



**UNIVERSITÀ
DEGLI STUDI
DI UDINE**

hic sunt futura

PRIN2020 Nirvana Project Update

PhD Dalila Ressi

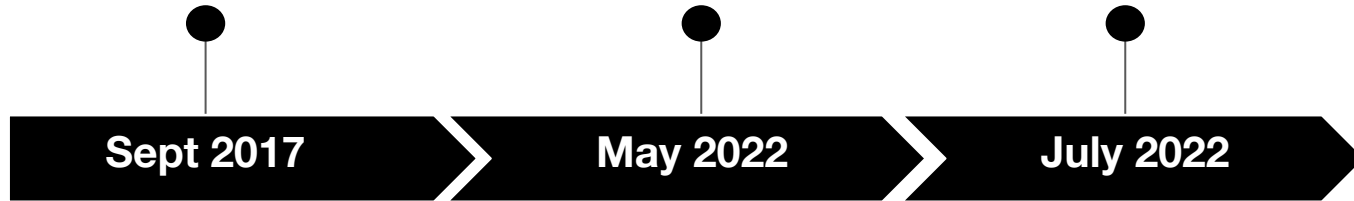
Supervisors:
Carla Piazza, Sabina Rossi

About Me

Started PhD in Ca' Foscari
in collaboration with Microtec Srl

PhD degree

Post-doc in University
of Udine



Università
Ca' Foscari
Venezia



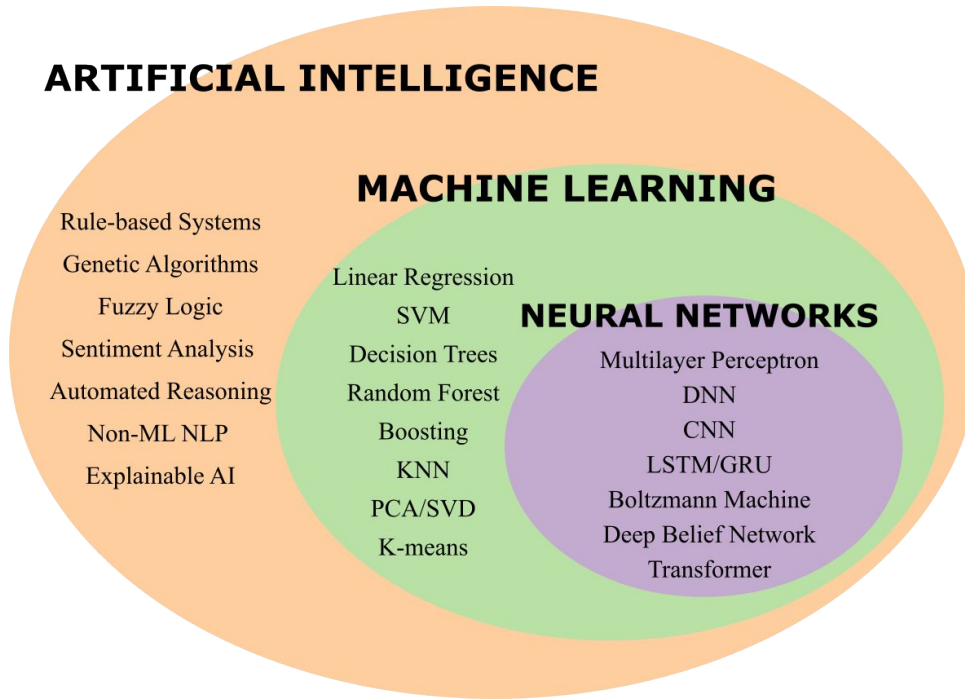
**UNIVERSITÀ
DEGLI STUDI
DI UDINE**
hic sunt futura

PhD thesis: *“Convolutional Neural Networks Compression for Embedded Industrial Applications”*

Research Works

- “Neural Networks Reduction via Lumping” D.Ressi, R. Romanello, S. Rossi, C. Piazza (Aixia 2022, working on journal edition)
- “AI-Enhanced Blockchain Technology: a Review of Advancements and Opportunities” D.Ressi, R. Romanello, S. Rossi, C. Piazza (Submitted to JNCA)
- “Security Verification of Ethereum Smart Contracts with ML Taking a Free Ride on Static Analysis” D.Ressi, R. Romanello, S. Rossi, C. Piazza, M. Bugliesi, S. Crafa (DLT2023, oral communication)

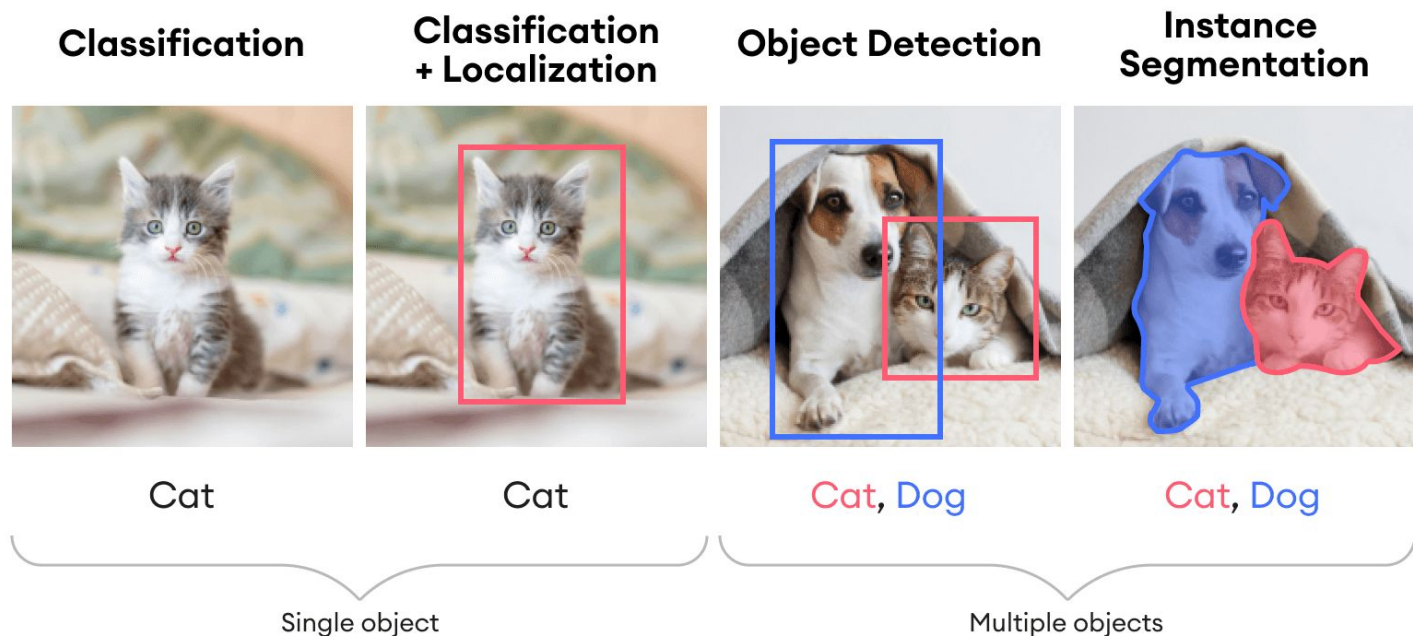
Artificial Intelligence Background



Tasks that can be solved with AI are mainly divided into three categories:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised Learning: Image Recognition



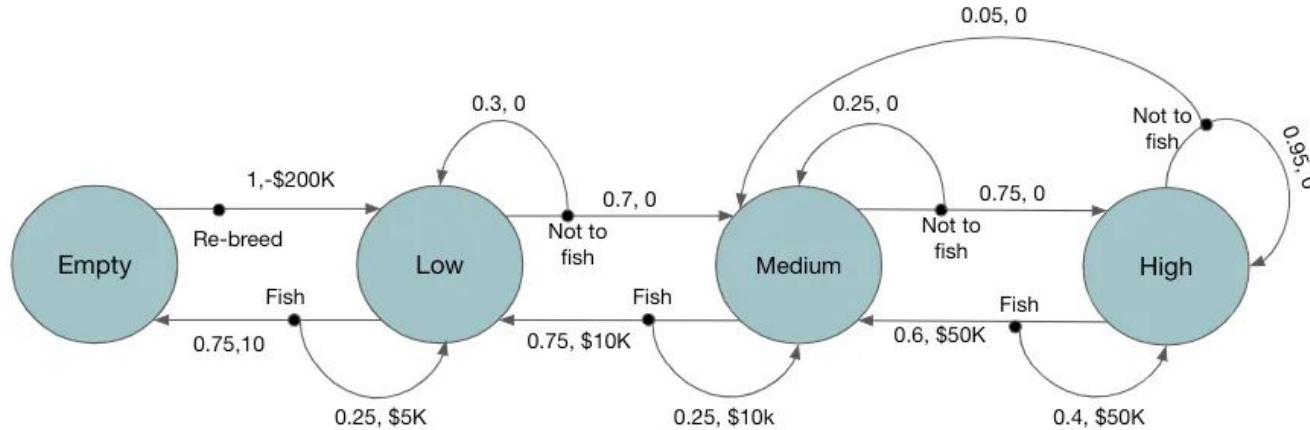
Unsupervised Learning: AI-generated content

ChatGPT

| ☀ Examples | ⚡ Capabilities | ⚠ Limitations |
|--|--|---|
| "Explain quantum computing in simple terms" | Remembers what user said earlier in the conversation | May occasionally generate incorrect information |
| "Got any creative ideas for a 10 year old's birthday?" | Allows user to provide follow-up corrections | May occasionally produce harmful instructions or biased content |
| "How do I make an HTTP request in Javascript?" | Trained to decline inappropriate requests | Limited knowledge of world and events after 2021 |



Reinforcement Learning: Markov Decision Process



Transition graph of fishing salmon markov decision process:

RL can be used to decide if to fish or not to fish to maximize reward over a certain period of time

AI-Enhanced Blockchain Technology: a Review

During this year we mainly worked on a survey about AI/ML and blockchain (BC)

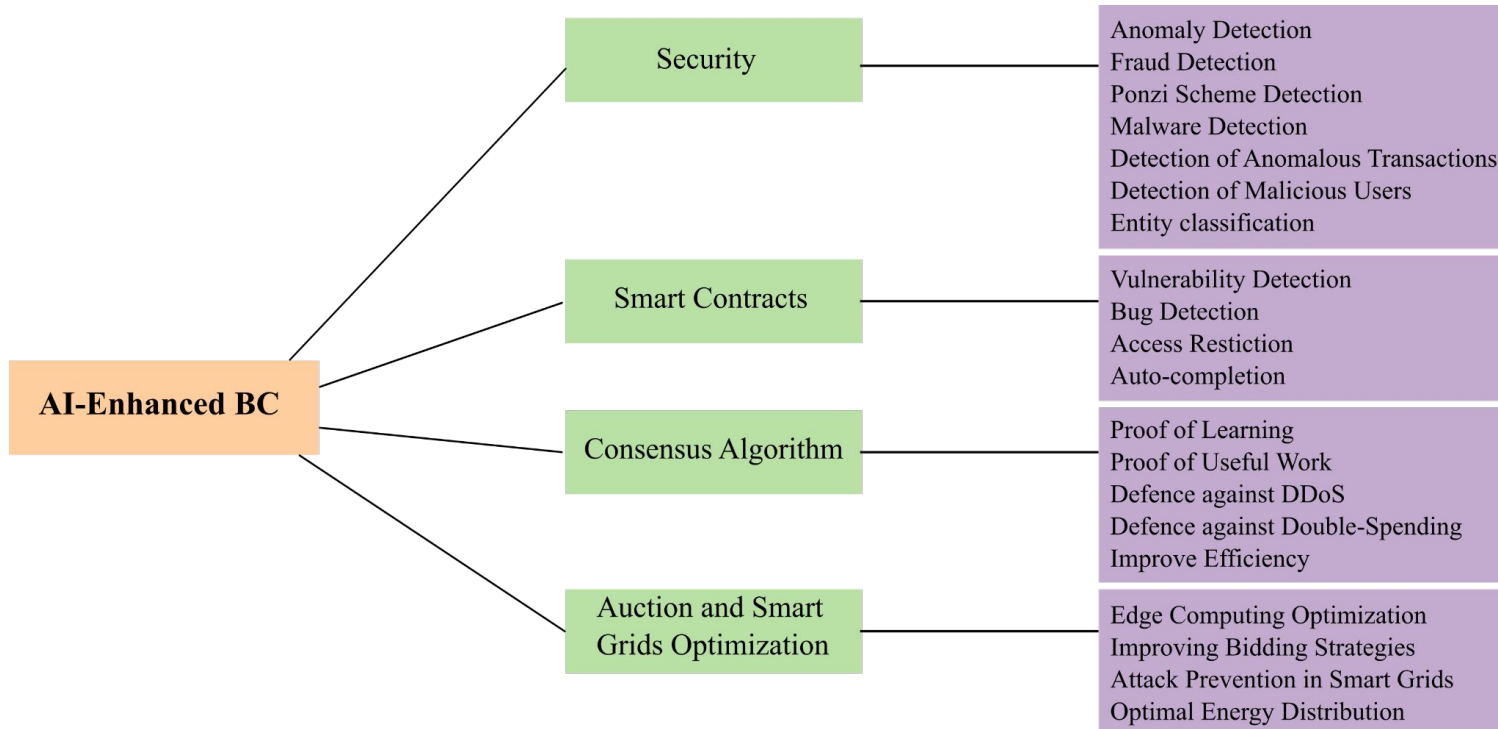
BC + AI in many application areas:

- Healthcare
- IoT/loV/Smart Cities
- DeFi
- Cryptocurrency

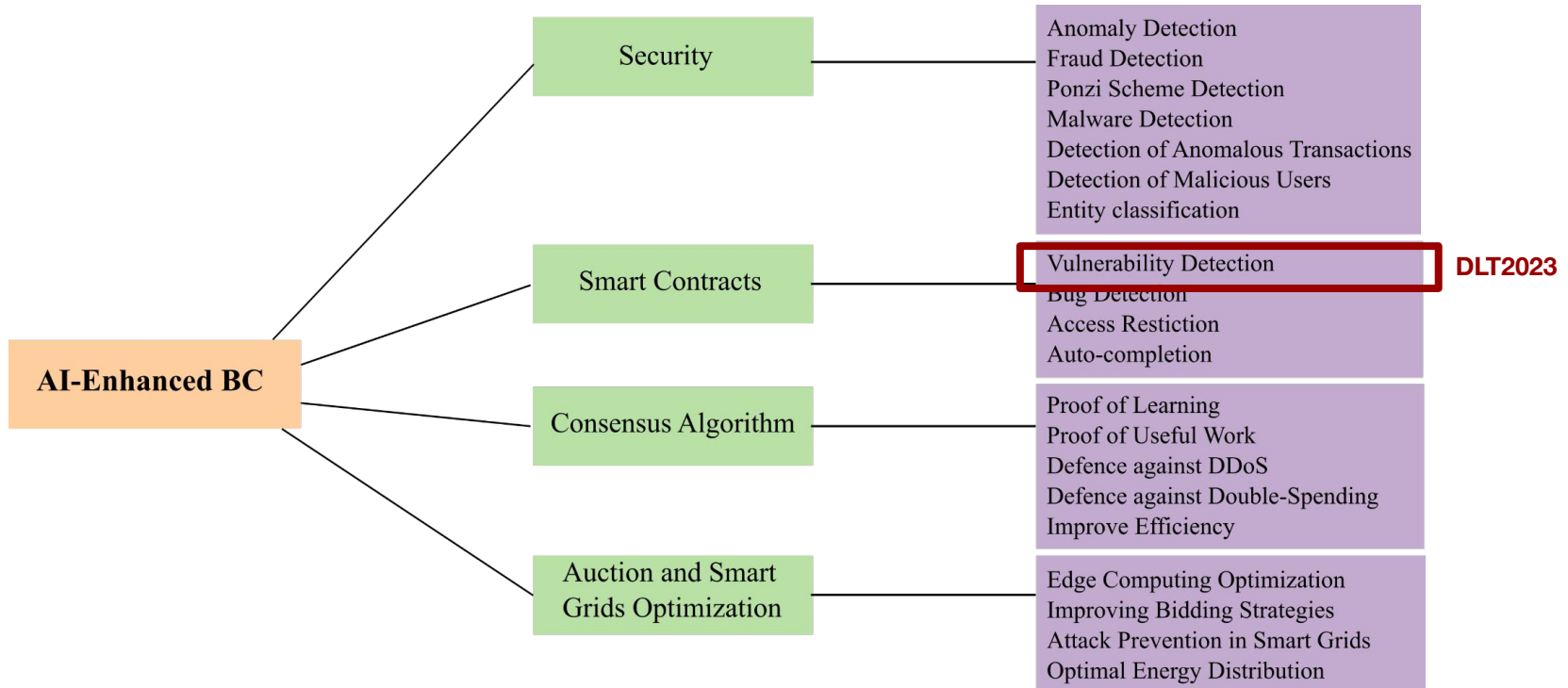
Areas improved through AI:

- Security
- Consensus Algorithm
- Auction and Smart Grids
- Optimization
- Smart Contracts

Areas Currently Improved Through to AI/ML



Areas Currently Improved Through to AI/ML



Ethereum Smart Contracts

```
pragma solidity ^0.8.0;
```

```
contract Storage {  
    uint256 data;
```

```
    constructor (uint256 _data) {  
        data = _data;  
    }
```

```
    function set(uint256 _data) public {  
        data = _data;  
    }  
}
```

Contract Code

```
PUSH1 0x80 PUSH1 0x40 MSTORE  
CALLVALUE DUP1 ISZERO PUSH2 0x10  
JUMPI PUSH1 0x0 DUP1 REVERT  
JUMPDEST POP PUSH1 0x40 MLOAD  
PUSH2 0x188 CODESIZE SUB DUP1 PUSH2  
0x188 DUP4 CODECOPY DUP2 DUP2 ADD  
PUSH1 0x40 MSTORE DUP2 ADD SWAP1  
PUSH2 0x32 SWAP2 SWAP1 PUSH2 0x54  
JUMP JUMPDEST DUP1 PUSH1 0x0 DUP2  
SWAP1 SSTORE POP POP PUSH2 0x9E  
JUMP JUMPDEST PUSH1 0x0 DUP2  
MLOAD SWAP1 POP PUSH2 0x4E DUP2  
PUSH2 0x87 JUMP JUMPDEST SWAP3  
SWAP2 POP POP JUMP JUMPDEST PUSH1  
0x0 PUSH1 0x20 DUP3 DUP5 SUB SLT  
ISZERO ....
```

OpCode

```
608060405234801561001057600080fd5  
b50604051610188380380610188833981  
810160405281019061003291906100545  
65b806000819055505061009e565b6000  
8151905061004e81610087565b9291505  
0565b6000602082840312156100665760  
0080fd5b60006100748482850161003f56  
5b91505092915050565b6000819050919  
050565b6100908161007d565b81146100  
9b57600080fd5b50565b60dc806100ac6  
000396000f3fe6080604052348015600f5  
7600080fd5b5060043610602857600035  
60e01c806360fe47b114602d575b60008  
0fd5b60436004803603810190603f9190  
6062565b6045565b005b8060008190....
```

ByteCode

Vulnerabilities in Ethereum Smart Contracts

Denial of Service

“tx.origin” usage

Integer Overflow/Underflow

Re-entrancy

Call to the unknown

Gasless “send”

...



Solidity Programming Language

Immutable bugs/mistakes

Ether lost in transfer

...



Ethereum Virtual Machine

Timestamp dependency

Transaction Ordering Dependency

...



Ethereum Blockchain Design

Vulnerabilities in Ethereum Smart Contracts

Denial of Service

“tx.origin” usage

Integer Overflow/Underflow

Re-entrancy

Call to the unknown

Gasless “send”

...



Solidity Programming Language

Immutable bugs/mistakes

Ether lost in transfer

...



Ethereum Virtual Machine

Timestamp dependency

Transaction Ordering Dependency

...



Ethereum Blockchain Design

⇒ More than 20 types of different vulnerabilities, we focus only on detection

Vulnerability Detectors: Formal Verification Techniques

Most of existing frameworks exploit static analysis, some examples are:

- Slither
- Mythril
- Oyente
- Securify
- SmartCheck
- SmartScan
- ...

Usually specialized on specific security properties,
running multiple frameworks is computationally expensive

Static analyzers can be divided into exact methods ('hardcore')
and approximated techniques

Vulnerability Detectors: ML Techniques

ML learning techniques take advantage of existing frameworks for dataset labeling, and then they simply perform classification with techniques such as:

- SVM
- **Boosting**
- Random Forest
- Decision Tree
- CNN
- GNN
- **LSTM**
- ...

Machine Learning techniques claim to have higher performance than static analyzers with respect to inference time and accuracy

Limitations and Open Problems

FORMAL
VERIFICATION

Lack of soundness
guarantees

Restricted number of
vulnerabilities

Different datasets:

Number/type of vulnerabilities

Number of contracts

Source code/Opcode/Bytecode

MACHINE
LEARNING

Scalability

Easily deprecated

Based on possible
mislabelling

Hard to compare detectors

Future Work: Dataset Creation

The availability of **benchmark dataset** would not only provide a solid base to develop new detection algorithms, but it would also allow us to evaluate and compare existing ones.

Key features:

- Large number of smart contracts
- Include all three representations of a contract (source code, opcode, bytecode)
- Metadata

Labeling process:

- Exploit only formal verification techniques providing soundness guarantees
- Represent as many as possible vulnerabilities (eventually augmenting the cardinality of heavily underrepresented classes)

Thank You!

dalila.ressi@uniud.it