June 28th, 2024

Estimating Path Lengths in Traveling Salesman Images

A Computer Vision project by **The Unsupervised Learners**

Tobias Frey
Maxime Drouhin
Randolf Nkimo

Table of contents

01

Problem Statement

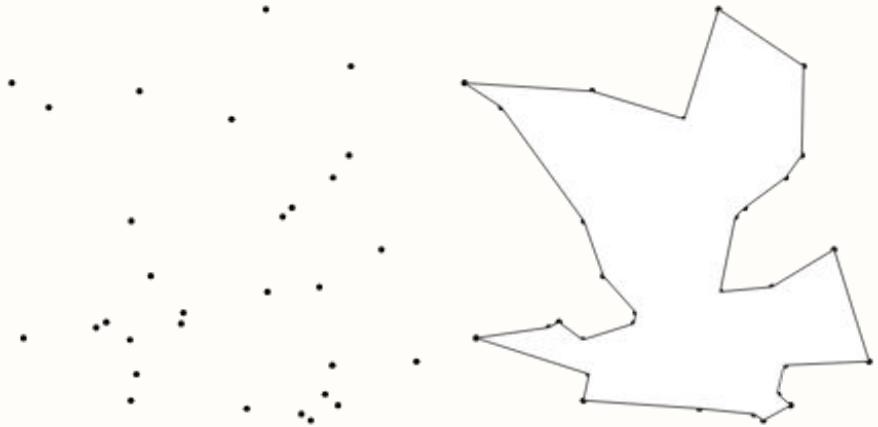
Understanding the challenge and the requirements

What is the Traveling Salesman Problem?

Goal: Find the shortest route that visits each city once and returns to the start (or not, depending on the variant).

Significance: Crucial for optimizing logistics and route planning.

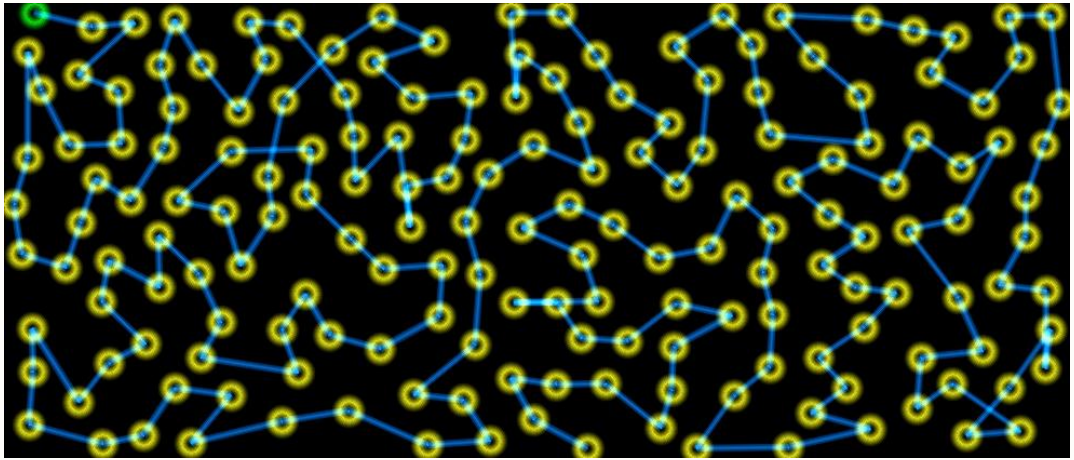
Challenge: NP-hard problem with no efficient solution for large datasets.



The objective of this project

The goal is **not** to write a TSP algorithm.

Actual goal: **estimating the total path length** from images.



Data provided

Images of maps with yellow circles representing cities, linked with blue edges

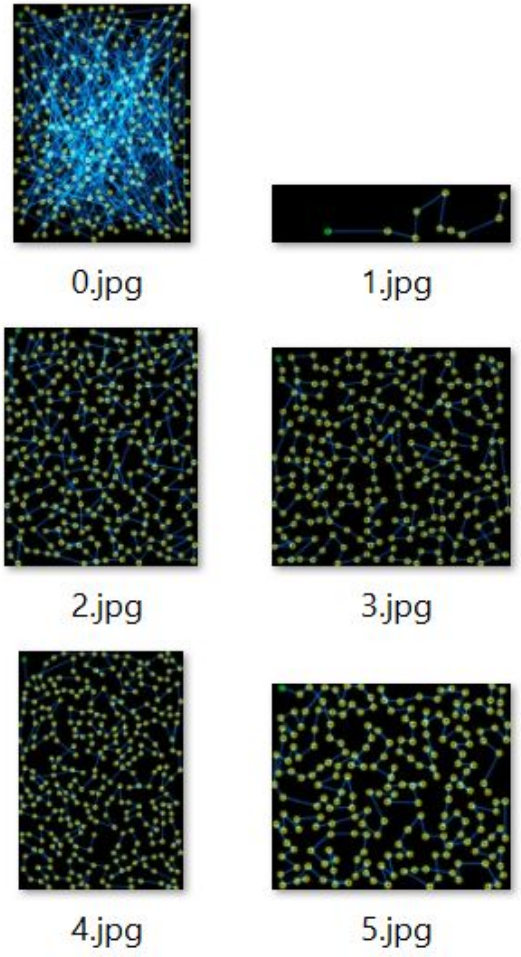
16 018 images labeled with path length

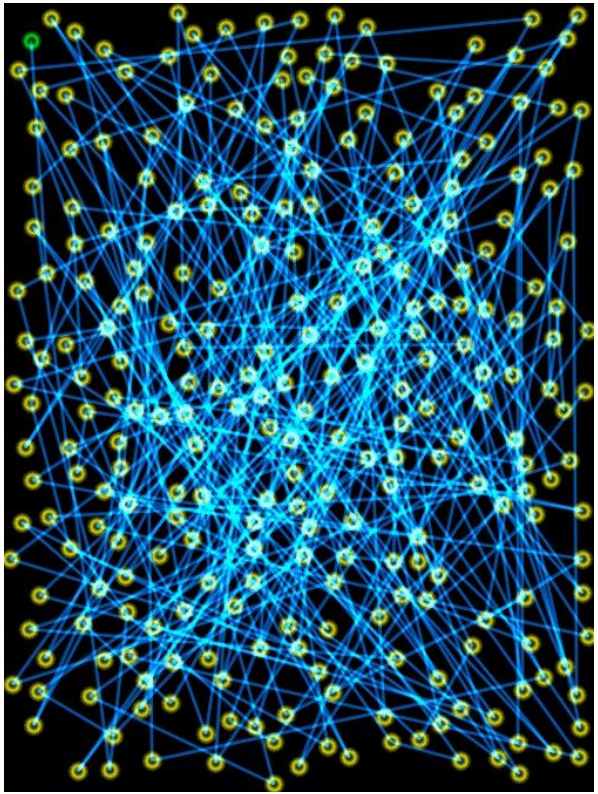
4 005 unlabeled images

Source: Traveling Salesman Computer Vision dataset

Variability:

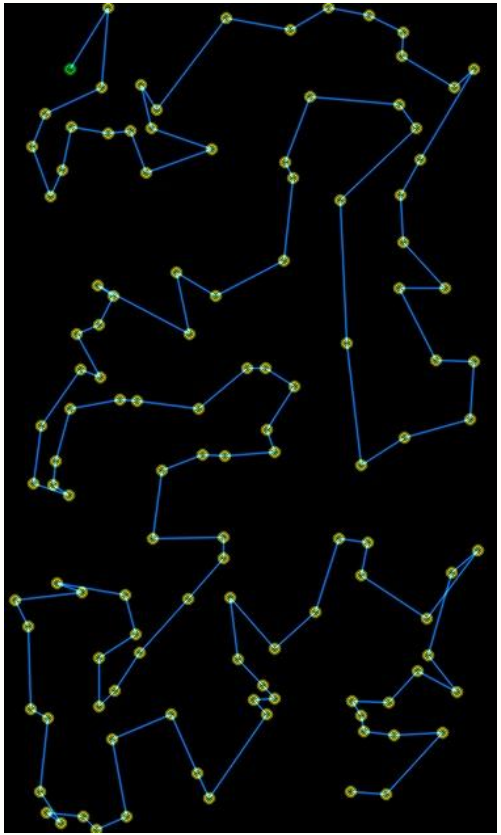
- Variable resolution
- Variable image dimensions
- Edges might cross or not depending on the image





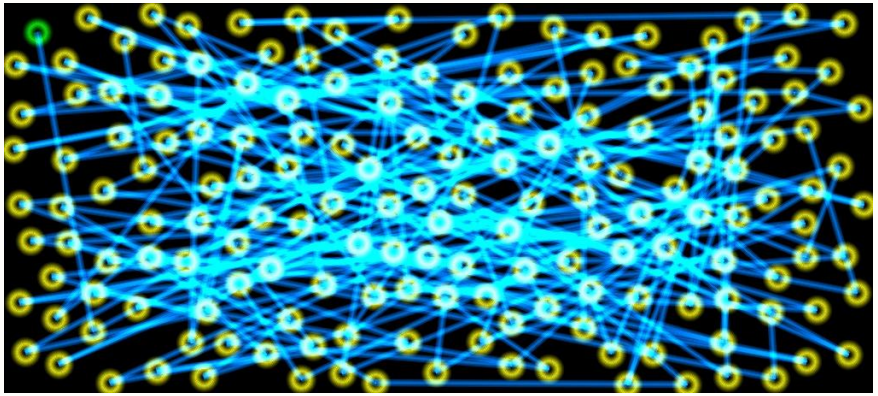
Distance

83 110



Distance

4 526



Distance

68 735

02

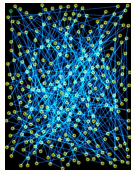
Machine Learning Approach

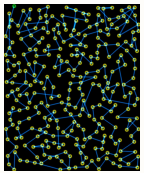
Exploring the method and its results

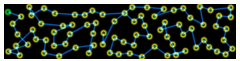
Relation between feature and target
Performance of ML Models

The idea

Images





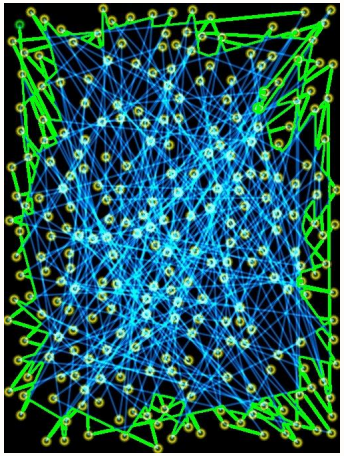


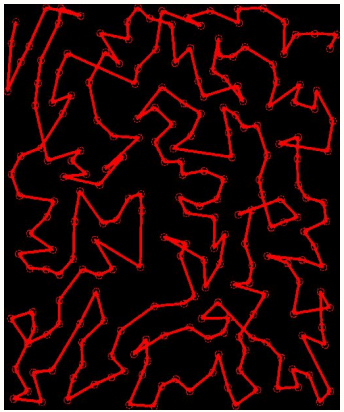
Model

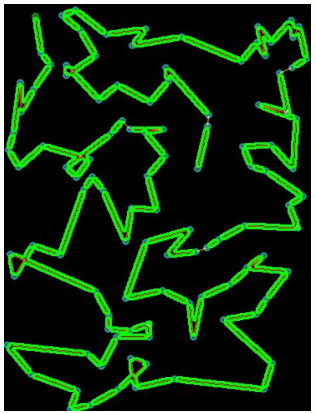
Extracted features					Path length estimation
width	height	res.	nb of blue pixels	...	
302	265	50	12 354		83 110
470	124	72	170		1 035
323	262	53	2 692		20 756
483	126	81	644		4 110

Methodology

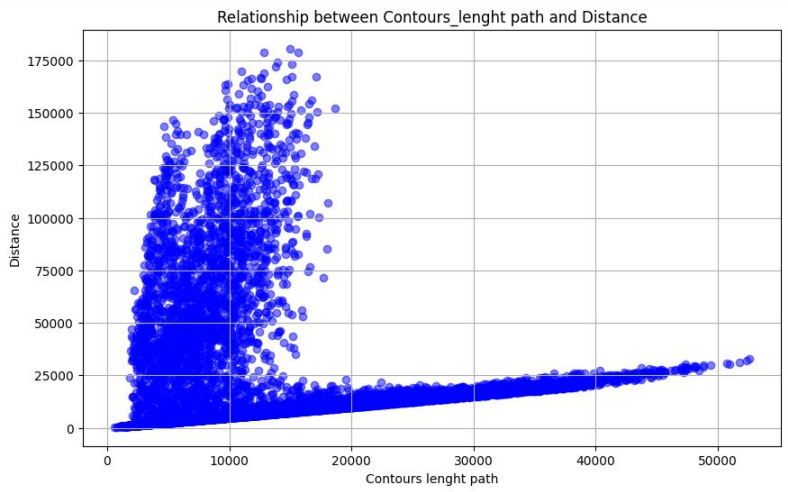
- **Data Preprocessing**
  - **Gray scale conversion**: Convert images to grayscale for less computation
  - **Contour detection**: identify contours in the image
- **Features extraction**
  - **Shape descriptor**
    - Calculate the total length of contours
    - Count **blue pixels**/width of a line

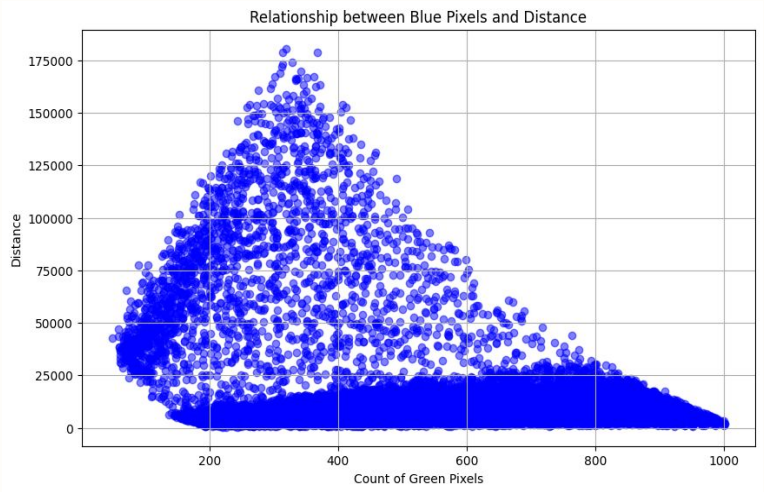






Relation between features and target



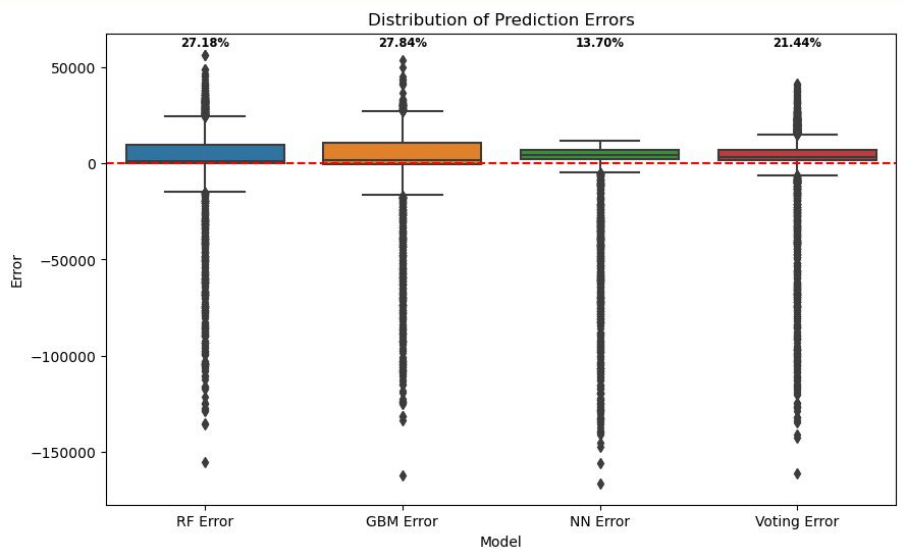


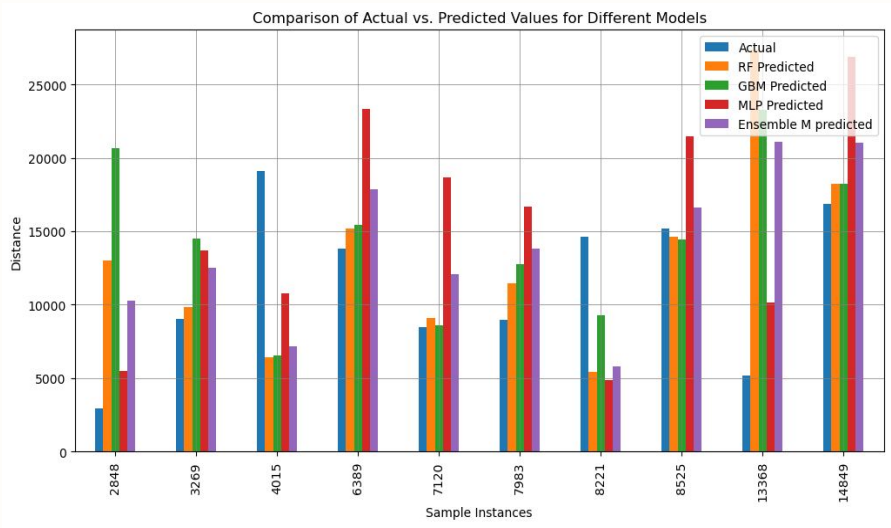
Comparative Performance of Machine Learning Models

Models	RMSE	R2
Random Forest*	22872.9	0.155
Gradient Boosting	22997.1	0.145
MLP	26600.6	-0.142
Ensemble	23349.11	0.11

*With the best parameters determined by grid search

Error distribution



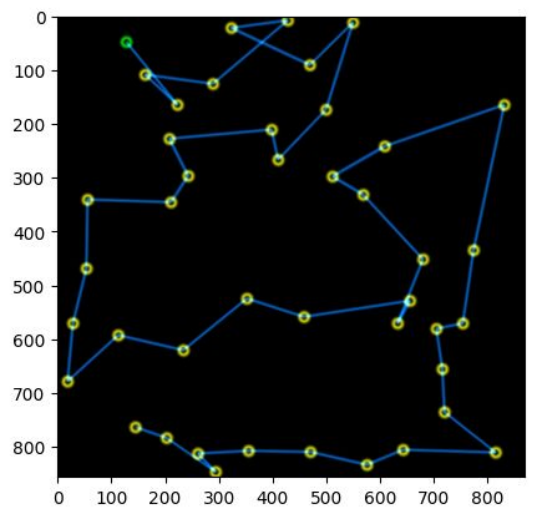


03

Algorithmic Approach

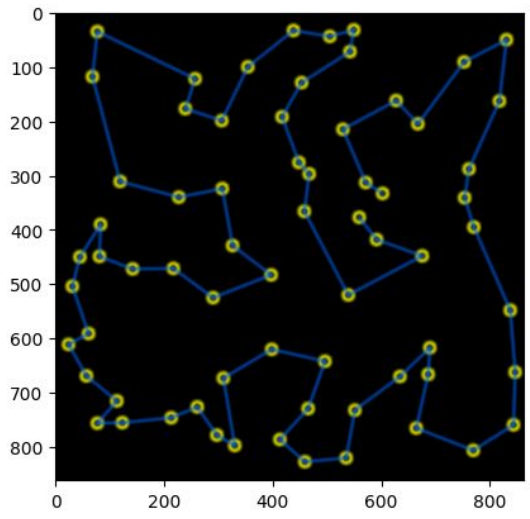
Exploring the method and its results

Data generation



original dataset

Length of the total path **only**



generated data

Length of the total path
Positions of the yellow dots
Connections between the dots

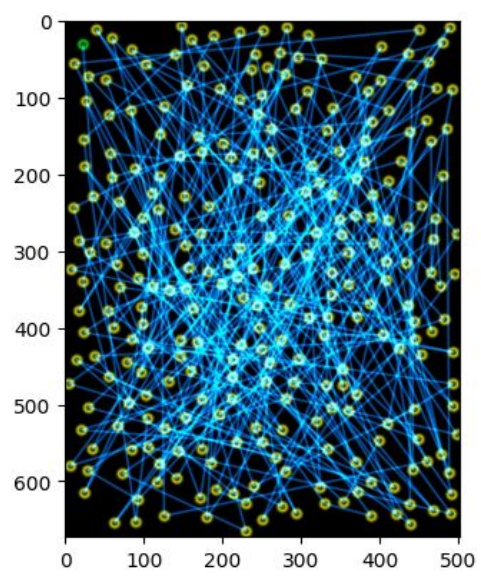
Finding the dots

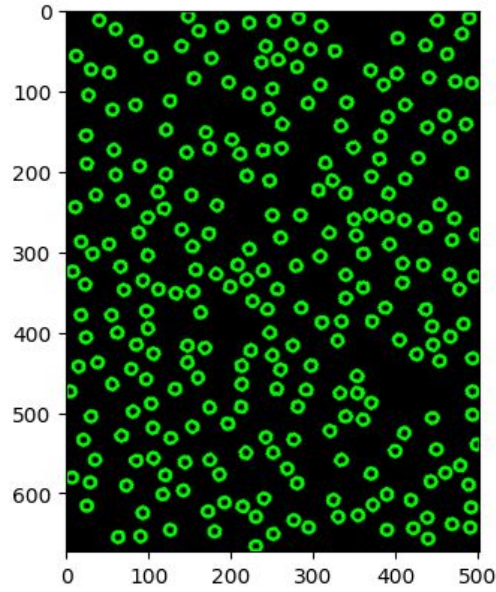
With opencv

- Masking the image to filter for yellow values
- finding contours
- getting the center of each contour

Finding the start point

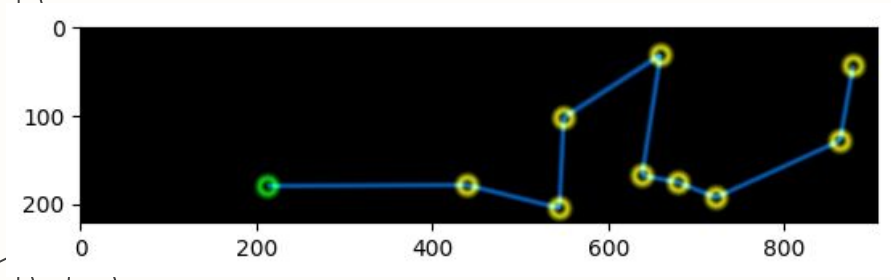
- getting all green points
- calculating the center





Connecting the dots

- Between each point combination
- checking the ratio of blue and black pixels in a line between the two points
- Make it a matrix



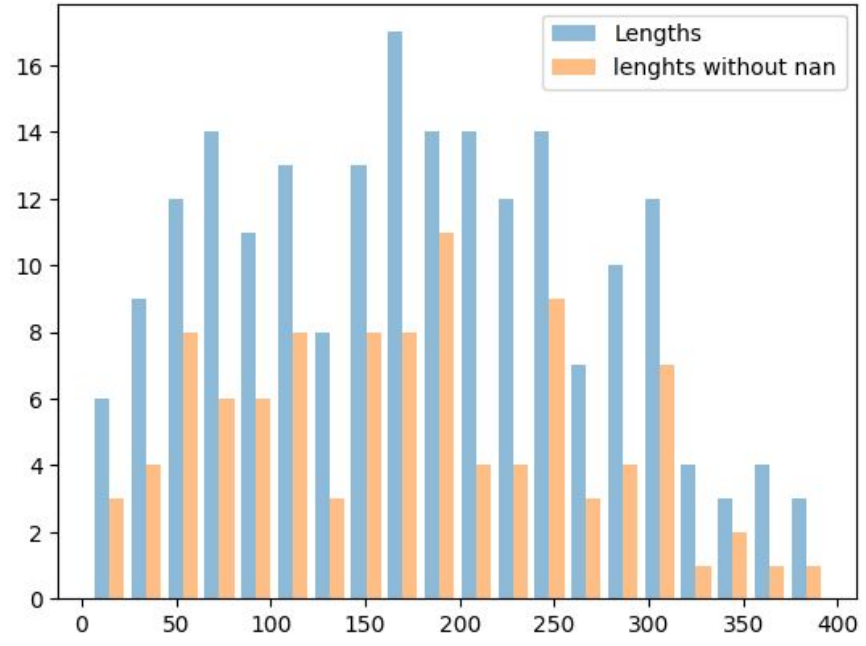
inf	0.82	0.68	0.00	0.45	0.67	0.91	0.95	0.95	0.96
0.82	inf	0.93	0.00	0.91	0.93	0.90	0.00	0.96	0.89
0.68	0.93	inf	0.87	0.00	0.11	0.00	0.76	0.93	0.92
0.00	0.00	0.87	inf	0.85	0.88	0.93	0.94	0.95	0.47
0.45	0.91	0.00	0.85	inf	0.00	0.90	0.84	0.97	0.85
0.67	0.93	0.11	0.88	0.00	inf	0.92	0.91	0.97	0.00
0.91	0.90	0.00	0.93	0.90	0.92	inf	0.94	0.00	0.97
0.95	0.00	0.76	0.94	0.84	0.91	0.94	inf	0.94	0.00
0.95	0.96	0.93	0.95	0.97	0.97	0.00	0.94	inf	0.98
0.96	0.89	0.92	0.47	0.85	0.00	0.97	0.00	0.98	inf

Finding the best path

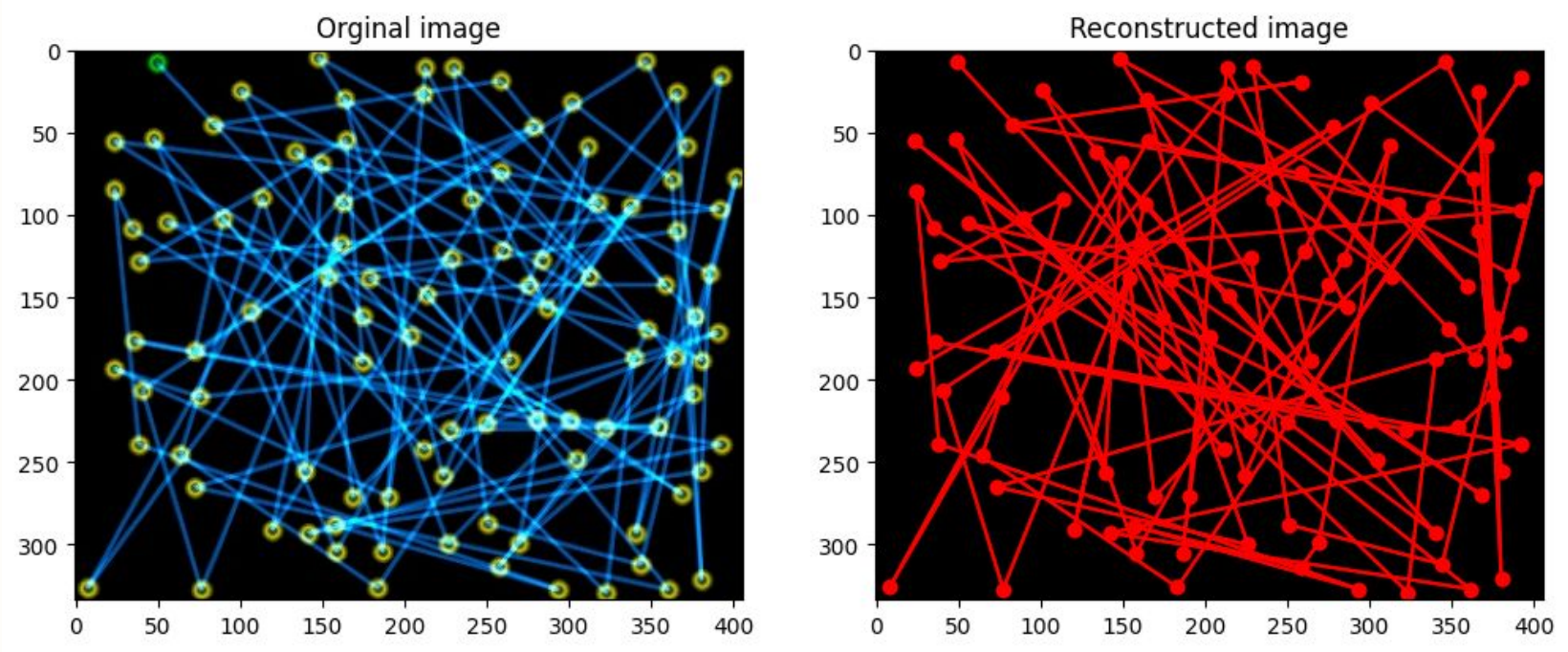
This is a TSP problem. Brute force is mostly not viable.

- Always take the best of the remaining paths
- Use a tsp library
- Write a custom function

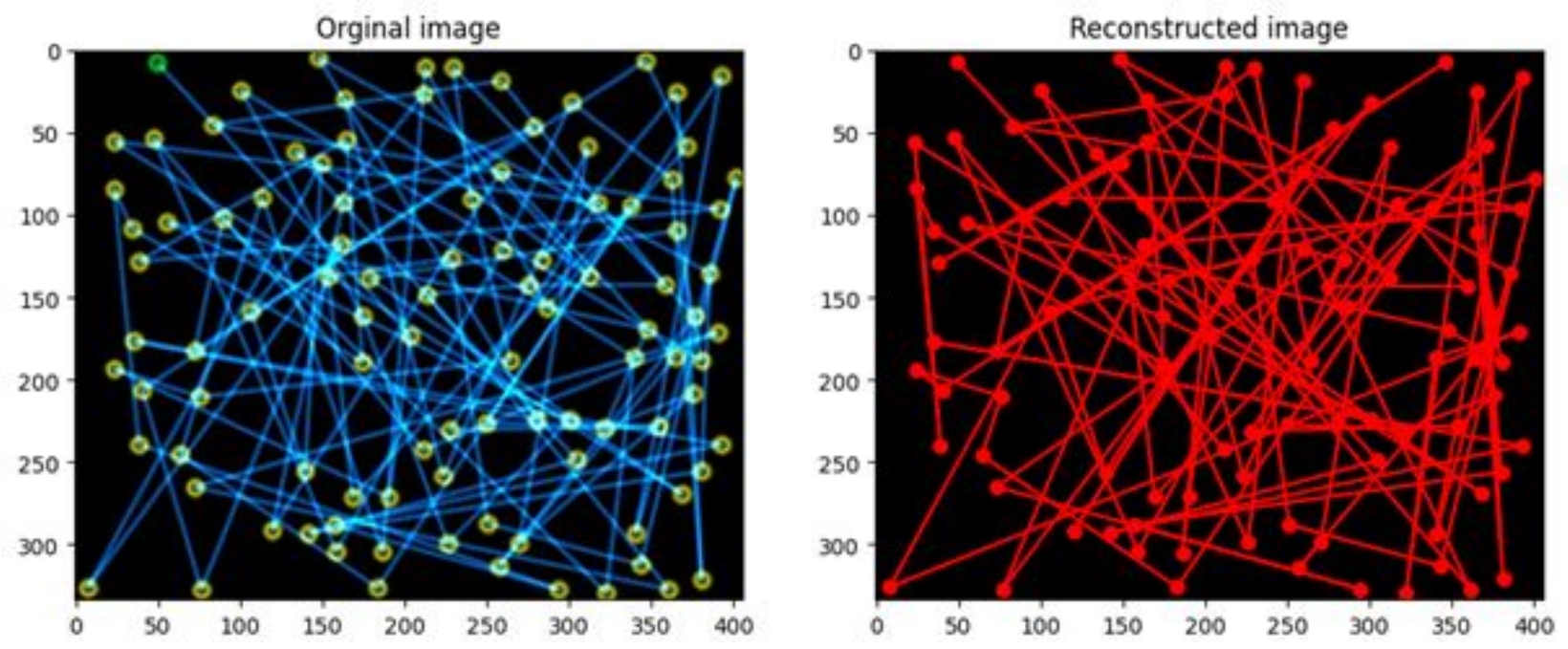
The custom function can only get a result for about half of the pictures



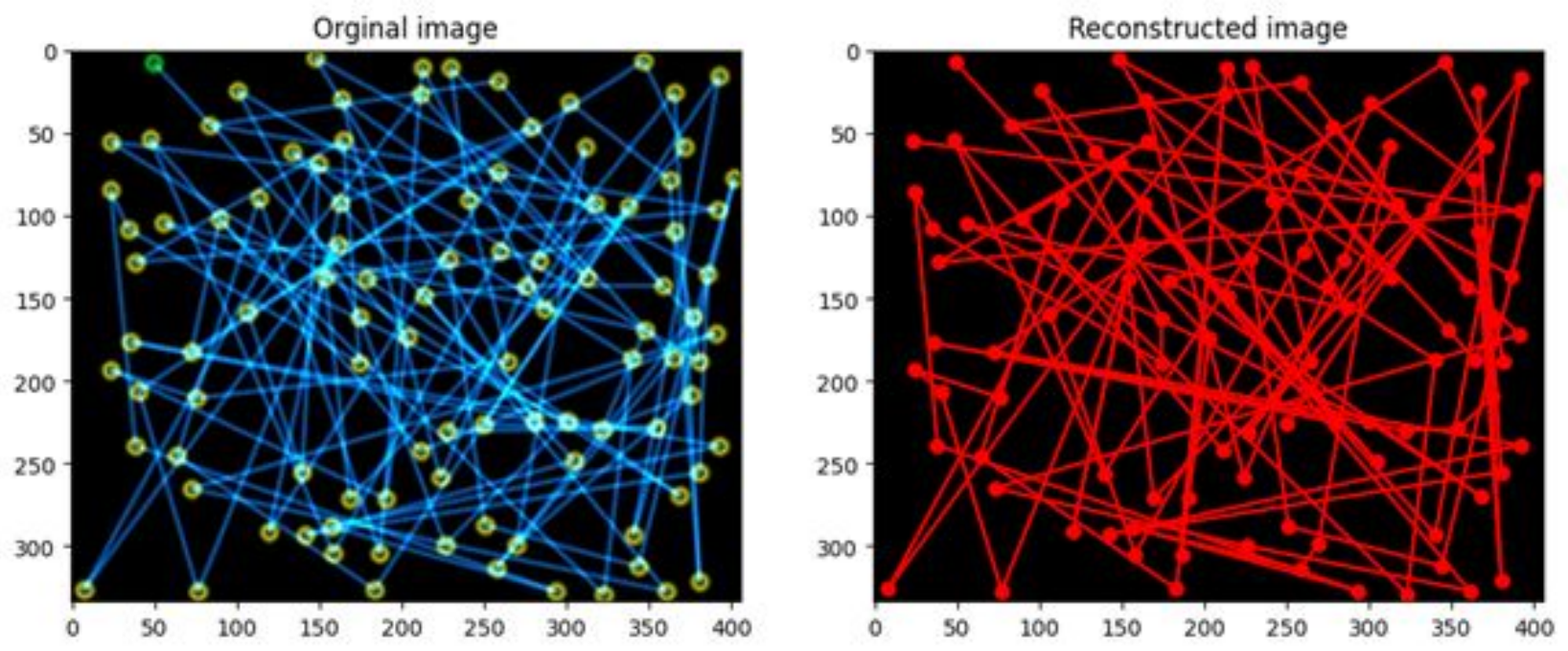
Finding the best path - Simple



Finding the best path - TSP - python_tsp

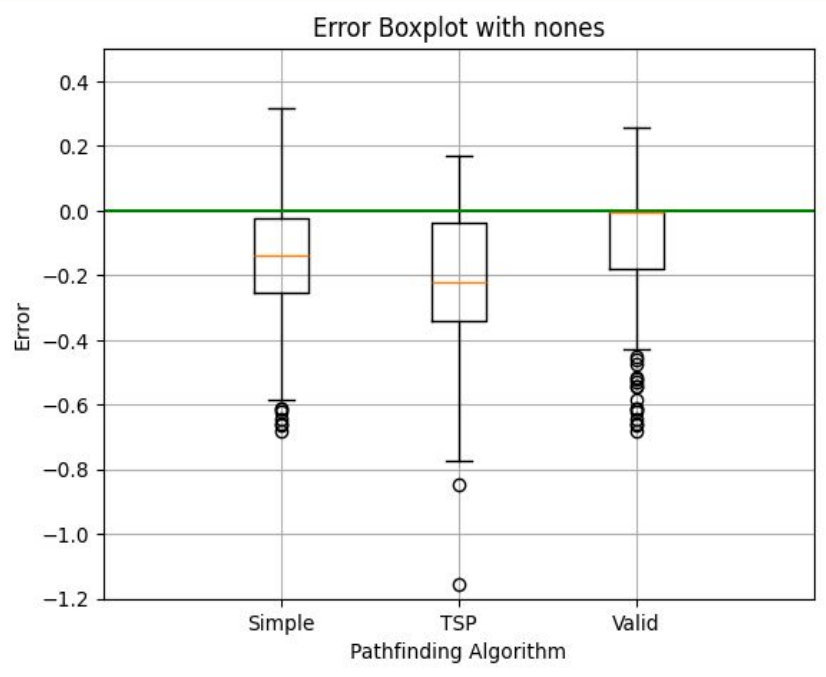


Finding the best path - Custom Pathfinding

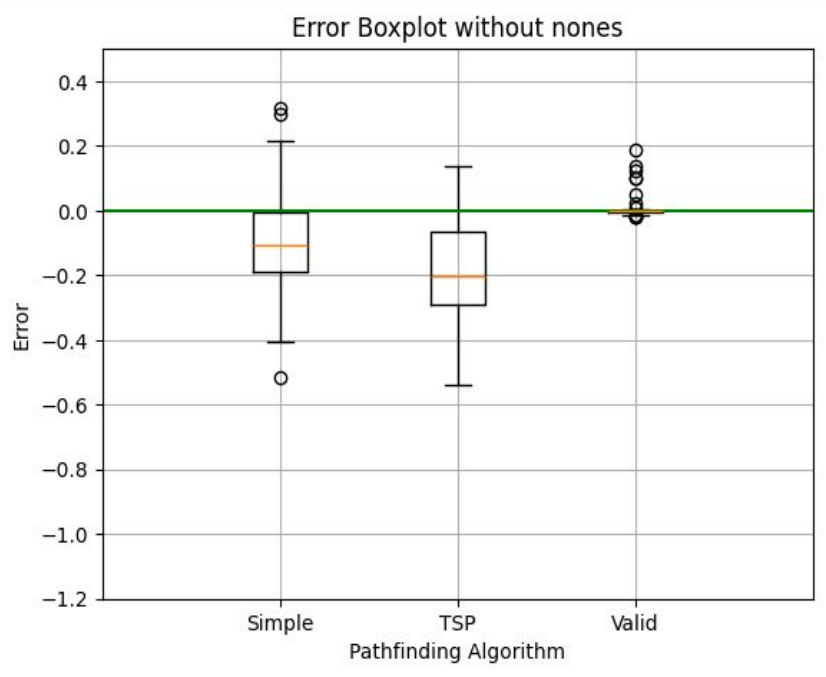


Results

Error: (true_length - predicted_length)/true_length



MSE: [0.0599, 0.0867, 0.0451]



MSE: [0.0303, 0.0611, 0.0009]

04

Conclusion

Summary of findings, lessons learned, and potential directions for further research

Conclusion

ML w/ feature engineering

Relatively accurate (13.7% average error)
Very fast

Most accurate:
🥇 MLP (~0.0028s)

Gradient Boosting (~0.0028s)
Random Forest (~0.03s)
Ensemble (~0.0028s)

Algorithmic

Very accurate (often finds the exact path)
Slow

1st step (find nodes and edges prob matrix)
~5s

2nd step (find the path)
🥇 “Custom” (0.5s -> 6 min)
🥈 “Simple” (0.1s)
🥉 “TSP library” (~1 min)

To go further...

Both approaches show relatively accurate results, but both can still be improved.

ML w/ feature engineering

Try more features
(number of blue pixels for each link, ...)

Improve contour detection

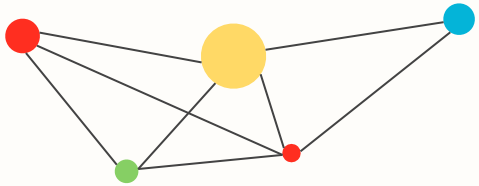
Algorithmic

For better speed, rewrite it in C++,
with parallelisation

Better scoring function for edges
(many ways to score, try multiple ones)

Pathfinding algorithm: logic can be improved

We used the generated dataset for testing
-> test it on the original dataset



Thank you for your attention!