Vrije Universiteit Amsterdam

KLM Ground Services

Master Thesis

Trajectory Prediction in a Digital Twin for Autonomous Ground Service Operations

Author: Bryan van Ingen (2685432)

1st supervisor: Rob van der Mei
daily supervisor: Peter Huisman (KLM Ground Services)
2nd reader: Paulo Jorge de Andrade Serra

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

April 29, 2024

"I am the master of my fate, I am the captain of my soul"
from Invictus, *by William Ernest Henley*

Abstract

Ground Services are a vital part of the operations of any airport. In order to make these operations more efficient, more safe, and more sustainable, KLM has started an autonomous operations project, which aims to automate as much of the ground service operations as possible. To this end, it is crucial to gain insights into the movement of the traffic around the airport grounds. To aid in this endeavour, a digital twin is being developed that can track all ground service equipment vehicles. This thesis' aim is to introduce methodologies that can be used to predict the location of the vehicles in this digital twin in a future state.

To this end, multiple sequence-to-sequence encoder-decoder model architectures were trained along with a transformer model and an mixture density network model. These models were trained on different scenarios to find the impact of changes to the experiment setups

Through this, we learned that the behaviour in movement for different types of vehicles used in ground service operation were too diverse to capture the information at once in a single model. We also studied the effect of changing the input data used for our models. We found that the performance of models could be improved by decreasing the input sequence length, and by selecting data to ensure a higher rate of diversity in the training data.

Unfortunately, we were unable to find a model structure that was able to predict trajectories that can be seen as a realistic representation of the true movements happening on the airport aprons using the proposed methods.

Acknowledgements

I would like to thank all colleagues of the KLM Ground Services business development department for welcoming me into their department and offering their insights and help in the moments I needed it. I especially want to express my gratitude towards Peter, my direct company supervisor, for offering me this assignment and giving me the chance to complete my Master in a company of such stature as KLM. I am also grateful for the trust, autonomy and resources I was offered during my time here.

Another special thanks goes out towards my VU supervisor Rob for guiding me throughout my internship and especially for making sure that I did not try to rush through important beginning steps in the research and making sure I took every step one at the time.

Contents

1

Introduction

To make sure that all travellers are able to fly according to schedule, it is of the utmost importance to prepare each aircraft before a flight. Ground service operations encompass all the actions upon which a timely departure and a smooth transfer of a passenger is dependent. These actions range from checking-in and boarding to unloading, to loading and refueling, ground services is responsible for the perfect pit stop for every aircraft. Each process that is part of the ground service operations is enabled by the use of specialized vehicles and well trained personnel.

There is continuous growth in the demand for air travel, but there is also difficulty in finding the personnel necessary to be able to accommodate this demand. To still be able to facilitate all passengers and ensure smooth and safe travel, the Royal Dutch Airlines (KLM) is starting to look into slowly automating as much of the ground service operations as possible. If this is done successfully, this can result in safer, more efficient, and cheaper handling of the necessary processes.

1.1 Problem statement

One of the first steps in this Autonomous Operations project is to get more insights into what is happening on the airport aprons on a daily basis. Where are all vehicles located? Is there enough equipment available? Which vehicles should be assigned to do a given task? These questions and more are important in order to run an automated airport smoothly. Thus, a digital twin is in production to track all activity around the Schiphol Airport premises.

In its current state, the twin is able to provide insights into past behavior of the vehicles, and track activity of the ground service equipment in real time and is mainly used as tool for visualization. In the setting of the airport, where there is limited direct communication between vehicles, real-time prediction of vehicle traffic may help alleviating some pressure on the traffic demands due to operations. Thus, knowing where vehicles may be in the future can help with planning, assigning vehicles to tasks, optimizing traffic flows and in the future perhaps with fully automating the ground service operations.

Machine learning can play a pivotal role in advancing the digital twin. By using specialized algorithms to their advantage, we may be able to move beyond simple visualization and explore more predictive analyses. Loads of data is being collected every day, and machine learning is a useful tool to derive underlying information from this data. For instance, machine learning models can analyze historical vehicle behavior and real-time data from ground service equipment (GSE) to forecast future traffic patterns at airports. A simple first step in this would be to use machine learning algorithms to predict the trajectories of the many vehicles in the system in order to gain extra information about the busyness of the environment and to locate possible bottleneck locations.

1.2 Information about the host organization

KLM is mainly a provider of air transportation services. The company is involved in various operations including transporting passengers and cargo, maintaining airframes, engines, and components, as well as operating charters and scheduled services with affordable fares. Founded in 1919, KLM is the oldest, still operating, airline company in the world and yearly they carry millions of passengers to locations all over the world with a fleet of over 100 aircraft. KLM is actively involved in finding more sustainable and efficient methods to run its operations.

Too show KLM possibilities the future that may hold, the KLM Ground Service Business Development department aims to demonstrate innovative ideas and technologies by means of various projects.

1.3 Research objective

The Autonomous Operations project of which this research is part is primarily used as a demonstration of what will be needed to achieve fully automated ground service operations. Therefore, the focus of this research is not in delivering the most accurate model to be used, but to show where there are possibilities that can be exhausted to further the developments towards autonomous operations.

Thus the objective of this research is to provide possible answers to the following question: *How can machine learning aid in predicting trajectories of multiple GSE vehicles in order to forecast the future state of an airport environment?*

In this report, we will look into how the provided data can be used to make predictions that are usable for prediction performed in the existing digital twin environment. This will be done by exploring different experimental setups and training different models which should be capable of capturing information from sequential data. In the end, conclusions will be drawn about the data needed to solve the proposed problem and possible approaches for solving the problem.

1.4 Structure of the report

This report consists of eight chapters followed by a list of references used to support the research. Three appendices containing extra information can be found at the end of the report

In Chapter 1 the reader will be introduced to the research topic and the problem that will be tackled in the report. Then, Chapter 2 will provide extra background information into some of the leading background topics of this research. Chapter 3 will discuss the methodologies used previously in research with similar goals as this project. In Chapter 4, the data that is used for this project will be described and explored. Following, the methodologies applied during this research are explained in Chapter 5. The resulting outcomes of this research are shown in Chapter 6 and will be discussed in Chapter 7. Finally, Chapter 8 will complete this report by summarizing the most important findings and offering ideas for future research.

2

Background

In this chapter, some background information about airport ground services and the underlying project this research is part of, will be explained.

2.1 Airport ground services

Airport ground services are an important aspect of the logistics of aviation. The ground handling operations oversees a diverse list of services that are aimed toward facilitating smooth, efficient and safe movement of aircrafts, passengers, and cargo around the airport. Ground services are divided into various tasks, each of which are performed by specially trained teams. The tasks that fall under ground services include ramp handling on the apron where the aircrafts are parked, baggage handling, aircraft servicing, cargo handling and passenger handling. All these tasks are performed under strict time pressure. This pressure comes from the necessity to meet tight turnaround times between flights, and thus requires seamless coordination between different forces to meet departure times and prevent flight delays and the disruption of flight schedules. Figure 2.1 gives a partial overview of tasks that have to be performed between the landing of an airplane and the next departure. Ground handling is responsible for the execution of all these tasks.

Each of these tasks comes with its own safety hazards and risks, as it entails working close to moving aircraft and heavy equipment. All personnel undergo training before they are permitted to work on the aprons, equipping them with the necessary knowledge and skills to mitigate these risks. Training programs teach about safety protocols, emergency procedures, and the proper use of protective equipment to ensure the safety of ground handling staff at all times. Additionally, specialized vehicles and equipment are employed

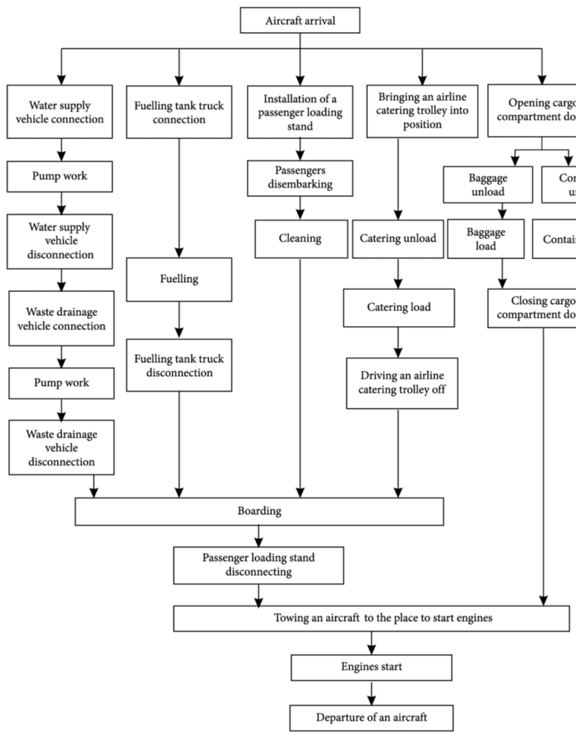

Aircraft arrival
Water supply vehicle connection
Fuelling tank truck connection
Installation of a passenger loading stand
Bringing an airline catering trolley into position
Opening cargo compartment doors
Pump work
Passengers disembarking
Baggage unload
Containers unload
Water supply vehicle disconnection
Cleaning
Catering unload
Baggage load
Containers load
Fuelling
Catering load
Closing cargo compartment doors
Waste drainage vehicle connection
Fuelling tank truck disconnection
Driving an airline catering trolley off
Pump work
Waste drainage vehicle disconnection
Boarding
Passenger loading stand disconnecting
Towing an aircraft to the place to start engines
Engines start
Departure of an aircraft

Figure 2.1: A schematic of a part of the turnaround process of an aircraft (23).

for each task, designed to enhance efficiency and safety. A list of the different vehicles that will be discussed in this project can be found in Appendix A.

2.2 Autonomous operations

In 2050 Amsterdam Schiphol airport aims to be the most sustainable airport in the world (3). Simultaneously, they aim to enhance the capacity utilization of the airports while ensuring safety. The sustainable and autonomous transformation of all vehicles and associated processes on the airport will contribute significantly to this objective.

Autonomous operations for vehicles on airport grounds are an upcoming area in the development in the airport industry. As mentioned, ground services include a large range of tasks which have to be performed in succession to ensure a fast turnaround time and a successful flight. Automating tasks underneath the wings of an airplane can offer many benefits such as safety, cost reduction and reduced emissions (2).

Airplanes are mostly similar, and have standardised configuration (30, 37). This makes it so that the same equipment can be used for many different airplanes, and thus making automation a feasible goal in terms of equipment and an interesting field of research. However, automation of the full ground handling operations is not a task as simple as just having the equipment. The American Federal Aviation Administration (FAA) has also included autonomous operations in their research landscape for the upcoming years (1). However, they also admit that there are still plenty challenges in this field. In their research landscape they aim to research multiple factors including, but not limited to:

- Infrastructure and methodologies to facilitate the operations of autonomous ground vehicles (such as navigation, charging, or refueling).
- Examination of the influence of weather conditions on equipment performance.
- Development of training protocols and guidelines for flight crews, dealing with handling emergency or unforeseen circumstances.
- Implementation of measures for the security and identification of autonomous vehicles.

There is also the problem that tasks need to be allocated to different vehicles and planned optimally in order to keep the operations running smoothly and efficiently. Research in this field is already advancing. For example, Chen et al. (9) already introduced a framework that merges task assignment and route planning for automating ground handling operations from a multi-agent viewpoint.

It is safe to say that in order to successfully automate the ground services operations, insights in the movements of the vehicles is necessary. For that, a digital twin is currently being developed.

2.3 Digital twins

A digital twin is a virtual representation of a real-world product, system, or process (its physical counterpart), used for a multitude of tasks. These tasks include simulation, testing, monitoring, and maintenance. In its initial introduction by Michael Grieves, the digital twin was designed to be used with Product Lifecycle Management and coexist with the physical entity it represents throughout its entire life cycle from creation to disposal (13). Nowadays, the intended use of a digital twin is to provide users with better ways to gain a better understanding of increasingly sophisticated and complicated systems. This will results in a reduction of failures and problems in the physical system. This in turn reduces expenses, time, and also risks to everyone involved with the physical system (14).

The digital twin being developed at KLM can be more accurately described as a Digital Twin Environment (DTE). A DTE is a digital space where besides topographic objects, specific objects (like GSE vehicles) are also represented. Furthermore, in a DTE, temporal relations between objects over time are also shown (22). DTEs serve a variety of purposes. This includes prediction and interrogation. The predictive purposes are evident when a digital twin is used to predict future behavior and performance of the system. A fully functional twin can take actual components and the history of said components in the system to predict the behavior of the system. By analyzing and combining data from multiple instances, one would be able to provide a range of possible future states. The interrogative purposes cover the idea that, since a digital twin is connected to its physical counterpart, regardless of the location of the counterpart, one is able to gather information about the system's current state and performance remotely. Data collected from multiple instances in a DTE may relate to different trends or patterns which may in turn help with the predictive objectives for which the twin was implemented (14).

3

Related Work

This chapter aims to explore what has been done before in research and projects with similar goals in trajectory prediction.

Trajectory prediction can be defined as the act of forecasting future states of dynamic agents in a system given their current and past states. This definition can be translated to two main different focal points in research: (1) vision based motion prediction and, (2) a more top-down approach with next location prediction in a Vehicle-to-everything (V2X) network. In the vision based approaches, computer vision is often used to detect and analyze surroundings of a vehicle to prevent dangerous situation and to allow more efficient routing (33). These approaches often use CNN-based models to extract features from images and then use different techniques such as RNN-based methods (24), generative methods (12) or statistical approaches such as Hidden Markov Models(8), to predict future states of their environment.

Even though these methods may give insights in how trajectory prediction is applied in general, the methodologies used are not applicable in the scope of this research. This project focuses primarily on using location data, and not computer vision. However, there is a plethora of research in different settings where a top-down location approach is used. We will divide these into three categories. First we will describe the sequence based temporal approaches, graph based approaches, and then a few generative methods that have been used previously.

3.1 Sequence-based approaches

Trajectories are often represented as a sequence of datapoints with given timestamps. Thus, models that are able to derive temporal relations from data are often considered in the trajectory prediction task. Most of the methods that fall into this category leverage the strengths of recurrent neural networks (RNN). Especially LSTMs and GRU-units are often mentioned as a solution towards dealing with the time-dependencies in the data. A simple application of these models can be found in the work of Kim et al. (17). They used a simple two-cell LSTM model to predict the trajectories of moving vehicles over an occupancy grid map. The method used consist of a compact model that is only used to predict trajectories of at most two seconds into the future. In the results we can also already find that, as may be expected, the prediction error increases significantly as we look more timesteps ahead. Alahi et al. (4) combined multiple LSTM models using a 'social' pooling layer in order to not only model the short term movement of a single person, but also to try and capture human-human interactions into their method. There method is shown to outperform previous state-of-the-art models on standard datasets. The results clearly show how the model is able to make intelligent route choices like yielding for other agents in the system. Trajectory prediction is also used in naval research. Suo et al. (34) used a GRU model to predict vessel trajectories. In their research, they have shown that their model is able to achieve similar results to models using LSTM layers, however, their model improves on computation efficiency.

Unfortunately, simple LSTM models are not enough to get a decent result in trajectory prediction. Since trajectory prediction in the simplest sense can be described as analyzing an input trajectory, and predicting its continuation, we can see this task as a sequence-to-sequence problem. Therefore, more often model architectures that are designed to capture the intrinsic information of sequences are used in for trajectory prediction. The most commonly seen architecture is an encoder-decoder model using LSTM blocks. Park et al. (26) try to predict the trajectory of vehicles across an occupancy grid map using the work of Kim et al. (17) as baseline. Here we can see that using the encoder-decoder model, Park et al. are able to achieve better results. If we look at the last predictions (so after two seconds), Park et al, are on average of by 0.93 grids, whereas this value was 1.31 for Kim et al. In their work, Wang et al. (40) also show that sequence-to-sequence models outperform the baselines. They use three different sequence-to-sequence architectures, and for each of them the MSE (over the whole data) is shown to be approximately half the value compared to the other models used. It does show that the sequence models appear to be a little behind the other models when it comes to one-step predictions, but when looking further ahead, the sequence model outperforms consistently.

Deo and Trivedi (11) show in their work that this architecture is able to predict the parameters of a bivariate Gaussian distribution instead of the trajectory itself. In their output sequence, each element corresponds to the means and variances of future locations for the vehicle. Using this strategy, they outperform state-of-the-art models on the same datasets, showing the viability of this approach. To take the sequence-to-sequence architecture a step further, Nadarajan and Sivanraj (25) use and encoder-decoder model but enhance it using a spacial, and a temporal attention mechanism. They do this to analyse the multiscale spatiotemporal dependencies in non-Euclidean space to forecast traffic. This way, they take into account similarities between different places in the road network. In their model, they also integrate data from external factors such as the weather, to get the final predictions. Their model is shown to have more accurate MSE scores compared to other models consistently up until two hours into the future.

In a vastly different direction, Qin et al. (27) use a CNN based approach for their trajectory predictions. Instead of predicting the exact locations, they split their area of interest into grids and follow their vehicles as a series of grid number. By converting each value into a one-hot vector and then concatenating all vectors in a sequence over a temporal dimension, they created a spatio-temporal grid representing each trajectory. These grids were then used to train a Capsule network. Capsule networks use groups of neurons that encode specific properties of its input. Using this method, they were able to achieve improved accuracy scores compared to LSTM and regular CNN models.

3.2 Graph-based approaches

Most traffic situations consist of intricate information that cannot be simply defined by linear dependencies. Thus, a graph representation, consisting of nodes and edges corresponding to to individual spatial units are often used to model these intricacies. Most graph based models use graph convolutional networks (GCN) in order to learn more complex topographical information that is more difficult to handle for convolutional networks. In forecasting tasks, GCNs are often combined with different methods that are able to handle temporal relations such as RNNs. Zhao et al. (44) show that by combining GCNs with GRU-units, one can achieve forecasting results which indicate a steady state under different prediction horizons. This means that these models are not only able to accurately predict the short-term dependencies in data, but that they are also applicable in long-term prediction tasks.

These so-called temporal GCNs also form the base of more complex applications of graph networks. For example, Li et al. (19) propose a dynamic graph convolutional recurrent network where dynamic graphs are generated using an encoder-decoder architecture that is often seen in the previously mentioned sequence-to-sequence models. They combine these generated graphs with previously defined static distance based graphs and use a GCN with GRU units for their final predictions. Zhi et al. (20) take this model a step further by proposing a method where they fuse multiple graphs with different similarity information to model dynamic spatio-temporal relations in traffic. They show that their proposed method is able to outperform multiple different baseline models on different datasets. However, a main limitation of this methodology is that their are high requirements for the datasets on which it is used. Since the model requires a lot of accurate traffic flow-, road-, POI- and weather data, the limited availability of data which meets the requirements may affect the generalization ability of the method.

Another use of GCN model is provided by Zhu et al. (45). They propose an attribute-augmented spatio-temporal GCN for traffic forecasting. In this work they derive augmented graphs in order to consider more external factors that may affect the movement of traffic in their GCN model. The main contribution of this model is that not only their prediction outperform commonly used baseline models, but they are also able to show attribute importance. Through this they have also shown that diversity of external factors can be used improve forecasting performance.

3.3 Generative methods

Another approach to predicting the trajectory for a vehicle is by using generative models. These models focus on generating new data samples that resemble the training data by learning the underlying distribution. An example of this approach can be found in the work of Rossi et al. (29). They propose an architecture using generative adversarial networks to generate different trajectories that portray different behavior. However, they also conclude that in most scenarios that LSTM outperforms GAN, meaning that the latter is not a replacement for the former. On the other hand, the scenario where GANs did outperform is when a multi-modal approach is necessary or in cases with high uncertainty.

Using a different model, Bhattacharyya et al. (5) predicted drone trajectories. Their conditional flow variational autoencoder model uses conditional priors based on conditional normalizing flows, which allow for the model to take conditional information into account when decoding the training data. This in turn allows for more complex multi-modal representations of the data in the latent space of the model. This approach is also shown to achieve results on the same level as other state of the art generative models used for this task.

Sørensen et al. (36) used a Bidirectional Long-Short-Term-Memory Mixture Density Network to characterise the underlying distributions of the movements of ships. This application of trajectory prediction is used not only to prevent dangerous situations at sea, but also to attempt to identify unknown vessels at sea. They applied multiple different model architectures in their project and where able to predict locations of vehicles with a mean difference of 2.53km 50 minutes into the future. This is, on the scale of the maritime trajectory prediction, accurate and on par, if not better, than state of the art attention models.

Finally, Zeng et al. (43) answer the question if transformer models are effective for time series forecasting. Their finding, using a relatively simple model, is that in this task, transformers may be susceptible to temporal information loss. However, Jiang et al. (16) propose a different structure compared to regular transformers where the model should be more viable in tasks involving sequence data by combining the transformer model with LSTM structures. They show that in busy maritime waters, their approach outperforms state of the art models for trajectory prediction thus indicating that adapted transformers can be used for trajectory prediction.

4

Data

In this chapter we will look at the data that has been used during this project. We will delve into the raw data and how it has been processed to obtain useful trajectories. We we also take a look the data itself by doing an exploratory analyses both on the raw data as well as on the obtained trajectory data.

4.1 Description

The data used in this project is a dataset for which data point have been collected since 2021. Over time, the GSE vehicles have been equipped with sensors that are able to register data about the vehicle at short time intervals. Approximately every minute, if a vehicle is active, an instance is saved with information about the vehicle's location and its status. The recording of the instances is a project that has been introduced in stages, thus, over time, more vehicles's movement is tracked and recorded. Different vehicles, may be equipped with one of three different types of sensors ('CTRACK', 'WEBFLEET', or 'TARGA'). The information retrieved from these different sensor types and transmitted to the dataset is consistent regardless of the type. At the end of each day, the dataset is updated with all data instances up until the day before.

The data has 17 different features that are recorder. However, many of these features, such as the company which owns the vehicles and the department to which the vehicle belongs, are all limited to only a single unique value in the data, and thus do not provide distinguishable information. Table 4.1 gives a short overview of the variables in the data that are have been used during this project.

Variable name	Description
EventId	An identifying string that is uniquely generated for each instance in the data.
EventSource	The sensor type used to collect the instance.
EventTimeUtc	The date and time (in UTC) at which an instance was recorded.
AssetName	Identifying name for each vehicle.
AssetId	Id assigned to each vehicle by the sensors.
AssetType	The type of vehicle.
Latitude	The latitude value of the location of the vehicle.
Longitude	The longitude value of the location of the vehicle.

Table 4.1: Overview of the relevant variables in the dataset.

Even though there are two variables that seem to be usable as identifier for the vehicles in the dataset (AssetName and AssetId), only AssetName can actually be used to distinguish vehicles. Because the assigning of AssetIds is done seperately by each tracker system (the three different EventSources), there is overlap in some ids between sensor types causing clashing AssetIds for different vehicles. So even though there are more unique AssetId values in the data, they can not be used to identify vehicles, which is why the AssetName is used as identifier.

4.2 Pre-processing

The first step before we are able to make predictions will be to process the data in a suitable form to feed it into our model architectures. This process involves cleaning the data and transforming it into a usable form.

4.2.1 Trajectory extraction

To get usable trajectory information from the raw GSE data, a few steps have been taken. First of all, the data is split by vehicle and sorted in the chronological order of the event-times. From there, we can obtain the first event for a vehicle as the start of the first trajectory. Next, we check the time difference between the current latest event and the next. If this differences is within a given threshold, the next event is added to the trajectory. In the scenario where the allowed difference is larger than one time-unit, the time steps in between the two events are interpolated with copies of the earlier event. This is done to ensure consistency in the time steps as missing data may drastically degrade the quality of the predictions (42). In case where the next event does not happen within the specified time range, the current trajectory is closed off and a new one is started with the now latest event as the beginning.

Each trajectory is represented as a list of pairs of (latitude, longitude) datapoints. With each trajectory, some key information is also tracked to get insights in the data. Aside from the name and type associated with the vehicle, we also measure the travelled distance, the start- and end times for each trajectory (and thus the total duration) and the number of time steps the vehicle remains stationary during the trajectory.

4.2.2 Data selection

Not all event data will be used in order to extract trajectories. There are two main reasons for not taking a data point into account. First of all, events for which both the vehicle name and vehicle type are unknown are taken out of the data. This is done because there is no way of knowing to what vehicle the data point belongs. This may cause events from different vehicles to be used together in a single trajectories which would naturally cause incoherence and mistakes in the our training data. The second filter to screen out data is based on the location at which the event is recorded. The scale of this project is limited to Schiphol airport grounds, however, some data points are recorded far from away. Even though this mostly concerns company car type vehicles, there are other vehicles that are also located outside of the premises in the data. For this reason, the decision was made to filter out all datapoints where the registered longitude was below 4.73 or above 4.81. For the latitude all datapoints before 52.285 and after 52.335 were filtered out.

4.2.3 Trajectory preparation

During training of the models, not all trajectories will be used as input. One can imagine that a trajectory consisting of only a few data points does not hold a lot of information compared to longer ones. The trajectories used are those that are of length of at least two thirds of the desired input sequence length. The base scenario discussed will use input trajectories of length 45. This means that for a trajectory to be considered for the training, it has to be of at least length 30. If a trajectory is too short for the experiment, but meets the minimum length requirement, these trajectories are assumed to have been stationary before they began. Thus, they are padded in front of the sequence using the first value of the true sequence. If a trajectory is longer than the desired input length, it is split into separate trajectories of the desired length. This is done both starting from the front of the trajectory, as well as starting from the back, to ensure that only limited sequential information is lost during the splitting of long trajectories of indivisible sizes.

4.2.4 Normalization

Normalization is used to rescale data such that different features are of similar scale, often between 0 and 1. This is done to minimise bias in a model caused by a difference of scale between explanatory features. For this project, min-max normalization will be applied to all numerical features that are used as input to a model.

Previous research shows that min-max normalization is a successful technique when dealing with a 2d GPS coordinate system (7). Similarly, Shi et. al. (31) used min-max normalization in a GPS based flight prediction task.

In min-max normalization, the data is rescaled towards a new range of values within predefined boundaries. This technique uses the following formula:

$$x'_i = y_{min} + \frac{x_i - x_{min}}{x_{max} - x_{min}}(y_{max} - y_{min}), \tag{4.1}$$

where x_{min} and x_{max} are the minimum and maximum value respectively of the variable to be normalized. y_{min} and y_{max} are the boundaries of the selected range towards which the data is to be rescaled.

4.2.5 Data representation

The trajectories will be used in two different representations during this project. The first representation is the straightforward sequence of normalized (latitude, longitude) pairs. This representation shows the exact location where the events have been recorded in the data and thus will be used to try and predict the exact continuations of the trajectories. Secondly, an area based representation will also be used. In this representation, the premises has been divided into multiple small subareas, which are used to give an approximation of where the vehicles are located at a given time. The area has been divided into 625 parts, where there a total of 25 divisions along both the longitude, as well as the longitude. This is also illustrated in Figure 4.1. When used in training, the sequence of areas representing the movement of a vehicle has been transformed into a sequence of one-hot vectors.

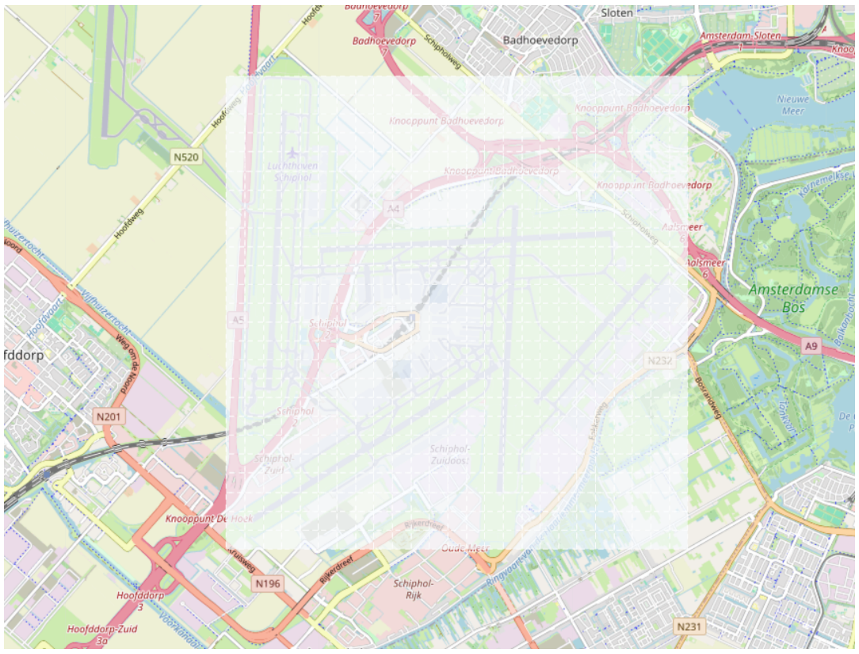


Figure 4.1: The total area of the Schiphol premises partitioned into multiple subareas.

4.3 Exploratory analysis

The data used can be split into two different sets: firstly, the raw event data, consisting of numerous instances of recorded events by the different GSE vehicles, that the has been used in to extract the trajectories, and secondly, the trajectories themselves represented as a sequence of consecutive events. We will look into both datasets to get a better idea of the underlying distributions of our data points. This will mainly be done visually as an aid in understanding the frameworks in which this project partakes.

4.3.1 Event data

The event data is an incredibly rich dataset consisting of millions of recorded events. When we only look at the data recorded in 2023, just under 50 million instances can be found in the data. Each instance representing a single event of a single vehicle recorded at a single moment. One would expect that on different moments in time, these instances would show more or fewer activity in the use of GSE vehicles. To get insights in this, the number of vehicles active in certain time frames is plotted in Figure 4.2.

There are a few conclusions to be drawn from the plots in Figure 4.2. First of all, we see a spike in activity at 0:57 daily when we look at the top-left plot. This spike is caused by one type of sensor (the CTRACK sensors) which give an automatic status update daily at this time. Furthermore, in the same plot we see two clear peaks. The first at around 6:00 and the second around 18:00. These peaks correspond to the beginning and the end of a working day and can be assumed to be caused by the vehicles being taken from and returned to designated areas.

Both in the distributions per day of the week and per day of the month, there appear to be no significant seasonality or trends. This indicates that the amount of use for vehicles is not dependent on those time frames. Even though one might expect more traffic in the summer, since there are more flights in the vacation months, the bottom-left plot actually appears to dip in July and August. After some investigation, this dip was found to be caused by errors in the communication between the software and the sensors. Thus, this pattern cannot be used as an indication of the use of the vehicles. There is a significant increase in instances in the data between the first two months. This is not a sign of seasonality or trend, but caused by the introduction of more recorded sensors on different vehicles.

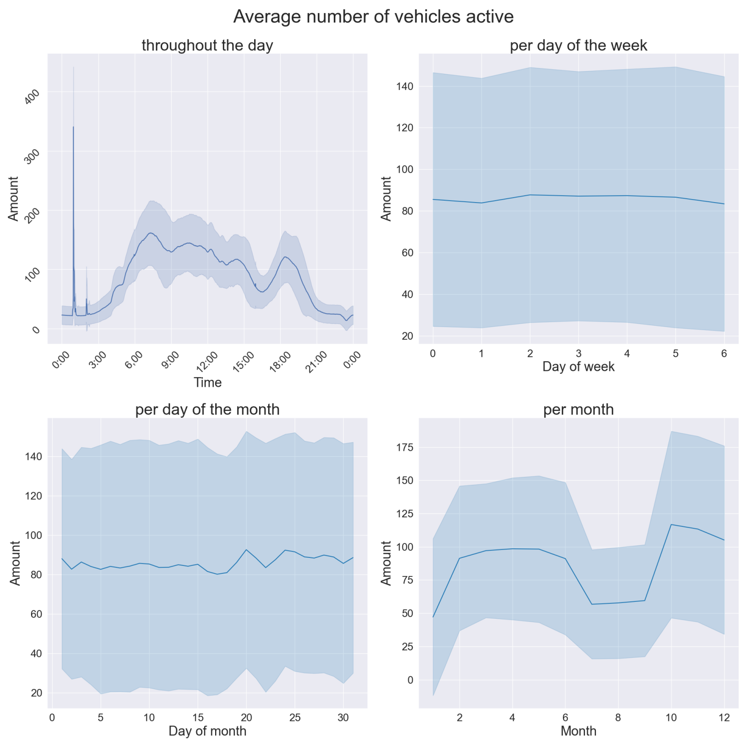


Figure 4.2: Temporal distribution of the events in the data represented per minute of the day (top-left), day of the week (top-right), day of the month (bottom-left), and per month (bottom-right). The blue line represents the mean values at those times, whereas the light blue area portrays the standard deviation.

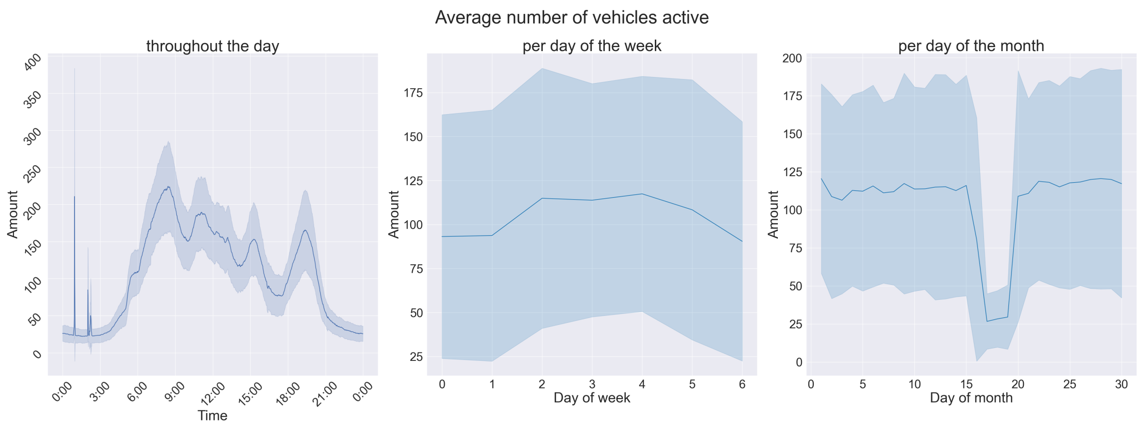
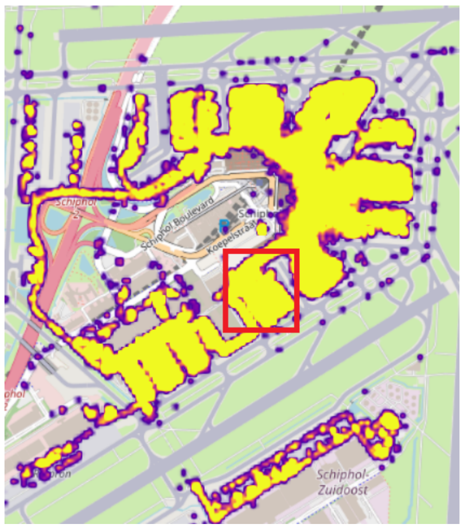


Figure 4.3: Temporal distribution of the events in the data of December 2023 represented per minute of the day (left), day of the week (middle), and day of the month (right). The blue line represents the mean values at those times, whereas the light blue area portrays the standard deviation.

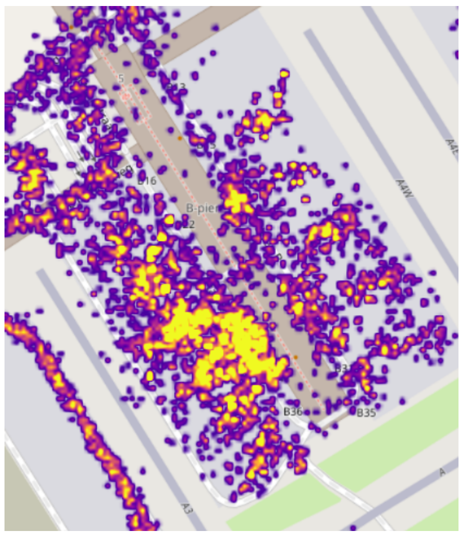
Because the number of data points is too large to use in an efficient manner. The decision was made to only use the data of December 2023 in the training process. This leaves us with just over 6 million instances of event data. The temporal distribution of this data can be found in Figure 4.3, which confirms the observation madein Figure 4.2. The one main difference, is that there is a suspicious lack of activity between the 17th and the 19th day of the month. This irregularity is again caused by a problem in the communication between sensors and software. However, as the time distributions seem to be independent on the day, this is not seen as a problem for this project.

The sensor data can be collected anywhere. For example, some vehicles like the company cars, are recorder to travel throughout the country. However, since this project focuses on the premises of the Schipol Airport grounds, we will only look at the vehicle present there. Figure 4.4 shows a heatmap of the events recorded on a single day (October 1, 2023). The most important conclusion we can draw from Figure 4.4 is that even though the grounds do have a defined road network, the vehicles are not always restricted in their movement. This is especially evident in Figure 4.4 (b). When we look at the lower-left corner, we see that all recorded events follow a straight line along the road, on the apron however, there is no clear structure on where the vehicles should move.

The data used is event data collected over multiple vehicles. Different vehicles can have different uses for the ground service process. In total there are 15 different vehicle types recorded in the data and there is also an 'unknown' token for vehicles for which the type



(a) Heatmap of the locations of events in the dataset.



(b) Zoomed in heatmap.

Figure 4.4: A heatmap of locations of the recorded events. Figure 4.4 (b) shows a zoomed in version focused on the red rectangle in Figure 4.4 (a).

has not been properly recorded. In Appendix A a description for each of the different vehicles types can be found. In total, 890 vehicles were recorded in use in December 2023. Figure 4.5 shows the distribution of the different vehicles types in the data. There is a clear imbalance between the different vehicle. Approximately 25 percent of the vehicles has no known vehicle type associate with the AssetName. From the other vehicles, a clear majority of the vehicles is either a company car or a GPU. It is logical that these vehicles are represented most in the data, as company cars are the most universally used means of transport for employees around the Schiphol grounds and each ramp is equipped with a GPU vehicle to power the parked aircrafts. The other vehicles are easier to share between aprons or ramps and thus a lower number of them is necessary in the vehicle fleet.

The final factor to look at is the utilization of the vehicles. Figure 4.6 shows for how many moments a vehicle has a recorded event on average. Important to consider when looking at Figure 4.6 is that the presented utilization is purely based on the number of events found in the data. However, there are situation where a vehicle is used, placed at a given location, and then turned off. For example, the Trap Pax vehicles which are used to help passengers board a plane, can be placed next to an aircraft for an extended period of time. When the vehicle is not powered, there are no events recorded by the sensor, but the vehicle is still in use. Thus, the presented utilization is an underestimation of the true use. Remarkable

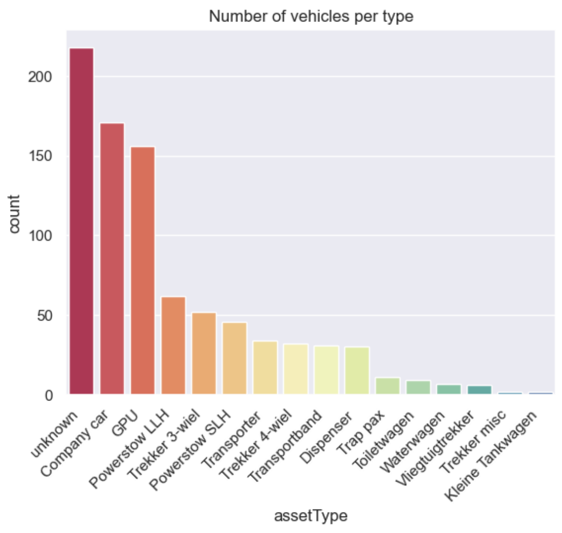


Figure 4.5: Distribution of the different vehicle types in the data.

in Figure 4.6 is that there is one vehicle recorded with an utilization of almost 1. This is the 'vehicle' that has been recorded as with the AssetName and AssetType 'unknown'. Thus this data point does not represent a single vehicle, both more a collection of vehicles whose information is not properly recorded. The data points this outlier represent will for this similar reasons also not be taken into account for the rest of the project.

Aside from the the distributions for all instances together, we also considered information as shown in Figures 4.3 and 4.6 for vehicles of the different types separately. These plots portray behaviour similar to that of the distributions for all vehicles combined, thus to prevent redundancy, the plots for only vehicle types 'unknown', 'Company Car', and 'GPU' can be found in Appendix B.

4.3.2 Trajectories

The extracted trajectories are shown to have vastly different types of behaviors. Figure 4.7 shows how diverse the trajectories are. The black trajectory depicts a route where the vehicle traveled around most of the airport buildings. Whereas the trajectory encircled in red shows mostly local movement almost in a single location. These different behaviours in movement can to a certain extend also be linked to the vehicle types and the tasks those vehicles have to perform. The encircled trajectory is characteristic for a GPU vehicle which is mostly stationary. Since there are plenty of these units around the airport, they also do not have to moved much and thus show limited movement. On the other hand company

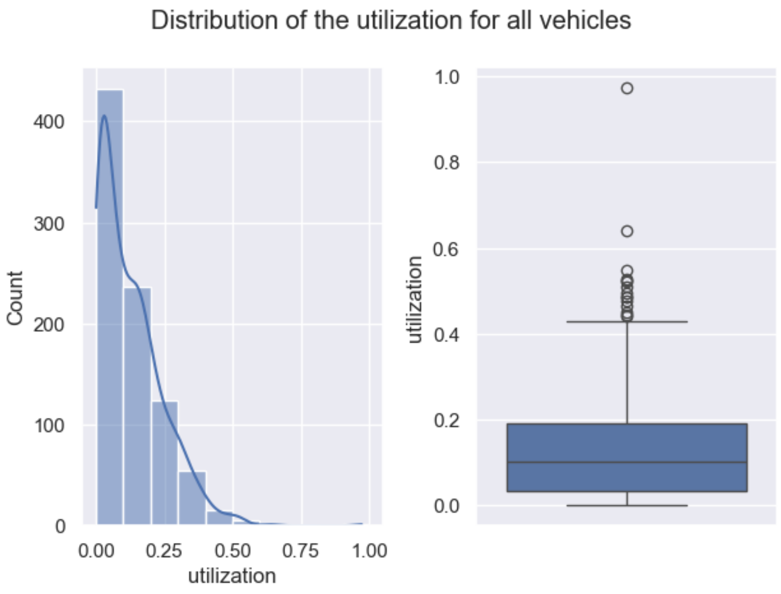


Figure 4.6: Utilization of the vehicles.

cars, or the 'trekker' vehicles, have to move around more to perform the tasks they are assigned to. Figure4.8 shows the total number of trajectories recorded for each vehicle type. We can see that, unsurprisingly the most trajectories are recorded for the most commonly seen vehicles. However, besides vehicle types 'Company Car and 'unknown' there appears to be a more balanced distribution of trajectories, decreasing with the number of vehicles of a type.

The trajectories have been extract with different idle windows, meaning different time spans allowed without an event before a trajectory is cut off. In Table 4.2, information about the trajectories is shown. Remarkable is the high stationary percentages shown. These mean that even though trajectories of a certain length have been seen in the data. The vast majority of the trajectories are stationary most of the time. Nonetheless, this percentage does decrease when we take a higher idle window, which is surprising as the gaps created by the idle window are filled with stationary sequences. Due to the higher number of eligible trajectories, and the lower stationary percentages, we will mainly focus on trajectories with an idle window size of 10 during this project.

In Figure 4.9, the number of useful trajectories, according to the length criteria described in Section 4.2.3 can be seen. In this figure we can clearly see that the number of trajectories is much more balanced after the selection process. When we look at the idle window of size 1, there is still a steady decline of the number of trajectories for different types. On the other

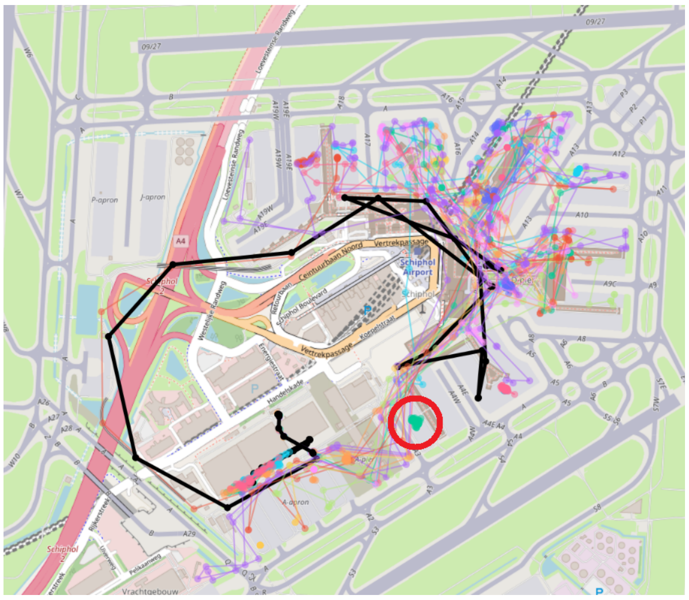


Figure 4.7: Examples of extracted trajectories. Encircled in red, a relatively stationary trajectory can be seen, while the trajectory highlighted in black is shown to go around the premises.

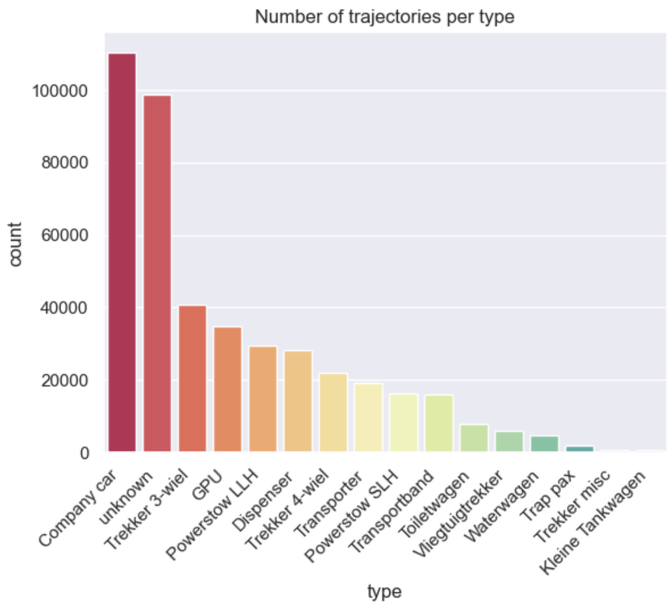


Figure 4.8: The number of trajectories recorded for each vehicle type.

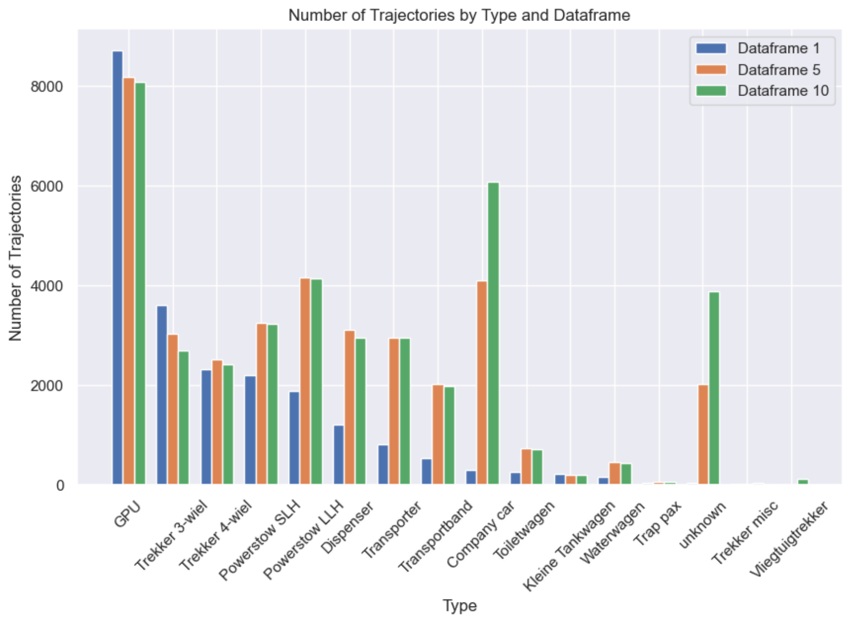


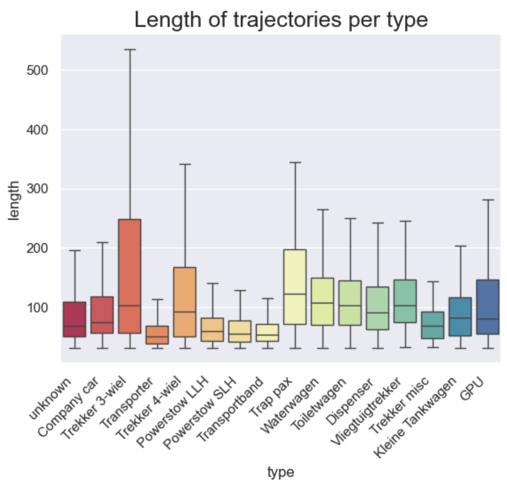
Figure 4.9: Number of useful trajectories separated by vehicle type and idle window.

Idle window	Number of trajectories	mean duration	mean stationary percentage
1	22590	71 minutes 10 seconds	0.87
5	37517	93 minutes 19 seconds	0.77
10	40866	97 minutes 1 second	0.76

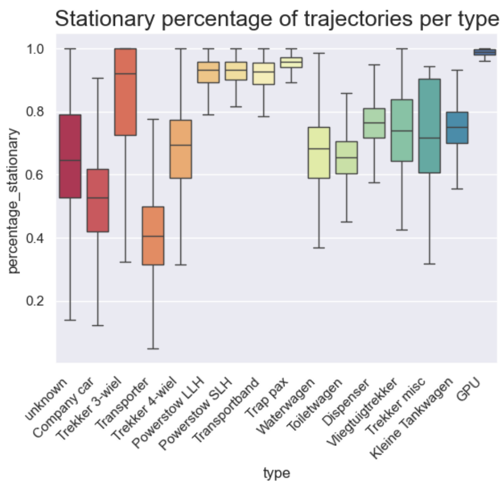
Table 4.2: Trajectory information per idle window size

hand, the trajectories extracted with an idle window of size 10 appear to be more equally divided over the different company types and are also more frequent. Unfortunately, there are still a few types for which there are barely any trajectories eligible to be used according to the defined criteria. Though, this was to be expected as we have seen that there are a few vehicle types for which there are only a few records in the data as seen in Figure 4.5.

Correspondingly, in Figure 4.10, the distributions of the trajectory lengths for the eligible trajectories and the percentage stationary per vehicle type are shown for the trajectories with an idle window of 10 minutes. In Figure 4.10 (a), we see that the trajectories consist for the majority of between the 50 and 150 instances. Interestingly, for the type 'trekker 3-wiel', we see a significantly larger spread in the boxplot. We can also see in Figure 4.10 (b) that there are a few vehicle types, like the GPU and 'Trap Pax' which display a consistently high stationary behavior. This is caused by the nature of these vehicles as for the tasks they are used for, the vehicles are required to remain on the same place while in use. For the vehicles that are required to move more, like the company cars, we still see that they are often found on the same location of multiple consecutive time steps.



(a) Length of trajectories per vehicle type.



(b) stationary percentage of trajectories per vehicle type.

Figure 4.10: Overview of trajectory length (a) and stationary percentage (b) per vehicle type.

5

Methodology

In this chapter, methodology for the trajectory prediction performed in this project will be explained. First the models used will be explained, then we will delve deeper into how the models were trained and how the results will be evaluated.

5.1 Models

In Chapter 4 we explained how the primary source of available data is limited to trajectory data which is derived from the raw GSE GPS data. Due to the nature of this data the choice was made to include three different types of models: (1) Recurrent Neural Networks (RNNs), (2) sequence-to-sequence architectures and (3) generative models. These models were chosen as they enable us to take the temporal nature of the sequences in the data into account. This should allow for the underlying structures in the data to be captured effectively.

5.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data by maintaining an internal memory. In RNNs extra connections are included that form directed cycles within hidden layers to capture temporal behavior. This architecture enables RNNs to model and predict sequences of data. In this project we will mainly use a variant of RNNs called long short-term memory (LSTM) models.

The idea behind RNNs is that they can use feedback connections in the network to store a representation of recent inputs in the model. However, they also have a high risk of either vanishing or exploding gradients. To solve this problem, Hochreiter and Schmidhuber

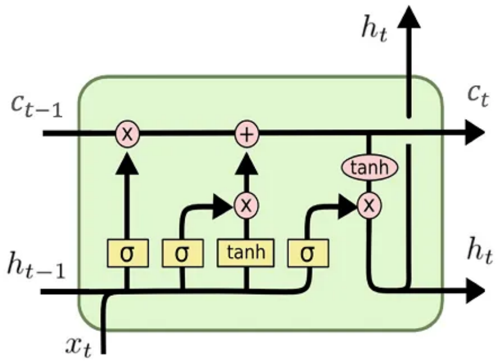


Figure 5.1: A single LSTM cell (28).

(15) proposed the LSTM cell to be introduced in neural networks. LSTM have obtained widespread adaptations in sequential data analysis, time series prediction and classification tasks across various domains. Their ability to retain temporal information over extended periods of time makes them outstandingly effective for handling time-series data.

LSTM units consist of multiple cell that use three multiplicative gates to control the flow of information through the model. The input gate determines which new information should be stored in the cell. The forget gate combines the previous hidden state and the current input, and determines which information can be discarded. And thirdly the output gate controls the output of the model at each time step. Figure 5.1 shows a single LSTM cell.

To clarify Figure 5.1, the LSTM cell is also provided in equation form in Equations 5.1-5.4

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.1}$$

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{5.2}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.3}$$

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{5.4}$$

Where:

t :	The current time step in the sequence with $t = 1, 2, ..., T$
T :	The sequence length
f_t :	Forget gate output
i_t :	Input gate output
$\tilde{C}_t$:	Candidate cell state
C_t :	Updated cell state
o_t :	Output gate output
h_t :	Current hidden state
W_f, W_i, W_c, W_o :	Weights
h_{t-1}, x_t :	Previous hidden state, current input
b_f, b_i, b_c, b_o :	Biases

In this project, an extension of this traditional architecture will also be used. The so-called Bidirectional LSTM (BiLSTM) incorporates two separate LSTM layers. One that processes a sequence from start to finish, and one that processes the data backwards, hence the name 'Bidirectional'. This extended architecture has previously been shown to improve performance of time-series forecasting problems (32). Therefore, in this project we will also investigate if the bidirectional learning can be leveraged to improve on the results.

Both the LSTM and BiLSTM models that are used in this project will be used as a baseline for the larger models to compare performance to.

5.1.2 Sequence-to-Sequence learning

Sequence-to-sequence learning is a deep learning framework used to take sequential inputs to generate a corresponding output. Often this architecture is used in combination with RNN variants like LSTMs. Most commonly these models will be found in tasks like machine translation and speech recognition, but as shown in Chapter 3, they are also proven to be effective models in trajectory prediction. The main idea behind sequence-to-sequence learning is to encode the input sequence step-by-step into a single vector which is then decoded by a second set of LSTM models (35).

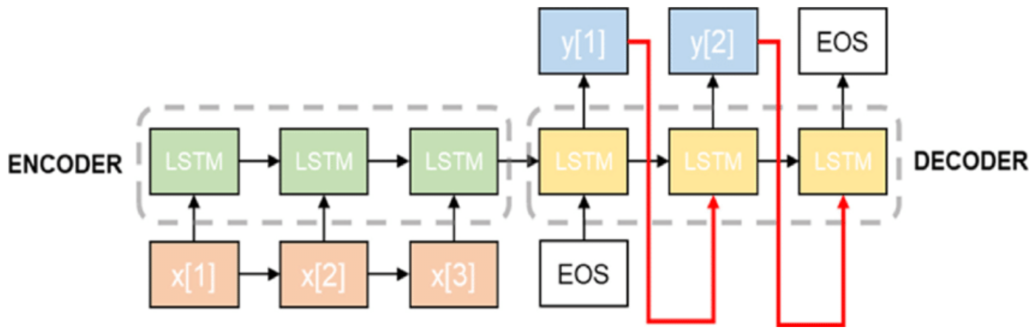


Figure 5.2: The structure of a sequence-to-sequence LSTM encoder-decoder model (18).

5.1.2.1 LSTM encoder-decoders

The sequence-to-sequence architecture that is used in this project is an LSTM encoder-decoder model. This means that the input trajectories are encoded and decoded using LSTMs. Figure 5.2 shows the general structure of this architecture. Here, the input X_t for each time step t is input into each own LSTM block in the encoder network. After each input has been processed, the hidden states of the LSTM block are carried over as the input of the next block. This is repeated until the full input sequence or trajectory has been encoded in the hidden states of the final LSTM block. In the decoding, the model is firstly fed with an arbitrary token that is meant to indicate the beginning end/or end of a sequence. This token is used to predict the first value(s) of the output sequence, next the final token in the output sequence is fed back into the model until a stopping condition has been met.

Because the predicted tokens are fed back into the model, it is not possible to predict the full output sequence at once and a special inference has to be done. This process is described in Algorithm 1. The stop condition is often either the maximum output sequence length has been reached, or the end token has been predicted as output, indicating the end of the sequence.

5.1.2.2 Attention

There are still challenges with the encoder-decoder models. For example, when the input sequences are too long, it might be hard to compress the sequence into a proper state vector without losing too much information. Another problem may be that for different steps in the decoding process, information from different points in the input may be relevant. In short, the encoded state vector can act as a bottleneck for information (39). To circum-

Algorithm 1 Inference Algorithm for Seq2Seq Encoder-Decoder Model

```
1: Input: Encoder model $E$, Decoder model $D$, input sequence $X$
2: Output: Generated output sequence $\hat{Y}$
3: Initialize empty output sequence $\hat{Y}$
4: $h \leftarrow E(X)$ Encode input sequence
5: Initialize decoder input token $y_0$ to start token
6: stop_condition ← False
7: while not stop_condition do
8:     $\hat{y}_t, h \leftarrow D(y_{t-1}, h)$ Decode with previous hidden state and previous output token
9:     Append $\hat{y}_t$ to $\hat{Y}$
10:    if $\hat{y}_t$ is end token then
11:        break
12:    end if
13: end while
14: return $\hat{Y}$
```

vent this, an attention mechanism can be built into the model. An attention mechanism determines the significance of various components of the inputs at every decoder step. The adapted architecture can be seen in Figure 5.3. In this architecture the encoder is not obligated to condense all information in the training data into a single vector. Instead, it can give a representation of the information from all source tokens. In this project, we use a simple dot product for the attention mechanism. However, other functions, such as the bilinear function used in the Luong model (21), could have been used as well. In Algorithm 2, we can see the adapted algorithm for the inference when attention is included in the model.

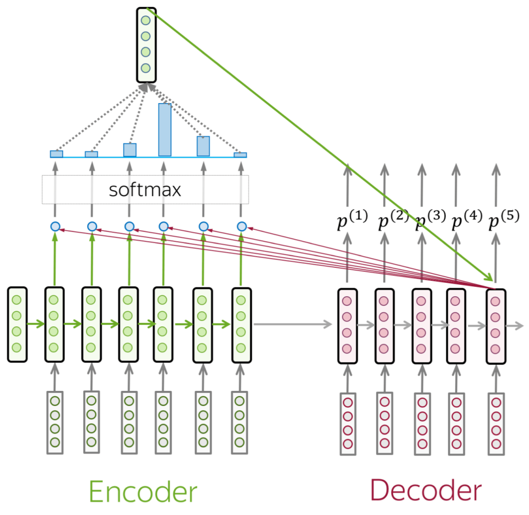


Figure 5.3: The structure of an encoder-decoder model with an extra attention mechanism (39). The blue circles represent the attention mechanism, in this case: a dot product.

Algorithm 2 Inference Algorithm for Seq2Seq Model with Attention Mechanism

```
1: **Input:** Encoder model $E$, Decoder model $D$, input sequence $X$, maximum output
   sequence length $T_{\text{max}}$
2: **Output:** Generated output sequence $\hat{Y}$
3: Initialize empty output sequence $\hat{Y}$
4: $h \leftarrow E(X)$ {Encode input sequence}
5: Initialize decoder input token $y_0$ to start token
6: $t \leftarrow 1$
7: stop_condition $\leftarrow$ False
8: **while** $t \leq T_{\text{max}}$ **and not** stop_condition **do**
9:     Compute attention weights $a_t$ using $h$ and $y_{t-1}$
10:    $\hat{y}_t, h \leftarrow D(y_{t-1}, h, a_t)$ {Decode with previous hidden state, previous output token,
       and attention weights}
11:    Append $\hat{y}_t$ to $\hat{Y}$
12:    **if** $\hat{y}_t$ is end token **then**
13:        stop_condition $\leftarrow$ True
14:    **end if**
15:    $t \leftarrow t + 1$
16: **end while**
17: **return** $\hat{Y}$
```

5.1.3 Generative models

Sequence-to-sequence models focus on mapping an input sequence to an output sequence, where the encoder processes the input sequence into a representation of a fixed dimension. Generative models differ from sequence-to-sequence models in that they try to replicate the underlying probability distributions of the training data. Once the underlying distribution is known, new datapoints can be sampled that resemble the original data. In this project, two generative models are used to see if we can get a close approximation of the true continuations of the given input trajectories using these models.

5.1.3.1 Transformers

Transformers can be used both as a sequence-to-sequence model, but also a generative model. Even though in the setting of trajectory prediction, transformers probably represent a sequence-to-sequence model more, we discuss them in this section as they are vastly different from the previously described models. Unlike the previously mention RNN based networks, transformers use a mechanism called self-attention to capture dependencies between input and output tokens (38). The structure of the model can be seen in Figure 5.4. The self-attention mechanism is the base of the transformer and it uses a query, key and value matrices to compute attention scores between all pairs of input tokens. This calculation is shown in Equation 5.5.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{5.5}$$

Where:

- Q represents the query matrix,
- K represents the key matrix,
- V represents the value matrix, and
- d_k represents the dimensionality of the key vectors.

This self-attention is then used in the multi-head attention layers as shown in Equation 5.6

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O, \\ \text{where} \quad \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \\ \text{for } i &= 1, 2, \dots, h. \end{aligned} \tag{5.6}$$

After the self-attention layers, each token's representation passes through a position-wise feedforward neural network. By leveraging self-attention mechanisms and multi-head at-

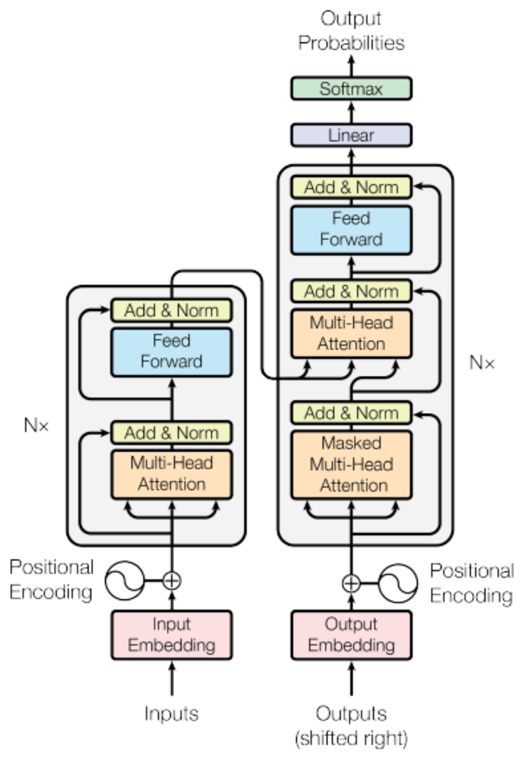


Figure 5.4: The structure of a transformer model (38)

tention, transformers can efficiently capture long-range dependencies in sequences, making them highly effective for various sequence processing tasks.

5.1.3.2 MDN models

Mixture Density Networks (MDN) are a type of neural networks where instead of assuming that we can get a deterministic output for each given input, it is assumed that the conditional distribution of the target data is, in fact, Gaussian (6). In MDN models, the output is not an exact target value, but the parameters of a mixture of Gaussian distributions which represent the data. The conditional probability is modelled by approximating the probability as a mixture of several known probability density functions (PDF) as shown in Equation 5.7.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i(\boldsymbol{x})\phi_i(\boldsymbol{y}|\boldsymbol{x}) \tag{5.7}$$

Where:

- m represents the number of mixtures,
- $\alpha_m(\boldsymbol{x})$ the mixture weight for the m'th mixture normalised using the softmax function, and
- $\phi_i(\boldsymbol{y}|\boldsymbol{x})$ represents the individual mixture component, which is modelled as a Gaussian PDF as seen in Equation 5.8.

$$\phi_i(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{(2\pi)^{c/2}\sigma_i(\boldsymbol{x})^c} \exp\left\{ -\frac{\parallel \boldsymbol{y} - \boldsymbol{\mu}_i(\boldsymbol{x}) \parallel^2}{2\sigma_i(\boldsymbol{x})^2} \right\} \tag{5.8}$$

With:

- f represents the number of target features,
- $\boldsymbol{\mu}_m(\boldsymbol{x})$ represents the mean vector of features the m'th mixture, and
- $\sigma_i(\boldsymbol{x})$ represents the standard deviation of the features.

An MDN model is a neural network where the final layer is actually a concatenation of three separate feed forward layers which each represent the values of either α, μ, or σ. For this experiment a model consisting of 10 mixtures was trained.

5.1.4 Model architectures

During this project multiple different architectures were trained to compare there performance in the end. All mentioned models were used in both the scenario where we predict the GPS locations directly, and also the area based predictions. However, they do differ in that the final fully connected layer in the area based models use a softmax activation instead of a linear activation in order to get a probability distribution over the possible areas.

As baseline models, both a simple one layer LSTM and a BiLSTM model were used. These models consisted of only an input layer, then one RNN layer and finally a time distributed fully connected layer to get the desired outcomes.

When we look at the more complex architectures, in all scenarios two models were trained, one with an LSTM encoder and one with a BiLSTM encoder. Because the decoders require the outputs to be generated one by one in these models, it is not possible to apply the backwards learning used in a BiLSTM in the decoder. Thus, all decoders consist of regular LSTM blocks.

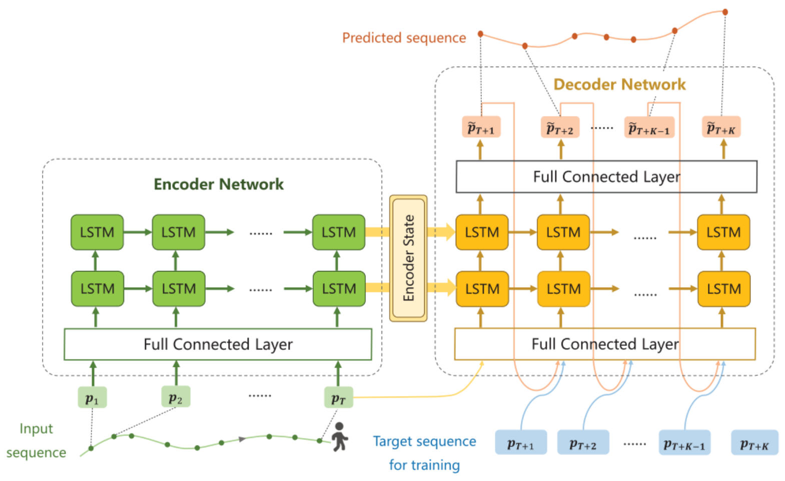


Figure 5.5: The sequence-to-sequence framework with two stacked LSTM layers preceded by a FC layer (40).

The simplest encoder-decoder model are built out of one layer of RNN blocks in both the encoder and the decoder. From there, the complexity of the models was increased step-by-step. The first step was to stack an extra layer of RNN blocks on top of the first in the encoder. These models will be referred to as the double_(Bi)LSTM models. Thirdly, we build models inspired by the architecture used in the works of Wang et al. (40). Their model consist of two stacked LSTM blocks as well, but they embedded their trajectories in a fully connected layer first. The layout of this model can be seen in Figure 5.5.

Finally, we also deployed a sequence-to-sequence model which incorporates an attention mechanism as shown in Figure 5.3. The transformer network that was also used was a standard transformer as explained earlier and shown in Figure 5.4. Lastly, the MDN model used to generate trajectories two LSTM layers followed by the MDN-layer. Table 5.1 gives an overview of the models used. As seen in Table 5.1, the MDN model was not trained on the area based trajectories. This decision was made because the nature of MDN models is to approximate a continuous distribution and in the scenario where the locations are split into subareas, the output is not continuous.

Model	GPS	area
LSTM	x	x
BiLSTM	x	x
Seq2seq_LSTM	x	x
Seq2seq_BiLSTM	x	x
Seq2seq_double_LSTM	x	x
Seq2seq_double_BiLSTM	x	x
Seq2seq_double_LSTM_FC	x	x
Seq2seq_double_BiLSTM_FC	x	x
Seq2seq_attention	x	x
Transformer	x	x
MDN	x	

Table 5.1: Overview of the models used during this project

5.2 Training and evaluation

For each experiment run, the data has been split into three separate sets, a training set, a validation set, and a test set. The training data is the data that is fed into the models during the training phase of the process. Simultaneously, the validation set is used to track how well the models perform during training on unseen data. Finally, the test set will only be used to make the final predictions which will be used to evaluate the final performance of the models. A 60/20/20 split has been applied on the full dataset to create the separate sets.

The training of the models was done using the Adam optimizer. This choice as made as Adam is the most commonly used optimizer in the literature seen (16, 25, 29, 36). During training, we used the mean squared error (MSE) loss function for the models where the GPS coordinates were predicted, and the categorical cross-entropy for the area based models. The formulae for these loss functions can be found in Equation 5.9 and 5.10, respectively.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.9}$$

Where:

- y_i represents true values,
- $\hat{y}_i$ represents the predicted values, and
- n is the total number of datapoints.

$$\text{Categorical Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(\hat{y}_{ij}) \tag{5.10}$$

Where:

- y_{ij} represents true probability of sample i being of class j,
- $\hat{y}_{ij}$ represents the predicted probability of sample i being of class j,
- n is the total number of datapoints, and
- m the number of possible different classes.

During the training of the sequence-to-sequence models, we also apply an algorithm called teacher forcing. Teacher forcing is an algorithm that involves feeding the true values of a sequence back into an RNN model instead of the predicted values. This method can be used to accelerate the convergence of the loss during training, as it stabilizes the learning process by not allowing it to deviate too far from the ground truth (41).

5.3 Experimental setup

During this project we aim to predict the location as accurately as possible. To measure this, we make predictions up until 15 minutes into the future. As the basis for this project, all models were trained towards this task using an input trajectory of 45 minutes with no allowed idle window on the full dataset. From here, we performed four different experiments to see how different factors would influence the performance of the model.

First of all, since the data only records events when a vehicle sends a signal, it is too harsh to say that a trajectory stops when only a single minute of data is missing. In some cases the vehicle may be stationary for a few minutes, but still in be use. This would cause the trajectory to be registered as ended, even though that is not necessarily true. To counteract this, scenarios where trajectories were allowed an idle window were tested as well. The idle windows represent the amount of time we can have no event happening

Experiment	Idle window	Data used	Input sequence length
Base	10	All	45
Idle windows	1, 5, 10	All	45
Split by vehicle type	10	Split by vehicle type	45
Stationary trajectory	10	Split by vehicle type Fewer stationary trajectories	45
Input sequence length	10	Split by vehicle type	15, 30, 45

Table 5.2: Overview of the setup for each experiment

in our data without the trajectory being ended. This was done for idle windows of 5 and 10 minutes.

The second experiment was to split the data by vehicle type. This had to be done, as different vehicles portray vastly different behavior when we look at the trajectories. Even though this could possibly be taken into account if we introduce an information vector into the models, the decision was made to create separate models for each vehicle type as that would be an easier way to counteract this problem without increasing the complexity too much.

Thirdly, it was noticeable that a vast majority of the trajectories depicted a lot of stationary behavior, that is to say, a lot of consecutive events for a given vehicle were recorded at the same location. This may cause bias in the models towards predicting stationary behavior. However, this behavior is in general less interesting for this task, thus the decision was made to test what would happen to the predictions if we were to limit the number of highly stationary trajectories in the training data. This was done by calculating the percentage of stationary time steps in a trajectory and using a linear probability to decide whether or not a trajectory is used in training.

The final experiment was aimed at examining the influence of the input trajectory length on the results. One can imagine that longer input sequences may mean that the information is harder to decode and thus result in worse performance. We used input sequences of length 15 and 30, next to the already mentioned 45 minutes to see how shorter input sequences may change the results. Table 5.2 summarizes the setup for each experiment.

Parameter	Tested values	Used value
Learning rate	0.01, 0.001, 0.0001	0.001
Batch size	64, 128, 256	128
Number of epoch	15, 25	25
Hidden layer size	16, 32, 64, 128, 256	64

Table 5.3: Overview of the hyperparameters used.

5.4 Hyperparameters

The hyperparameters were chosen using a gridsearch method. Parameter tuning is an important optimization step as without proper tuning, models may be more susceptible to either over- or underfitting. This in turn makes the models worse at generalization and thus can cause worse performance on the test data. An overview of the hyperparameters that were tuned, with the tested values, and the final choice is given in Table 5.3. The parameters were tuned on the base scenario where all vehicle were taken into account at once with input trajectories of 45 minutes an idle window of 10 minutes.

5.5 Evaluation metrics

In testing, the models will not compared using the loss functions. Since we want to evaluate the results in a temporal manner, the loss functions, which are taken over the full sequences, do not say much about the performance. For example, a model that is very accurate in the first five time steps, may have way worse results that other models later on in the trajectory. The loss function in training averages this, however in practice, it may be more desirable to have more accurate results in earlier time steps. Towards this end, we computed metrics separately for each time step for our models. For the models that predict the GPS locations directly, the mean absolute difference (in km) was taken as metric. This metric is shown in Equation 5.11.

$$MD = \frac{1}{n}\sum_{i=1}^{n} d(y_i - \hat{y}_i) \tag{5.11}$$

Where:

- $d(y_i - \hat{y}_i)$ represents the distance between the true values y_i and the predicted values $\hat{y}_i$, However, since the data is originally presented as normalised longitude, latitude pairs, the distance values cannot be taken directly from the outputs. After denormalizing our data, we can use the Haversine formula (Equation 5.12) to calculate the true distances in km (10).

$$\begin{aligned} a &= \sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat}_1)\cdot\cos(\text{lat}_2)\cdot\sin^2\left(\frac{\Delta\text{long}}{2}\right) \\ c &= 2\cdot\text{atan}^2\left(\sqrt{a}, \sqrt{1-a}\right) \\ d &= R\cdot c \end{aligned} \tag{5.12}$$

For the area based models, we use categorical accuracy as evaluation metric. The categorical accuracy measures the frequency of how often the true values and predicted values match. The accuracy measure can simply be described as the percentage of correctly predicted values as seen in Equation 5.13.

$$\text{Categorical accuracy} = \frac{1}{n}\sum_{i=1}^{n}(y_i = \hat{y}_i) \tag{5.13}$$

6

Results

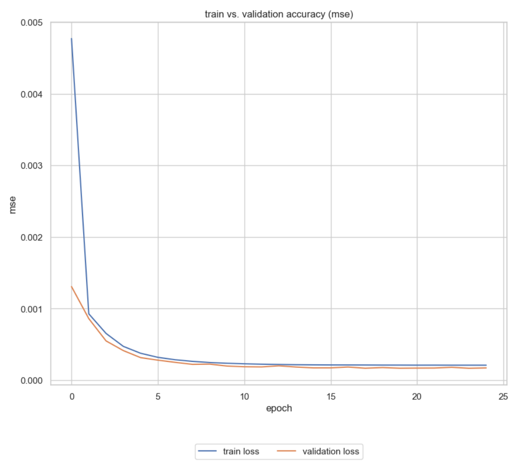
In this chapter we will discuss the results obtained by the trained models. These results were all obtained by testing the models on a separate test dataset that was left out of the training of the model. We will elaborate on the results for each experiment and highlight the key outcomes.

There are four different experiments that will be considered. First of all, the results achieved in the base scenario, as described in Section 5.3, will be highlighted. Next, we will delve into the effect of changing the idle windows for the trajectories. Subsequently, we will discuss the effect of separating the data and training a model for each vehicle type apart. The third experiment portrays the effect of a decreased number of stationary trajectories in the training data. And finally, we will analyze the effect of different lengths of input sequences.

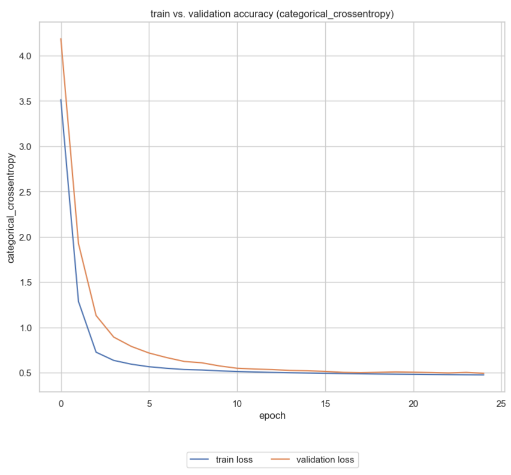
The results of each model will be discussed based on how well they are able to predict up to 15 time steps into the future. For all scenarios we will consider both the direct GPS location predictions, as well as the area based predictions. For the GPS locations we will be considering the true difference in distance between prediction and ground truth, for the area based predictions, the accuracy will be used as metric.

6.1 Base scenario

In the base scenario, all of the models were trained on the training data as a whole. This setup is mainly used to see the impact of the changes proposed in the attempted experiments on the results. In Figure 6.1 the learning curves for both the GPS model and



(a) Learning curve for the GPS model.

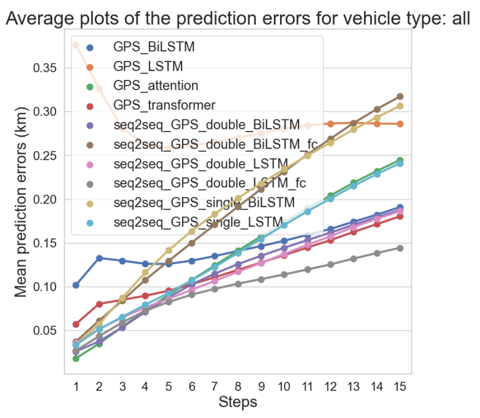


(b) Learning curve for the area model.

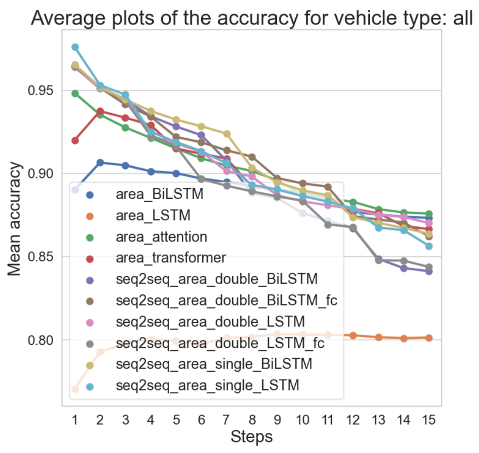
Figure 6.1: Learning curves for the single LSTM seq2seq model for both the GPS model (a) and the area model (b).

the area model are shown. The learning curves in Figure 6.1 show that the training process is mostly converse towards a stable training and validation loss. One might still argue that there is still a somewhat descending trend ongoing for the losses shown, especially in Figure 6.1 (b) for the area based model. However, a significant amount of extra training epochs would be needed for the possible decrease in learning loss to have a large impact. Thus, the choice of using 25 epochs to train the data seems to be an acceptable choice in this scenario. Some of the other learning curves do show a behavior that is characteristic for overfitting. For the basic LSTM model and the transformer model, the validation loss does not seem to decrease over time whereas the training loss does, widening the gap between the two. The plots for these learning curves can be found in Appendix C.

Figure 6.2 shows the results from the prediction made by the models. For both the GPS predictions, as well as the area based predictions, the basic LSTM model used as baseline. Both plots also clearly show that the predictions are more accurate at the first few time steps compared to later time steps. For the models besides the LSTM model, there appears to be no clear deciding factor that makes one of the models stand out more than the other. In both cases, all models, both sequence-to-sequence, as well as the transformer and the regular BiLSTM model are very close together in terms of performance.



(a) Distance errors for the GPS predictions.



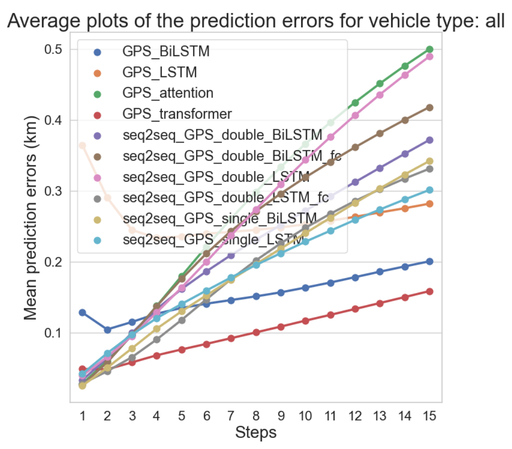
(b) Accuracy of the area predictions. per vehicle type.

Figure 6.2: Prediction performance for each model for the base scenario. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.

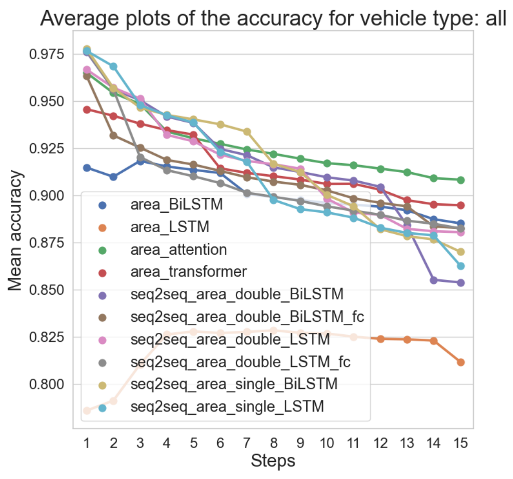
6.2 Idle windows

Three different idle windows were used in this project to see what the results would be if we allowed for more slack in the time the trajectories were allowed to have no recordings before being cut short. The base experiment uses a window of 10 minutes. However, we also experimented with windows of five minutes and a window of one minute, the latter meaning that there was no break allowed in between instances. Figures 6.3 and 6.4 show the performance of these two experiments, respectively.

Interestingly, Figures 6.3 and 6.4 show that even though the basic LSTM model still has the worst performance in the first few steps, when we look at both GPS predictions, the baseline LSTM model seems to perform average on the later steps in the prediction. At time step 15, the LSTM is actually able to predict, on average over 100 meter more accurate than some of the other models. Another observation is that in the area models, the accuracy seems the decrees slower as the idle window gets smaller. Where we saw a decrease of approximately 10% in the base scenario, this only seems to be around 5% for some of the models trained on data with an idle window of 1. But, when combining these results together with the values in Table 4.2, this may seem to be an occurrence caused by the higher stationary percentages.

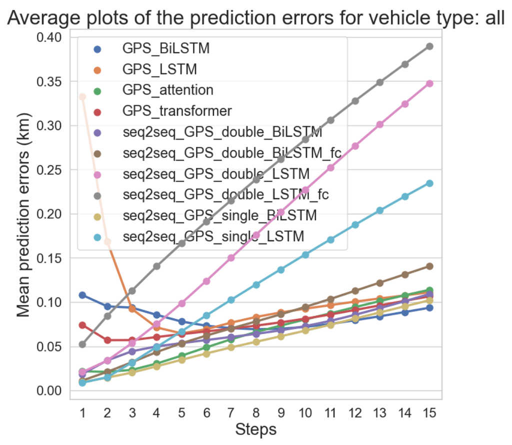


(a) Distance errors for the GPS predictions.

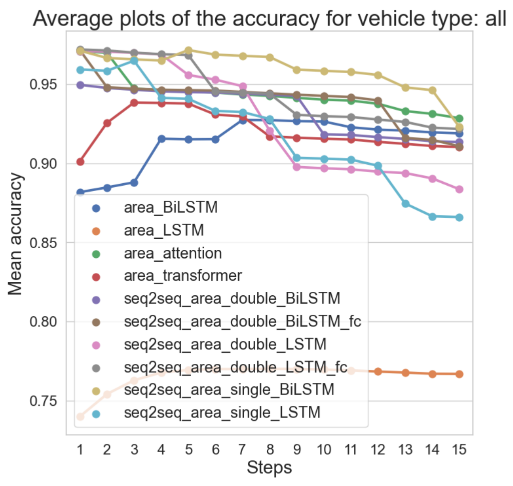


(b) Accuracy of the area predictions. per vehicle type.

Figure 6.3: Prediction performance for each model with an idle window of 5. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.

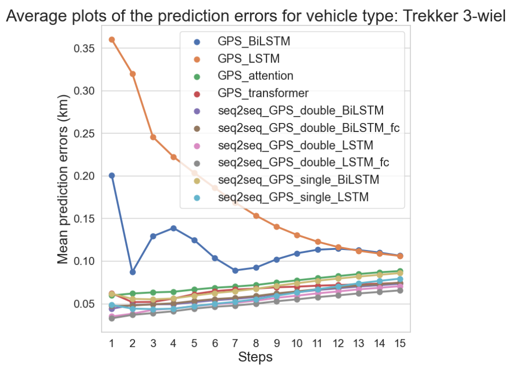


(a) Distance errors for the GPS predictions.

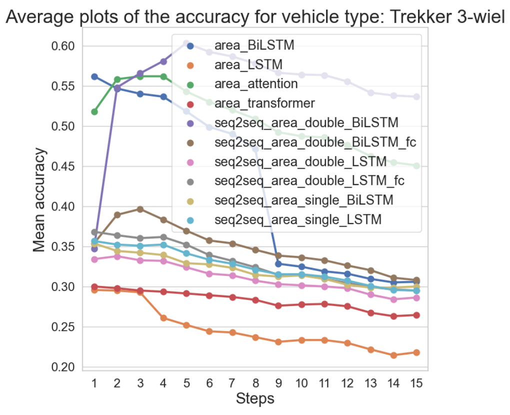


(b) Accuracy of the area predictions. per vehicle type.

Figure 6.4: Prediction performance for each model with an idle window of 1. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.



(a) Distance errors for the GPS predictions.



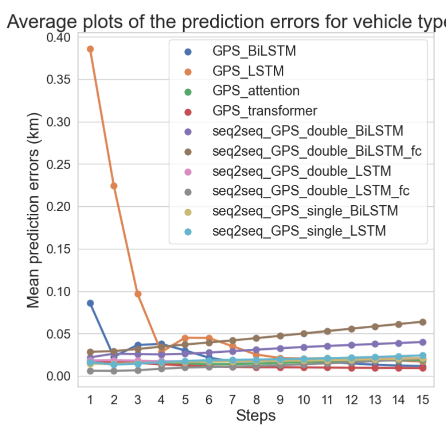
(b) Accuracy of the area predictions. per vehicle type.

Figure 6.5: Prediction performance for each model trained on data of only 'Trekker 3-wiel' vehicles. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.

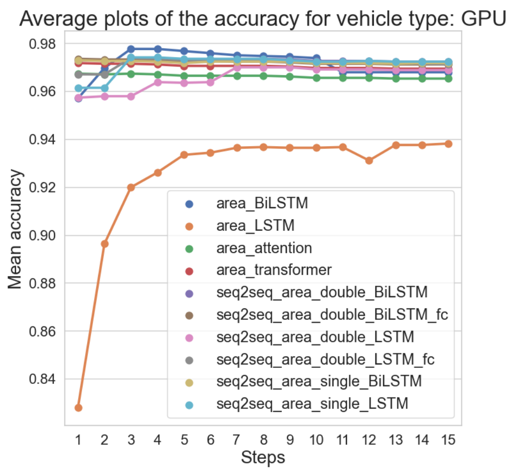
6.3 Split by vehicle type

The next step was to train models for each vehicle type separately. This decision was made because, as we have seen previously, different vehicle types may portray different behaviors in their trajectories. Thus, training a model for each vehicle type separately may lead to more accurate predictions. Also, since Figure 4.10 (b) shows that there is a relation between the the ratio of stationary activity in trajectories and the trajectory types. As the results in the previous experiment led us to believe that stationary trajectories may be too prevalent, splitting the vehicles by type when training may solve this problem. We will compare the results from vehicle types 'Trekker 3-wiel' and GPU as they are shown to be respectively the least and most stationary vehicles according to Figure 4.10 (b). The resulting model performances for 'Trekker 3-wiel' can be found in Figure 6.5 and those for the GPU vehicles can be found in Figure 6.6.

When comparing Figure 6.5 to Figure 6.2, we clearly see improvement when looking at the GPS predictions. As Figure 6.5 shows that most models are able to predict the location 15 steps ahead with an accuracy of between 50 and 100 meters on average. However, the area based prediction seems to be all over, with only a single accuracy score higher than 0.6 being seen in the plot. On the other hand, the GPU performance seen in Figure 6.6 shows clear improvements in performance. With the location being consistently predicted



(a) Distance errors for the GPS predictions.



(b) Accuracy of the area predictions. per vehicle type.

Figure 6.6: Prediction performance for each model trained on data of only GPU vehicles. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.

within 10 meters by some models and the accuracy for the area based models being above 0.96 continuously for all models except the baseline LSTM model.

Even though it is interesting to see the how the different models perform in terms of metrics, the plots themselves do not translate well to actually real world interpretation of the results. To get an idea of the real implications of the model outcomes, in Figure 6.7 some of the actual predicted trajectories are shown. These predictions were made on trajectories for vehicles of type 'Trekker 3-wiel'.

Figure 6.7 does seem a somewhat cluttered, but we can make some clear observations based on what we see. First of all, we see that each of the different models predicts similar continuations for the trajectories, regardless of the input. The light-orange trajectories for example all seem to move in a north-eastern direction at first, before making a turn and moving back towards where the vehicle came from. Similarly, we can observe that each of the green trajectories are predicted to first go north, before turning towards the west. Even though they are not shown for the other experiments, similar observations could be made when plotting the predictions for those models.

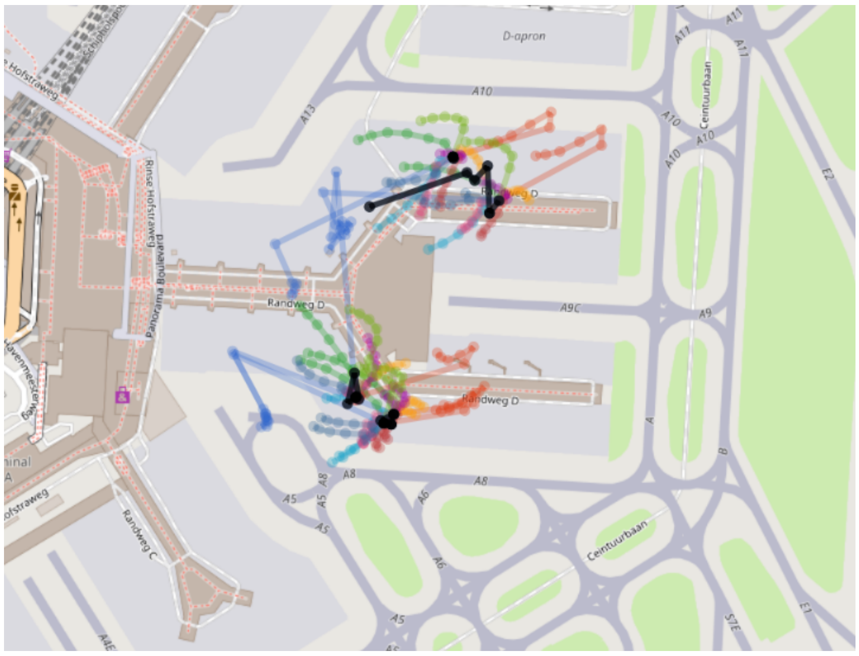
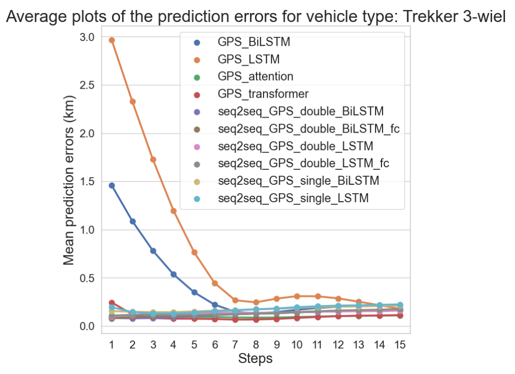


Figure 6.7: Actual and predicted trajectories based on vehicle type 'Trekker 3-wiel'. The black trajectories indicate the true trajectories, whereas each colored trajectory represents a prediction of a given model.

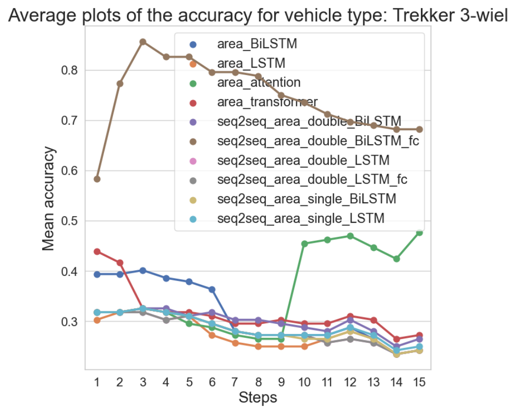
6.4 Stationary trajectories

As splitting the trajectories by vehicle type when training the models clearly revealed a relation between the performance of the models and the percentage of stationary behaviour in the data, the next experiment was to see what would happen when fewer stationary trajectories were allowed in the training data. This was done by applying a simple linear filter which lowered the probability of a trajectory being used the more stationary sequences were seen in a trajectory. Since the splitting of the data by vehicle type did seem to improve the performance of the models for the more active vehicle types as well, the decision was made to keep the vehicle types separate in this experiment and the next. For comparison we will again, look at the results for the 'Trekker 3-wiel' vehicles and the GPUs. The performances can be found in Figure 6.8 and 6.9, respectively.

Figure 6.8 reveals that reducing the number of trajectories in the training data does not appear to do much good for the performances. In Figure 6.8 (a) we see that the baseline models are off by on average 1.5 and 3 kilometers respectively for the BiLSTM and the LSTM model. The other models do appear to still be able to predict somewhat reasonable predictions where the 15th step predictions are all on average still around 250 meter off or

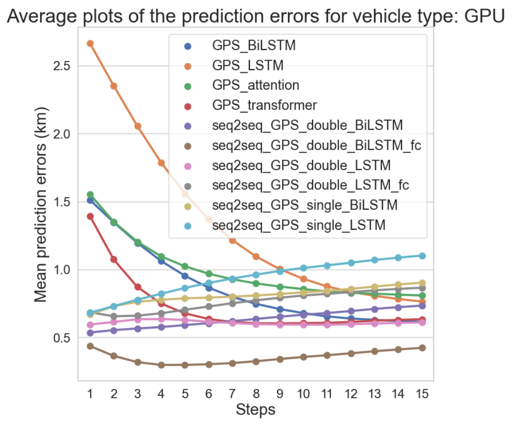


(a) Distance errors for the GPS predictions.

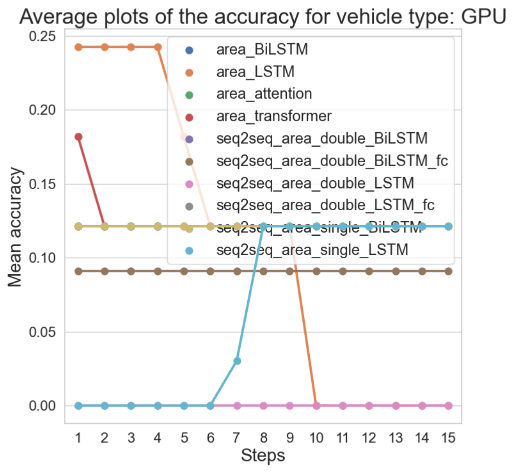


(b) Accuracy of the area predictions. per vehicle type.

Figure 6.8: Prediction performance for each model trained on data of only 'Trekker 3-wiel' vehicles with fewer stationary trajectories. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.



(a) Distance errors for the GPS predictions.



(b) Accuracy of the area predictions. per vehicle type.

Figure 6.9: Prediction performance for each model trained on data of only GPU vehicles with fewer stationary trajectories. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.

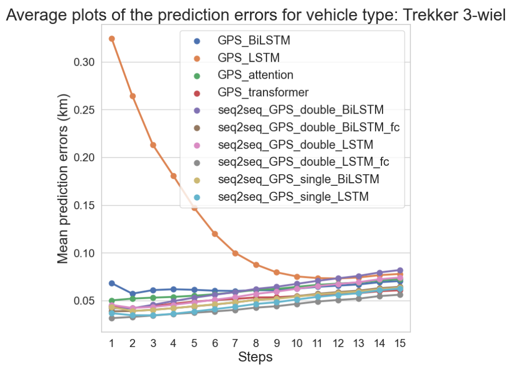
less. An interesting development in the area based model, is that the sequence-to-sequence model with a double BiLSTM decoder and a preceding fully connected layer actually is actually able to predict with at least 70% accuracy for all but the first time step. This is approximately doubles that of the other models.

Cutting the number of instances in the training data down did not work well for the model that predict the movement of the GPU vehicles. The best GPS predictions are still almost half a kilometer off after 15 time steps. The area based model shows interesting straight patterns in the accuracies. This is a clear indication that there was too little data left over to properly train a model and make predictions. Also all of the accuracies shown are below 0.25 which is a significant decrease compared to what was seen before.

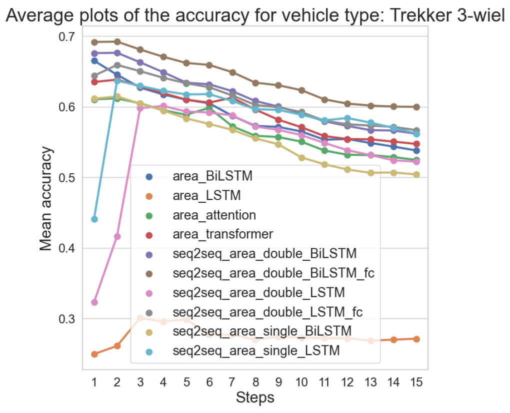
6.5 Input sequence length

The final experiment run, was to see how the results would change if we used shorter input sequences to predict the same length of output as before. As limiting the number of trajectories with high stationary behavior, did not appear to have a possible effect on the predictions, we did go back to using all trajectories for this experiment. However, we did keep the split in vehicle types. In Figure 6.10 and 6.11, the performance metrics are shown for vehicles of type 'Trekker 3-wiel' with an input trajectory length of 30 and 15 minutes, respectively.

Both figures show that the LSTM model still under-performs compared to the others. Especially in the first step of the GPS base locations, the distance errors of the LSTM model appears to be on average around 200 meters more compared to the other models. The GPS predictions do appear to have very similar results with both input length showing a slight upward trend in distance error from between 0 and 50 meters to 50 to 100 meters. In Figure 6.10 (b) we notice that for the area based prediction, most of the models start between an accuracy of 60 to 70 percent, decreasing with around 10 percent of the time steps. We do see that the sequence-to-sequence models that use regular LSTMs in the encoder, do have significantly more trouble making accurate predictions in the first time step. Figure 6.11 shows that the area based predictions all start with an accuracy of approximately at 70 or just percent, which in this scenario, also decrease at a similar rate as with the input length of 30 data points.

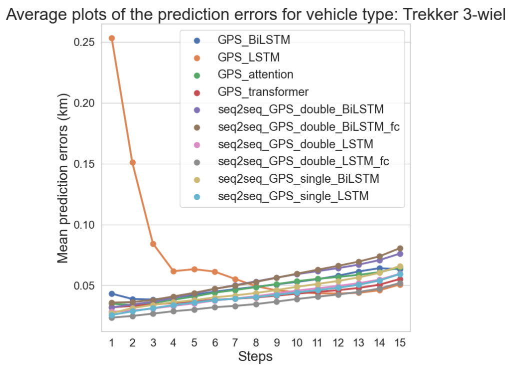


(a) Distance errors for the GPS predictions.

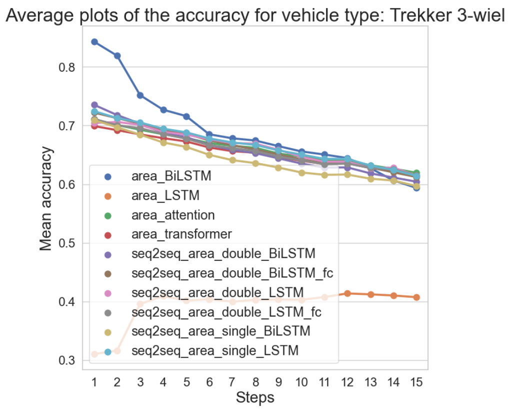


(b) Accuracy of the area predictions. per vehicle type.

Figure 6.10: Prediction performance for each model trained on data of only 'Trekker 3-wiel' vehicles with input length 30. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.



(a) Distance errors for the GPS predictions.



(b) Accuracy of the area predictions. per vehicle type.

Figure 6.11: Prediction performance for each model trained on data of only 'Trekker 3-wiel' vehicles with input length 15. (a) shows the differences in distance (in km) for the difference in GPS location and (b) shows the accuracy at each step for the area based prediction.

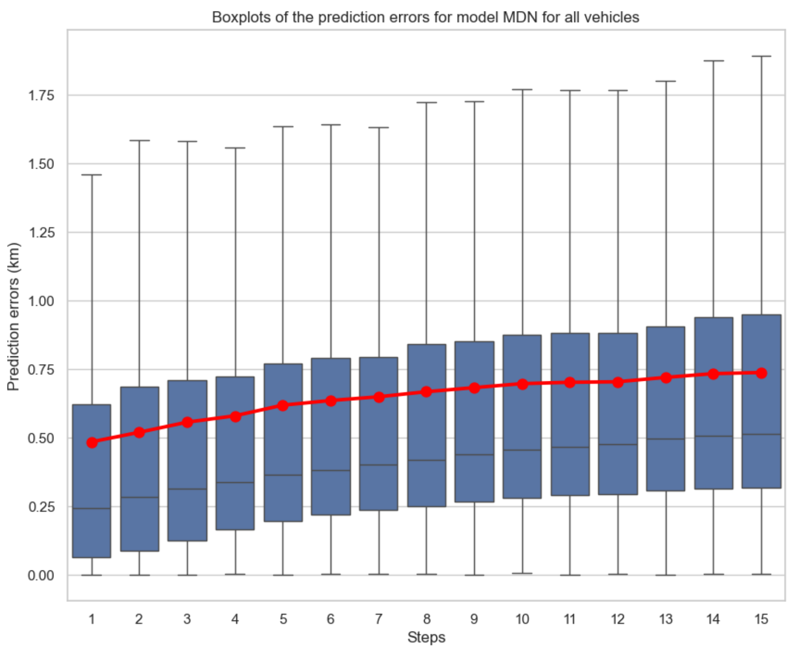


Figure 6.12: Distance error for trajectories generated by the MDN model.

6.6 MDN trajectory generation

The final model to be discussed is the result of the MDN trajectory generation. Figure 6.12 shows the distribution of the distance error for each time step between the true trajectories, and the generated continuations of the given input trajectories.

The distances are on average further off for the MDN model than the results we have seen for the sequence-to-sequence models. However, contrarily to the encoder-decoder models, it is important to consider that generative models are not used to directly map an input into a given desired output. Instead, this model tries to approximate the underlying distribution of the input data as a combination of Gaussian distributions.

To see how the data is modelled by this model, a single generated sequence is shown in Figure 6.13. The generated trajectory shows very erratic movements all over the Schiphol premises. Since this model was trained on the full data, it is thus not hard to explain the large differences in distances between the generated data and the true data shown in Figure 6.12, as the data consist of a lot stationary behavior, which is the contrary of what is seen here. Because of this, the generated trajectories do not represent the data well. However, as seen previously in Figure 4.7, the erratic movement is not necessarily uncharacteristic for some of the seen behavior in the data.

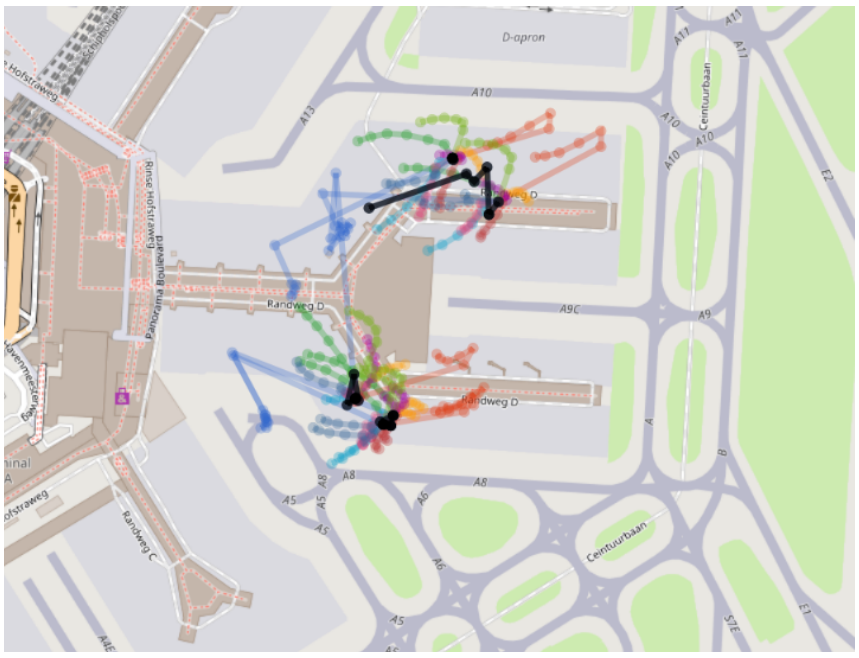


Figure 6.13: A single trajectory generated by the MDN model.

7 Discussion

In the previous we have seen how different setups for the experiments could change the performance of the resulting trajectories. From these results, we can draw a few main conclusions.

1. Splitting the data for each vehicle type will improve prediction performance for specific data points.
2. Having more diverse trajectories, in terms of movements, in the training data may result in overall better performance.
3. Shorter input sequences may lead to more accurate outputs.
4. The resulting predicted continuations of the trajectories do not resemble the trajectories.

As the first point states, we saw that by training models separately on data for each vehicle type, we could improve the accuracy of the resulting predictions. This makes sense as different vehicles are used for different tasks, and thus portray different behavior. Furthermore, since the data was not well balanced in terms of the number of trajectories per vehicle type, some of the behaviors were more prevalent in the data, causing the models to favor that behavior in the trajectory prediction. This would make it so that vehicles for which the normal movement is relatively stationary, some movement is still being expected by the models. The other way around, vehicles that one would expect to move around the premises more, appear more suppressed. Thus using the data for different vehicle types separately has proven to aid in diminishing this effect.

In a similar trend, we can conclude from the results that having more diversity in the behavior betrayed by the trajectories may improve the performance for some of the models. Especially when we look at the more actively moving vehicles, in Figure 6.8 we saw that at least one of the models was able to outclass the other and achieve higher accuracies when making predictions. Coincidentally, this was also the largest model trained, in terms of complexity and the number of parameters. This could be seen as an indication that it is possible for these types of models to make more accurate predictions than what was shown during this research, but that more complex architectures may be needed to achieve this. In turn, this also is a logical result as the erratic movements some of the vehicles portray, may be too hard to discern properly for the smaller models.

The most surprising result found was seen in Figures 6.10 and 6.11. Here we found that smaller input sequences led to more accurate performances from the models. Two possible underlying reason could be causing this. First of all, it may be that the encoders were not able to properly encode the information given to them by the longer input sentences, resulting in less accurate prediction given by the decoder. Secondly, since the trajectories were shown to contain a lot of stationary behavior, decreasing the input size may have caused the active movements to be better separated from the stationary, causing more diversity in the input trajectories. This would mean that there is more useful information to be learned by the model.

Unfortunately, as shown in Figure 6.7, the accuracies shown in the figures exhibited in this report, do not directly translate to realistic trajectories predicted by the models. It is strongly suspected that this is cause by the large imbalance between active movement and stationary trajectories in the data. As seen in 4.2, a vast majority of the trajectories portray stationarity. As this is also the information that is fed into the models during training, it is highly likely that the models suffered from exposure bias and thus are unable to properly discern when a vehicle should be expected to move larger distances.

It is hard to say of one specific model used for the sequence-to-sequence predictions outperformed the others. Both the transformer and the encoder-decoder models have shown similar results in when talking about accuracy of the models. Each model trained did outperform the basic LSTM model in most, if not all, scenarios presented. This means that there is merit to using more complex models for this tasks. However, there will always be a trade-off between complexity and computational speed. As indicated before, more complex models may be able to better grasp the underlying information from the data, but more research will be needed to conclude if the extra complexity is worth the outcome in terms of performance.

These results seem to differ from what was found in previous research. In Chapter 3, we explained how different project have already shown that similar approaches can be effective for trajectory prediction. However, since none of those projects were either on an openly traversable area, like the airport aprons, or on a way larger scale, like naval predictions, it is hard to directly compare the results. For example, in ship trajectory prediction, locations that were off by one kilometer could be seen as accurate (36), whereas in our case, that would mean we expect a vehicle to be on the other side of the premises. In terms of actual distance, our model do seem to be able to predict within a similar distance as shown in previous research on which our approach was partly based (40).

The generative approach towards finding a continuation for the trajectories that was shortly explored with MDN models did show, complete contrarian behavior compared to the sequence-to-sequence predictions. In the MDN model, the generated trajectories seemed to be too chaotic. However, admittedly, the options explored for the generation of models was severely limited. Thus it is unjust if we conclude based solely on the results seen here, that his approach is not a viable one. Additionally it was also to be expected that these models would not result in accurate reconstructions of the true trajectories, as generative models like the MDN model are more aimed towards modelling the underlying distribution of the data instead of being able to map a given input into a desired output.

8

Conclusion

This project has focused on answering the question of to what extent different deep learning methods can be used in order to augment a digital twin environment for the GSE processes with the ability to do future predictions for traffic present on the Schiphol airport premises.

8.1 Conclusions

For this task, multiple models were trained on different experimental setups to find out to what extend they are able to predict the continuations of trajectories for different GSE vehicles. Through data analysis and preparation, a process of extracting trajectories from raw location data was defined and these trajectories were used to describe and predict the movements of the vehicles. The first model used was a simple LSTM which was seen as a baseline to compare the other models to. The other models we used were a collection of different sequence-to-sequence encoder-decoder models and a transformer network. Between these networks, there was not a single standout model that was able to consistently make better predictions than the others.

By doing multiple experiments we were able to draw some conclusions about how the setup of the experiment influences the performance. We have shown that not training the models on the full data, but training a separate model for each vehicle type, made to so that the predictions for the isolated vehicles improved. Also using a process that increases the diversity of the data has been shown to potentially improve the forecasting performances. Thirdly, we have seen that using shorter input sequence allowed us the improve on the accuracy of the predictions.

Unfortunately, even though our models were statistically able to perform predictions with relative accuracy, the predicted trajectories were shown to not be a realistic representation of actual movement of the GSE vehicles. This was most likely caused by the inherent characteristics of the data that was used to train the models. However, the complexity of the setting of the airport premises may also have had a significant influence on the results.

8.2 Future research

Even though the results showed that the models trained during this project do not translate to a directly applicable use for the prediction of the trajectories of GSE vehicles, and thus the traffic on the Schiphol grounds. We did provide groundwork toward the implementation of a digital twin that is usable to gain insights in future states of the environment. There are two main directions in which future research can make steps towards an end product.

First of all one may attempt to build further from this project and further develop these or similar models. in that case, we would recommend to critically consider what data is to be used. We expect that the characteristics of the current data are one of the major limiting factors in this project. Thus making sure that the data used consists of more diversity would possibly enhance the performance models significantly.

Another possibility would be to augment the data used by using extra data that describe factor that have an influence on the movements around the premises. One could for example think about somehow training models that also take flight schedules and/or inter-vehicle interactions into account. Since the area used contained both restricted movement over road networks, but also open movement around the gates, the decision was made to train models that are able to predict without these movement restrictions. However, looking into defining the roadnetwork as graph and using graph based models to do further forecasting may also be a viable option for developing a suitable product to help in the Autonomous Operations project.

References

[1] *Research Landscape for the National Airspace System 2020 – 2030.* Federal Aviation Administration, 2020. 6

[2] *Sustaining your world Vision and strategy towards the most sustainable airports 2022.* Royal Schiphol Group, 04 2022. 5

[3] *Een autonome luchthaven in 2050.* 2023. URL `https://www.schiphol.nl/nl/innovatie/blog/een-autonome-luchthaven-in-2050/`. 5

[4] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016. doi: 10.1109/cvpr.2016.110. 9

[5] Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional flow variational autoencoders for structured sequence prediction, 2020. 12

[6] Christopher M. Bishop. Mixture density networks. 1994. URL `https://publications.aston.ac.uk/id/eprint/373/`. 33

[7] Leyla Cakir and Berkant Konakoglu. The impact of data normalization on 2d coordinate transformation using grnn. *Geodetski Vestnik*, 63:541–553, December 2019. doi: 10.15292/geodetski-vestnik.2019.04.541-553. 16

[8] Álvaro Castro-González, Masahiro Shiomi, Takayuki Kanda, Miguel Salichs, Hiroshi Ishiguro, and Norihiro Hagita. Position prediction in crossing behaviors. pages 5430–5437, 10 2010. doi: 10.1109/iros.2010.5651144. 8

[9] Szu-Tung Chen, Gülçin Ermiş, and Alexei Sharpanskykh. Multi-agent planning and coordination for automated aircraft ground handling. *Robotics and Autonomous Systems*, 167:104480, 2023. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2023.104480. URL `https://www.sciencedirect.com/science/article/pii/S0921889023001197`. 6

[10] Nitin R Chopde and Mangesh Nichat. Landmark based shortest path detection by using a* and haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2):298–302, 2013. 40

[11] Nachiket Deo and Mohan M. Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. Ieee, June 2018. doi: 10.1109/ivs.2018.8500493. URL `http://dx.doi.org/10.1109/IVS.2018.8500493`. 10

[12] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Tracking by prediction: A deep generative model for mutli-person localisation and tracking, 2018. 8

[13] Michael Grieves. Sme management forum completing the cycle: Using plm information in the sales and service functions. 10 2002. 6

[14] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary perspectives on complex systems: New findings and approaches*, pages 85–113, 2017. 6, 7

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`. 27

[16] Dapeng Jiang, Guoyou Shi, Na Li, Lin Ma, Weifeng Li, and Jiahui Shi. Trfm-ls: Transformer-based deep learning method for vessel trajectory prediction. *Journal of Marine Science and Engineering*, 11(4), 2023. ISSN 2077-1312. doi: 10.3390/jmse11040880. URL `https://www.mdpi.com/2077-1312/11/4/880`. 12, 36

[17] ByeoungDo Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference*

on Intelligent Transportation Systems (ITSC), pages 399–404, 2017. doi: 10.1109/itsc.2017.8317943. 9

[18] Woohyun Kim, Yerim Han, Kyoung Kim, and Kwan-Woo Song. Electricity load forecasting using advanced feature selection and optimal deep learning model for the variable refrigerant flow systems. *Energy Reports*, 6:2604–2618, 11 2020. doi: 10.1016/j.egyr.2020.09.019. 29

[19] Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Depeng Jin, and Yong Li. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution, 2021. 11

[20] Zhi Liu, Jixin Bian, Deju Zhang, Yang Chen, Guojiang Shen, and Xiangjie Kong. Dynamic multi-view coupled graph convolution network for urban travel demand forecasting. *Electronics*, 11(16), 2022. ISSN 2079-9292. doi: 10.3390/electronics11162620. URL `https://www.mdpi.com/2079-9292/11/16/2620`. 11

[21] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015. 30

[22] Thomas Machl, Andreas Donaubauer, and Thomas H Kolbe. Planning agricultural core road networks based on a digital twin of the cultivated landscape. *J. Digit. Landsc. Archit*, 4:316–327, 2019. 7

[23] Kristina Marintseva, Gennadiy Yun, and Sviatoslav Kachur. Resource allocation improvement in the tasks of airport ground handling operations. *Aviation*, 19:7–13, 03 2015. doi: 10.3846/16487788.2015.1015291. 5

[24] Anton Milan, S. Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. doi: 10.1609/aaai.v31i1.11194. URL `https://ojs.aaai.org/index.php/AAAI/article/view/11194`. 8

[25] Jeba Nadarajan and Rathi Sivanraj. Attention-based multiscale spatiotemporal network for traffic forecast with fusion of external factors. *ISPRS International Journal of Geo-Information*, 11(12), 2022. ISSN 2220-9964. doi: 10.3390/ijgi11120619. URL `https://www.mdpi.com/2220-9964/11/12/619`. 10, 36

[26] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678, 2018. doi: 10.1109/ivs.2018.8500658. 9

[27] Yan Qin, Yong Liang Guan, and Chau Yuen. Spatiotemporal capsule neural network for vehicle trajectory prediction. *IEEE Transactions on Vehicular Technology*, 72(8): 9746–9756, 2023. doi: 10.1109/tvt.2023.3253695. 10

[28] Rahuljha. Lstm gradients, Jun 2020. URL `https://towardsdatascience.com/lstm-gradients-b3996e6a0296`. 27

[29] Luca Rossi, Andrea Ajmar, Marina Paolanti, and Roberto Pierdicca. Vehicle trajectory prediction and generation using lstm models and gans. *Plos One*, 16(7):1–28, 07 2021. doi: 10.1371/journal.pone.0253868. URL `https://doi.org/10.1371/journal.pone.0253868`. 11, 36

[30] Michael Schmidt, Philipp Nguyen, and Mirko Hornung. Novel aircraft ground operation concepts based on clustering of interfaces. 09 2015. doi: 10.4271/2015-01-2401. 5

[31] Zhiyuan Shi, Min Xu, Quan Pan, Bing Yan, and Haimin Zhang. Lstm-based flight trajectory prediction. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi: 10.1109/ijcnn.2018.8489734. 16

[32] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292, 2019. doi: 10.1109/BigData47090.2019.9005997. 28

[33] Apoorv Singh. Trajectory-prediction with vision: A survey, 2023. 8

[34] Yongfeng Suo, Wenke Chen, Christophe Claramunt, and Shenhua Yang. A ship trajectory prediction framework based on a recurrent neural network. *Sensors*, 20(18), 2020. ISSN 1424-8220. doi: 10.3390/s20185133. URL `https://www.mdpi.com/1424-8220/20/18/5133`. 9

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. 28

[36] Kristian Aalling Sørensen, Peder Heiselberg, and Henning Heiselberg. Probabilistic maritime trajectory prediction in complex scenarios using deep learning. *Sensors*, 22 (5), 2022. ISSN 1424-8220. doi: 10.3390/s22052058. URL `https://www.mdpi.com/1424-8220/22/5/2058`. 12, 36, 55

[37] Diego Tabares and Felix mora camino. Aircraft ground handling: Analysis for automation. 06 2017. doi: 10.2514/6.2017-3425. 5

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 32, 33

[39] Elena Voita. Sequence to sequence (seq2seq) and attention, Nov 2023. URL `https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html`. 29, 31

[40] Chujie Wang, Lin Ma, Rongpeng Li, Tariq S. Durrani, and Honggang Zhang. Exploring trajectory prediction through machine learning methods. *IEEE Access*, 7: 101441–101452, 2019. doi: 10.1109/access.2019.2929430. 9, 35, 55

[41] Wanshun Wong. What is teacher forcing?, Oct 2019. URL `https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c`. 37

[42] Hongwu Yuan, Guoming Xu, Zijian Yao, Ji Jia, and Yiwen Zhang. Imputation of missing data in time series for air pollutants using long short-term memory recurrent neural networks. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, UbiComp '18, page 1293–1300, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359665. doi: 10.1145/3267305.3274648. URL `https://doi.org/10.1145/3267305.3274648`. 14

[43] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. 12

[44] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2020. doi: 10.1109/tits.2019.2935152. 10

[45] Jiawei Zhu, Qiongjie Wang, Chao Tao, Hanhan Deng, Ling Zhao, and Haifeng Li. Ast-gcn: Attribute-augmented spatiotemporal graph convolutional network for traffic forecasting. *IEEE Access*, Pp:1–1, 02 2021. doi: 10.1109/access.2021.3062114. 11

Appendix A

Overview of vehicle types

Company Car: Regular cars or minivans that are driven by KLM personnel around the premises of the airport.

Dispenser: Vehicles able to dispense fuel from underground systems into an aircraft.

Ground Power Unit (GPU): Vehicles which are able to provide power to parked aircraft.

Kleine Tankwagen: Vehicles used to carry fuel and fuel aircraft.

Powerstow LLH: Vehicles equipped with a snake-type rollertrack conveyor belt to help ground personnel load and unload luggage.

Powerstow SLH: A smaller version of the Powerstow LLH.

Toiletwagen: Vehicles used to empty and refill lavatories on board the planes.

Transportband: Vehicles carrying conveyor belts used to load and unload baggage onto an aircraft.

Transporter: Vehicles used to transport containers and pallets from dollies to maindeck loader and vice versa.

Trap pax: Vehicles equipped with a staircase to help with the boarding process for passengers.

Trekker 3-wiel: Small vehicles used to tug the dollies that carry loose baggage, mailbags, and other small cargo.

Trekker 4-wiel: Vehicles used to tug the dollies that carry loose baggage, mailbags, and other small cargo.

Trekker misc: Vehicles used to tug the dollies that carry loose baggage, mailbags, and other small cargo.

Vliegtuigtrekker: Pushback tugs used to move aircraft to and away from the gates upon arrival or departure.

Waterwagen: Vehicles used to store and provide water to the aircraft.

Unknown: Token used to indicate that it is unknown what vehicle type is linked to the event in the data.

Appendix B

Plots of the exploratory analysis for different vehicle types

In this appendix, we present plots illustrating the temporal distributions and utilization of specific vehicles within the observed dataset. To prevent redundancy, the plots for only vehicle types 'unknown', 'Company Car', and 'GPU' are shown.

Unknown

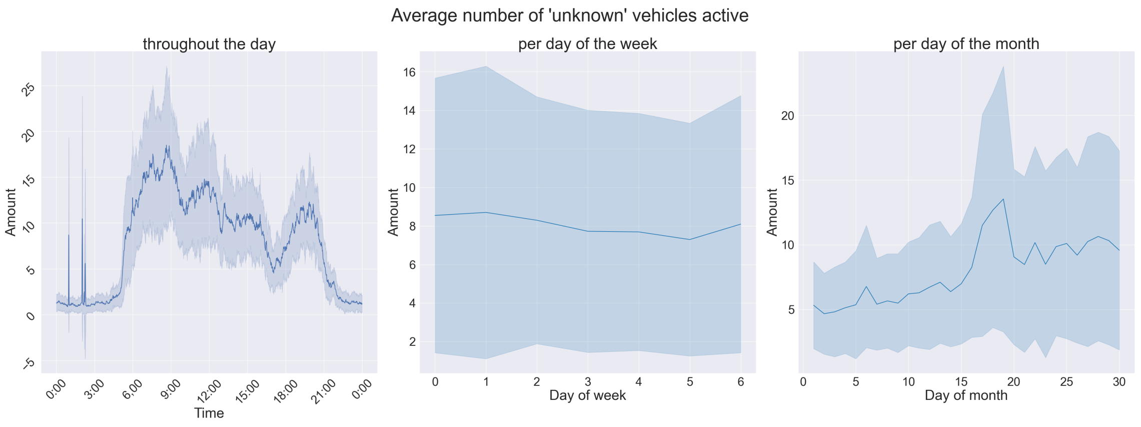


Figure B.1: Temporal distribution of the events in the data of December 2023 filtered by vehicle type 'unknown', represented per minute of the day (left), day of the week (middle), and day of the month (right). The blue line represents the mean values at those times, whereas the light blue area portrays the standard deviation.

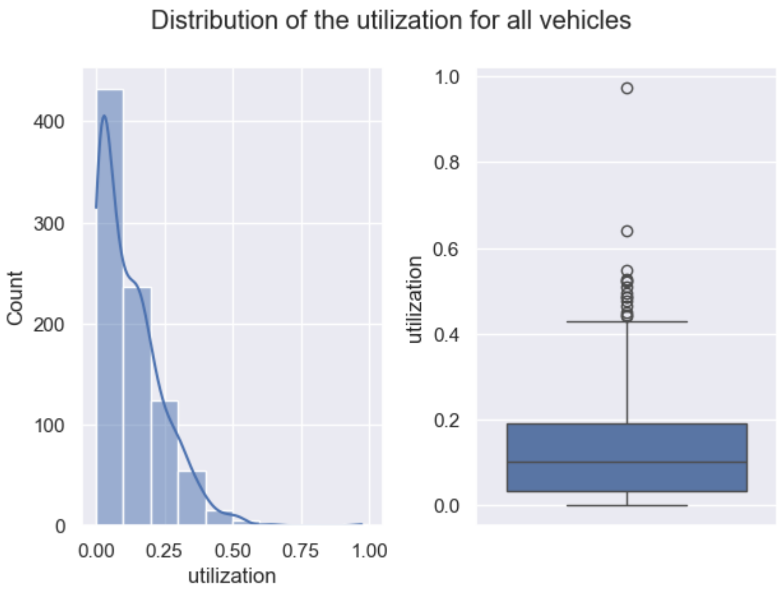


Figure B.2: Utilization of the vehicles of vehicle type 'unknown'.

Company Car

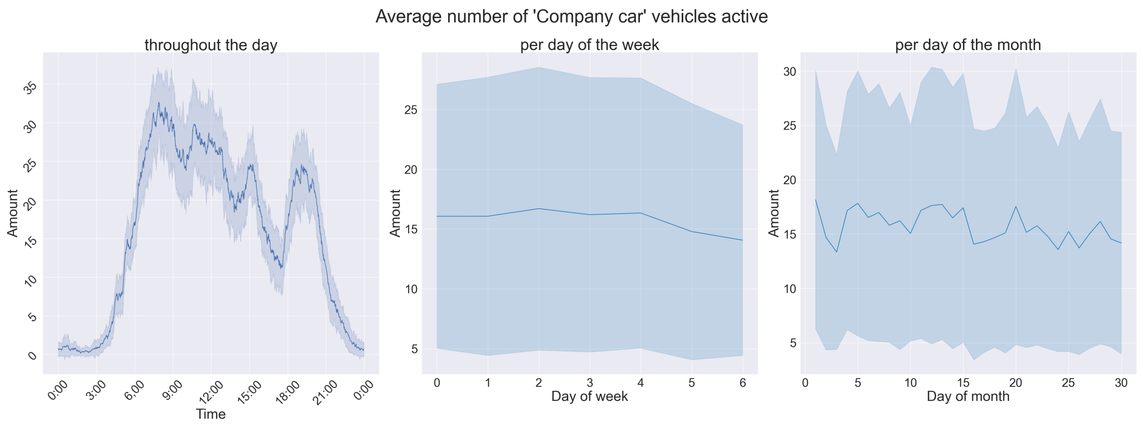


Figure B.3: Temporal distribution of the events in the data of December 2023 filtered by vehicle type 'Company Car', represented per minute of the day (left), day of the week (middle), and day of the month (right). The blue line represents the mean values at those times, whereas the light blue area portrays the standard deviation.

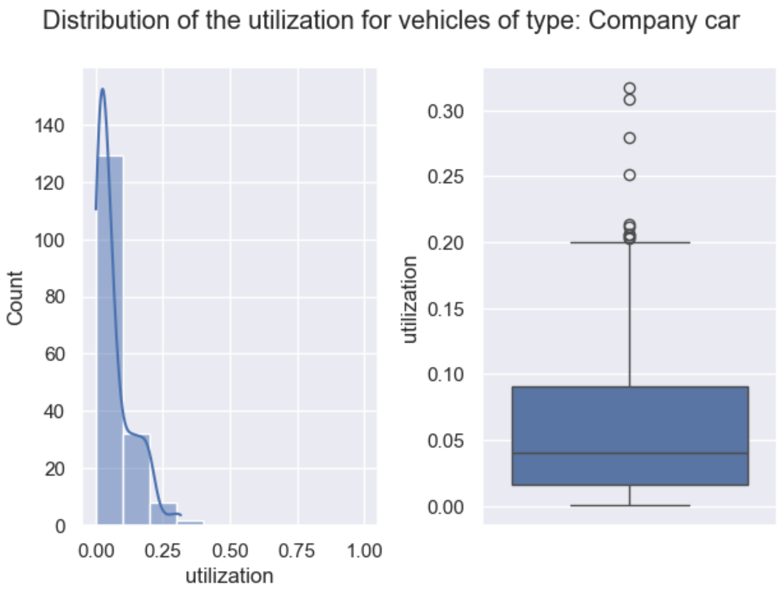


Figure B.4: Utilization of the vehicles of vehicle type 'Company Car'.

GPU

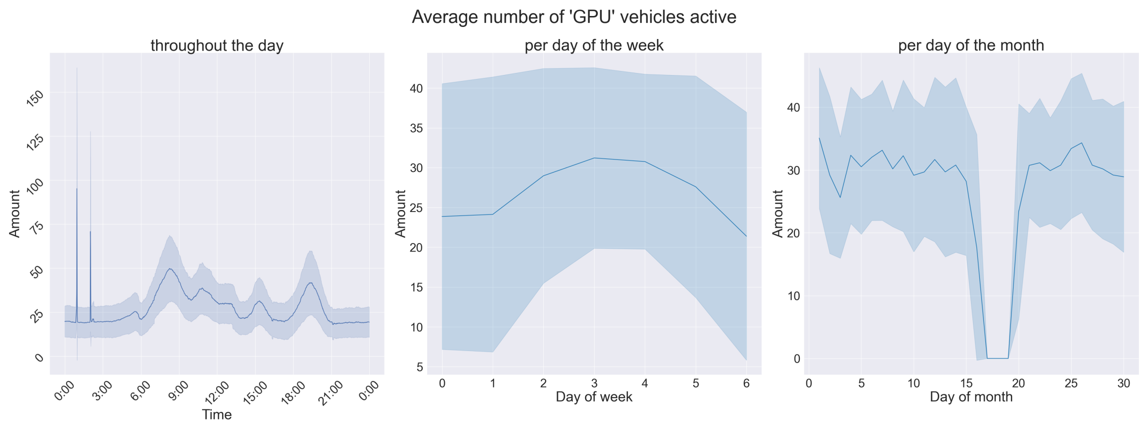


Figure B.5: Temporal distribution of the events in the data of December 2023 filtered by vehicle type 'GPU', represented per minute of the day (left), day of the week (middle), and day of the month (right). The blue line represents the mean values at those times, whereas the light blue area portrays the standard deviation.

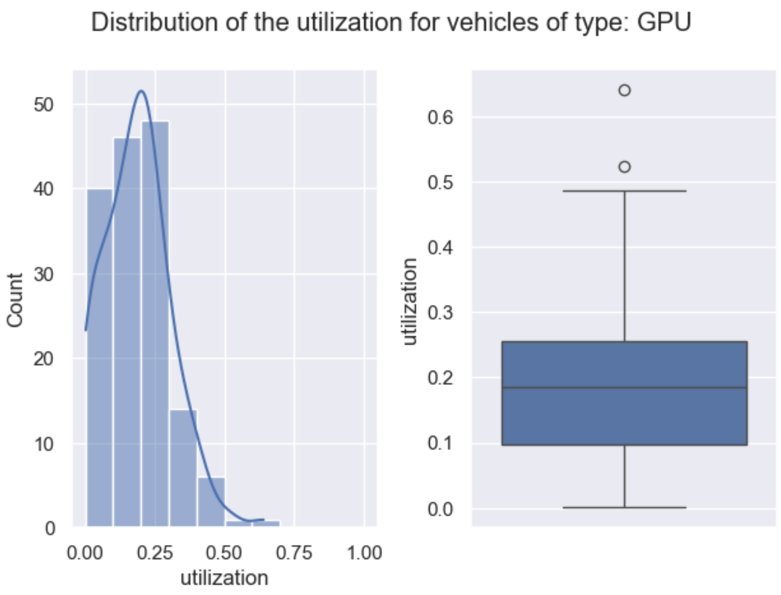
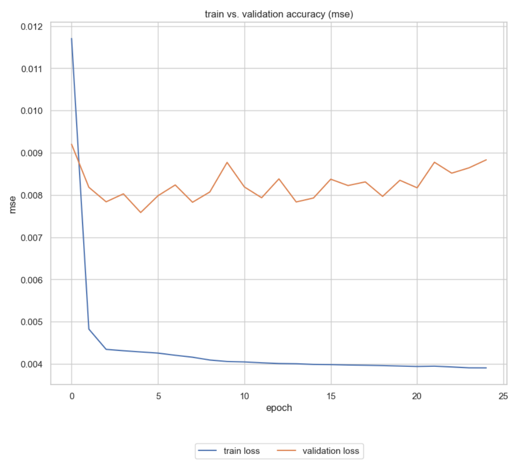


Figure B.6: Utilization of the vehicles of vehicle type 'GPU'.

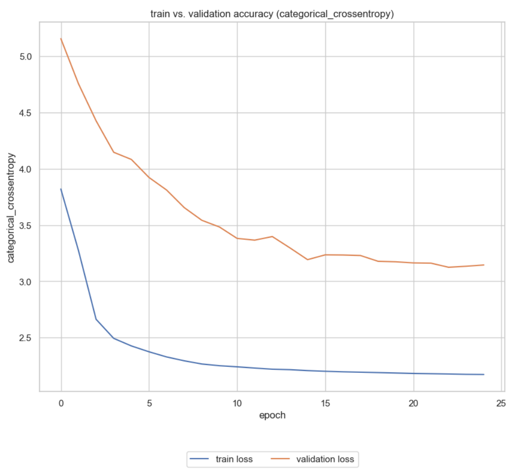
Appendix C

Learning curves for the base scenario

In this appendix, we provide the plots depicting the learning curves for the LSTM model and the transformer model. These plots represent the learning process for models trained on the base scenario.

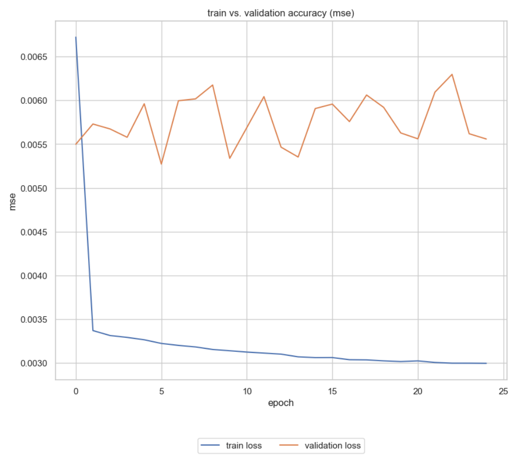


(a) Learning curve for the GPS model.

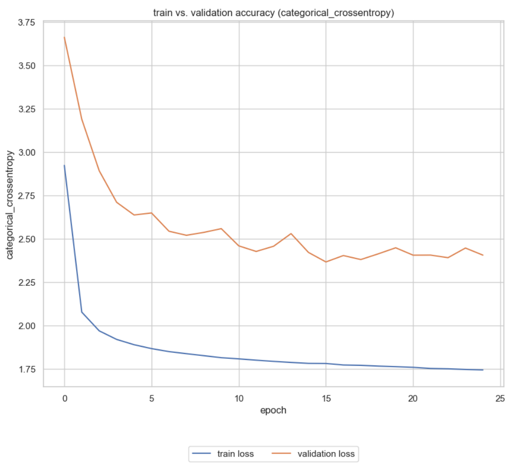


(b) Learning curve for the area model.

Figure C.1: Learning curves for the LSTM model for both the GPS model (a) and the area model (b).



(a) Learning curve for the GPS model.



(b) Learning curve for the area model.

Figure C.2: Learning curves for the transformer model in the base scenario for both the GPS model (a) and the area model (b).