UNIVERSITY OF TARTU
Institute of Computer Science
Data Science Curriculum

**Aap Vare**

# Vision-Based Optimization for Snowplowing on Estonian Roads

**Master's Thesis (15 ECTS)**

Supervisors:
Kallol Roy, PhD
Jaan Übi, PhD

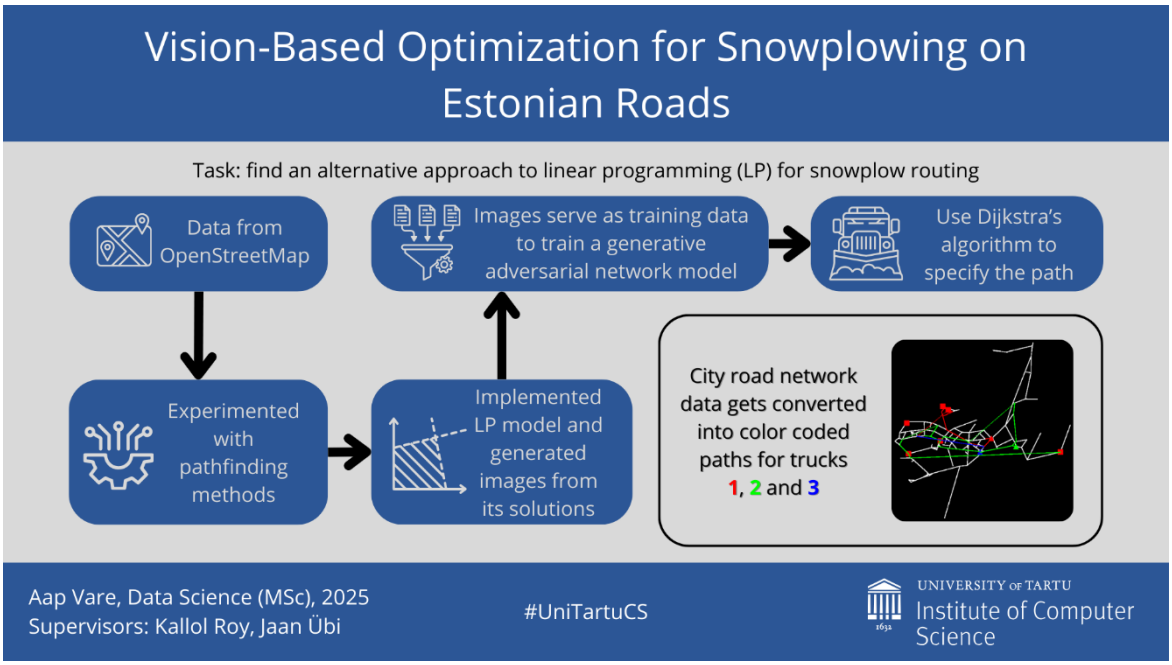
Tartu 2025

# Vision-Based Optimization for Snowplowing on Estonian Roads

**Abstract:**

Efficient snowplow route planning is critical for ensuring road safety and minimizing operational costs during winter maintenance in cold climate regions like Estonia. Traditional approaches to route optimization, such as mixed integer linear programming (MILP), offer high-quality solutions but are computationally intensive and difficult to scale. This thesis explores alternative, data-driven methods for approximating optimal snowplow routes by using images. The snowplowing optimization is formulated as a mixed integer linear programming task (MILP). The road network data from Estonian cities are extracted from OpenStreetMap (OSM) for training. The MILP solutions are converted into labeled images, either as bounding boxes or colored path masks, depending on the target model. We have implemented two deep learning-based vision models: object detection and segmentation using the You Only Look Once (YOLO) architecture, and image-to-image translation using the pix2pix framework. Fine-tuning and transfer learning were used with YOLO on a custom dataset, and pix2pix was trained to produce full route overlays from input maps. Both models were evaluated on their ability to predict plowable paths that closely approximate MILP-generated routes. The results demonstrate that computer vision neural models can serve as fast and approximate alternatives to optimization solvers but with limited applicability in real world scenarios.

**Visual Abstract:**




**Keywords:**

Snowplow Route Optimization, Mixed Integer Linear Programming, Vehicle Routing Problem, Generative Adversarial Network, Computer Vision, Convolutional Neural Network

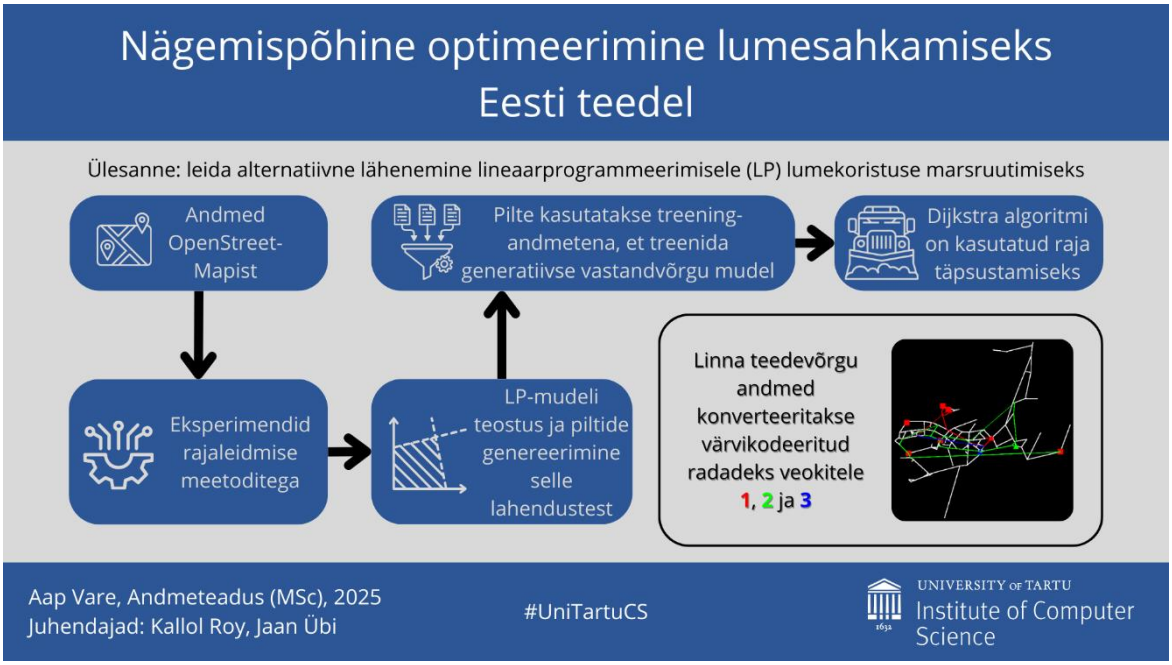
**CERCS:** P160 Statistics, operation research, programming, actuarial mathematics, P170 Computer science, numerical analysis, systems, control, P176 Artificial Intelligence

# Nägemispõhine optimeerimine lumesahkamiseks Eesti teedel

**Lühikokkuvõte:**

Tõhus lumesaha trassi planeerimine on kriitilise tähtsusega liiklusohutuse tagamiseks ja talihoolde tegevuskulude minimeerimiseks sellistes külma kliimaga piirkondades nagu Eesti. Traditsioonilised marsruudi optimeerimise lähenemisviisid, nagu segatäisarvuline lineaarne programmeerimine (MILP), pakuvad kvaliteetseid lahendusi, kuid on arvutusmahukad ja neid on raske skaleerida. See lõputöö uurib alternatiivseid andmepõhiseid meetodeid optimaalsete lumesaha marsruutide lähendamiseks piltide abil. Lumesaha optimeerimine on formuleeritud segatäisarvulise lineaarse programmeerimise ülesandena (MILP). Eesti linnade teedevõrgu andmed võetakse treenimiseks OpenStreetMapist (OSM). MILP-i lahendused teisendatakse märgistatud piltideks kas piirdekastide või värviliste teemaskide näol, olenevalt sihtmudelist. Oleme implementeerinud kaks sügavõppel põhinevat nägemismudelit: objektide tuvastamine ja segmenteerimine, kasutades YOLO (You Only Look Once) arhitektuuri, ning pildist pildiks teisendamine pix2pix raamistiku abil. Peenhäälestus- ja ülekandeõpet kasutati YOLO-ga kohandatud andmekogumil ning pix2pixi mudelit treeniti väljastama terviklikke teid katvaid marsruute sisendkaartide põhjal. Mõlemat mudelit hinnati nende võimekuse järgi ennustada sahatavaid teid, mis sarnanevad MILP-iga genereeritud marsruutidega. Tulemused näitavad, et masinnägemise närvimudelid on võimelised pakkuma kiireid ja ligikaudseid alternatiive optimeerimisülesannete lahendajatele, kuid nende rakendatavus pärismaailmas on piiratud.

**Visuaalne kokkuvõte:**



**Võtmesõnad:**

Lumesaha marsruudi optimeerimine, osaliselt täisarvuline lineaarprogrammeerimine, sõiduki marsruudiprobleem, generatiivne vastandvõrk, masinnägemine, konvolutsiooniline närvivõrk

**CERCS:** P160 Statistika, operatsioonanalüüs, programmeerimine, finants- ja kindlustusmatemaatika, P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria), P176 Tehisintellekt

# Table of Contents

# 1 Introduction

Effective snow removal on roads is vital for ensuring accessibility and safety during winter months. Snowplow routing, however, is costly and complex. It involves balancing objectives such as minimizing operational time, fuel consumption, and travel distances, while ensuring priority areas are serviced first.

In the past, people have tried to solve the issue by utilizing modeling techniques like linear programming (LP). This, however, tends to be computationally expensive and not scalable. Currently there is considerable research thrust to solve the snowplowing optimization task by using deep learning models e.g. reinforcement machine learning. Neural network-based solutions such as this have the potential to provide a faster inference compared to linear programming. That is why a neural network approach is proposed in this thesis.

The goal of the thesis is to use computer vision techniques to find snowplowing optimization solutions that the MILP constraint-based approach is capable of. The neural network and the MILP approaches are then compared against one another using metrics like percentage of matching roads taken and difference in path length. The models used in this thesis, which are YOLO and pix2pix, are also evaluated on their ability to fulfill their respective tasks of object detection and image-to-image translation. We propose and prove by experimental results the following hypothesis

**Hypothesis 1.1:**

Vision based deep learning models produce paths or areas for snowplowing that are close to MILP solution.

Road network information is retrieved from OpenStreetMap (OpenStreetMap contributors, 2025). The dataset includes road geometries, connectivity, and attributes like length and priority classifications. Previous approaches include breadth first search (BFS) and linear programming to find a path between two nodes. There are path traversal methods that were attempted with the goal of finding a way to cover the road network with minimal computational cost like injecting noise into an LP model and probability-based traversal through road intersections.

This thesis explores the optimization of snowplowing routes through a two-stage approach. The first stage employs mixed integer linear programming to provide a mathematical foundation for the snowplow routing problem, using techniques inspired by prior work in the

field (Hajibabai et al., 2014). The second stage transitions the MILP framework into a deep learning computer vision problem, where many output samples of MILP solutions are generated to provide training data for a neural network model. A couple of different deep learning models are tested. One is a pretrained YOLOv12 object detection model. Instance segmentation is also attempted with a pretrained YOLOv11 model. Besides YOLO, there is the pix2pix model which is used to produce colored paths on a road network map input. These models are used to provide estimates for the areas of movement of snowplow trucks. The areas can then be used to find the specific route by connecting the nodes within them by utilizing the Dijkstra pathfinding algorithm.

The remainder of this thesis is structured as follows. Chapter 2 reviews related work in optimization-based and learning-based pathfinding methods. Chapter 3 presents the methodology for data extraction, model and dataset formulation, and neural network training. Chapter 4 details the experimental setup and results. Chapter 5 provides a comparative analysis between MILP and deep learning approaches. Finally, the thesis concludes key findings and directions for future work.

The Python programming language and Jupyter Notebooks have been used to carry out the practical part of this thesis. These can be found in the publicly available GitHub repository[1].

[1] GitHub repository: https://github.com/Fa1nt/Optimizing-Snowplowing-Costs

# 2 Background and Related Work

Winter road maintenance, especially snowplowing, is a critical public service in Estonia that directly affects traffic safety, economic activity, and daily mobility. The Estonian Transport Administration has announced that the budget for maintaining national roads will be reduced by 30 million euros in the coming years (Mäekivi, 2024). This may result in lower service levels, including increased plowing intervals from every two hours to every four hours during snowfall. Longer intervals mean that roads will remain snowy and slippery for extended periods, increasing the risk of accidents and delays. In order to keep snowplowing costs down, there is a need to create optimal snowplow routes.

Snowplow routing problems are often modeled using mathematical optimization techniques, such as Mixed Integer Linear Programming (MILP). Snowplow routing problems belongs to the same class of optimization as Vehicle Routing Problem. In the snowplow routing the main objective is to clear specific road segments while accounting for constraints such as vehicle capacity, fuel, and plowing priorities. The following subsections describe some of the relevant research that has been conducted in recent years. A glossary explaining the terminology used here and in the thesis in general can be found in Appendix I.

## 2.1 Constraint-Based and Heuristic-Based Optimization Approaches

One snowplow routing optimization approach that uses MILP is presented in the paper Network Routing of Snowplow Trucks with Resource Replenishment and Plowing Priorities (Hajibabai et al., 2014). It introduces a model that incorporates several variables and constraints that are relevant to optimizing snowplowing costs (Hajibabai et al., 2014). The inputs of the optimization model are the set of plowing tasks, salt domes, trucks, truck depots, truck capacities, time needed to travel between depots and tasks, and priority of each task (Hajibabai et al., 2014). The main output is the movement of the trucks which are determined by binary decision variables (Hajibabai et al., 2014). The objective function simultaneously minimizes the total deadhead travel time and the longest individual truck cycle time for all priority classes with their associated weights (Hajibabai et al., 2014). The constraints include ensuring that the trucks enter and leave the task links, salt domes, and depots (Hajibabai et al., 2014). The constraints are also set up to establish a relationship between a plowing task or salt replenishment start times and truck routes, and to eliminate subtours (Hajibabai et al., 2014). The overall solution has other heuristics besides the optimization model. They use K-Means clustering to divide up the tasks between the trucks and the Traveling Salesman Problem (TSP)

algorithm to minimize the travel time of each truck and ensure that each of the assigned task links is visited exactly once (Hajibabai et al., 2014).

The authors describe this solution as being capable of minimizing the total travel time of a fleet of 25 snowplow trucks in a road network of over 5000 connected nodes (Hajibabai et al., 2014). Therefore, it seems like a suitable model to utilize concepts from. Unfortunately, there is no code available the solution. The optimization model is described in good enough detail though that it is possible implement using a linear optimization solver.

There are other papers that propose a linear programming method to solve the snowplow routing problem. In the paper by Kinable et al. (2016), they develop and analyze the performance of a mixed integer programming (MIP) model, a constraint programming (CP) model, and a constructive heuristic procedure. The MIP solution used in this paper is not unlike the one proposed by Hajibabai et al. (2014). The inputs, outputs and constraints are quite similar. They also introduce refueling as a possible job to perform on top of salt resupply (Kinable et al., 2016). The CP method relies on interval variables which represent intervals during which an activity can be performed (Kinable et al., 2016). This is a more compact and suitable solution for sequencing and resource constraints. The heuristic procedure consists of two parts. One is a constructive heuristic to quickly build an initial feasible schedule (Kinable et al., 2016). The other is a late acceptance heuristic for improvement that swaps or moves jobs between vehicles while tracking acceptance criteria (Kinable et al., 2016). The authors find that the heuristic solution scales better than CP or MIP (Kinable et al., 2016). In their experiments, both CP and MIP ran into memory issues when trying to tackle large instances of routing problems (Kinable et al., 2016). This is something that was observed while working on this thesis as well. The heuristic approach scales better but generally gives a less optimal solution compared to CP (Kinable et al., 2016).

The study by Nguyen & Tran (2024) was designed to incorporate a combination of four routing optimization approaches and tools, including the vehicle routing problem's objective function, Dijkstra's pathfinding algorithm, constraint-based programming, and ArcGIS' network analyst. The authors aimed to develop a constraint-based optimization model for snowplow routing that minimizes the number of snowplow trucks required, maximizes the Level of Service (LOS) satisfaction in winter maintenance operations, and addresses challenges such as deadhead time, fleet size, and resource allocation (Nguyen & Tran, 2024). The VRP is modeled as a directed graph with plowable arcs, treating time, and truck capacity constraints (Nguyen & Tran, 2024). Dijkstra's algorithm is used to find the shortest paths between depots and snow

routes (Nguyen & Tran, 2024). The constraints encode the plowing rules, like continuity of the snow and ice routes in the network, deadhead time, and truck capacity (Nguyen & Tran, 2024). ArcGIS Network Analyst serves as the tool which is used to run the optimizations and visualize the network (Nguyen & Tran, 2024). The authors simulate two main optimization options: minimize fleet size while maintaining LOS or maximize LOS with the current fleet (Nguyen & Tran, 2024). The paper provides pseudocode to give an idea of the implementation (Nguyen & Tran, 2024). The overall approach is similar to the previous papers regarding the constraint programming. This paper provides two options for optimization which is a little more complex. The authors claim to have managed to save 29 hours in total travel time compared to the previously existing solution (Nguyen & Tran, 2024).

Rasul et al. (2021) aimed to optimize snowplow routing in residential municipalities by minimizing total travel distance, reducing undesired maneuvers like U-turns and sharp turns while meeting real-world constraints. They used a hybrid method that combines the Chinese Postman Problem (CPP), Dijkstra's algorithm, and Tabu Search Optimization (Rasul et al., 2021). CPP ensures that each edge in the traversable graph is visited at least once in an optimal way (Rasul et al., 2021). It also converts non-Eulerian graphs to Eulerian (Rasul et al., 2021). Dijkstra's algorithm computes the shortest paths for the CPP solution (Rasul et al., 2021). Tabu search optimization refines the initial CPP-based routes (Rasul et al., 2021). Tabu search is a meta-heuristic optimization technique for optimizing model parameters (Rasul et al., 2021). It starts with an initial solution and progresses iteratively by searching immediate neighborhoods (Rasul et al., 2021). The authors use OSM and NetworkX (Rasul et al., 2021), which are also utilized in this thesis. Their solution focuses on similar points as the prior papers, which are minimizing the total distance traveled and minimizing U-turns. They find that the Tabu optimization helps improve the initial CPP solution in terms of reducing total distance and number of U-turns (Rasul et al., 2021).

In the paper by Rao et al. (2011), they aimed to solve the single-plow snow route optimization problem in small towns, focusing on minimizing the total distance, reducing the number of U-turns and repeated road segments, and servicing higher priority roads first. They decompose the road graph into smaller bi-connected components and trees to reduce complexity (Rao et al., 2011). After that they applied the A* search algorithm to find the optimal path (Rao et al., 2011). A* is customized by the modeling of a problem-specific state, defining a set of operators, start and goal states and devising cost and heuristic functions to guide the search (Rao et al., 2011). The states consist of partial or complete walks through the graph (Rao et al.,

2011). The operators signify valid moves to adjacent vertices (Rao et al., 2011). A* needs to be able to guide its exploration by identifying the most promising state at each step of the search (Rao et al., 2011). This is where the cost function and heuristic come in. The cost function combines distance, number of U-turns, repeats, and priority misplacements, with user-defined penalties (Rao et al., 2011). The heuristic estimates the remaining distance and future priority misplacements (Rao et al., 2011). The results show an up to 12% travel distance reduction with zero increase in U-turns or priority violations compared to manually designed routes (Rao et al., 2011). Dividing up the road network into smaller spaces where a pathfinding algorithm is run on, is not unlike the solution proposed in this thesis. The paper presents a single-plow only approach though (Rao et al., 2011), not multi-plow coordination. A method for deciding which path to proceed with at each step was also attempted during the work for this thesis.

While these LP and heuristic-based methods produce strong results, they require significant computational resources and cannot be generalized so easily across different road networks. This motivated the use of deep learning in this thesis, as to take advantage of MILP-derived solutions and use them as training data for a model that can be applied to various road networks.

## 2.2 Neural Network-Based Pathfinding

Others have attempted to overcome the limitations of approaches such as LP by utilizing deep learning methods. Joshi et al. (2019) tried to approximately solve the Traveling Salesman Problem (TSP) using deep learning. Their paper focuses on the 2D Euclidean TSP, where cities are represented as points in 2D space (Joshi et al., 2019). The goal is to find the shortest path that visits each city once and returns to the start (Joshi et al., 2019). The authors propose a non-autoregressive deep learning approach using a Graph Convolutional Network (GCN) that outputs a probabilistic adjacency matrix, representing the likelihood of each edge being part of the TSP tour (Joshi et al., 2019). The model is trained in a supervised manner using pairs of problem instances and optimal solutions (Joshi et al., 2019). The solutions are provided by an existing TSP solver (Joshi et al., 2019). The authors claim that the model performs faster than autoregressive models due to computing the whole adjacency matrix in one shot and not having to predict one node at a time (Joshi et al., 2019). The presented approach arrives at a solution relatively quickly compared to many other methods, but it is a bit less accurate compared to traditional solvers (Joshi et al., 2019). The authors also point out that their models are trained on graphs with fixed numbers of nodes, and they do not generalize well to other sizes (Joshi et al., 2019).

Methods like MILP can be used as a basis for constructing neural network solutions to solve the snowplow routing problem. Neural networks, namely reinforcement learning agents, have been used to compute solutions to MILP problems. Lee & Kim (2024) tackled the problem of solving Mixed-Integer Linear Programming problems, particularly those involving non-binary integer variables, using a reinforcement learning (RL)-based approach. Their goal was to create an end-to-end RL-based solver that finds fully feasible solutions to MILP problems without relying on traditional solvers like Gurobi and improves solution quality over time (Lee & Kim, 2024). The RL environment is a MILP instance converted into a bipartite graph consisting of variables and constraints (Lee & Kim, 2024). The agent uses a Graph Neural Network (GNN) with a transformer encoder to learn dependencies between variables and constraints (Lee & Kim, 2024). For each variable, the agent can increase, decrease, or leave the value unchanged (Lee & Kim, 2024). The reward system has two phases. The first continues until any feasible solution is found (Lee & Kim, 2024). The second is designed to improve the feasible solution toward the optimal (Lee & Kim, 2024). The authors test the model on MILP datasets of varying sizes and compare it with baselines including Gurobi and CNN-based methods (Lee & Kim, 2024). They find that the result is scalable and can learn from experience without needing labeled data (Lee & Kim, 2024). The solution reaches optimal solutions with less than 1% gap (Lee & Kim, 2024).

Nazari et al. (2018) aimed to solve the VRP using RL. Specifically, they wanted to develop a general framework that can solve multiple instances of VRP without retraining and eliminate the need for classical heuristics or hand-engineered rules (Nazari et al., 2018). The solution for this was a sequence-to-sequence neural network architecture with reinforcement learning (Nazari et al., 2018). Its input consists of static info, which is the locations of the destinations, and dynamic info, which is the demand at each of the locations (Nazari et al., 2018). The demand will change as the vehicle visits the destinations (Nazari et al., 2018). The model contains an RNN decoder with attention mechanism to generate the next node in the vehicle's route (Nazari et al., 2018). The encoder is removed to handle unordered inputs since the nodes that need to be visited are in a set, not a sequence (Nazari et al., 2018). The authors use the policy gradient algorithm to train the network (Nazari et al., 2018). It contains two networks: an actor network that predicts a probability distribution over the next action at any given decision step, and a critic network that estimates the reward for any problem instance from a given state (Nazari et al., 2018). There are masking procedures in place to prevent infeasible choices from happening during decoding (Nazari et al., 2018).

Like the RL solution by Lee & Kim (2024), this one is quite generalizable, does not require retraining, and offers competitive performance. Inference is fast once the model is trained (Nazari et al., 2018). The VRP focuses on utilizing just one vehicle though (Nazari et al., 2018). Even though RL is not used in this thesis, there is an overlap in the objective. The goal is to speed up inference from LP and come up with a generalizable and scalable solution. The hope for this thesis is that the training can be performed more easily using an existing computer vision model.

CNNs have been used to solve pathfinding, but not vehicle routing. The authors of one such paper aimed to develop a fast, data-driven method for 2D path planning using CNNs (Sartori et al., 2021). It works on image-based map representations with obstacle information and start and goal points (Sartori et al., 2021). The output is a sequence of waypoints from start to goal that can be used for robotic navigation (Sartori et al., 2021). The training data consists of paths generated using the Theta* algorithm (Sartori et al., 2021). The CNN model uses a modified VGG16_bn architecture (Sartori et al., 2021). The inputs are 320 × 320 images that depict free space in black and obstacles in white (Sartori et al., 2021). The start position is green, and the end position is red (Sartori et al., 2021). The output contains coordinates of up to 10 waypoints forming a path (Sartori et al., 2021). The training data consists of paths generated using the Theta* algorithm (Sartori et al., 2021). The authors claim that the inference of the trained model is fast and there is a high correlation between the predicted and the true waypoints (Sartori et al., 2021). The path length is fixed at a maximum of 10 waypoints though which could limit the adaptability for longer and more complex paths (Sartori et al., 2021). The CNN models are trained for specific environments as well, so they do not generalize well across different layouts without retraining (Sartori et al., 2021).

Caffagni (2023) also explored whether a CNN could effectively perform 2D path planning. The task involved finding the shortest path from a start point to a goal point on a 100 × 100 grid, avoiding obstacles (Caffagni, 2023). The author created a synthetic dataset of approximately 230,000 samples by generating random occupancy grids with varying obstacles (Caffagni, 2023). Start and goal positions were randomly assigned (Caffagni, 2023). Ground truth paths were computed using a custom implementation of the D* Lite algorithm (Caffagni, 2023). The model used is an encoder-decoder CNN architecture that resembles U-Net (Caffagni, 2023). The encoder compresses the input map into a latent representation, while the decoder reconstructs a score map indicating the likelihood of each cell being part of the optimal path (Caffagni, 2023). To address the CNN's inherent position invariance, the input was

expanded to three channels: the occupancy grid, a gaussian map centered at the start of the position, and a gaussian map centered at the goal position (Caffagni, 2023). This relative positional encoding helps the network understand the spatial relationship between the start, goal, and obstacles (Caffagni, 2023). After obtaining the score map, a bidirectional search algorithm is used to extract the path from the start to the goal by following the highest probability values (Caffagni, 2023). 87% of the total test samples presented a valid solution (Caffagni, 2023). The model was trained on synthetic occupancy grids though, and not real-world road networks.

The approach described by Caffagni (2023) inspired the general methodology that is used in this thesis. He was trying to find a path between two points, not a route between many nodes. The underlying process utilizing a CNN reminiscent of U-Net is still quite similar though.

# 3 Methodology

This thesis explores in solving the snowplow cost optimization problem from the lens of computer vision. An approach is proposed that combines optimization and deep learning techniques to approximate snowplow routing solutions. There were a few other pathfinding methods that were explored before deciding on the final approach.

## 3.1 Data Extraction

The work in this thesis revolves around road network data. This is extracted from OpenStreetMap using the OSMnx Python package (Boeing, 2025). OSM data is open data that is free to use for any purpose, as long as they and their contributors are credited (OpenStreetMap contributors, 2025). The street networks are extracted as graphs, and they are comprised of roads that are navigable by vehicles. The edges in the network are undirected.

OSM graph data consists of nodes and edges. Nodes represent geographic coordinates such as intersections, endpoints, or turns. Edges represent the roads connecting those nodes. Each edge in the graph includes attributes such as length, road type, and speed limit. Given that this thesis tackles snowplowing, only edges that are navigable by motor vehicles were retained, filtering out paths such as footways, cycleways, and pedestrian zones.

For the purposes of training computer vision models, the extracted road network graphs were rendered into black-and-white images using the Matplotlib[2] and NetworkX[3] libraries, where roads appear as white lines on a black background. Additionally, graphical markers were overlaid to indicate special features like the starting locations and destination nodes.

## 3.2 Tested Approaches Besides Computer Vision

### 3.2.1 Pathfinding Heuristics

We have explored on different pathfinding approaches before converging to our proposed method. First, we convert the OSM graph into a grid of pixels and setting a start and end point on it. The task was to find a path among the navigable road pixels. Breadth First Search (BFS) was used to find the optimal route. An example of this is depicted in Figure 1.

[2] Matplotlib library: https://matplotlib.org/
[3] NetworkX library: https://networkx.org/

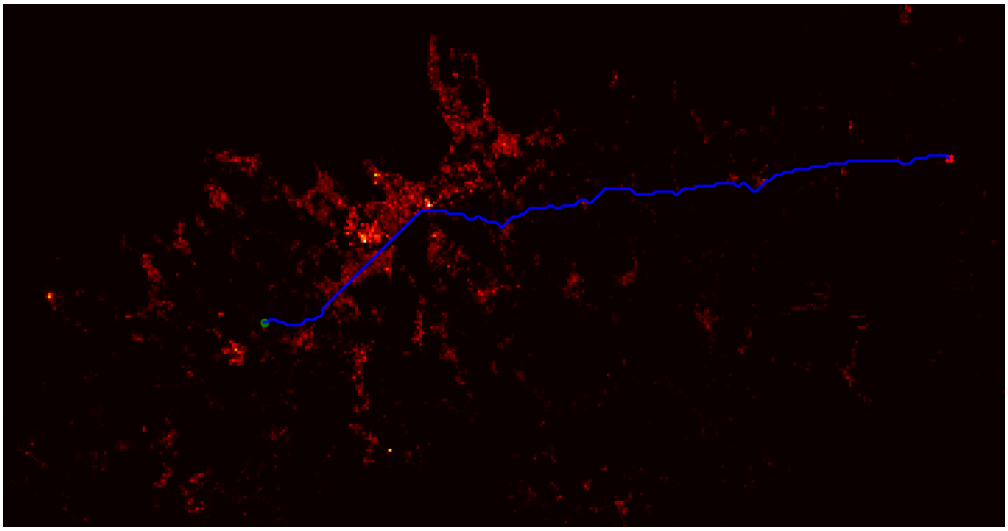


Figure 1. Path that was found between two random points using BFS.

After that, linear programming is tested to find the optimal path. This is inspired by several of the papers mentioned earlier. The graph network data was converted to an adjacency matrix that was populated with the weights of the edges, representing the length of the paths between any two directly connected nodes. This matrix is sparse since most entries are missing, because not every node is connected to every other.

An ILP model was created using the Gurobi optimizer[4] in Python to formulate the shortest path problem as flow optimization. The goal of this model is to minimize the total distance traveled. The flow starts at a source node and ends at a target node. The variables are binary for each edge: 1 if the edge is used, 0 otherwise. The constraints are formed so that any node that is entered needs to be exited. Also, there can only be one path from source to target. The components of the ILP model are described in Table 1. Figure 2 provides an example of a path found using this solution.

Table 1. Overview of the components of the shortest path flow ILP model.

| Component | Description | |
|---|---|---|
| Objective Function | Minimize the total travel cost of selected edges | $min \sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$ |
| Decision Variables | Binary variable indicating whether edge $(i, j)$ is included in the shortest path | $x_{ij} \in \{0, 1\}$ |

[4] Gurobipy library: https://pypi.org/project/gurobipy/

| | | |
|---|---|---|
| Parameters | Cost (distance) associated with edge $(i, j)$ | $c_{ij}$ |
| | Set of all directed edges with positive weights in the graph | $E$ |
| Flow Conservation Constraints | For each node $i \in V$: | |
| | Source node | $\sum_j x_{ij} - \sum_j x_{ji} = 1$ |
| | Target node | $\sum_j x_{ij} - \sum_j x_{ji} = -1$ |
| | Intermediate nodes | $\sum_j x_{ij} - \sum_j x_{ji} = 0$ |

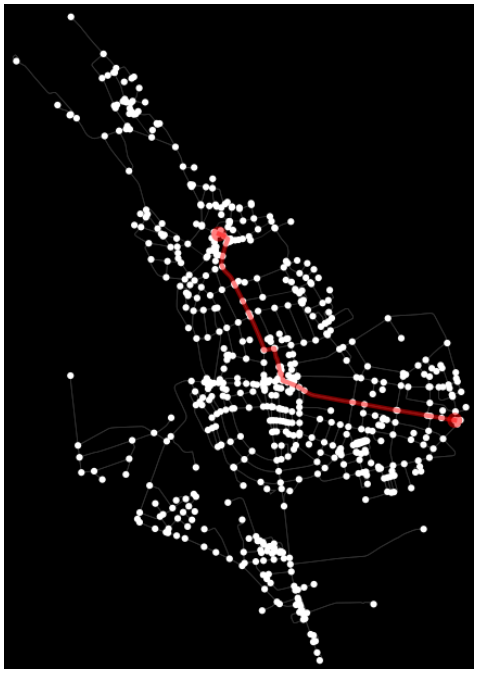


Figure 2. Path found using the shortest path flow ILP model, highlighted in red.

The idea of exploring complementary space was explored after creating the initial flow optimization with ILP. Two subspaces of a vector space are said to be complementary if their direct sum gives the entire vector space as a result (Taboga, 2021). One subspace in the context of the pathfinding problem is assumed here to be the optimal path. To find the complementary space, the non-optimal paths would have to be found. After solving the initial shortest path, the weights of the edges, that were part of the solution, are multiplied by a noise factor of 1.5. Solving the problem again with the updated weights enables the finding of alternative paths that avoid reusing the same roads as before. An example of finding several paths between two

points with this method is presented in Figure 3. This technique encourages path diversity while still favoring short and reasonable paths.

Figure 3. Complementary paths found between two random points, highlighted in different colors.

A maximum likelihood-based approach was tested to predict paths through a road network in an efficient manner. Instead of finding the shortest path with Dijkstra or ILP, the probabilities are modeled for choosing edges. The idea is to approximate how a snowplow might probabilistically move through a city network toward a goal. It starts with initializing verifier values for each edge at each node in the graph. The verifier values are essentially probabilities which edge would be the best one to take. These values can be initialized in different ways. One way is to distribute the probabilities evenly, so that each edge has an equal probability to start out with. Another would be to give a higher value to the shorter edge and a lower value to the longer edge. After that, a thousand random paths are found in the network using the A* pathfinding algorithm. Every time a path is found, the verifiers at each node are updated based on which edges were in the path. If an edge is part of the optimal solution, its verifier is increased by 0.1 and the probabilities at that node are normalized, so that they would all add up to 1. After this process, the paths can be found in the network by always choosing the path with the highest probability of leading to the most optimal solution.

Utilizing verifiers to find the path is quite efficient since the path can be found by simply following the probability values. However, this typically leads to suboptimal solutions and

dead ends in the graph, which is why this did not find use in the end. The other approaches listed in this section, while being more functional, did not end up being used to achieve the final results of this thesis either.

### 3.2.2 Pathfinding with Object Detection CNN

There was the idea of utilizing computer vision techniques to find the path or the approximate area of the path in an image of the road network. This involves the use of object detection or instance segmentation to highlight where the optimal path is located.

The first attempt at doing this was done on images of simple road network segments sampled randomly from the map of Estonia. The roads are white, and the background is black. The start and end points are marked at random points on the road. The input images do not have the connecting path marked on them but the ground truth bounding box surrounds it as well as the start and end points. Segmentation was also tested. The ground truth segment mask also covers the path and the start and end points.

The training data consisting of images and bounding boxes, or images and segment masks, were used to fine tune the pretrained YOLOv8n model (Jocher et al., 2023). The training set contained 800 images; the validation set contained 200 images. The images are re-sized to 128 × 128 for the training and the number of epochs was 50 and the size of the batch was 16. Both the object detection and segmentation models provided good results in terms of accurately highlighting the area on the image where the path is supposed to be. Examples of this can be seen in Figure 4 and 5.

There was another attempt at finding bounding boxes on maps of Estonian cities and towns. This time the images had the size 640 × 640 and the number of epochs was increased to 100. The labels were also given categories based on the order that they are supposed to be driven through to get from the starting point to the end. The trained model was able to find paths with relative success. An example of this is Figure 6. The bounding boxes did not always cover the whole path area, but they were generally in the right ballpark. The order of the labels was also generally accurate, heading from start to finish incrementally.

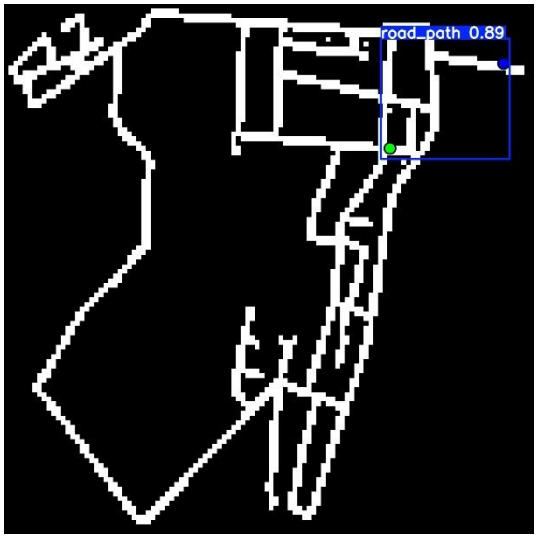



Figure 4. Bounding box prediction of the fine-tuned YOLOv8n model.


Figure 5. Segmentation mask prediction of the fine-tuned YOLOv8n model.

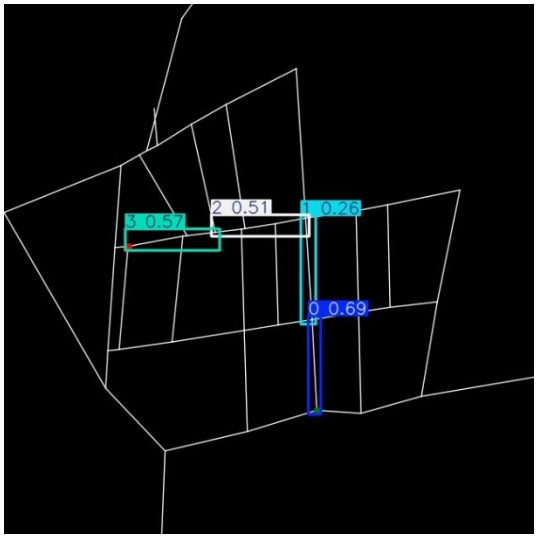



Figure 6. Bounding box predictions of the fine-tuned YOLOv8n model on a larger graph of an Estonian town.

## 3.3 Computer Vision to Optimize Snowplow Costs

Given that the CNN-based computer vision pathfinding has shown promise, we decided to proceed with this for tackling the snowplow routing as well. YOLO was utilized for this again, but an image-to-image translation approach was also attempted.

### 3.3.1 Ground Truth MILP Data

The ground truth data comes from solutions that are derived from a MILP model. This model is the one that is described in the paper by Hajibabai et al. (2014). The objective function,

constraints and variables are shown in Table 2. This MILP model is used to solve snowplow routing, given certain task nodes and depots on a map. The optimization model has been implemented in Python using the PuLP[5] library. The maps used for the training data are all 47 cities in Estonia. Each image in the data has the drivable roads of a city. In each instance, 10 nodes are randomly picked from the graph. Between 1 and 3 of them are designated as depots or the start and end points of the trucks. 1 node is assigned to be the salt dome, and all the remaining are task nodes.

Table 2. MILP model optimization objective function and constraints from the paper by Hajibabai et al. (2014).

| | | |
|---|---|---|
| **Objective Function** | Minimize total deadhead travel time and the longest individual truck cycle time for all priority classes with their associated weights | $\min \sum_{p \in P} c_{LOS,p} z_p + c_{fuel} \sum_{k \in K} \sum_{i \in I \cup S \cup \{d_k\}} \sum_{j \in I \cup S \cup \{d_k\}} t_{i,j} x_{i,j,k}$ |
| **Constraints** | Any truck $k \in K$ exits from the depot | $\sum_{i \in I} x_{d_k,i,k} = 1$ |
| | Any truck $k \in K$ goes back to the depot | $\sum_{i \in I} x_{i,d_k,k} = 1$ |
| | Any truck $k \in K$ that enters a link or satellite salt location will exit that location | $\sum_{j \in I \cup S \cup \{d_k\}, i \neq j} x_{i,j,k} - \sum_{j \in I \cup S \cup \{d_k\}, i \neq j} x_{j,i,k} = 0$ |
| | Every plowing task is performed exactly once | $\sum_{k \in K} \sum_{j \in I \cup S \cup \{d_k\}} x_{i,j,k} = 1$ |
| | Establish the relationship between a plowing task or salt replenishment start times and truck routes, and eliminate subtours | $u_i + t_{task,i} + t_{i,j} + U(x_{i,j,k} - 1) \leq u_j$ |
| | When a truck $k \in K$ moves from task $i \in I$ to a task or satellite salt location $j \in I \cup S$, the amount of salt will decrease by $l_i$ | $v_{i,k} - l_i - L_k(x_{i,j,k} - 1) \geq v_{j,k}$ |
| | Salt replenishment for vehicle $k \in K$ | $v_{j,k} \leq L_k$ |
| | Define the longest individual truck cycle time $z_p$ in the objective function | $z_p \geq \delta_{i,p}(t_{task,i} + u_i)$ |
| | Define the binary variables that determine which path to take | $x_{i,j,k} \in \{0, 1\}$ |

[5] PuLP library: https://coin-or.github.io/pulp/

| | | |
|---|---|---|
| | Set the initial times of the tasks to 0 | $u_i \geq 0$ |
| | Indicate the nonnegativity of remaining salt on truck | $v_{i,k} \geq 0$ |

The goal of the MILP optimization is to have all of the trucks start from their respective depots, drive through the task nodes in the most efficient manner and return to their depots. The MILP solution has been set up in a simplistic way where many of the input variables do not have an impact on the path finding. This was done because it is not possible to input these variables when trying to find the best paths with a computer vision type solution. The trucks' capacities are set to such a large number that they never need to visit the salt dome. This is why the salt dome does not appear on the training images. It is simply defined for running the MILP optimization. Fuel consumption and road priorities are also non-factors.

### 3.3.2 Training Images and Annotations

The generated MILP solutions get converted into images. The images are set to the default image size in YOLO which is 640 × 640. The node coordinates from each city are extracted and normalized to fit within the dimensions of the image. The task and depot nodes are depicted on the image as red squares and green triangles respectively. These are supposed to serve as reference points for the model since the path will pass through them. The color and shape have been chosen to make them stand out from the white road lines and the black background. The optimal path generated by MILP is not included in the training image because this is what the model is supposed to help find from the input. The images are created using the Matplotlib library[6]. Examples of the training images can be found in Figure 7.

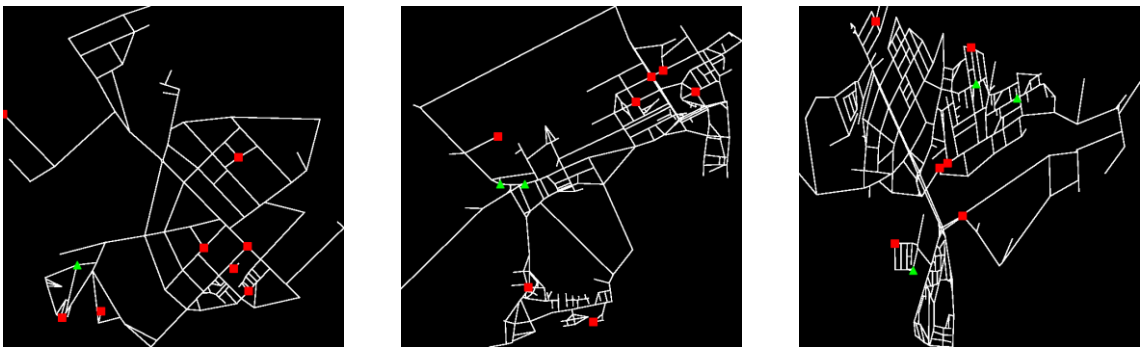


Figure 7. Examples of training images for YOLO object detection. The image on the left has 1 depot, the one in the middle has 2, and the one on the right has 3.

[6] Matplotlib library: https://matplotlib.org/

The annotations for the training images are created automatically instead of labeling by hand. The bounding boxes are placed so that the start and end node and the optimal path between them are inside the boxes. The segmentation masks are created in the same manner, covering two nodes and the path. The classes for the annotations are created so that they would reflect the order in which the boxes or masks should be driven through to follow the best snowplow route. A few different methods were attempted for the placement and classification of the image labels to try to improve the model's performance. In the case of bounding boxes, box 0 contains the path between depot 1 and the following task node, box 10 contains depot 2 and the following task node, and box 20 contains depot 3 and the following task node. There are 30 classes in total. The other classes contain the task nodes and the paths in between. This classification approach was used for segmentation as well. The masks, however, were attempted to be placed so that one segment covers just one edge in the path. This was done to avoid creating masks with complicated shapes due to having to cover turns. Bounding box labels and optimal snowplow routes are visualized on a training image in Figure 8. Segmentation mask labels are visualized for the same road network in Figure 9.

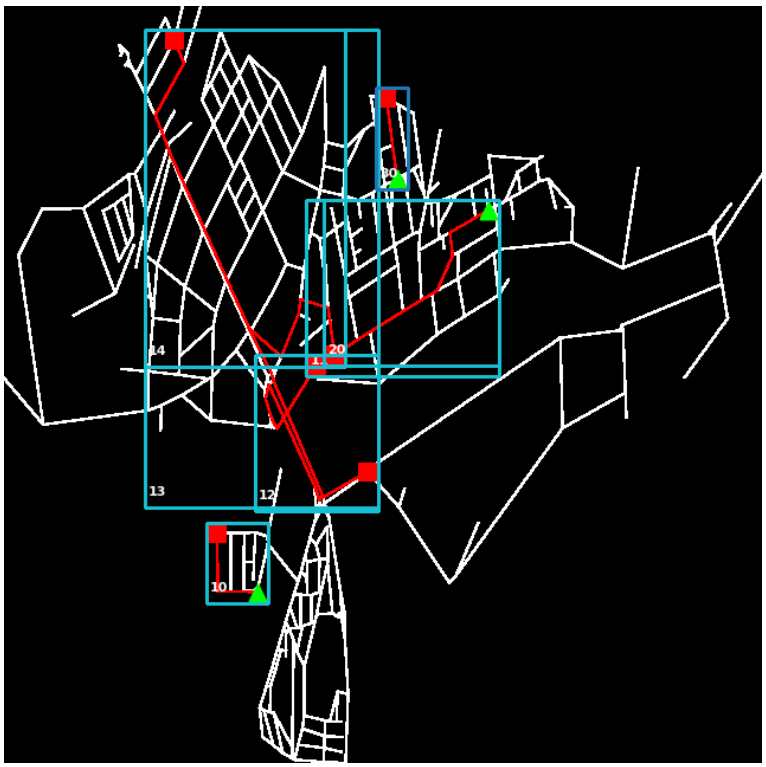



Figure 8. A ground truth image of the optimal snowplow routes with the training bounding boxes plotted.

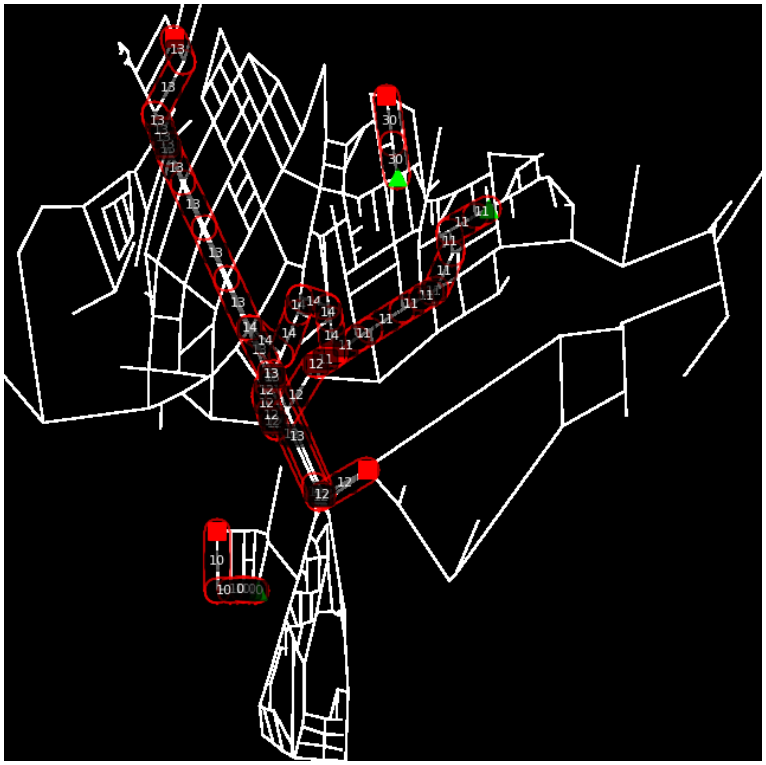


Figure 9. A ground truth image of the optimal snowplow routes with the training segmentation masks plotted.

The size of the object detection training set was 1600, and the validation set was 400. The size of the training set for segmentation was 2400, and the validation set was 600.

### 3.3.3 Converting Outputs to Optimal Paths

To retrieve the information about the optimal path from the output of trained CNN model, the plan was to extract the nodes from the predicted labels. Based on the node mapping that was used to create the initial input images of the graph representations, the graph nodes can be extracted from the image.

The bounding boxes or segments can be overlayed on an image by reading in the outputs of the trained YOLO model. The coordinates can be used to find the area of each label and they can be combined to create a unified area where the optimal path is located. Nodes inside of this area are linked together using Dijkstra's pathfinding algorithm in OSMnx (Boeing, 2025). The order given by the classes included in the output set the order of how to navigate the nodes within the labels.

### 3.3.4 Implementation & Experiments

Pretrained YOLO models were used to try to predict the boundaries of the optimal snowplow path. YOLO (You Only Look Once) is a real-time object detection framework that consists of a single convolutional neural network that predicts object bounding boxes and class probabilities directly from input images (Joseph et al., 2015). It is known for its speed and accuracy, making it highly suitable for time-sensitive tasks such as road network analysis. In this thesis, the latest version of the YOLO architecture, YOLOv11 (Jocher & Qiu, 2024) and YOLOv12 (Tian et al., 2025a), were leveraged for the task of predicting optimal snowplow paths in road networks.

YOLOv11 and YOLOv12 represent the latest iterations in the YOLO object detection family, incorporating significant architectural enhancements for improved accuracy and efficiency. YOLOv11 introduced the C3k2 (Cross Stage Partial with kernel size 2) block, SPPF (Spatial Pyramid Pooling - Fast), and C2PSA (Convolutional block with Parallel Spatial Attention) components (Khanam & Hussain, 2024). These contribute to improving the model's performance in several ways such as enhanced feature extraction, spatial attention, and enhanced speed (Khanam & Hussain, 2024). YOLOv12 further refined the architecture by introducing A2 (Area Attention), R-ELAN (Residual Efficient Layer Aggregation Networks), and FlashAttention (Tian et al., 2025b). These modules enhance the use of attention mechanisms, model convergence speed, and memory efficiency (Tian et al., 2025b).

Using a fine-tuning approach, the training process began with taking pretrained weights from a large-scale dataset and then fine-tuning the model on a custom dataset composed of road network images with bounding boxes generated from MILP-optimized snowplow routes. Transfer learning was utilized soon after due to not seeing favorable results from fine-tuning. This was done by freezing the backbone layers of YOLOv12. It was trained for 100 epochs with a batch size of 32. The same approach was done with instance segmentation but YOLOv11 was used in that instance because v12 does not currently have pretrained weights for segmentation. The models were trained in Google Colab using T4 GPU.

## 3.4 Generative Adversarial Network to Optimize Snowplow Costs

After working on fine tuning YOLO models, one more computer vision approach was attempted to find snowplow routes. This was to use a GAN model to paint the optimal paths to an input image of a road network. The method used here is inspired by the article written by

Caffagni (2023). The approach of generating ground truth data is the same here as it was for creating the images for YOLO. This time, however, the output of the model is supposed to contain the optimal paths themselves.

### 3.4.1 Training Input and Target Images

As with the YOLO approach, the generated MILP solutions get converted into images that the GAN can learn from. The images are set to a similar size to what was used in YOLO but to be compatible with pix2pix, they are shrunk to 512 × 512. The node coordinates from each city are extracted and normalized to fit within the dimensions of the image. The task and depot nodes are depicted on the image as dark green and red dots on the path respectively. The optimal path generated by MILP is depicted on the target image because this is what the model is expected to predict. Example images are in Figure 10. The GAN takes the input image with the depot and task nodes and outputs and image with the optimal paths.

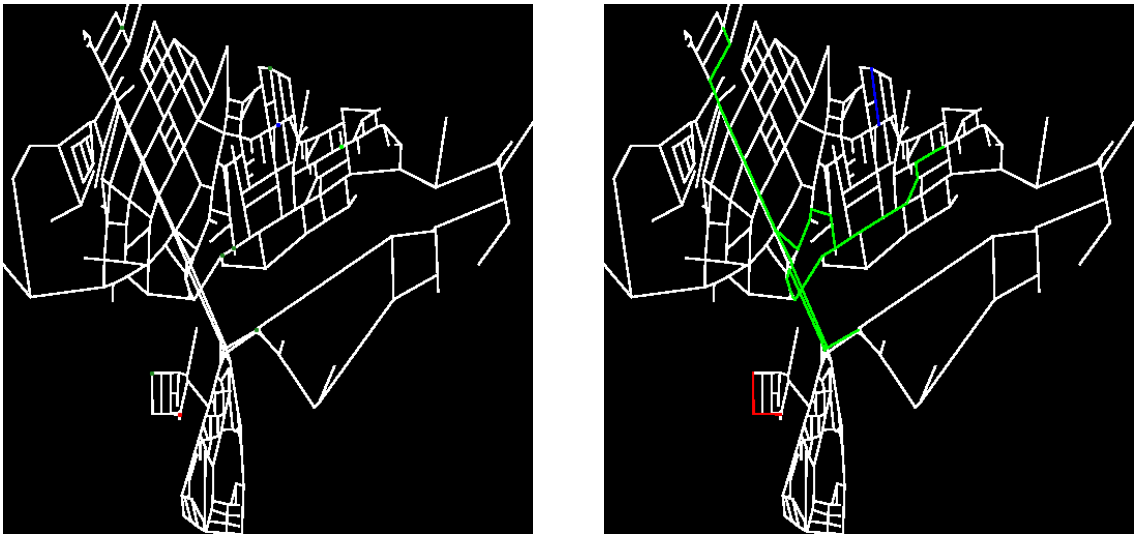


Figure 10. One of the image pairs used for training the GAN. The input image is on the left and the target image is on the right.

A second dataset was also created to train another GAN model. This one uses positional encodings. The idea is based on the work done by Caffagni (2023). It involves adding encodings to the second and third channels of the image. These encodings are made by creating 2 feature maps from a 2d gaussian function centered on the depots and task nodes, respectively. Equation 1 shows the function. σ is the standard deviation which controls the spread of the Gaussian. This was chosen to be 40, so that they would appear visible while not covering too much of the area. $(\mu_x, \mu_y)$ is the center of the Gaussian. (x, y) is the position of the pixel.

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}} \quad (1)$$

The purpose of the positional encodings is to help the image-to-image translation to recognize what the distances are from the important nodes to other pixels in the image. With this solution, the pixel values in the channel get smaller the further the path moves away from the node. The first channel of the image consists of the road network, the second consists of the depot nodes, the third consists of the task nodes. An example of an image pair is shown in Figure 11.

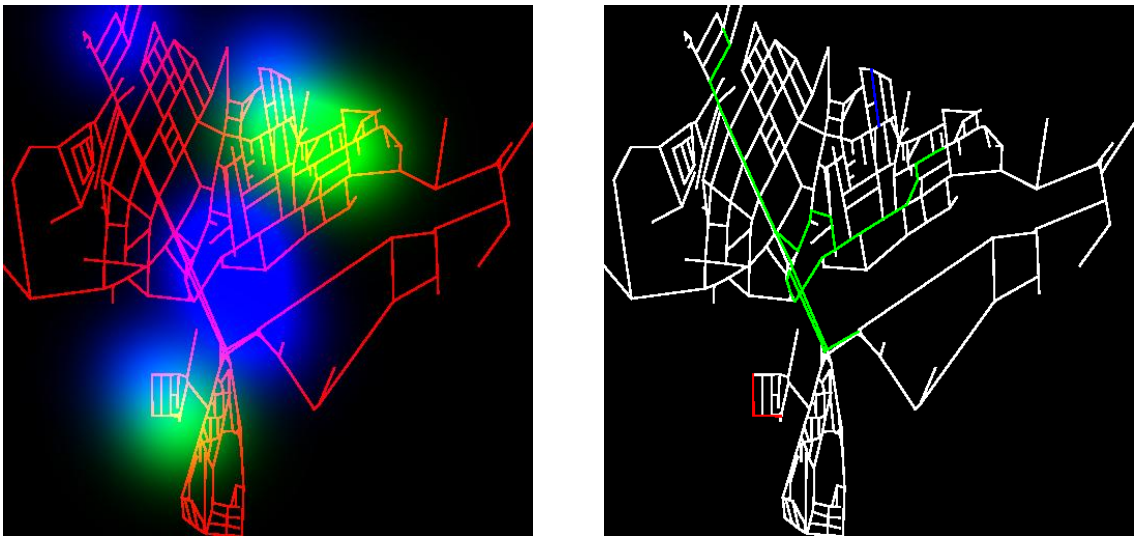


Figure 11. One of the image pairs used for training the GAN. The input is positionally encoded this time. The input image is on the left and the target image is on the right.

There is no classification involved with the GAN approach. That is why it cannot output a specific order for following the route. Color coding is used to differentiate between the paths of each truck though. Every vehicle has its own color and different overlaps between the routes have their own colors as well. The size of the training data is 3000 image pairs.

### 3.4.2 Converting Outputs to Optimal Paths

The approach to getting information about the optimal path from the output is similar to what it was before. The relevant graph nodes are extracted from the output image based on the node mapping used to create the input. This time there are no coordinates to work with though. This is where the color coding of the paths in the output comes into play.

The routes in the output have distinct colors that can be separated from the rest of the image. These can be used to extract the paths from the image as image masks. After that the masks can be utilized in the same manner as the segments in the previous YOLO approach. The mask

of each truck's path gets extracted from the image based on the RGB values, then the graph nodes covered by the mask are found. While the GAN model does not output an order to the nodes, there is still the input which gives the starting positions and the destinations. Dijkstra's algorithm can be used to start from the depot, then proceed moving along the mask boundaries until it has passed through the task nodes, then return to the beginning.

It should be noted that the solution of the GAN might sometimes output paths for more trucks than there are depots in the image. In these cases, the overabundant trucks routes are re-assigned to trucks that do exist. After this is completed, Dijkstra's algorithm can be run.

### 3.4.3 Implementation & Experiments

The model that is trained based on the generated input and target images, is pix2pix. pix2pix is a conditional Generative Adversarial Network (cGAN) designed for image-to-image translation tasks, where the goal is to learn a mapping from an input image to a corresponding output image (Isola et al., 2016). It consists of a generator that tries to produce realistic target images from inputs, and a discriminator that evaluates how close the generated images are to the ground truth. The generator follows a U-Net structure, which includes an encoder-decoder pipeline with skip connections between corresponding layers (Isola et al., 2016). The discriminator uses a PatchGAN architecture, which classifies whether each patch in the generated image is real or fake, focusing on local consistency rather than global realism (Isola et al., 2016).

In this thesis, pix2pix was experimented with to predict optimal snowplow paths by training the model on paired images. These are the input images consisting of road network maps with marked start and task points, and the target images showing the same maps with the optimal path segments derived from MILP overlaid in color. Two models were trained. One used the dataset of images without positional encodings, the other used the images with positional encodings.

The parameters used for the training include a batch size of 32 and a learning rate of 0.0002. The model was trained for 100 epochs in Google Colab using T4 GPU. The last 50 epochs were trained with learning rate decay. Code from the pix2pix repository was used to run the training and inference (Isola et al., 2016).

# 4 Results and Analysis

## 4.1 Performance of YOLO

Despite extensive experimentation with fine-tuning and transfer learning of the YOLOv12 and v11 architectures, the object detection and segmentation models were ultimately unable to learn to predict bounding boxes corresponding to the optimal snowplow paths. The training datasets were constructed using road network images with overlaid bounding boxes and segmentation masks generated from MILP-derived routes. The models were trained with pretrained weights. In the case of object detection, the size of the dataset is increased to 6000 from 2000. This was done through augmenting the data using techniques such as blur, rotation, flipping the image, and rescaling. This marginally improves the precision of the model, but the model still failed to produce meaningful detections on the validation and test sets. The training was performed with the nano versions of the YOLO models which are the ones with the smallest number of parameters. YOLO11x was tested for segmentation as well but this did not improve the results much.

The final metrics for the YOLO models are presented in Tables 3 and 4. The training graphs of both models are also included in Appendix II. The evaluation metrics of object detection show the performance of the bounding boxes. The segmentation ones show it for the segmentation masks as well as the bounding boxes surrounding them. Precision refers to the accuracy of the detected objects, indicating how many detections were correct (Ultralytics, 2025). Recall is the ability of the model to identify all instances of objects in the images (Ultralytics, 2025). mAP50 means the mean average precision calculated at an intersection over union (IoU) threshold of 0.50 (Ultralytics, 2025). mAP50-95 refers to the average of the mean average precision calculated at varying IoU thresholds, ranging from 0.50 to 0.95 (Ultralytics, 2025).

Table 3. Performance metrics of YOLOv12n object detection after performing transfer learning on it.

| **Object detection model YOLOv12n** | |
|---|---|
| Precision | 0.2622 |
| Recall | 0.2806 |
| mAP50 | 0.0686 |
| mAP50-95 | 0.0509 |

Table 4. Performance metrics of YOLOv11n instance segmentation after performing transfer learning on it.

| **Segmentation model YOLOv11n** | | | |
|---|---|---|---|
| **Box** | | **Mask** | |
| Precision | 0.0354 | Precision | 0.0167 |
| Recall | 0.0327 | Recall | 0.0145 |
| mAP50 | 0.0204 | mAP50 | 0.0091 |
| mAP50-95 | 0.0125 | mAP50-95 | 0.0042 |

These results indicate that the models were not able to generalize the relationship between road structure and route patterns from the training data. This failure can be attributed to several factors: the abstract and visually sparse nature of road network images, the absence of visually distinct features associated with the target routes, and the difficulty of learning spatial context in a task that requires understanding of graph connectivity rather than visual appearance alone. Due to the inability of the models to create predictions on test images, the performance cannot be compared to the MILP model.

## 4.2 Performance of pix2pix

Two pix2pix models were trained using paired input-output images to evaluate the potential of generative image-to-image translation for snowplow route prediction. The input images contained road networks with marked start and task points, while the target images contained the corresponding optimal snowplow paths, generated using a MILP solver. One model was trained with input images featuring paths and marked nodes. Another model was trained with input images that featured positional encodings instead of marked nodes. After training, the pix2pix model successfully learned to produce paths on road networks, although the predicted routes were often imprecise or disconnected compared to the ground truth. The training graphs are depicted in Appendix II.

Qualitatively, the outputs of both of the model variants look more or less like they should at first glance. The target images that the pix2pix models were trained on, had color coded roads painted over them according to which truck the path belongs to. This can be seen on the images produced by the trained models as well. The predicted colored paths have decent continuity, but sometimes short sections of colored roads can appear in random locations where they do

not need to be. Visually, it seems that both models have fulfilled their task to an extent. Visualizations of outputs from the models can be found in Figure 12.

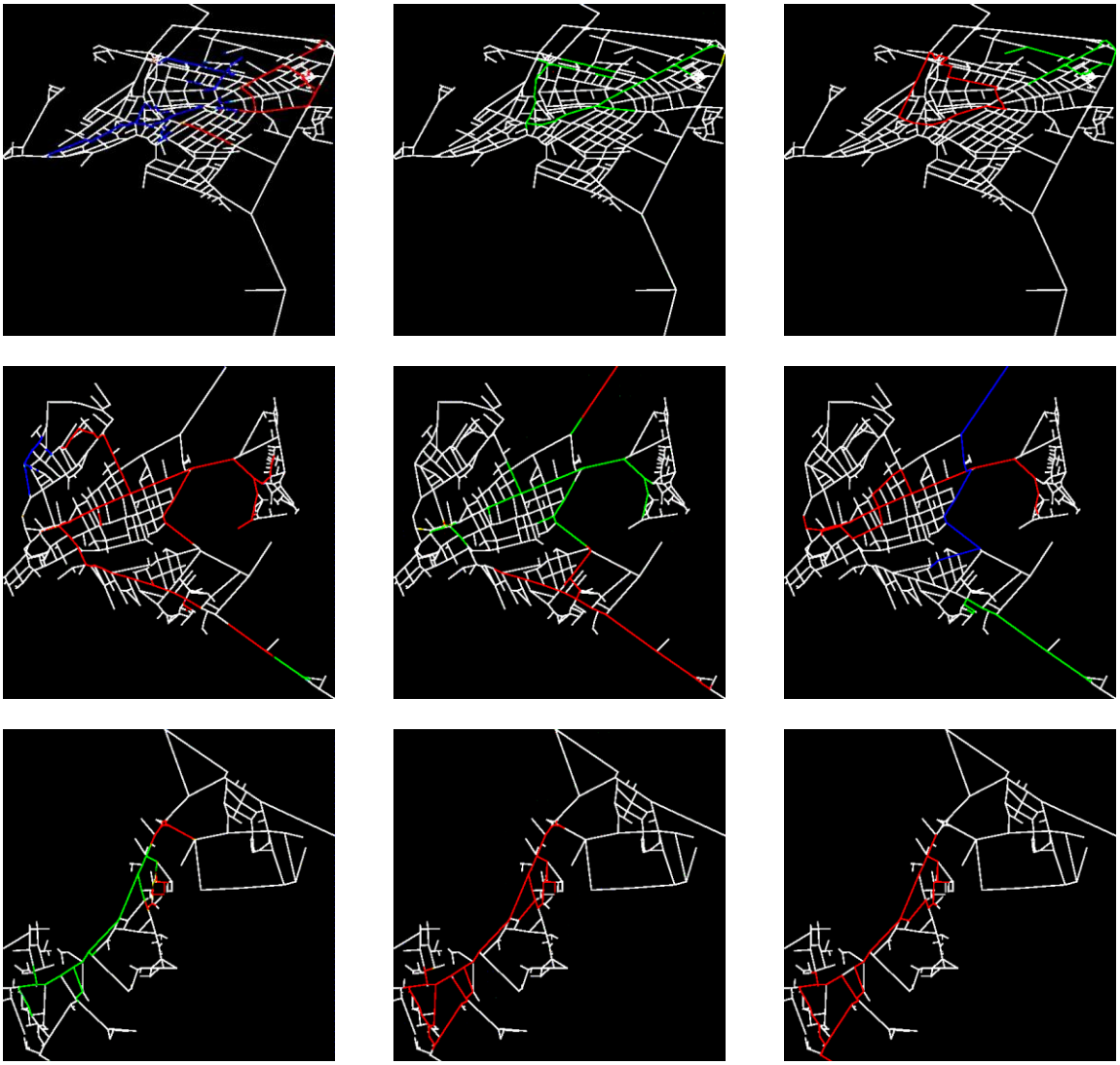


Figure 12. Examples of output images from the trained pix2pix models and the ground truth images. Images on the left are from the model without positional embeddings. Images in the middle are from the model with positional embeddings. The images on the left are the ground truth images generated from the MILP model.

Quantitative evaluation was conducted by comparing the predicted route masks to the ground truth using pixel-level metrics. The image pixel accuracy is calculated by taking the percentage of matching pixels between the ground truth image and the generated image. There are also two mask pixel accuracies. Mask refers to the colored paths that have been painted over the white road network. Since these are the important bits that is supposed to be predicted, it seemed like the appropriate metric to use for the evaluation of the models. One mask pixel accuracy is calculated by taking the percentage of how many pixels of the ground truth mask match with the generated mask. This metric only considers the shape of the mask. The other

mask pixel accuracy does the same comparison but also takes the accuracy of the colors in the mask into account. This way it is possible to see how accurate the generated image is, how accurate the overall shape of the predicted snowplow route is, and how accurately the route is divided up between the trucks. These metrics can be found in Table 5.

Table 5. Pixel-level performance metrics of the trained pix2pix models with and without positional embeddings.

| | **pix2pix without positional embeddings** | **pix2pix with positional embeddings** |
|---|---|---|
| Mean percentage of pixels in pix2pix output image that match the ground truth image | 93.83 | 96.71 |
| Mean percentage of pixels in pix2pix output mask that match the ground truth mask | 45.60 | 57.19 |
| Mean percentage of colored pixels in pix2pix output mask that match the colored ground truth mask | 15.76 | 28.18 |

Evaluation of the paths produced by pix2pix and Dijkstra's algorithm consists of several metrics. The mean, maximum, and minimum inference time is compared between the models to see if the cGAN helps with finding the snowplow routes faster. The traveled distances of the entire test set are combined and compared between the models. The same is done with the distances driven by each truck. The number of edges in the graph that have been used for each path are compared along with the number of edges that match with the solutions given by MILP. There is a comparison of the number of task nodes that were driven through to see how well the pix2pix models with Dijkstra's algorithm can solve the problem. Number of teleports indicates the number of times Dijkstra's pathfinding algorithm was not able to continue along the colored path provided by pix2pix and had to teleport to the next closest node that had not been traversed yet. All these metrics have been calculated across a test set of 200 different random snowplowing tasks using the 47 cities in Estonia. The testing was performed on a HP EliteBook 840 G10 Notebook PC equipped with a 13th Gen Intel(R) Core(TM) i5-1335U CPU, 16 GB of RAM, and 512 GB SSD.

Table 6. Performance metrics comparing the two pix2pix models trained with and without positional embeddings, as well as the implemented MILP.

| | **pix2pix without positional embeddings + Dijkstra's algorithm** | **pix2pix with positional embeddings + Dijkstra's algorithm** | **MILP** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Inference time in seconds (mean, max, min) | 0.93, 8.30, 0.38 | 1.07, 11.79, 0.44 | 16.54, 459.39, 0.17 |
| Total travel distance in meters | 1791802.27 | 2029773.80 | 2425246.75 |
| Number of graph edges in the routes | 19994 | 4961 | 20135 |
| Total travel distance per truck in meters | 1435356.10, 314436.85, 42009.32 | 1339966.45, 497981.36, 191825.99 | 1556543.10, 599745.90, 268957.75 |
| Number of graph edges that match the ones in the MILP solutions | 5620 (27.91%) | 627 (3.11%) | 20135 (100%) |
| Number of task nodes passed through | 908 (65.23%) | 703 (50.50%) | 1392 (100%) |
| Number of teleportations | 1954 | 4467 | 0 |

The results presented in Table 6 offer a comparative analysis between the MILP solution and the two pix2pix-based approaches for snowplow route prediction. The MILP model, as expected, achieved the highest route quality, covering all task nodes. However, this came at the cost of significant computational effort, with inference times reaching up to 459 seconds in the worst case. In contrast, the pix2pix models were substantially faster, with mean inference times under 1.1 seconds. Among them, the version without positional embeddings performed better in terms of route overlap with the MILP baseline (27.91% edge match vs. 3.11%) and task node coverage (65.23% vs. 50.50%). Interestingly, the model with positional embeddings exhibited more teleportation artifacts or disconnected route segments. This suggests that the positional cues may have introduced unintended spatial biases. Despite their reduced accuracy, the pix2pix models demonstrate some potential as fast approximators of optimal snowplow paths. However, their lower graph edge and task node coverage, along with the high number of teleportations, shows a significant trade-off between speed and route reliability.

# 5 Summary and Discussion

## 5.1 Feasibility of Computer Vision for Optimization of Snowplowing Costs

This study explored the use of computer vision deep learning models as alternatives to traditional optimization solvers for predicting snowplow routes. The work specifically experimented with YOLO object detection and segmentation and image-to-image translation with pix2pix. While classical MILP models remain the most accurate and constraint-compliant solutions, their computational demands and lack of scalability limit their practicality for large-scale problem solving. The pix2pix model trained in this thesis combined with the Dijkstra algorithm was able to approximate optimal routes on road networks to a limited extent, offering a more lightweight prediction mechanism. This demonstrates that, under specific conditions, computer vision models can support or even partially replace optimization solvers in route planning tasks.

Computer vision models cannot fully replace traditional solvers. Image-to-image translation with pix2pix only worked to a degree and its solutions did not match the ones provided by MILP well enough. YOLO, despite its effectiveness in general object detection and segmentation, failed to learn meaningful features in this highly abstract spatial prediction task. These results suggest that while computer vision can play a role in optimization, it can used as an augmentation technique, such as generating route priors or classifying regions, rather than replacing solvers outright in operational planning.

## 5.2 Challenges in Real-World Implementation

Translating the proposed deep learning models into real-world deployment presents several challenges. Even the best performing pix2pix implementation proposed in this thesis is rather simplistic for practical use. It does not consider the variables that are in many of the constraint-based models listed in Literature Review. This means that fuel consumption, salt replenishment, truck capacities are not taken into account which are quite important in the real world. There is also the matter of what are the actual characteristics of the roads being traversed. There can be one-way or two-way streets, two or three or more lanes. This determines if a road would need to be plowed once or several times.

## 5.3 Future Research Directions

Given the limitations of the proposed solution in this thesis, there are several areas that remain open for improvement and further exploration. One step toward real-world feasibility is incorporating more detailed road data into the image representations of the cities. Currently, the roads are represented as white lines with no differentiation between them. Integrating attributes like lane count and plowing priority by making them more visually distinct could help the deep learning model find more suitable paths.

Constraint-based models excel in part because they consider variables such as salt dome locations, vehicle fuel levels, and storage capacity. These inputs are currently absent from the computer vision deep learning models, limiting their ability to reflect realistic operational constraints. A possible direction for future research is to inject these features into the neural network as conditioning variables, influencing the prediction process alongside the image input.

The YOLO-based approach in this study struggled to converge. Future work could focus on increasing both the volume and variability of the training samples. Additionally, a more nuanced training regime could be implemented by incorporating some sort of path-based loss. That way the model could evaluate how well the predicted bounding boxes align with the actual route geometry. This would encourage the model to focus on continuity and connectivity, key characteristics of valid routes.

Although pix2pix models showed better results compared to YOLO, they could be capable of even more. The model that did not use positional embeddings performed quite poorly at first but the last 14 epochs made a big difference in its capabilities. Since the training took a long time, 100 epochs were spared for each pix2pix model. The output could be significantly improved by extending the training even further.

# 6 Conclusion

This thesis set out to explore alternative methods for optimizing snowplow routes on road networks in Estonia, with a particular focus on reducing computational costs and improving the feasibility of large-scale deployment. Traditional optimization techniques, like mixed integer linear programming, offer precise and constraint-aware solutions to snowplow routing problems. However, their computational complexity and limited scalability pose challenges when sufficient computational resources are not available.

To address these limitations, this research investigated deep learning approaches that approximate optimal routes using computer vision models. The thesis implemented a MILP model to generate high-quality snowplow routes and used the resulting solutions to train two YOLO models for object detection and segmentation, and two pix2pix models for image-to-image translation. While YOLO failed to converge on meaningful detections due to the abstract nature of the input data and limited visual signal, the pix2pix model was able to learn and generate approximate path structures, demonstrating the viability of generative models for route prediction.

The experimental results show that, although computer vision neural models cannot yet replace traditional solvers in precision or reliability, they offer a fast alternative for approximate path estimation. This capability could be valuable in scenarios where timely decisions on route planning are more critical than strict optimality.

In addition to the technical findings, the thesis highlights challenges and opportunities for future research. These include better integration of road data, more nuanced training regime, and incorporation of optimization variables into vision models. Ultimately, the fusion of linear programming and machine learning shows promise as a direction for developing scalable and time efficient tools for winter road maintenance planning.

# References


Boeing, G. (2025). Modeling and analyzing urban networks and amenities with OSMNX. *Geographical Analysis*. https://doi.org/10.1111/gean.70009

Caffagni, D. (2023, January 22). 2D path planning with CNN. *Towards AI*. https://towardsai.net/p/l/2d-path-planning-with-cnn

*GAN Variations*. (2025, February 26). Google for Developers. Retrieved May 13, 2025, from https://developers.google.com/machine-learning/gan/applications

Hajibabai, L., Nourbakhsh, S. M., Ouyang, Y., & Peng, F. (2014). Network Routing of Snowplow Trucks with Resource Replenishment and Plowing Priorities. *Transportation Research Record Journal of the Transportation Research Board*, *2440*(1), 16–25. https://doi.org/10.3141/2440-03

*Integer Linear Programming*. (2021, February 27). APMonitor. Retrieved May 13, 2025, from https://apmonitor.com/wiki/index.php/Main/IntegerProgramming

Isola, P., Zhu, J., Zhou, T., & Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1611.07004

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8* (8.0.0) [Software]. https://github.com/ultralytics/ultralytics

Jocher, G., & Qiu, J. (2024). *Ultralytics YOLO11* (11.0.0) [Software]. https://github.com/ultralytics/ultralytics

Joseph, R., Santosh, D., Ross, G., & Ali, F. (2015). You only look once: Unified, Real-Time Object Detection. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1506.02640

Joshi, C. K., Laurent, T., & Bresson, X. (2019). An efficient graph convolutional network technique for the travelling salesman problem. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1906.01227

Khanam, R., & Hussain, M. (2024). YOLOV11: An overview of the key architectural enhancements. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2410.17725

Kinable, J., Van Hoeve, W., & Smith, S. F. (2016). Optimization models for a Real-World snow plow routing problem. In *Lecture notes in computer science* (pp. 229–245). https://doi.org/10.1007/978-3-319-33954-2_17

Lee, T., & Kim, M. (2024). RL-MILP Solver: A Reinforcement Learning Approach for Solving Mixed-Integer Linear Programs with Graph Neural Networks. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2411.19517

*Linear Optimization*. (2024, August 28). Google for Developers. Retrieved May 13, 2025, from https://developers.google.com/optimization/lp

Mäekivi, M. (2024, December 18). Rahanappuse tõttu võib riigiteede lumekoristus tulevikus venima hakata. *ERR*. https://www.err.ee/1609554508/rahanappuse-tottu-voib-riigiteede-lumekoristus-tulevikus-venima-hakata

*Mixed integer nonlinear programming*. (2021, August 31). APMonitor. Retrieved May 13, 2025, from https://apmonitor.com/wiki/index.php/Main/IntegerBinaryVariables

Nazari, M., Oroojlooy, A., Snyder, L., V., & Takáč, M. (2018). Reinforcement learning for solving the vehicle routing problem. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1802.04240

Nguyen, P. H., & Tran, D. (2024). Constraint-Based snowplow optimization model for winter maintenance operations. *Transportation Research Part a Policy and Practice*, *179*, 103911. https://doi.org/10.1016/j.tra.2023.103911

OpenStreetMap contributors. (2025). *OpenStreetMap*. Retrieved May 10, 2025, from https://www.openstreetmap.org/

*Overview of constraint programming*. (n.d.). indexIBM® Decision Optimization CPLEX® Modeling for Python (DOcplex) V2.25 Documentation. Retrieved May 13, 2025, from https://ibmdecisionoptimization.github.io/docplex-doc/cp.html

*Overview of GAN Structure*. (2025, February 26). Google for Developers. Retrieved May 13, 2025, from https://developers.google.com/machine-learning/gan/gan_structure

Rao, T., Mitra, S., & Zollweg, J. (2011). Snow-plow route planning using AI search. *2011 IEEE International Conference on Systems, Man, and Cybernetics*. https://doi.org/10.1109/icsmc.2011.6084095

Rasul, A., Seo, J., Xu, S., Kwon, T. J., MacLean, J., & Brown, C. (2021). Snowplow route optimization using Chinese Postman Problem and Tabu search algorithm. *Proceedings of the 38th ISARC*. https://doi.org/10.22260/isarc2021/0056

Sartori, D., Zou, D., Pei, L., & Yu, W. (2021). CNN-based path planning on a map. *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1331–1338. https://doi.org/10.1109/robio54168.2021.9739331

Taboga, M. (2021). *Complementary subspace*. Lectures on Matrix Algebra. Retrieved May 10, 2025, from https://www.statlect.com/matrix-algebra/complementary-subspace

Tian, Y., Ye, Q., & Doermann, D. (2025a). *YOLOv12: Attention-Centric Real-Time Object Detectors* [Software]. https://github.com/sunsmarterjie/yolov12

Tian, Y., Ye, Q., & Doermann, D. (2025b). YOLOV12: Attention-Centric Real-Time Object Detectors. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2502.12524

Ultralytics. (2025, March 30). *YOLO Performance Metrics*. https://docs.ultralytics.com/guides/yolo-performance-metrics/#object-detection-metrics

*Vehicle routing*. (2025, January 22). Google for Developers. Retrieved May 13, 2025, from https://developers.google.com/optimization/routing

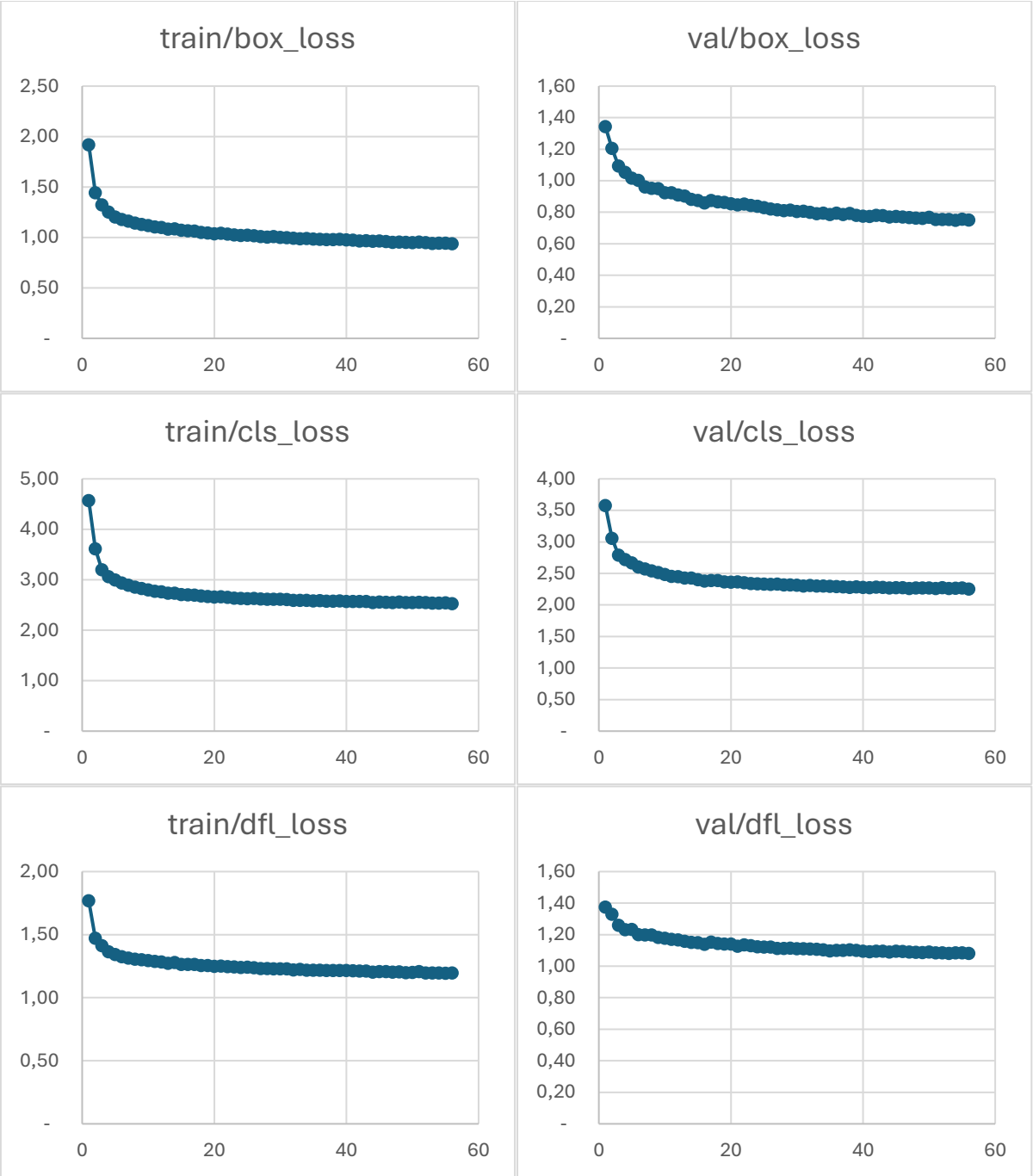
What are Convolutional Neural Networks? (n.d.). *Think | IBM*. Retrieved May 13, 2025, from https://www.ibm.com/think/topics/convolutional-neural-networks

# Appendices

## I. Glossary

| Term | Definition |
|---|---|
| Linear Programming (LP) | It refers to computing the best solution to a problem modeled as a set of linear relationships (*Linear Optimization*, 2024). |
| Integer Linear Programming (ILP) | Optimization problem where the variables are integer values and the objective function and equations are linear (*Integer Linear Programming*, 2021). |
| Mixed Integer Linear Programming (MILP) | Optimization problem that has continuous and integer variables and the objective function and equations are linear (*Integer Linear Programming*, 2021). |
| Mixed Integer Programming (MIP) | Optimization problem where some of the variables are continuous and some are discrete (*Mixed Integer Nonlinear Programming*, 2021). |
| Constraint Programming (CP) | Approach which is used to find solutions to scheduling and combinatorial optimization problems and is based primarily on computer science fundamentals, such as logic programming and graph theory, in contrast to mathematical programming, which is based on numerical linear algebra (*Overview of Constraint Programming*, n.d.). |
| Traveling Salesman Problem (TSP) | Optimization task where the goal is to find the shortest route for a salesperson who needs to visit customers at different locations and return to the starting point (*Vehicle Routing*, 2025). |
| Vehicle Routing Problem (VRP) | Optimization task where the goal is to find the best routes for a fleet of vehicles visiting a set of locations (*Vehicle Routing*, 2025). |
| Convolutional Neural Network (CNN) | Neural network that is typically used for image recognition tasks and where the three main types of layers are the convolutional, pooling, and fully-connected layer ("What Are Convolutional Neural Networks?", n.d.). |
| Generative Adversarial Network (GAN) | Network that consists of a generator that learns to generate plausible data, and a discriminator that learns to distinguish the generator's fake data from real data and penalizes the generator for producing implausible results (*Overview of GAN Structure*, 2025). |
| Conditional Generative Adversarial Network (cGAN) | GAN that trains on a labeled data set and allows to specify the label for each generated instance (*GAN Variations*, 2025). |

## II. Training Graphs



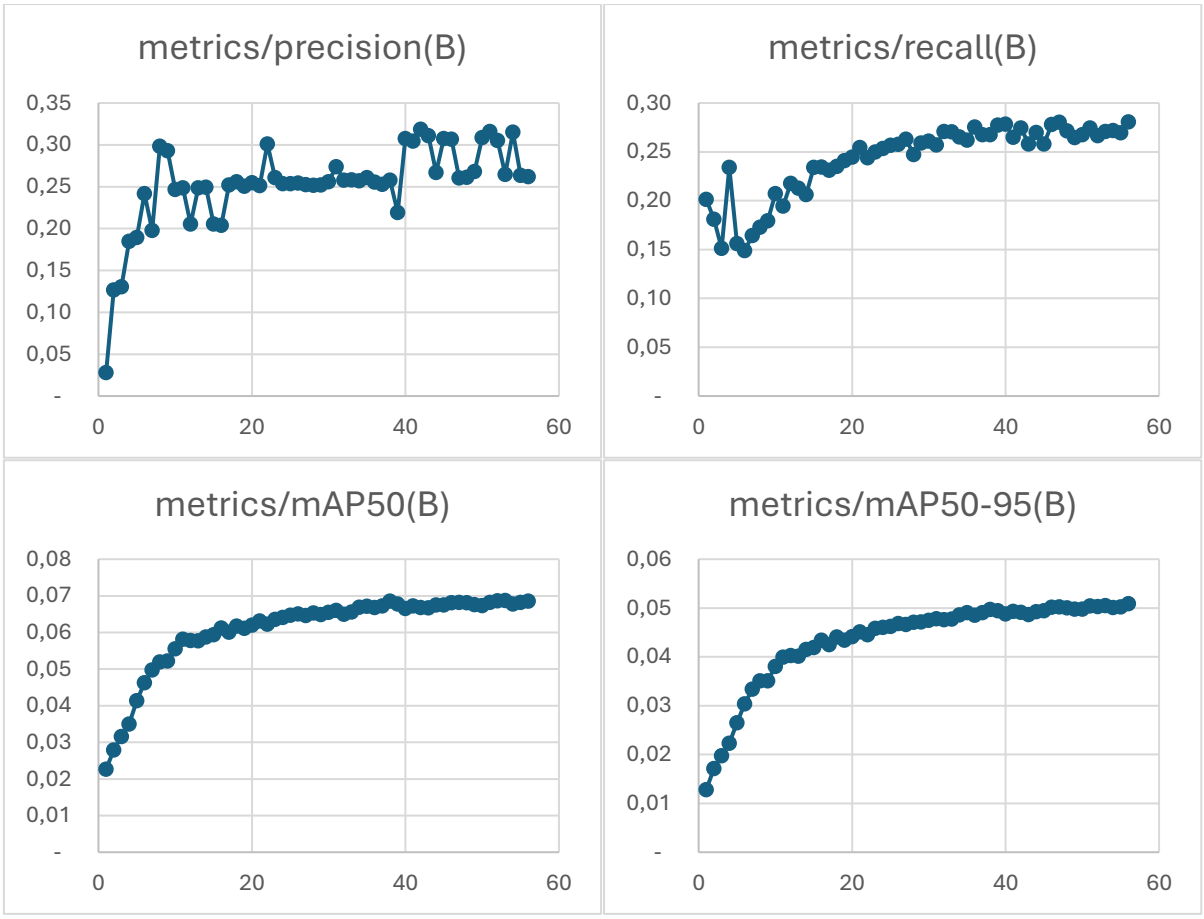
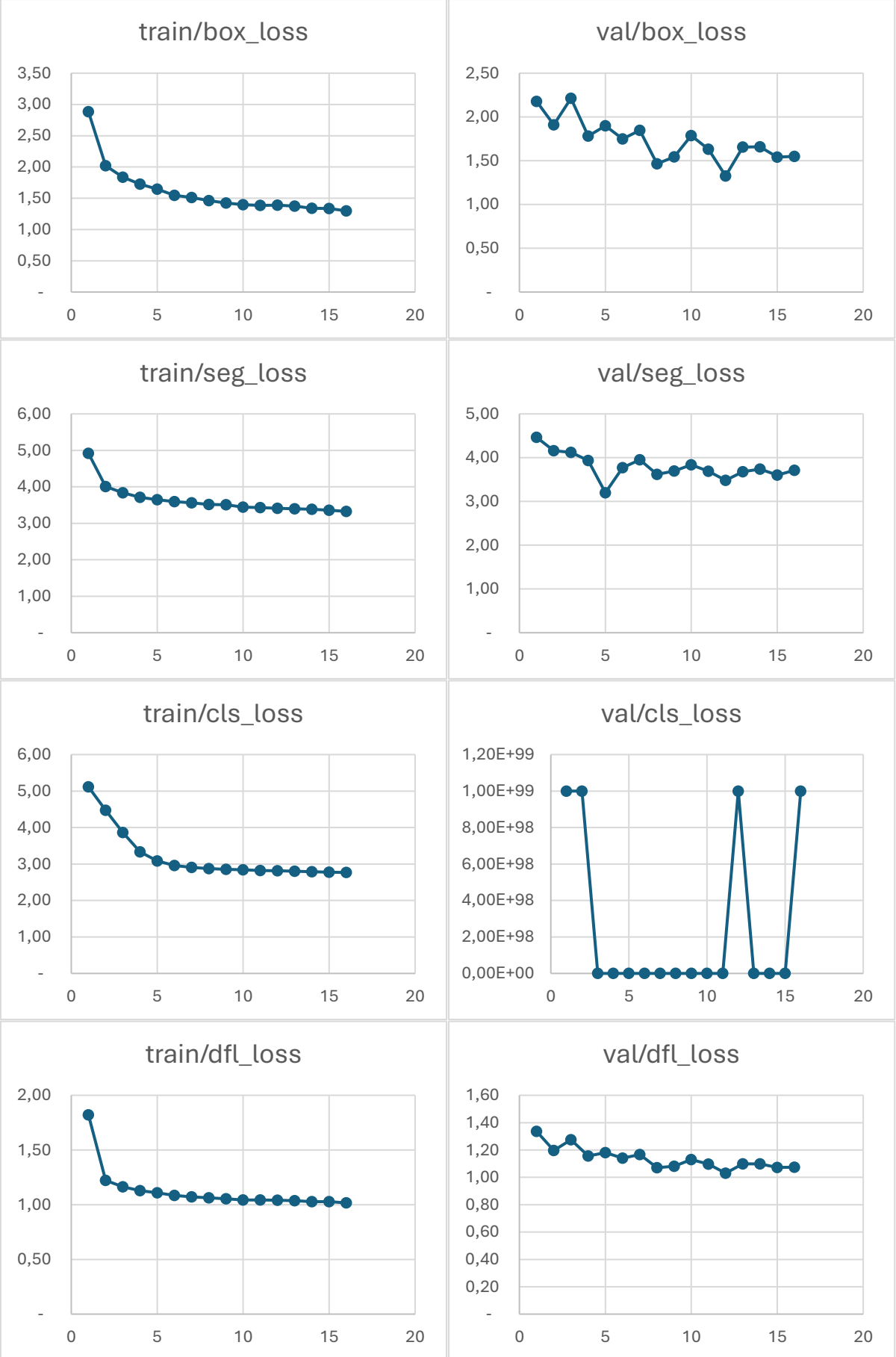



Figure 13. YOLOv12n object detection training graphs.


train/box_loss
3,50
3,00
2,50
2,00
1,50
1,00
0,50
-
0
5
10
15
20
val/box_loss
2,50
2,00
1,50
1,00
0,50
-
0
5
10
15
20
train/seg_loss
6,00
5,00
4,00
3,00
2,00
1,00
-
0
5
10
15
20
val/seg_loss
5,00
4,00
3,00
2,00
1,00
-
0
5
10
15
20
train/cls_loss
6,00
5,00
4,00
3,00
2,00
1,00
-
0
5
10
15
20
val/cls_loss
1,20E+99
1,00E+99
8,00E+98
6,00E+98
4,00E+98
2,00E+98
0,00E+00
0
5
10
15
20
train/dfl_loss
2,00
1,50
1,00
0,50
-
0
5
10
15
20
val/dfl_loss
1,60
1,40
1,20
1,00
0,80
0,60
0,40
0,20
-
0
5
10
15
20

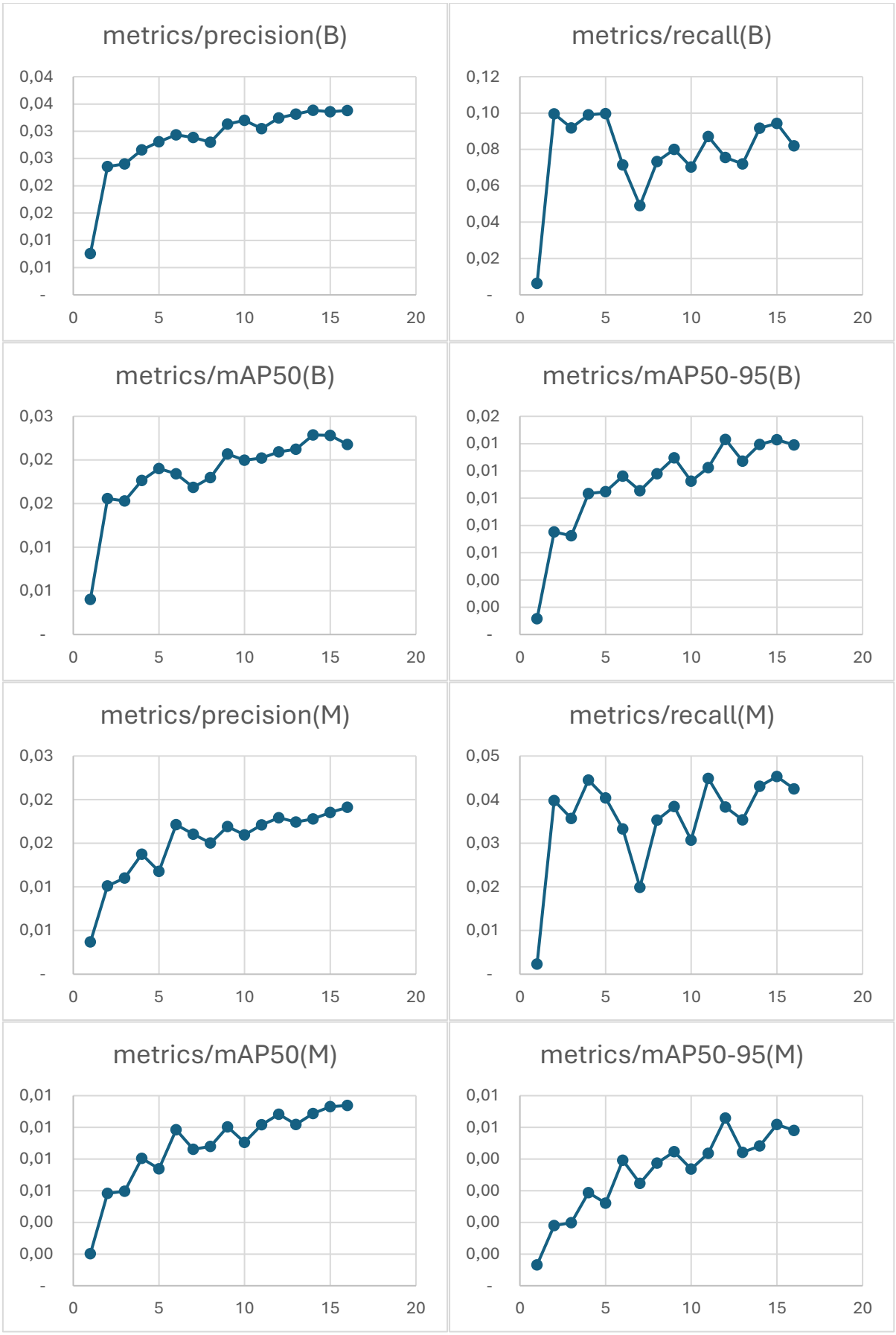



Figure 14. YOLOv11n instance segmentation training graphs.

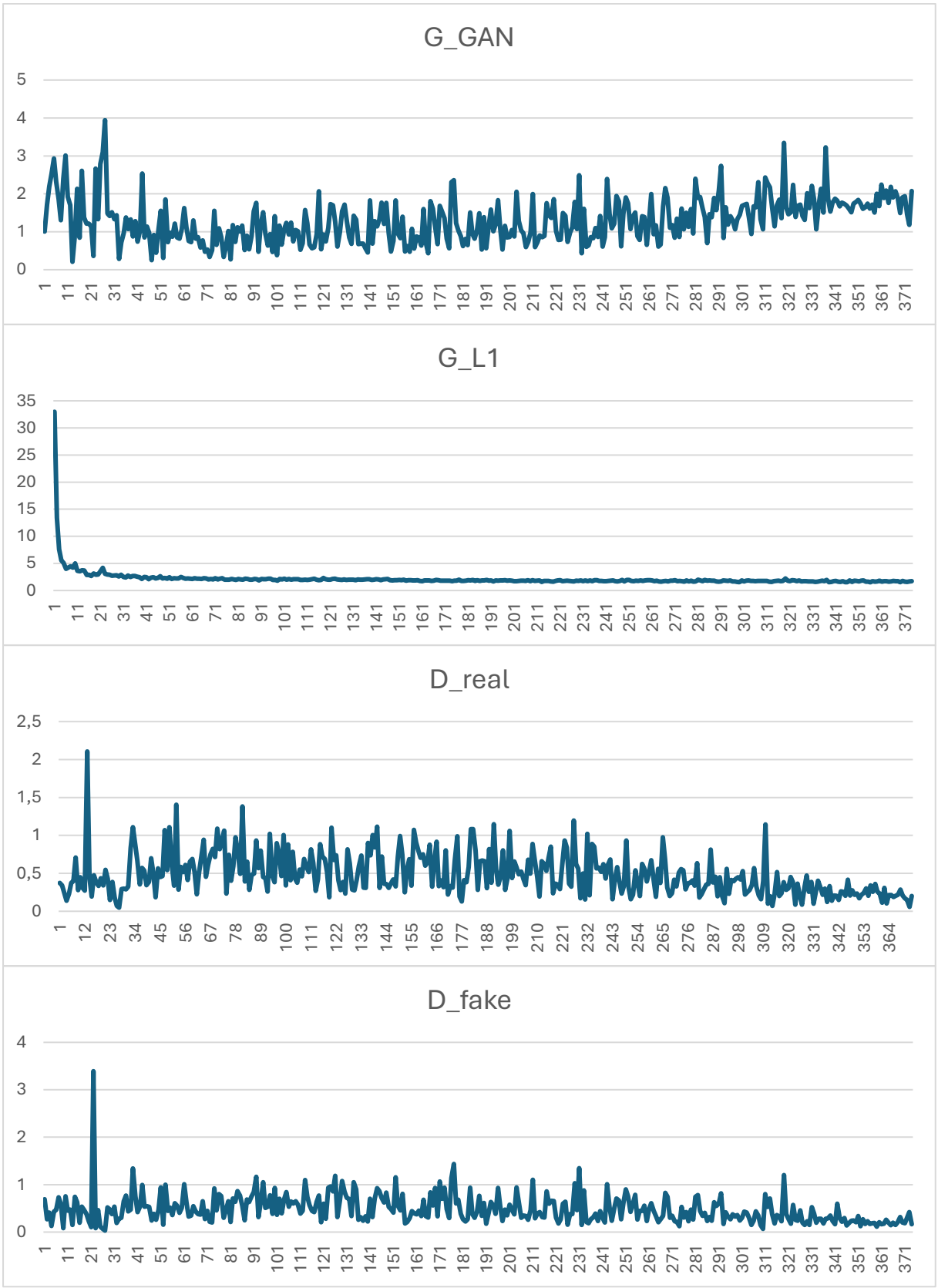



Figure 15. pix2pix training graphs without positional embeddings.

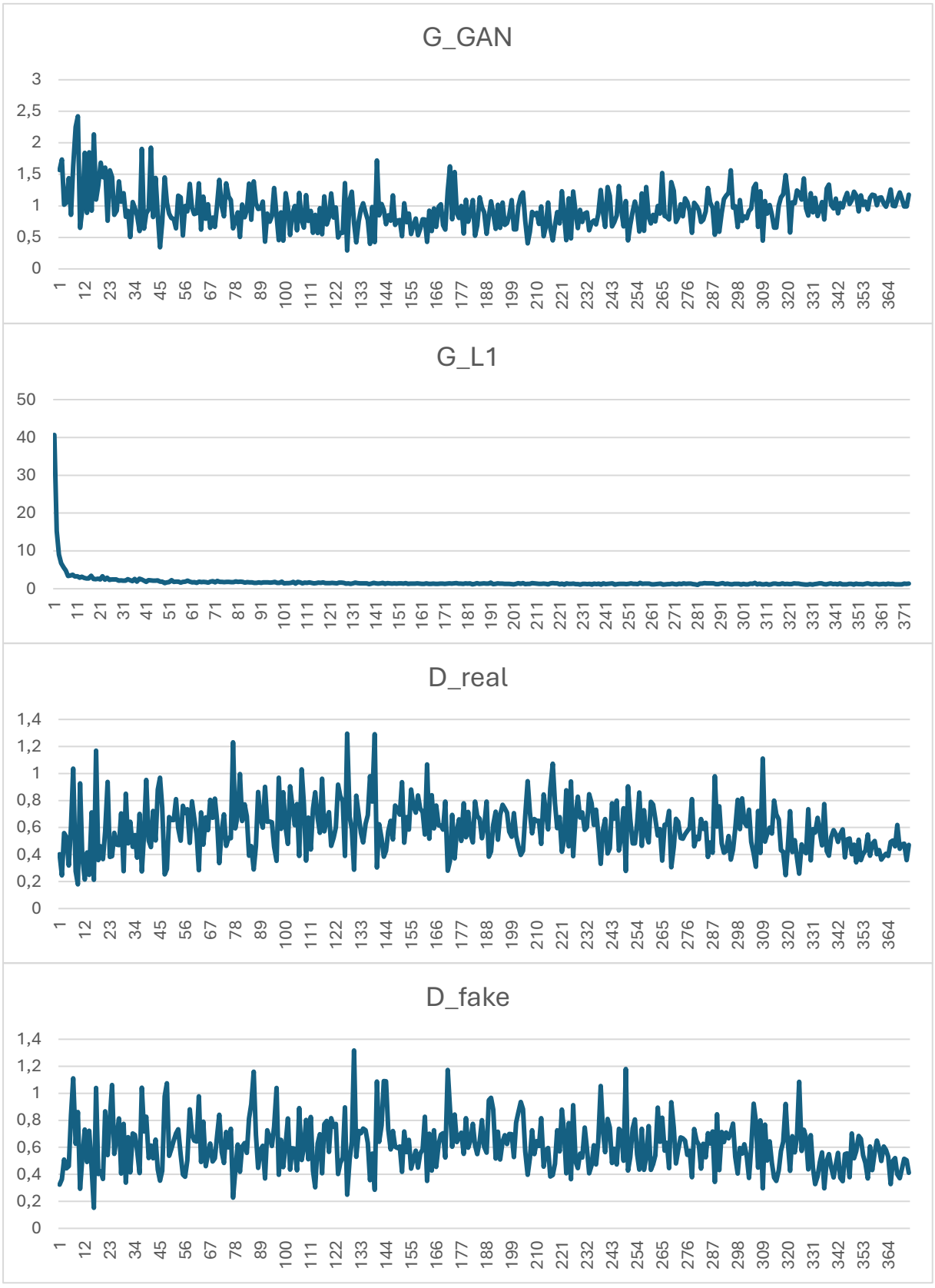



Figure 16. pix2pix training graphs with positional embeddings.

# License

**Non-exclusive licence to reproduce thesis and make thesis public**

I, **Aap Vare**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the Dspace digital archives until the expiry of the term of copyright,

   **Vision-Based Optimization for Snowplowing on Estonian Roads**,

   supervised by Kallol Roy and Jaan Übi.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Aap Vare

15/05/2025