University of Antwerp

Image painting using Genetic Algorithms and Classifiers

Arno, Giorgi, Ilias, Mohammed & Viktor

Table of contents

- Introduction
- Initial goal
- Project
  - Genetic Algorithm
  - CNN Classifier
  - First Integration
  - Siamese Networks
  - Second Integration
- Conclusion

Introduction

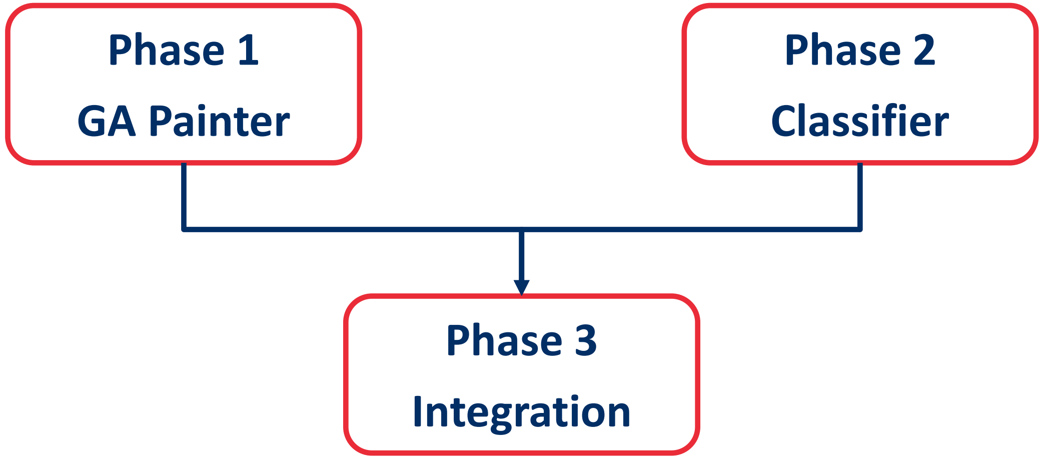
Introduction

- **Genetic algorithms project**
- **Professor Oramas and Ms. Haidar**
- **Weekly meeting**
- **Milestones**

Initial goal

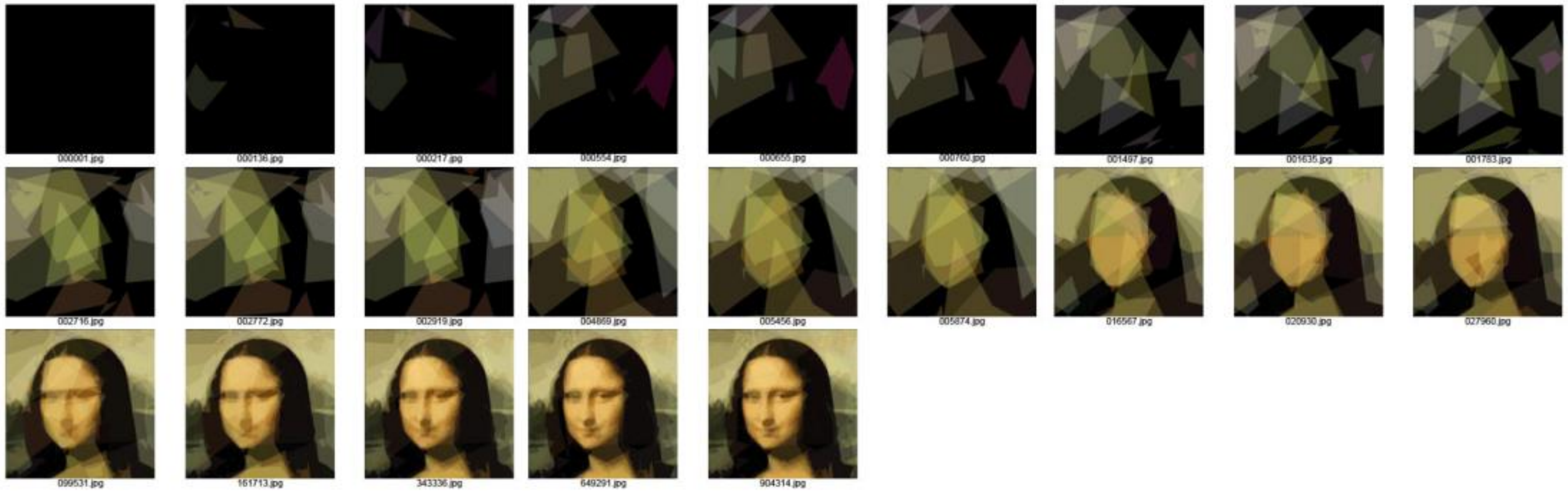
The plan

- Divide and conquer
- Three phase plan
- First two phases simultaneously



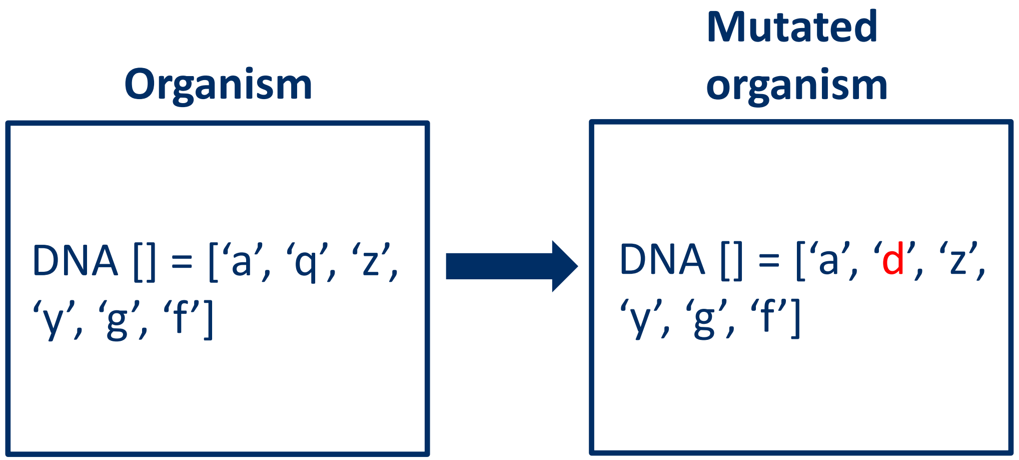
Phase 1 - The painter

- **Approximating a reference image**



Phase 1 - the painter

- **Approximating a reference image**



Organism

DNA [] = [‘a’, ‘q’, ‘z’, ‘y’, ‘g’, ‘f’]

Mutated organism

DNA [] = [‘a’, ‘d’, ‘z’, ‘y’, ‘g’, ‘f’]

Phase 2 – The classifier

- Fitness function for genetic algorithm
- Training to recognize
- Evolve

INPUT

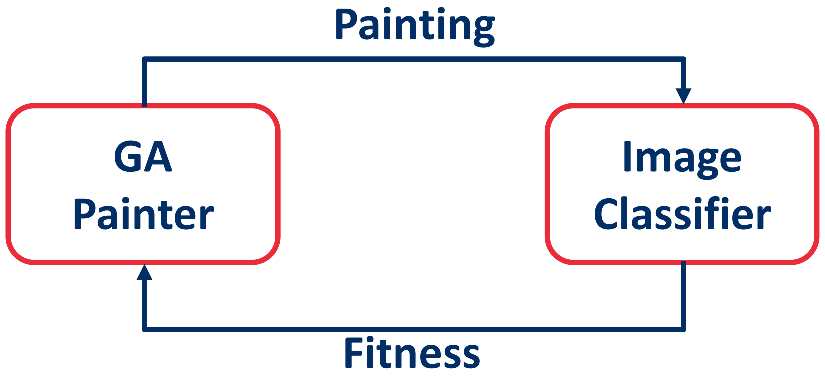
. . .

Predicted: American Lizard
Confidence: 95%

OUTPUT

Phase 3 – Integrating the parts

- **Combine classifier and genetic algorithm painter**



Project progress

Genetic algorithm

Genetic algorithm - practical considerations

- **Giorgi, Ilias & Viktor**
- **Constraints:**
  - Greyscale
  - Low resolution images
  - Simple dataset
- **Orientation**

Genetic algorithm - Pipeline

Organism

DNA [] = list of genes

Example

DNA [] =
[‘a’, ‘q’, ‘z’, ‘y’, ‘g’, ‘f’]

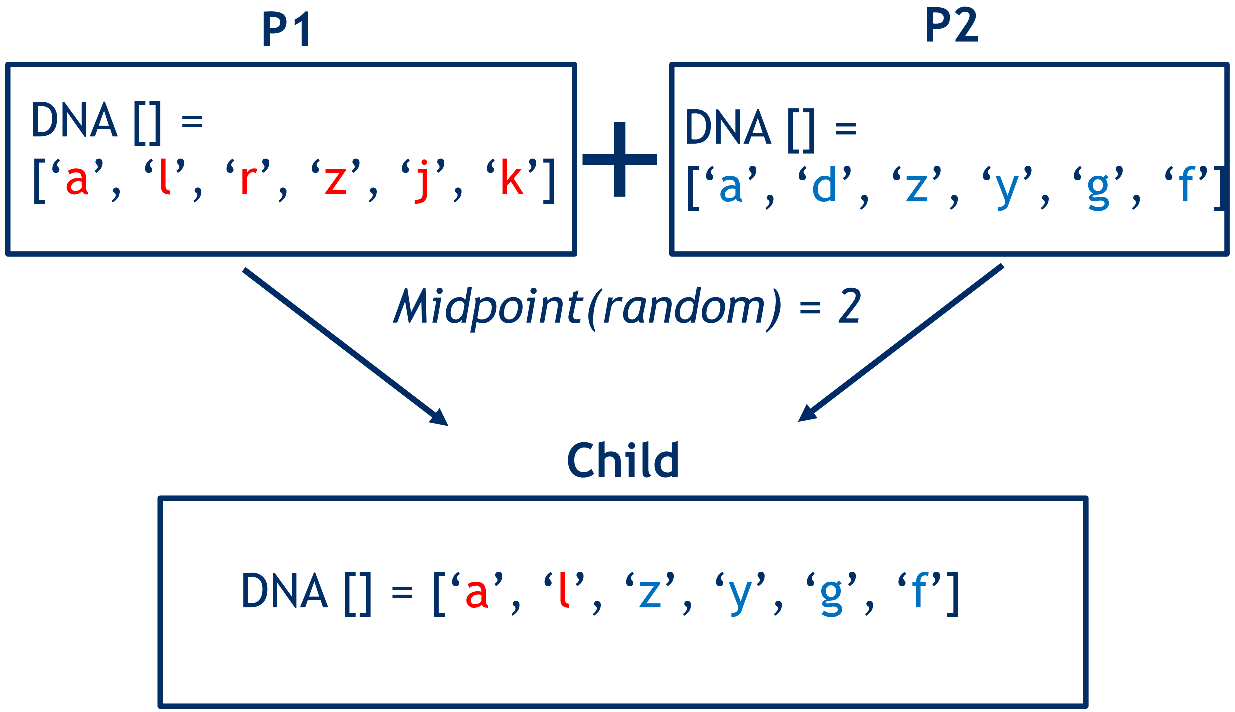
Mutation phase

Organism

DNA [] =
[‘a’, ‘d’, ‘z’, ‘y’, ‘g’, ‘f’]

*Each gene has a {**mutationRate**} % chance to mutate*

Crossover phase



P1

DNA [] =
[‘a’, ‘l’, ‘r’, ‘z’, ‘j’, ‘k’]

+

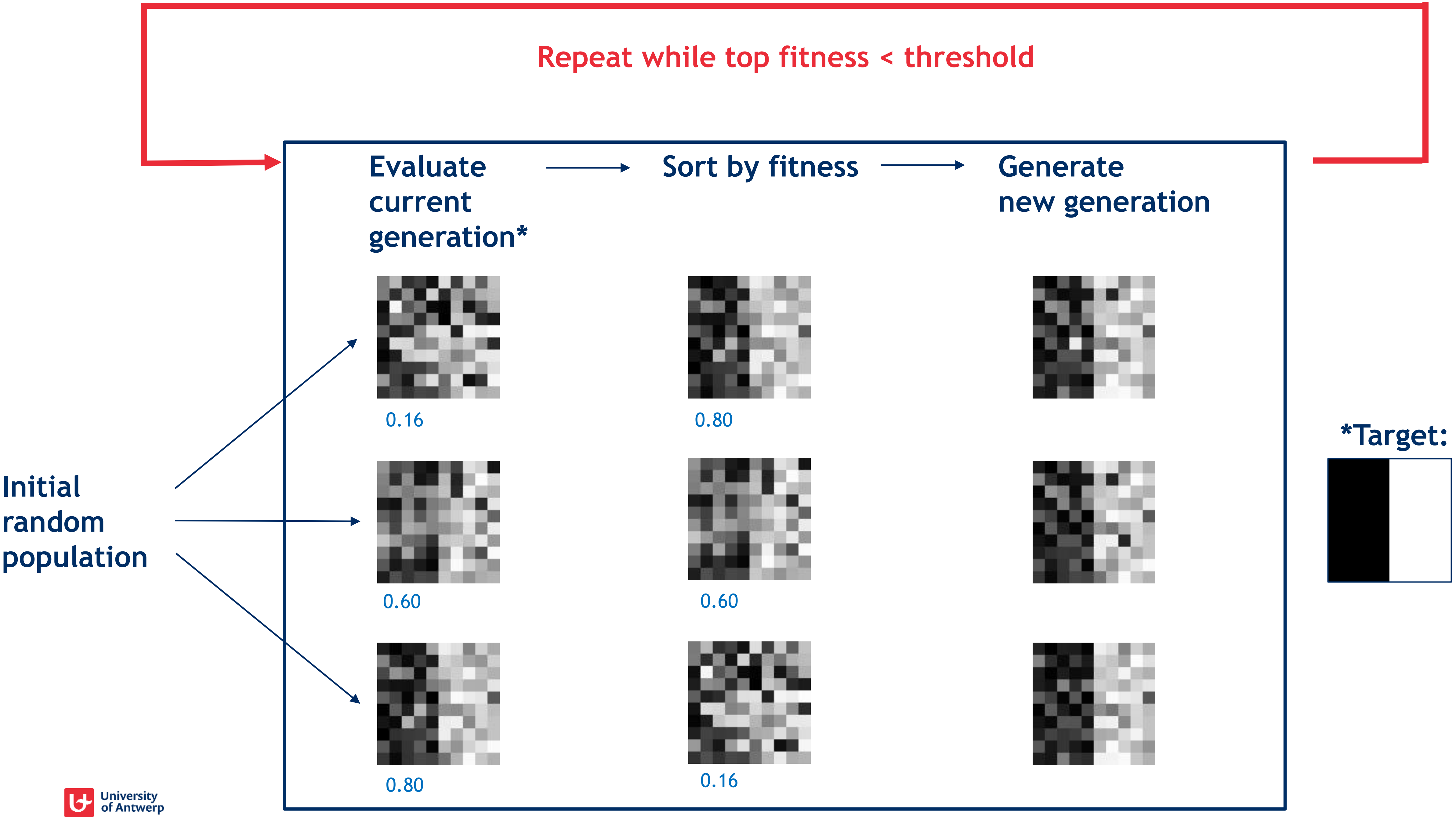
P2

DNA [] =
[‘a’, ‘d’, ‘z’, ‘y’, ‘g’, ‘f’]

Midpoint(random) = 2

Child

DNA [] = [‘a’, ‘l’, ‘z’, ‘y’, ‘g’, ‘f’]



Repeat while top fitness < threshold

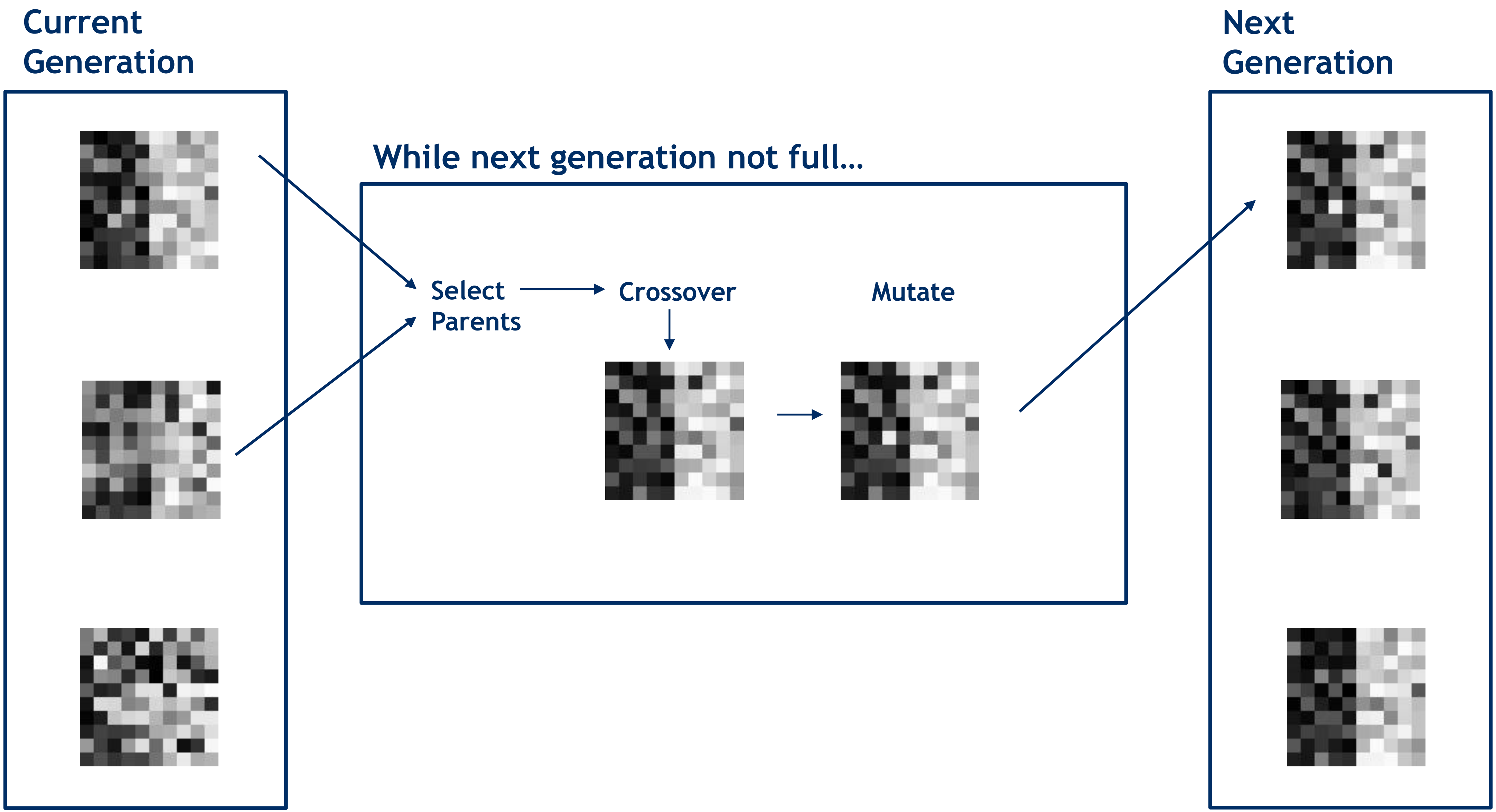
Initial random population

Evaluate current generation*

Sort by fitness

Generate new generation

*Target:



Current Generation

While next generation not full...

Select Parents

Crossover

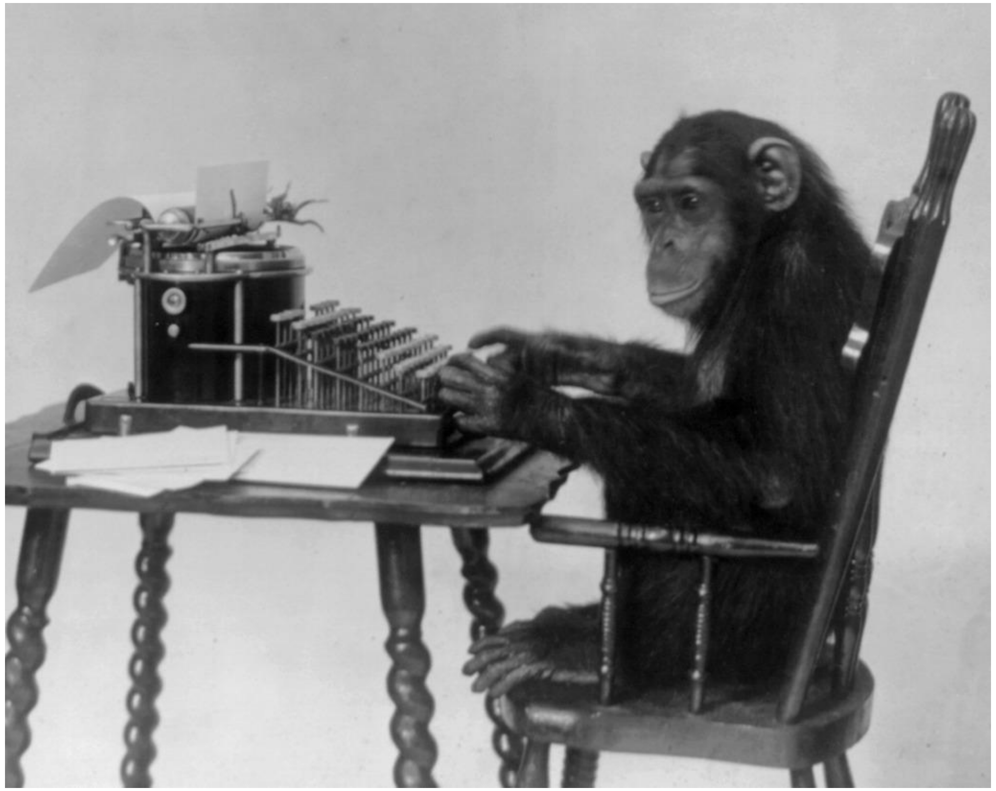
Mutate

Next Generation

Genetic algorithm – basic implementation

- **Implemented with strings**
- **Infinite Monkey Theorem**

```
DNA [] =
[‘t’, ‘o’, ‘b’, ‘e’, ‘o’, ‘r’, ‘n’, ‘o’, ‘t’, ‘t’, ‘o’, ‘b’, ‘e’]
```



Genetic algorithm – basic implementation

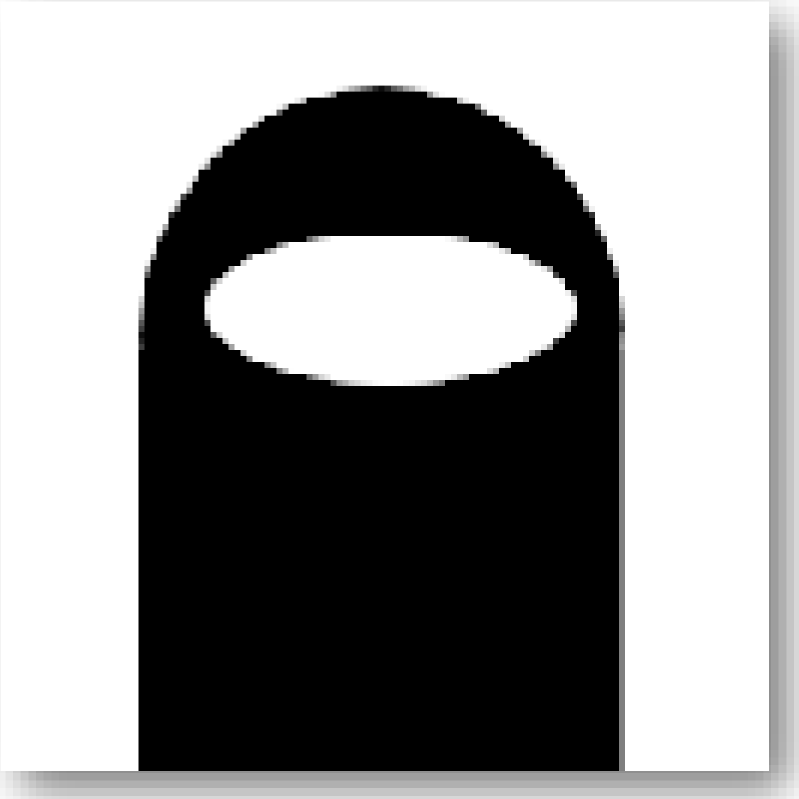
- **Ref Target:** to be or not to be

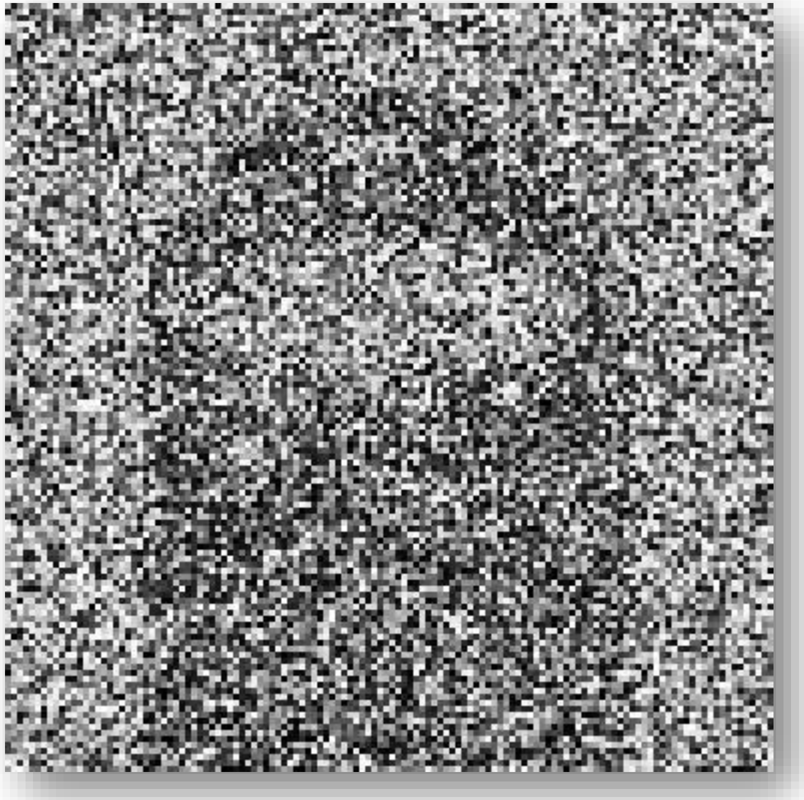
GENERATION	AVERAGE FITNESS	BEST FITNESS	BEST ORGANISM
0	0.033889	0.222222	`trybpuolsjlgjwpcue`
1	0.081667	0.222222	`trybpuolsjlevfgree`
2	0.130556	0.277778	`trybpuo  dzvsnc tn`
3	0.176111	0.333333	`trybe eoxmmy   cue`
4	0.222222	0.388889	`trybpuolsjsr ty je`
5	0.270278	0.444444	`tr drlmn ndr ty bf`
6	0.320556	0.5	`trybe obngotxwe te`
7	0.366667	0.555556	`trybe ovwnolaty te`
8	0.416111	0.555556	`trybe ovwnolaoo je`
9	0.455556	0.666667	`trybe objzot ty be`
10	0.503889	0.666667	`trybe objzot ty be`
11	0.55	0.722222	`tocbe ohynotety be`
12	0.592778	0.722222	`tocbe objzot ty be`
13	0.632222	0.722222	`trybe objzot to be`
14	0.663611	0.777778	`toybe ovwnot ty be`
15	0.690833	0.833333	`toybe oh vot to be`
16	0.726667	0.833333	`toybe ab not to be`
17	0.754722	0.888889	`torbe oh not to be`
18	0.792222	0.888889	`tocbe ob not to be`
19	0.822222	0.944444	`toybe or not to be`
20	0.853611	0.944444	`tocbe or not to be`
21	0.880556	1	`to be or not to be`

University of Antwerp

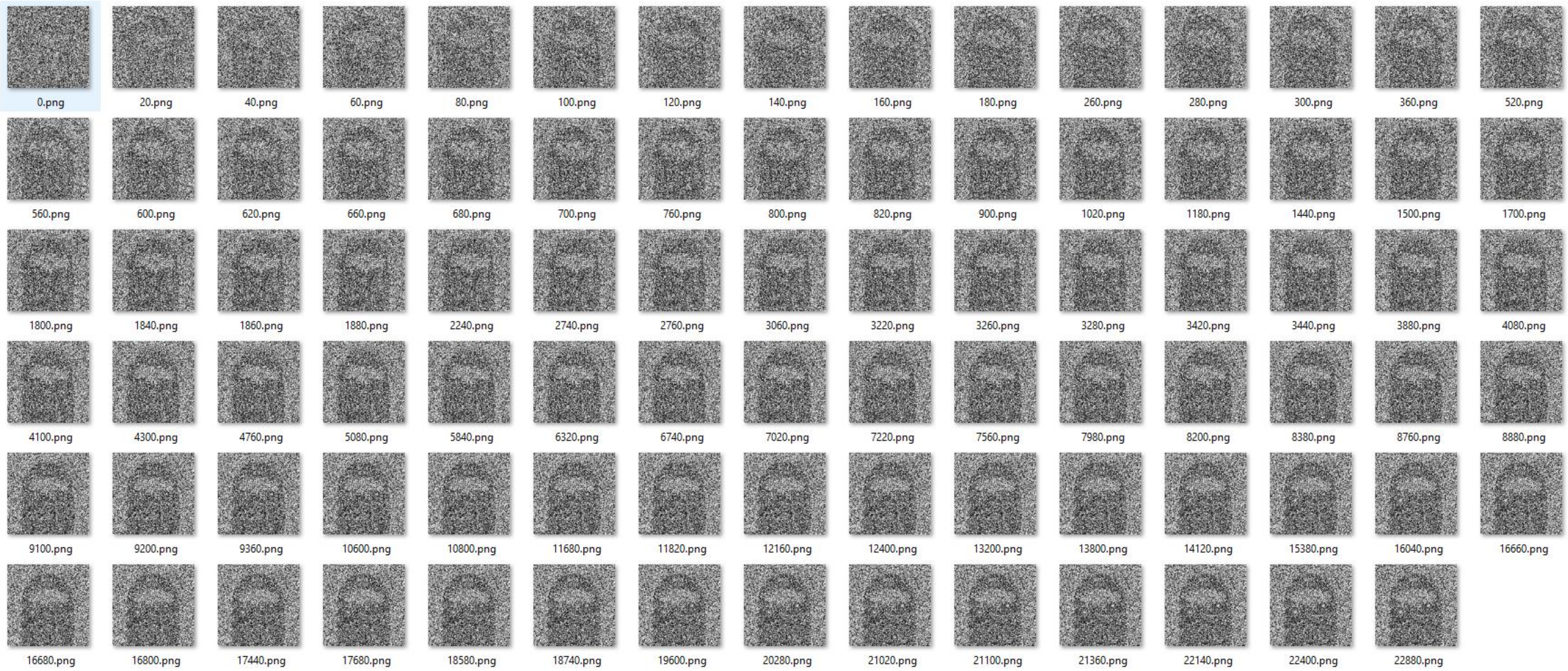
Genetic algorithm – basic implementation

- **First pixel test**
- **It’s getting there, but suffers *Premature Convergence***



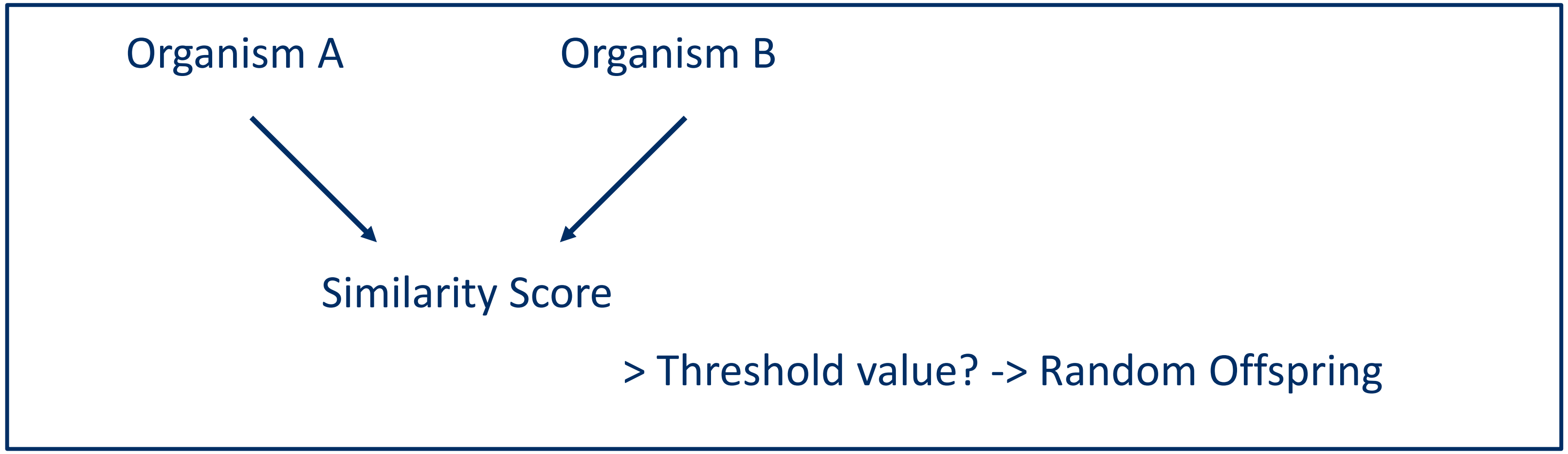


Genetic algorithm – basic implementation



Genetic algorithm – basic implementation

- **Preventing Premature Convergence:**
    - Adding special algorithms to counteract the problem
        - Random Offspring Generation (ROG)



Organism A

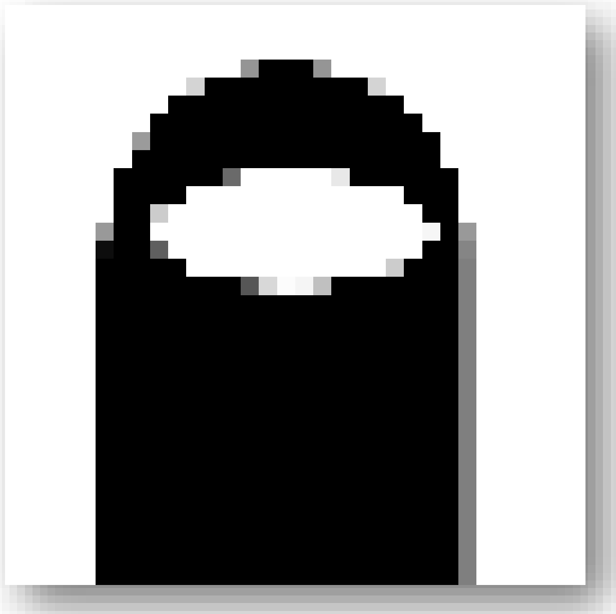
Organism B

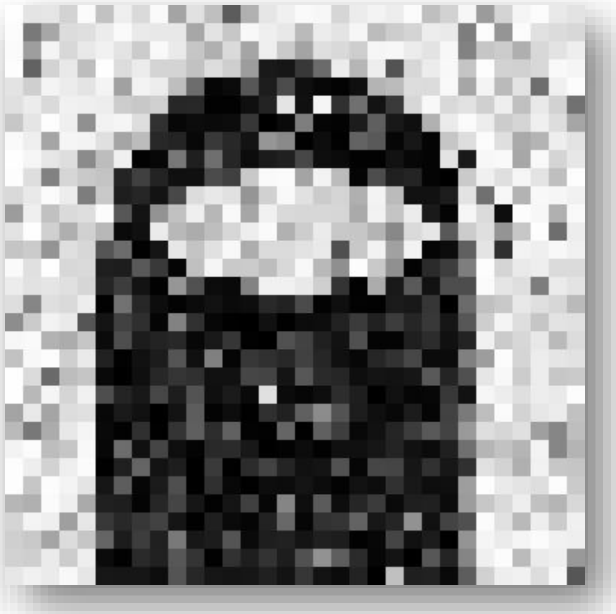
Similarity Score

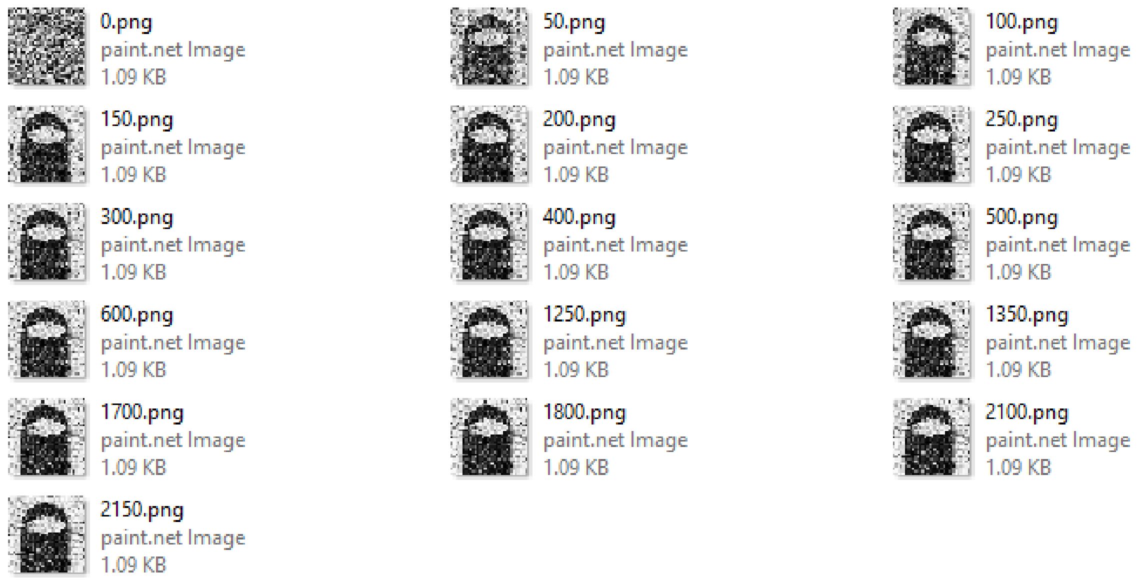
> Threshold value? -> Random Offspring

- Increasing population size / decreasing search space

Genetic algorithm – decreasing search space







0.png
paint.net Image
1.09 KB

50.png
paint.net Image
1.09 KB

100.png
paint.net Image
1.09 KB

150.png
paint.net Image
1.09 KB

200.png
paint.net Image
1.09 KB

250.png
paint.net Image
1.09 KB

300.png
paint.net Image
1.09 KB

400.png
paint.net Image
1.09 KB

500.png
paint.net Image
1.09 KB

600.png
paint.net Image
1.09 KB

1250.png
paint.net Image
1.09 KB

1350.png
paint.net Image
1.09 KB

1700.png
paint.net Image
1.09 KB

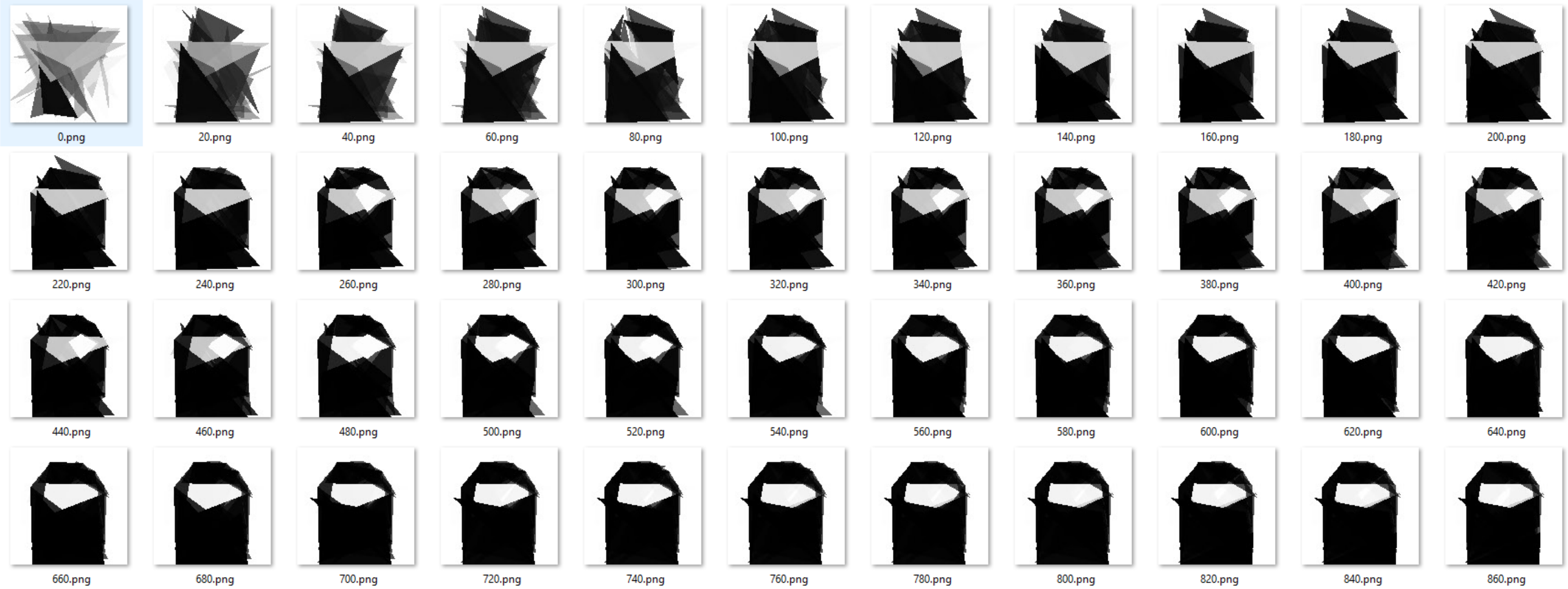
1800.png
paint.net Image
1.09 KB

2100.png
paint.net Image
1.09 KB

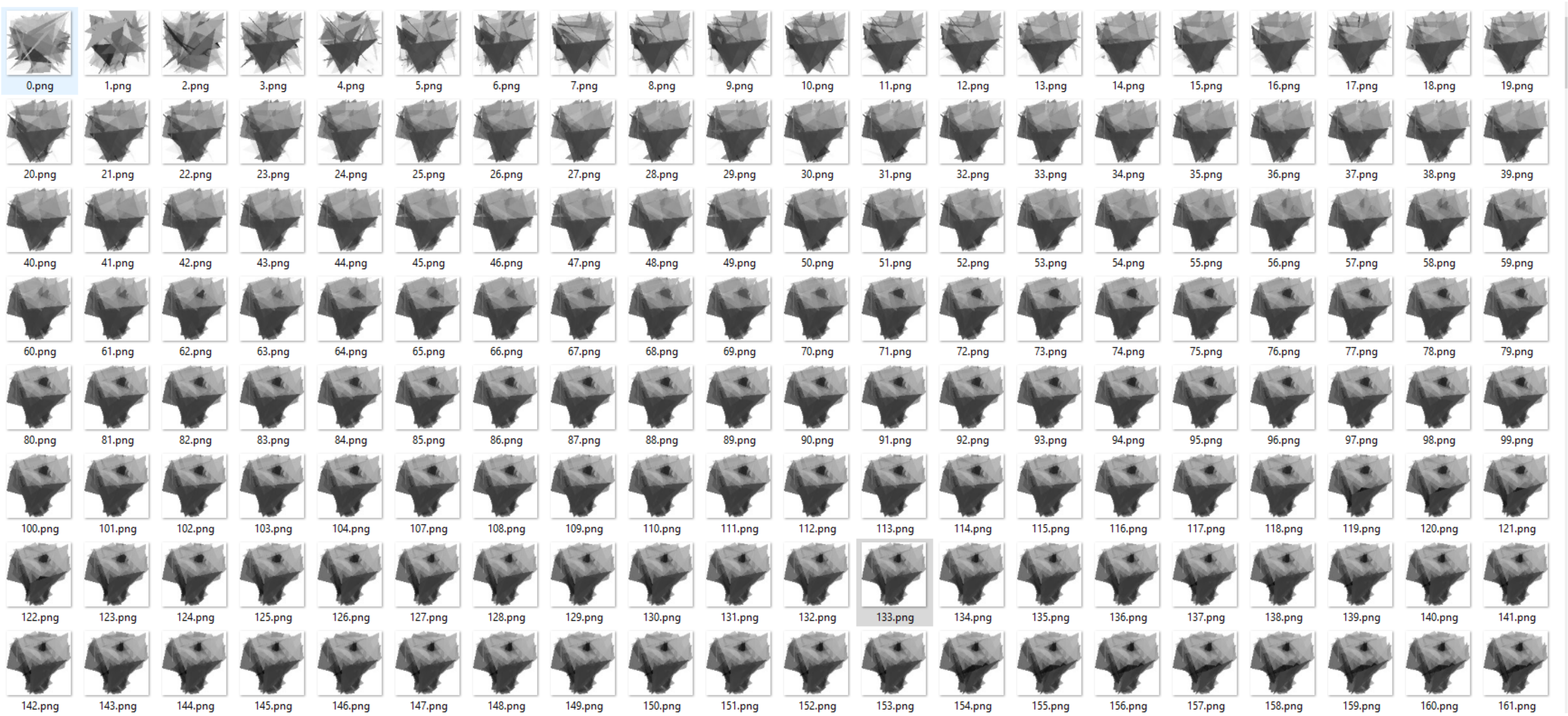
2150.png
paint.net Image
1.09 KB

Genetic algorithm – decreasing search space

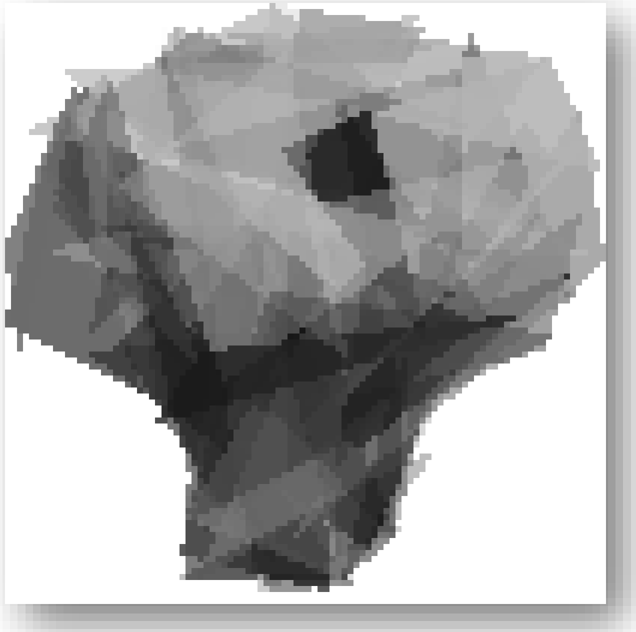
- **Triangles:**



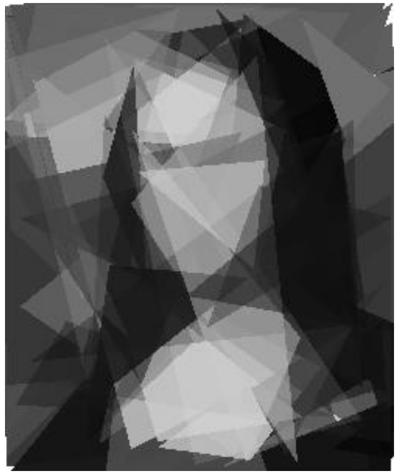
Genetic algorithm – decreasing search space

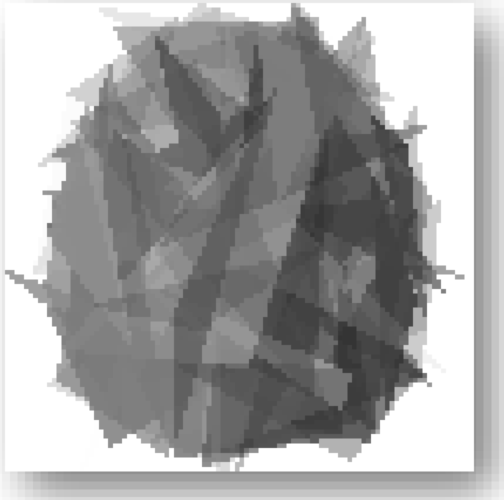


Genetic algorithm – decreasing search space



Genetic algorithm – decreasing search space





Project progress

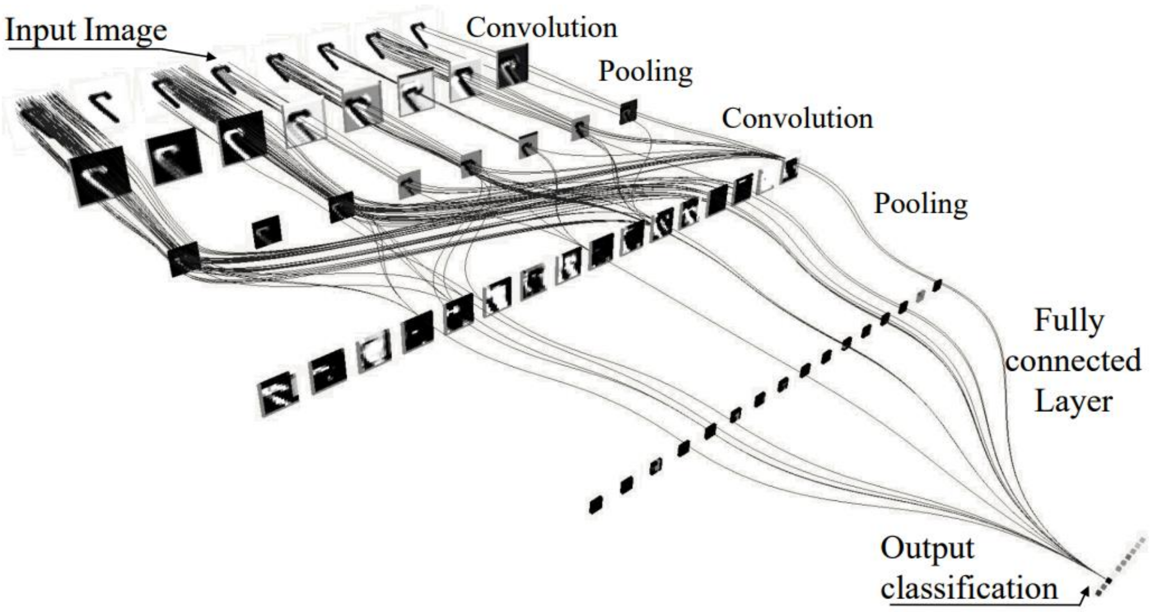
CNN Classifier

Classifier

- Arno & Mohammed
- No experience ⇒ research needed

Classifier - Research

- **Learn the basics of CNN's**
  - Mentors provided resources
- **Get acquainted with PyTorch**

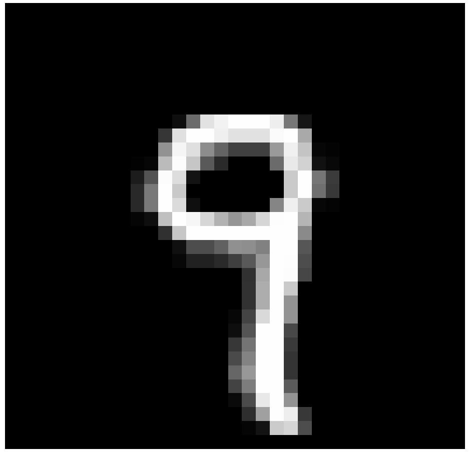


Classifier - Start

- **Experiment with existing models**
  - Trying EfficientNet – Is a more complex model feasible?
- **Come up with own model**
- **Deciding on the dataset**
  - CIFAR-10, CIFAR-100, MNIST & Fashion MNIST

Classifier – First implementation

- **Decided on MNIST digit dataset**
  - Easy training ⇒ faster development & testing
- **Good accuracy (± 99% TOP-1)**



INPUT

. . .

Predicted: 9
Confidence: xx%

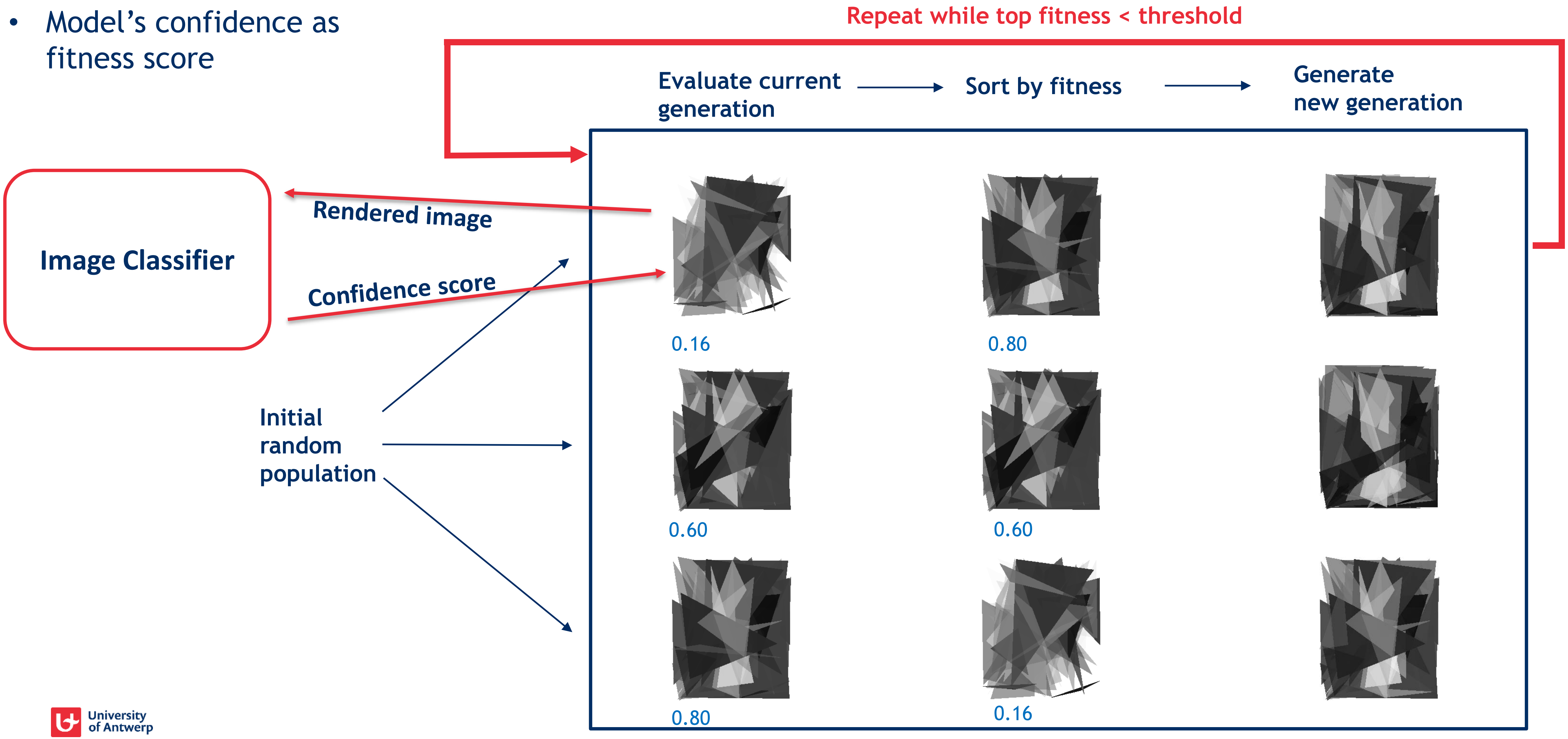
OUTPUT

Project progress

First Integration

First Integration – The straightforward way

- Model’s confidence as fitness score



Repeat while top fitness < threshold

Evaluate current generation → Sort by fitness → Generate new generation

Image Classifier

Rendered image

Confidence score

Initial random population

0.16

0.80

0.60

0.60

0.80

0.16

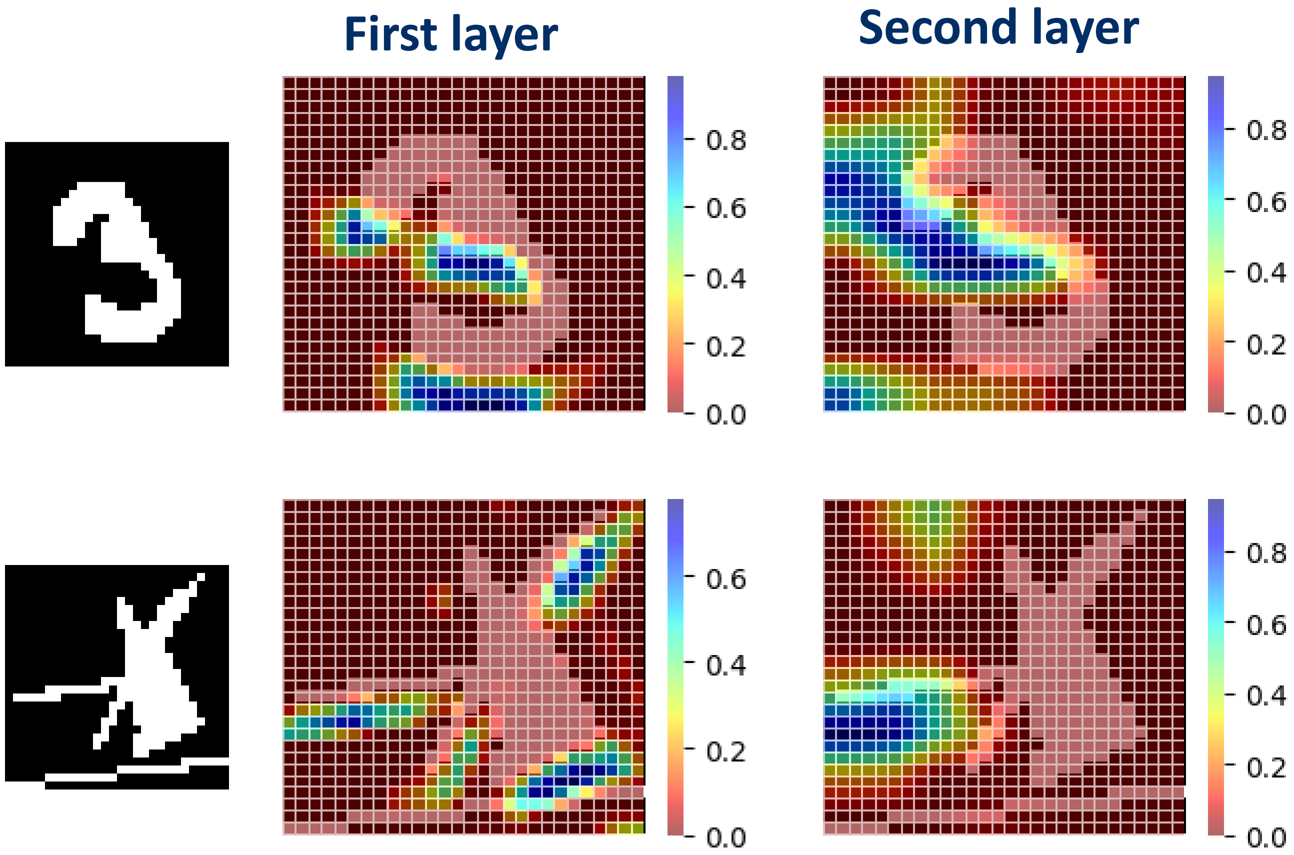
First Integration – Problems

- **Model learns difference between classes**
  - What makes a “3” a “3” instead of an “8”?
  - Obvious in hindsight, but an important intuition
- **Universe composed of the training data**
  - Everything is decided based on the data

First Integration – Mitigations

- **Add an extra class to the model**
  - Use representative data

  ⇒ Still overconfident
- **Try to find an explanation**
  - E.g., Using GradCAM

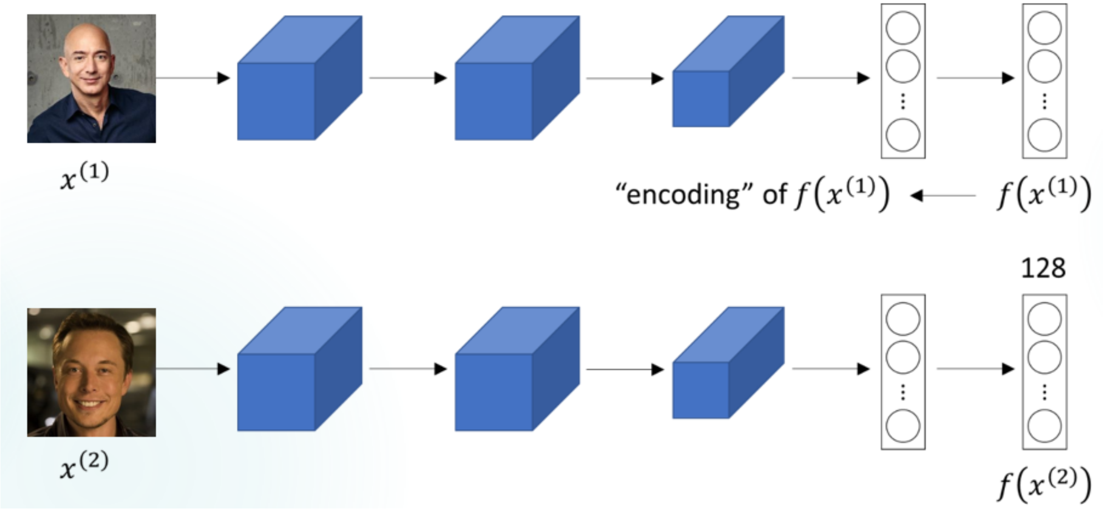


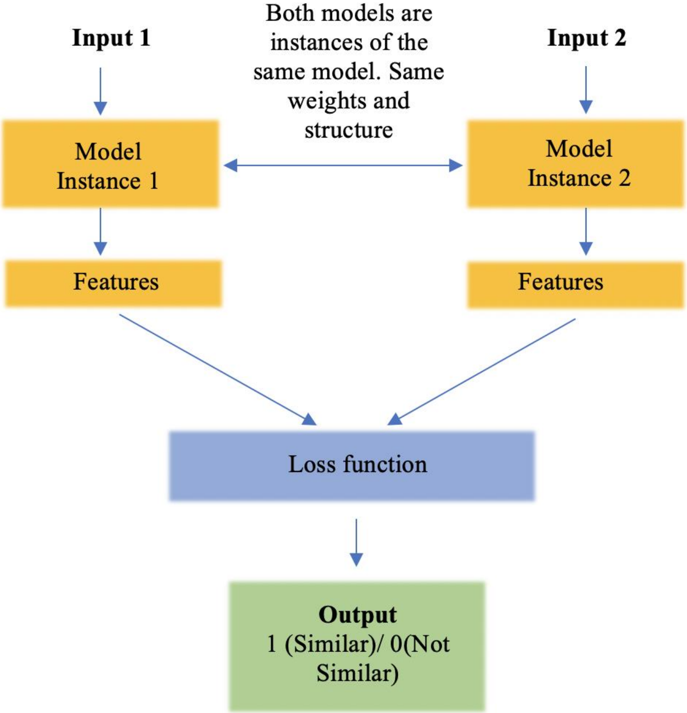
Project progress

Siamese Networks

Siamese Networks

- Predict similarity input – output
- Two inputs, painter & reference





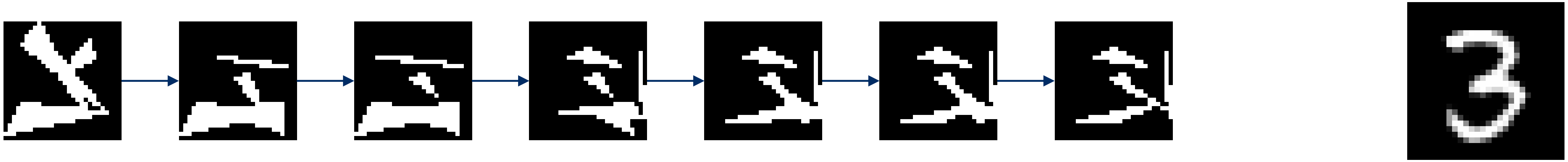
Input 1
Both models are instances of the same model. Same weights and structure
Input 2
Model Instance 1
Model Instance 2
Features
Features
Loss function
Output
1 (Similar)/ 0(Not Similar)

Project progress

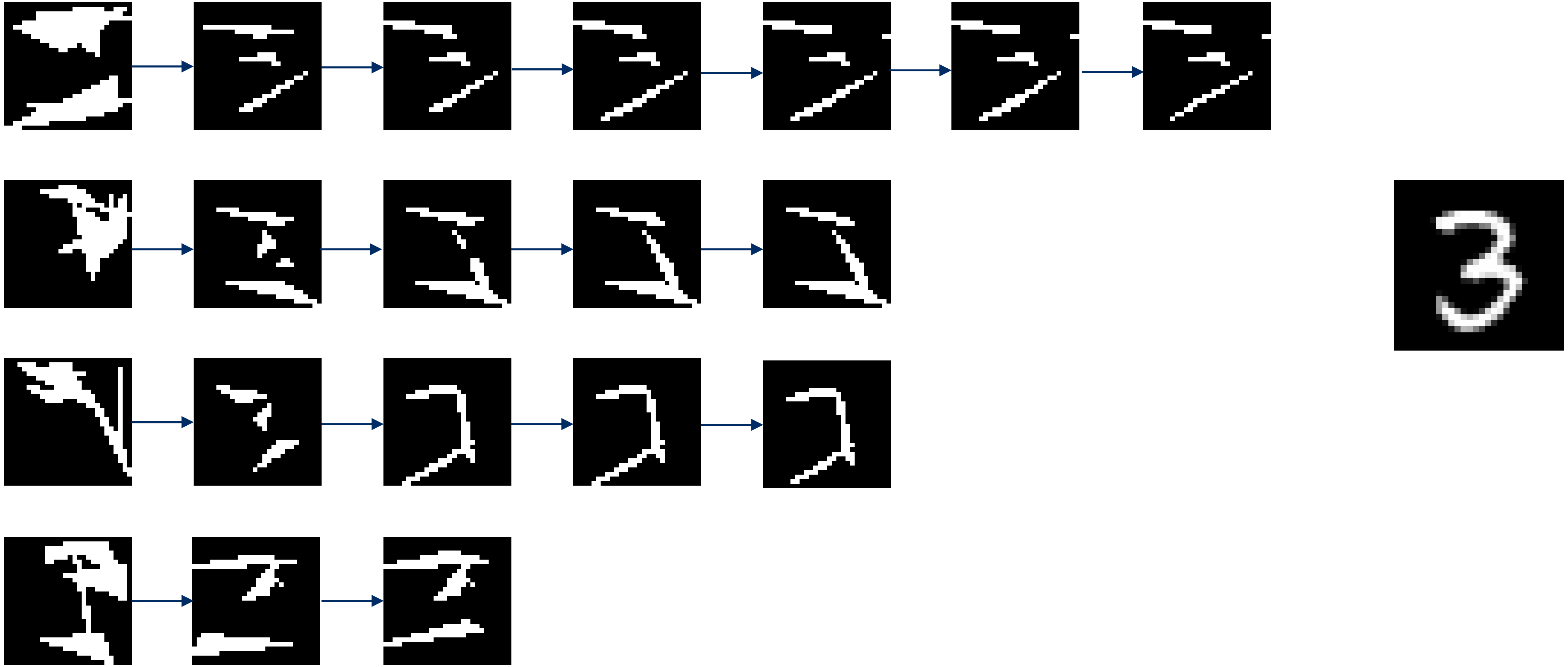
Second Integration

Second Integration – The Siamese network way

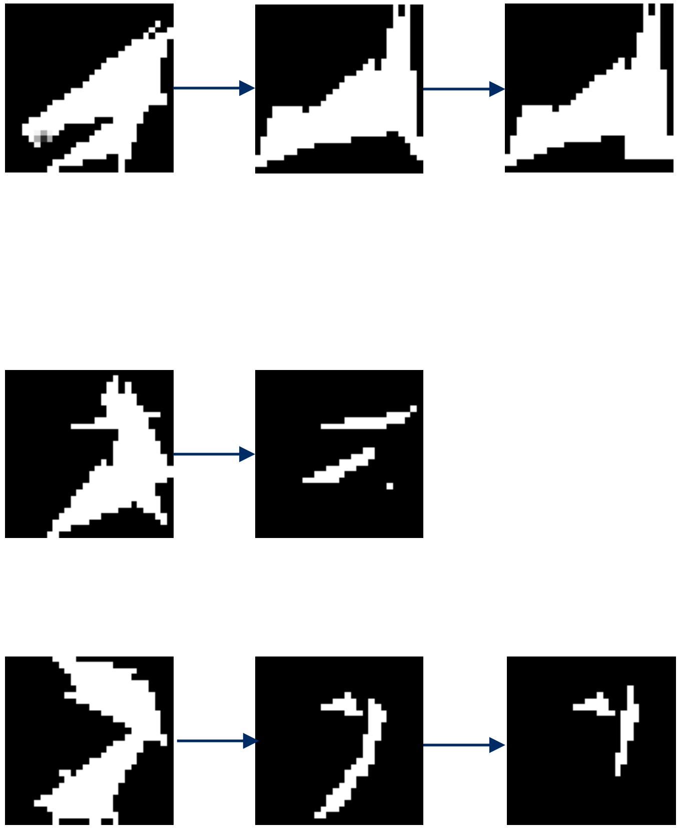
- **Use similarity as fitness score**
- **Intuition: Should act more like a comparator network**
  - Compares features not pixels



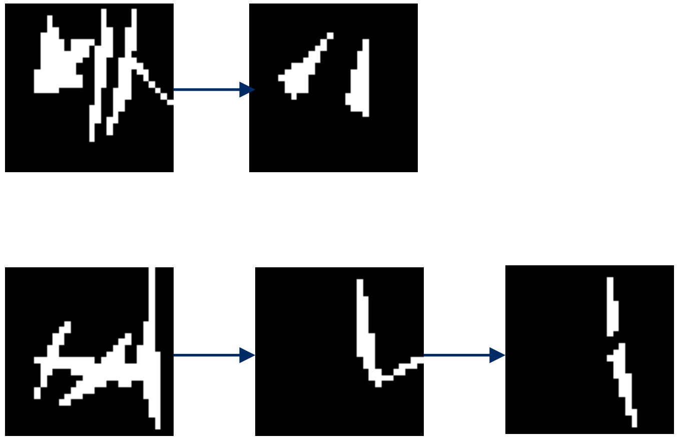
Second Integration – The Siamese network way

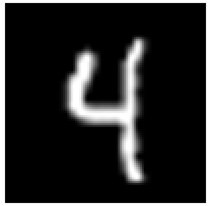


Second Integration – The Siamese network way



Second Integration – The Siamese network way





Second Integration – Limitations

- **Siamese networks add computation overhead**
- **Hard to find representative data**
- **Performance at scale unknown**
  - Could not test it

Second Integration – Further experiments

- **Siamese network with multiple reference images**
  - Average out the features and calculate loss
  - More computation overhead
- **MNIST Fashion dataset**
- **Binary Siamese network**
  - Hypothesis: Works better and easier to optimize
    - Easier training
    - Needs further research

Conclusion

Conclusion – What we learnt

- **We learnt a lot, especially**
  - Classifier: derivative image generation
  - Possible Siamese network visualization method?

Conclusion – Feedback

- Open for further research
- Good teamwork
- Great mentors
- Satisfying results

Thank you

The end