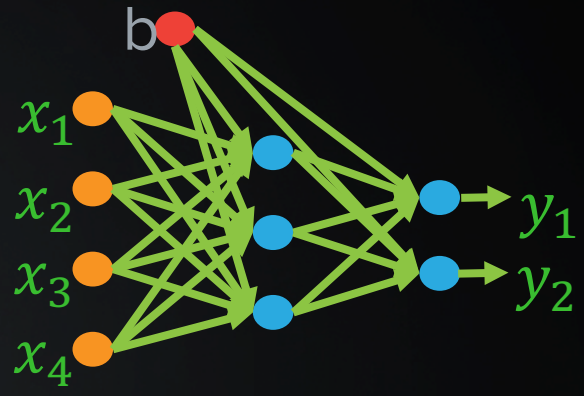






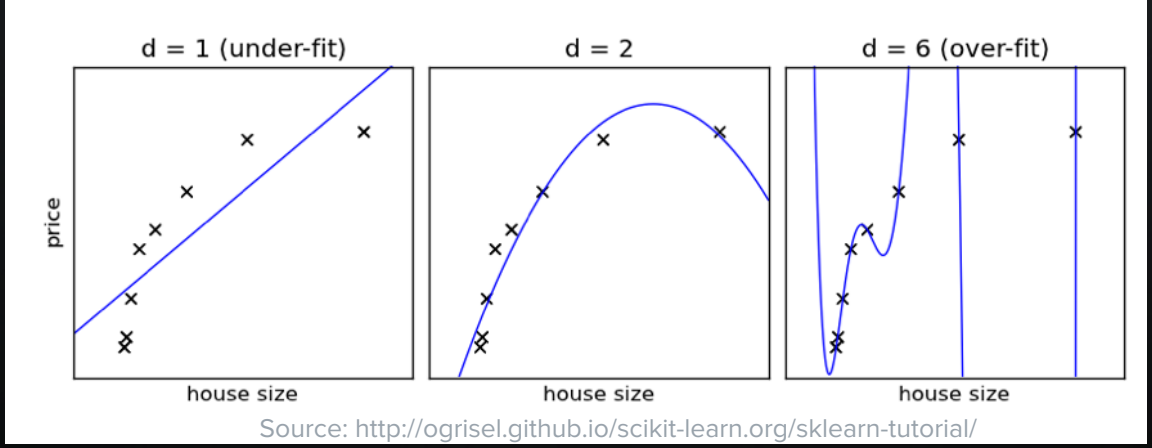
Neurons: Stacking them Together

- Neurons can be combined into multiple layers forming a simple ANN:



Input vector **x**
Output vector **y**

- The architecture will depend on the specific use-case, however:
- The more neurons / weights and layers within the network, the more complex the problems the network can solve
- However: more weighs need more training data and careful handling: overfitting



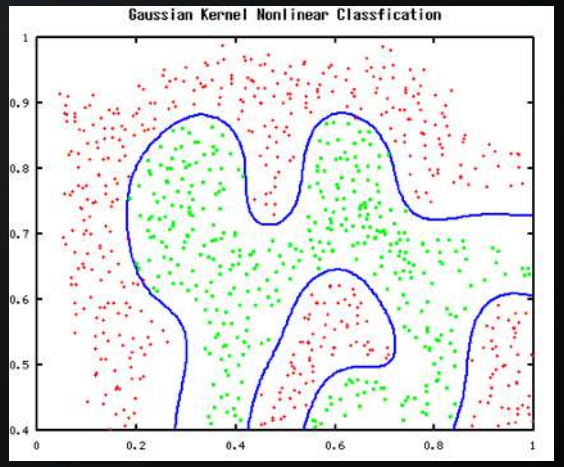
Source: http://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/

Training ANNs

- Objective: The network should be able to generalize and give a good prediction for unknown input vectors
- What can be adjusted within the network?
- ➔ Neuron weights!

- We are searching for a set of weights that enables our network to make accurate predictions, that is, to give a correct output (category/value) for given inputs
- As this can be formulated as an optimization problem, ANNs are typically trained using optimization techniques, such as error-backpropagation (which is actually a gradient descent method)

- Objective: minimizing prediction / classification error

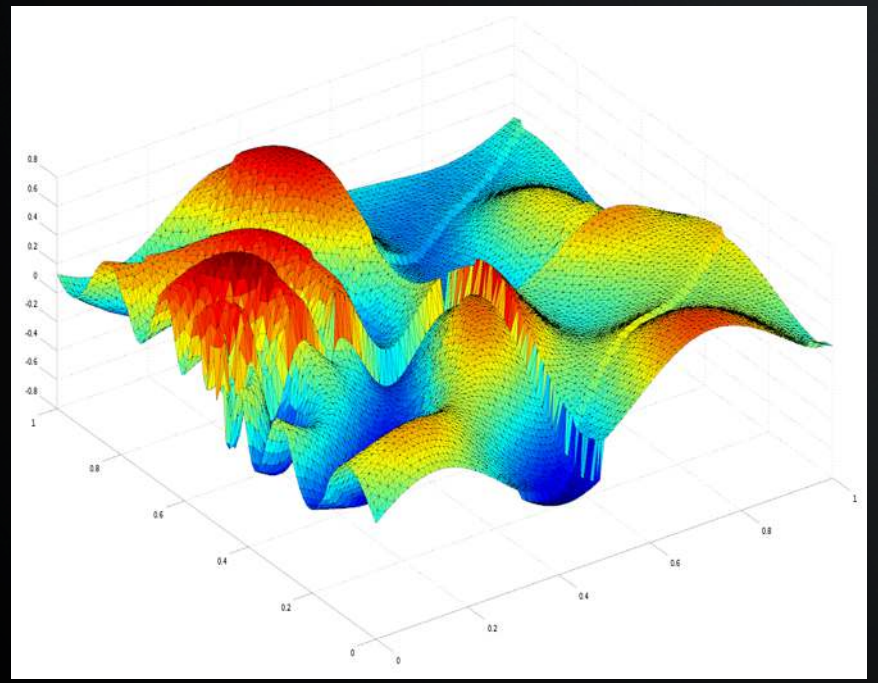


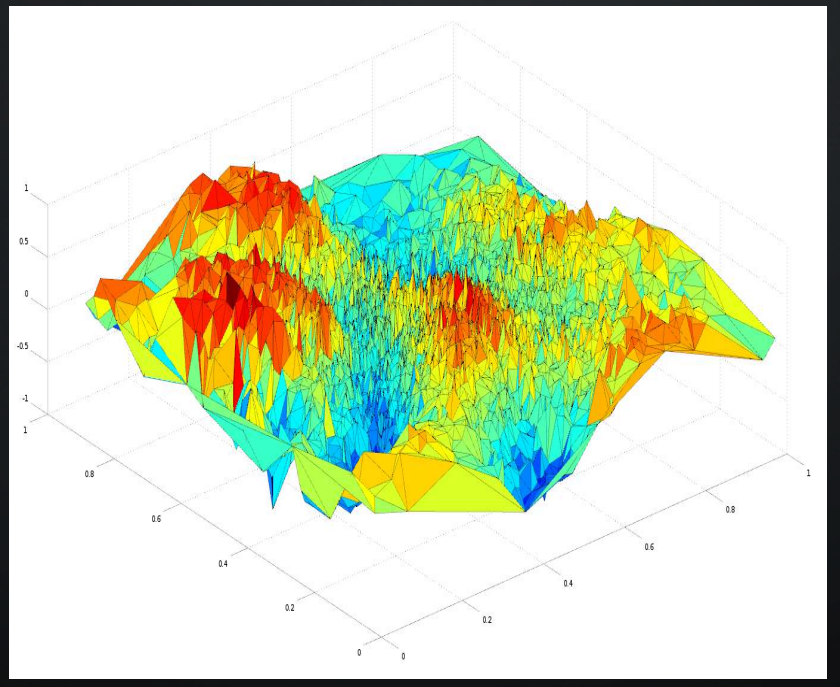
Source:
http://xenon.stanford.edu/~jianzh/ml/kernel.PNG

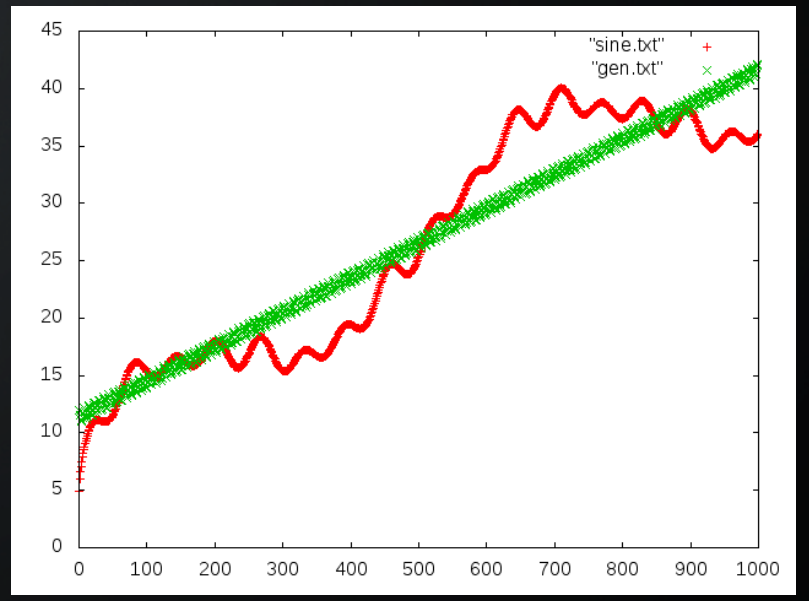
Some Training Details

- A batch of training samples (input vector x and correct output „teacher“ value t or class label) is fed into the network one by one
- Deviation between output vector y and t ➔ prediction error E

- Training will try to find a minimum in the error surface over all training samples by adjusting neuron weights







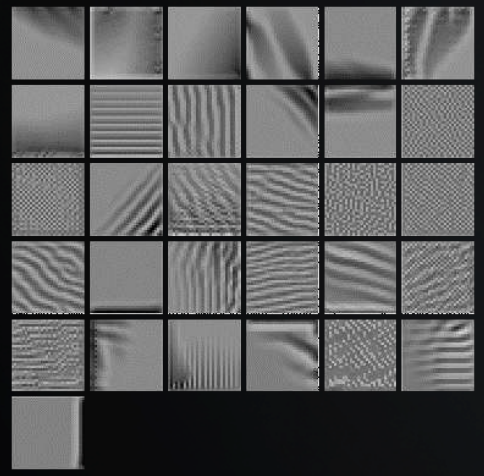
I need your help:

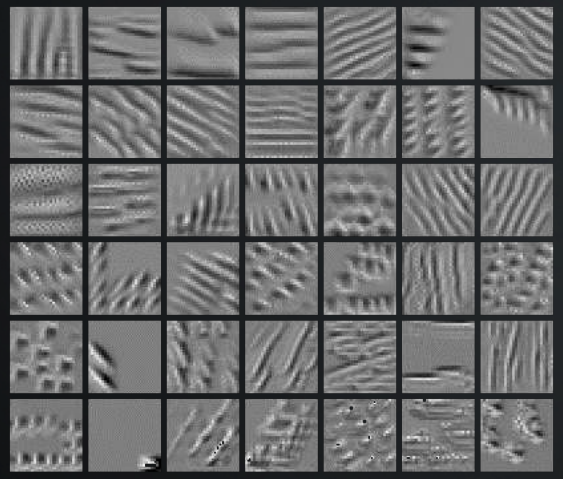
Let’s beat the image classifier!

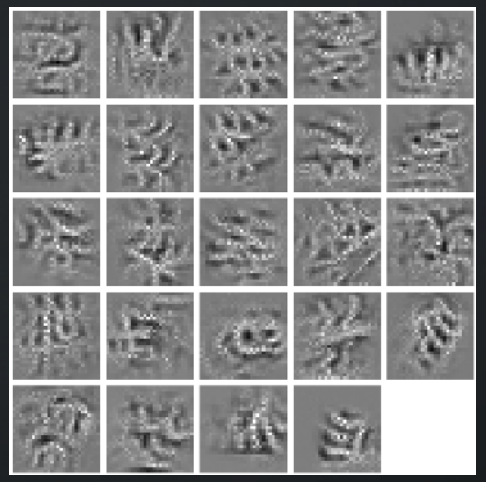
Deep Learning: Motivation

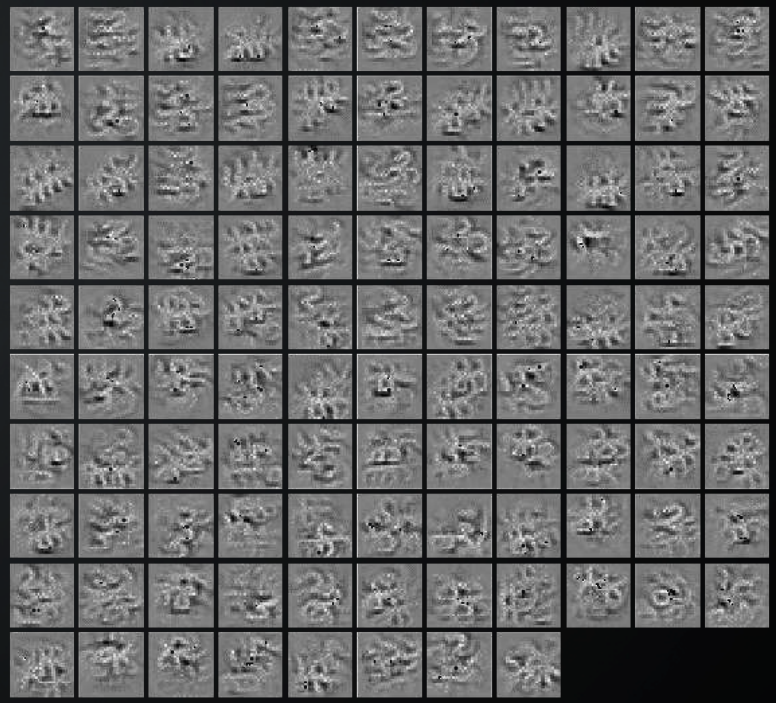
- In recent years, computing power has increased dramatically
- Ever-growing amount of digital data that can be used for training

- Researchers found out that adding more hidden layers to an ANN will enable it to learn hierachical abstraction levels of the data









Source: Aufarbeitung aktueller Deep-Learning-Techniken – Daniel Seichter 2016 TU Ilmenau, NIKR

- Important difficulties: vanishing / exploding gradients, overfitting
- Solutions: Relu activation, dropout, batch normalization

Breaking Records

- Deep Learning is breaking records in many disciplines nowadays

Examples:

- 2009: Speech Recognition: A system based on Deep Learning could transform speech to text better than previous algorithms → Today Apple Siri and Android speech recognition both utilize DL

Source: A. Krizhevsky, et al.: Imagenet classification with deep convolutional neural networks, NIPS, 2012

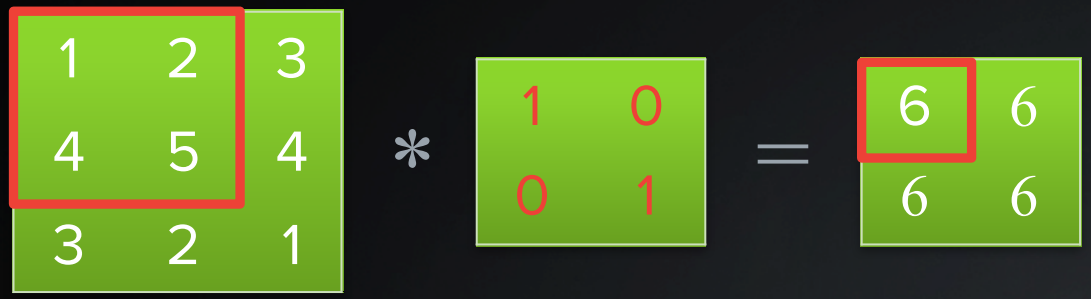
- 2012: Large Scale Visual Recognition Challenge 2012: Team around Geoffrey Hinton (Univ. Toronto, Google) introduced AlexNet and it had a top5-misclassification rate of only 15.3% (the winner one year before: 26,2%)

- 2016: Google DeepMind's AlphaGo won against Lee Sedol, one of the best Go-players world-wide (it uses only 1920 CPUs and 280 GPUs, you see Go is difficult ;-))

Source: http://www.latimes.com/business/la-fi-alpha-go-victory-20160315-htmlstory.html

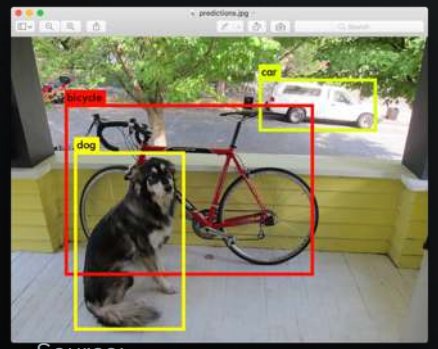
Convolutional Neural Networks

Recap – Image Convolution:

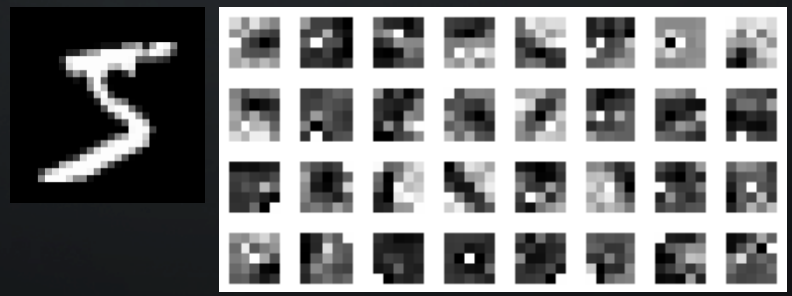


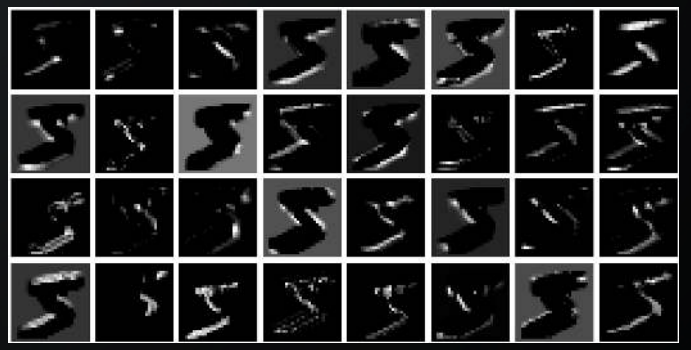
Step1: $(1 * 1 + 2 * 0 + 4 * 0 + 5 * 1) = 6$

- Convolutional Neural Networks consist of several convolutional layers
- The next layer receives only the filter responses from the layer before
- Filter kernel weights are optimized during training as well



Source: https://pjreddie.com/darknet/yolo/

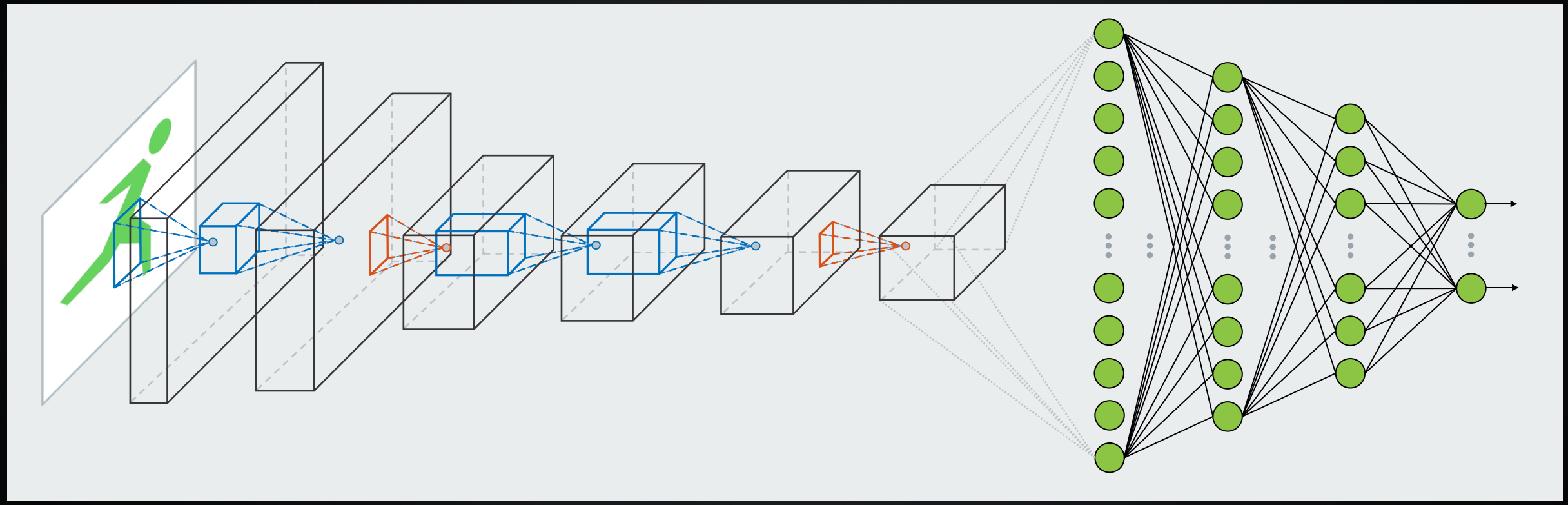




Source: Aufarbeitung aktueller Deep-Learning-Techniken – Daniel Seichter 2016 TU Ilmenau, NIKR

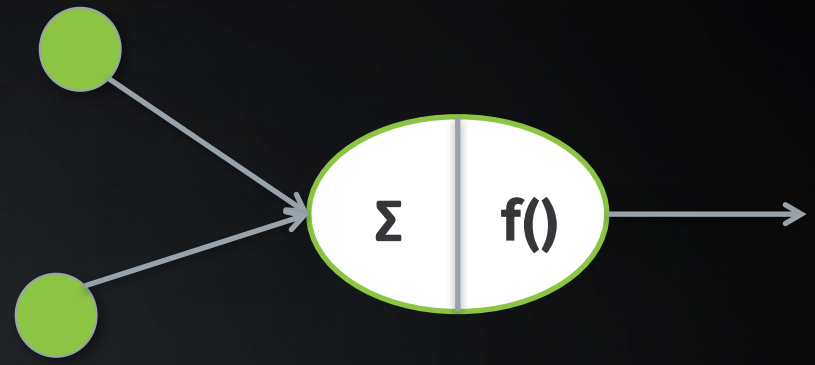
Convolutional Networks [2]

Example Network Structure:



Source: Aufarbeitung aktueller Deep-Learning-Techniken – Daniel Seichter 2016 TU Ilmenau, NIKR

Machine Learning: Recap



- Algorithms that learn to solve problems
- Artificial Neural Networks consist of neurons
- They can be used to classify input vectors (and for regression and clustering, etc.)
- Deep Neural Networks consist of many hidden layers and can learn hierachical abstractions within the input space
- Convolutional Neural Networks possess convolutional layers that apply trainable filter kernels on the data (e.g. images)

Checkpoint

- Blockchain theory 
- Machine Learning Overview 

Defining Our Objective

- Build a decentralized message service
- Send text messages and images to a blockchain
- Use Convolutional Neural Networks to extract image tags
- Subscribe to tags and receive the posts you're interested in
- Do all that in about 350 lines of Python code

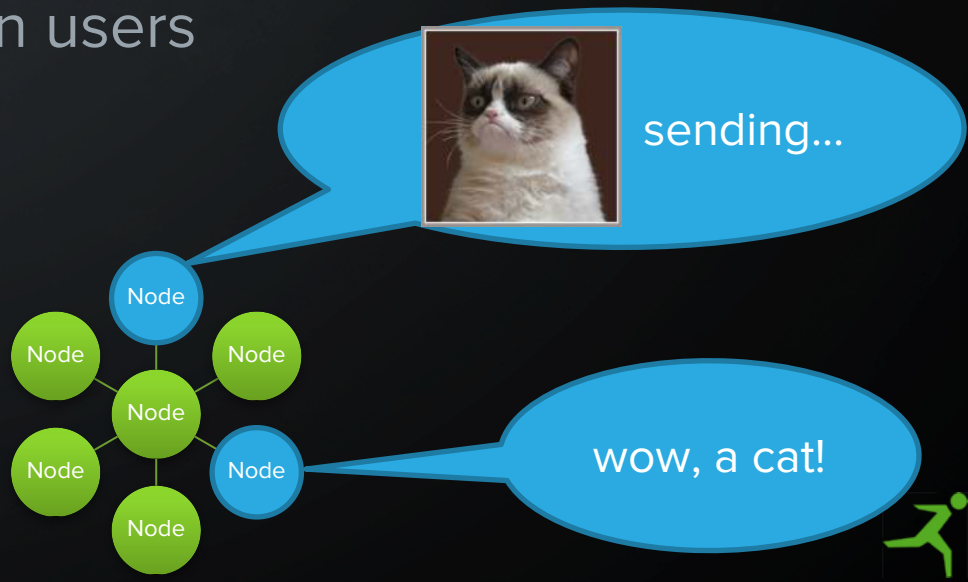
```
----------------------------------------
found contract on blockchain!
----------------------------------------
starting chat command line...
>> send
----------------------------------------
[composing new message]
message....: Hi, my name is Thomas
image file.: img.jpg
custom tags: #start #reply #test
----------------------------------------
sending...
```

python powered

```
print("Hello, world!")
```

Why is that interesting? [1]

- Give you the building-blocks to create decentralized applications on your own
- The network is fully functional and tamper-proof as long as enough nodes are using it
- Data is timestamped, automatically analyzed, structured and easy to find
- Fast, intelligent and secure collaboration between users



Why is that interesting? [2]

- We can apply this technology to companies, e.g. for improved document and contract handling
- Documents can be analyzed for topics, images, reasonable recipients via Machine Learning
- Using a company-wide Blockchain, they will be recorded and timestamped, are immune to tampering and will be available in seconds
- By the way: we can do that for you ;-)

Our Demo Architecture

How would you do it? Let's consider the following pieces:
(Please fill in the diagram on your table, 5min)

User

BC-
Node

Blockchain

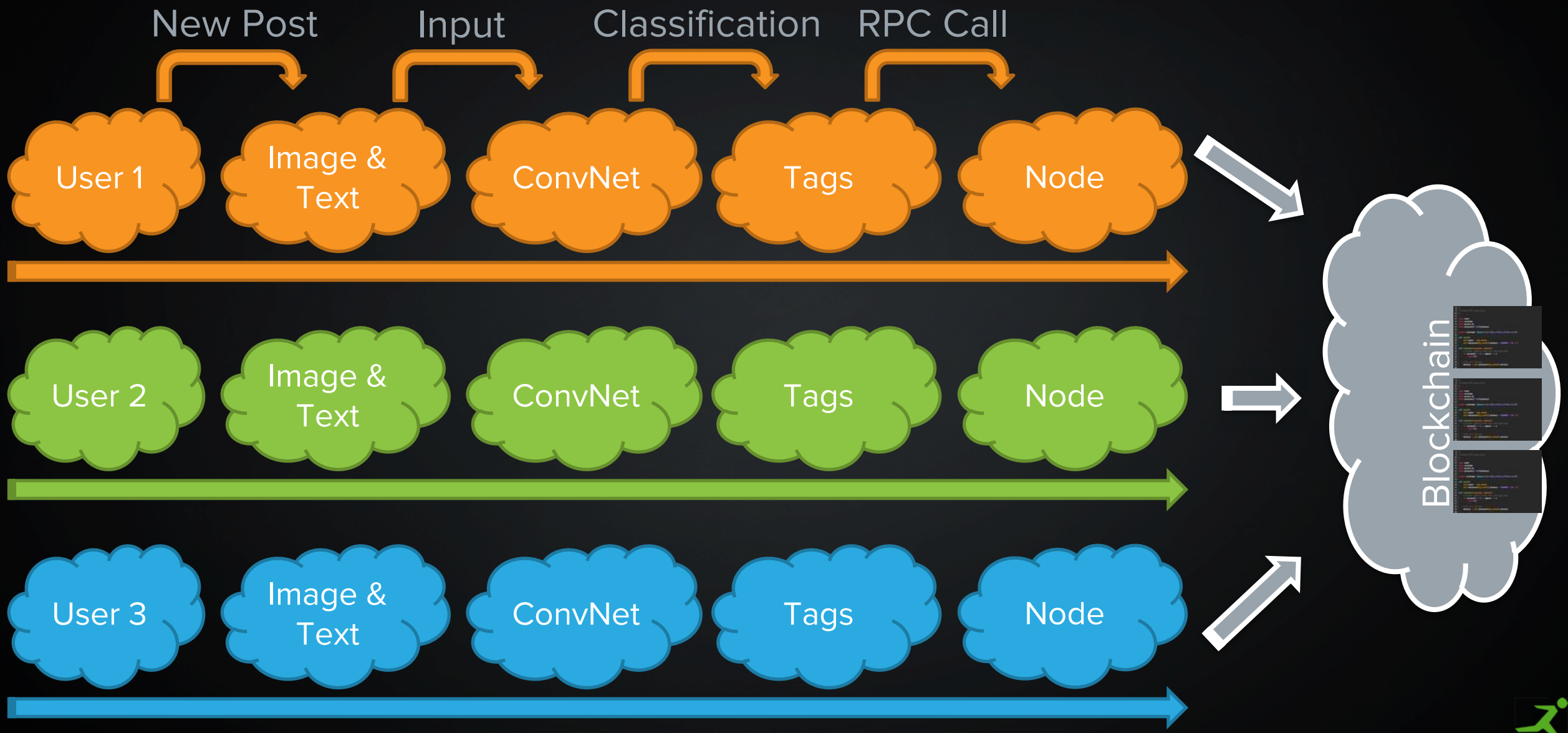
ConvNet

Smart
Contract

Tags

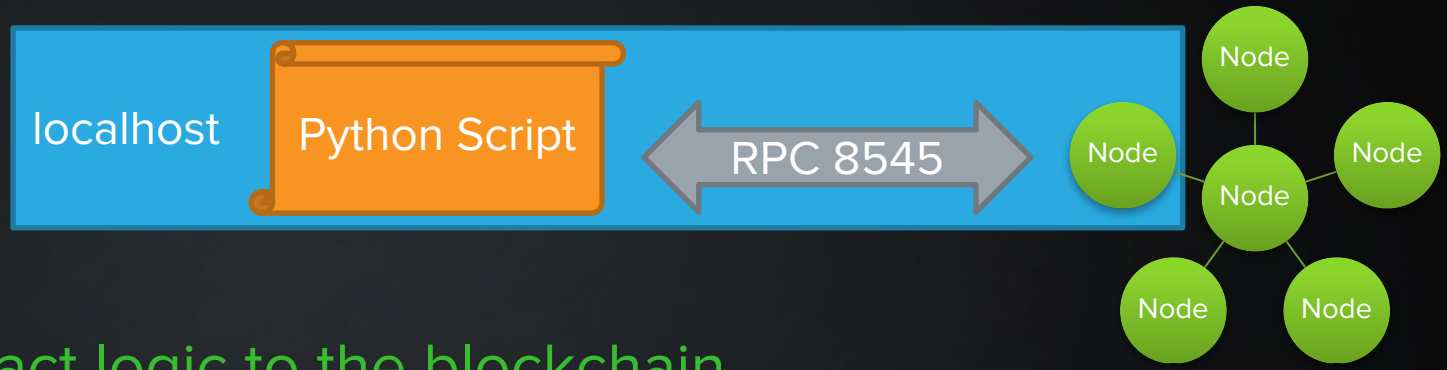
Image &
Text

Our Demo Architecture



Our Demo Architecture

- Each user will run a blockchain node, together they will form the network
- Communication with the local blockchain node is done via RPC (remote procedure call)



- We need to put our smart contract logic to the blockchain
  - it contains a mapping for every user account to [message, image, tags]
  - for every new post, this content is changed, but the history remains inside the chain

- Finally, we need a client tool for composing messages, subscribing to topics and receiving posts

Development Environment Setup

- We'll be using a private development Ethereum blockchain here (→ https://github.com/ethereumjs), since testing and setup is very easy

```
~/workspace/wrk/workshop$ testrpc
```

- Local testnet that simulates real blockchain network
- Identical interface and behavior
- 10 user accounts automatically created
- Listening for RPC requests on port 8545

```
EthereumJS TestRPC v3.0.3

Available Accounts
==================
(0) 0x727ac9f9a3a470cd1437dc3cfb0bf55566b13e01
(1) 0x51a0039aa4ef8b2408bbdbcda83f53deb42be874
(2) 0x09e956ffc8fba560befc175eeca55ee26542cffa
(3) 0x24d3d62dad12a8dd27ad3532ccb5a1c005a3eafe
(4) 0x5ca962eecae2a50bb7679277fced10faddac81a3
(5) 0x059f0391ea6102e565559110a05001db9503f099
(6) 0x2e9203da725ba08877925e3cf8dad28df9ed2844
(7) 0x9f2ef7919af6442b66afd281cd7422653d0bf923
(8) 0xa0cb7e68bd6744c4fd6916846ce3bb4e3e3fb4ca
(9) 0x0c1202e576f83fe09ab1584212eddcea80d1b6f7

Private Keys
==================
(0) 8296df3e1b8f5f42a67503c779efd385b234c416d8578ce40bf9a4e2cec4b9a7
(1) 757d480a9425422de317f03dceec554ecc516f671df265bc697624631ecde9da
(2) f81fd407a79744a457ae7d0bc433ab0d1ac69909d25f729b6fff4be3f559a443
(3) 1658310940d424ba5437c343b4fe09a998c8d48ed4278fc30d5f0db180de9250
(4) 48a681c6d34c7cd3b98454f8c5fd35bb16d24a2e6a0d0705d87c598d339ae429
(5) 61cafbee004113a6e1dc27b0e9b96c251cf48bbb9ccdd0b38c7bf11f35d3f29d
(6) e1fc0c0df432d6081ee88ba1b2210288e8ed878f2296be83c965571c14df2b27
(7) 60a26e44fb7fcf7911993bb920229679e539b40647d8c5ec7651ce220be999da
(8) 8ebab6fa63fad7c65ab74a0a25cff49a0bfefd7b20abbc774d90a2a6749f75d3
(9) 5245e9e7b9183fc59f87c6ed7e9abd886ce8201ec72ed9858701319c89fc17bf

HD Wallet
==================
Mnemonic:      girl cry suggest minor valve orient mouse elder evolve animal puppy pumpkin
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

Package: Python

- We'll be using the Python programming language here, because it's cool (➔ https://www.python.org/)


```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

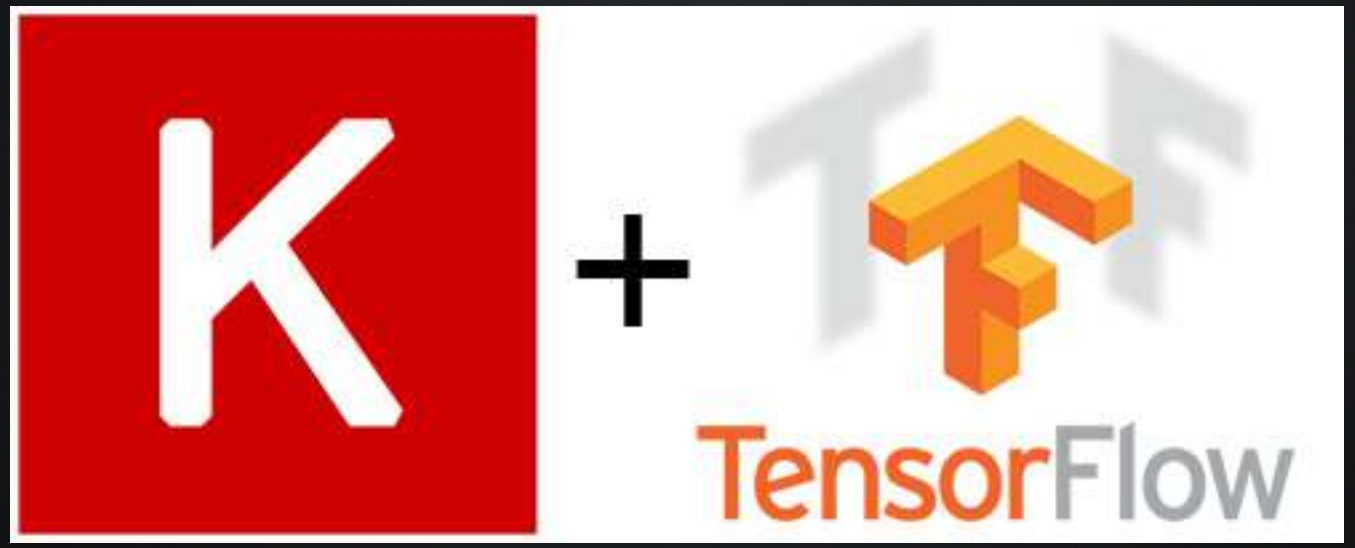
Package: ethjsonrpc

- For communication with the blockchain node, let's install the ethjsonrpc package
- It will provide an interface for communicating with our local blockchain node

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ethjsonrpc import EthJsonRpc
>>> rpc = EthJsonRpc('localhost', 8545)
>>> rpc.eth_blockNumber()
19
>>> rpc.eth_blockNumber()
19
>>> rpc.eth_blockNumber()
20
>>> rpc.eth_blockNumber()
20
>>> rpc.eth_blockNumber()
21
>>>
```

Packages: keras + tensorflow

- Keras with Tensorflow backend will take care of image classification for us, in 6 lines of code



Now, let's get our hands dirty.
Step-by-Step review of the client scripts:

Smart Contract

Deployment Script

Messaging Client

That was just an Overview

- More Machine Learning stuff: Natural Language Processing, Document Topic Extraction, Optical Character Recognition, …
- More Blockchain stuff: Smarter Smart Contracts, Blockchain Analytics (the chain has all the data = topics = messages = images, …), …
- Any questions? Ask them now, or drop me an e-mail. 

t.schmiedel@reply.de