Skoltech

Skolkovo Institute of Science and Technology

MASTER'S THESIS

Vehicle-Pedestrian Motion Prediction

Master's Educational Program: Data Science

Student: _____ Mohammed Deifallah
signature

Research Advisor: _____ Gonzalo Ferrer
signature

PhD, Assistant Professor

Moscow 2022

Skoltech

Skolkovo Institute of Science and Technology

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Прогнозирование Движения Пешеходов и Транспортных Средств

Магистерская образовательная программа: Математика и Компьютерные Науки

Студент: _____ Дейфалла Мохамед
подпись

Научный руководитель: _____ Гонзало Феррер
подпись
д.ф.-м.н., Доцент

Москва 2022

Vehicle-Pedestrian Motion Prediction

Mohammed Deifallah

Submitted to the Skolkovo Institute of Science and Technology on June 18, 2022

ABSTRACT

This work is dedicated to the development of a classification-based pipeline for the trajectory prediction, with focus on the behavior of autonomous vehicles and all of their surrounding objects. It is not common to tackle this problem using classification techniques, so this thesis proposes a new way to answer such a question, and whether it can outperform the other widespread used methods. In general, this study shows how such a pipeline can be built, including the structure of the dataset, design of the model architecture, and the loss functions and metrics used for this kind of problems.

In order to produce a reliable pipeline for such a problem, we suggest this classification model, SkolNet. We use a Deep Learning [21] (DL) model that uses a semantic map and a state history of all objects in the scene. The data required for the proposed model can be collected with any method, such as cameras, radars, or light detection and ranging (LiDAR). In this work, we decided to use the largest public dataset to date, Shifts [25].

Provided that this hypothesis has shown signs of success in some special cases, it proves that this way is promising and will give future prosperity on a larger scale (e.g., using more past steps from the agents or even multiple predictions simultaneously). This will directly reflect on the industry of self-driving vehicles and driverless agents.

Keywords: trajectory prediction, deep learning, classificationو mobile robotics, self-driving, autonomous vehicles

Research advisor:
Name: Gonzalo Ferrer
Degree, title: PhD, Assistant Professor

Contents

Chapter 1
Introduction

Relevance. Motion Prediction has been one of the research topics that has tremendous surveys and works since the start of the 21st century because of its various applications. For the sake of producing a reliable autonomous vehicle, this thesis is dedicated to vehicle-pedestrian trajectory prediction, so that the driverless agent can foresee the movements of all surrounding objects given their past steps and the map context.

Main purpose of the research. This work is mainly devoted to reproduce the model from [29], which is a classification model for trajectory prediction. The limitations of this model should be analyzed in order to mitigate them in the proposed model. Subsequently, this research work aims to produce a robust classification pipeline for motion prediction.

Scientific novelty. As it will appear later in the literature review, classification is rarely used to address this kind of problems. If used, it becomes experimental and is discussed along with the other more common approaches. Hence, this work will discuss a model architecture [29], which is completely designed for a classification algorithm, which will be discussed later in detail.

Practical value. As this thesis started as a part of a joint project between SberAutoTech and Mobile Robotics Lab from Skoltech, the results of this thesis are supposed to directly impact the industrial market because SberAutoTech has their own unit FLIP, shown in 1.1, that is yet being tested in closed tracks and I hope this may help them release it in open streets.

Figure 1.1: Prototype of the fully autonomous vehicle, FLIP

Individual contribution. As a member of the trajectory prediction team in Mobile Robotics Lab, I worked on CoverNet [29] model as mentioned before in purpose statement, following the GitHub repository [27] of the original authors.

Let us summarize the main contributions of this work as follows:

- Supporting a variable number of channels instead of sticking to only 3-channel RGB images.
- Providing an easy-to-follow implementation of the *trajectory set generator* (TSG).
- Adding non-linearity to the original model. Kindly refer to this pull request (PR) on the original repository.
- Adding batch normalization [18] (BN) to the linear layers.

Statements for defense. How to formulate the problem of trajectory prediction as a **classification** problem? Can this classification technique outperform the regression techniques which are commonly used in this research area?

Literature review. Trajectory prediction problem is mainly solved by regression methods. So, it would be important to enrich it with new approaches and techniques, especially if they (e.g., classification) are seldom used. Thus, it would be remarkable to apply classification to this problem in order to analyze the advantages and limitations of it.

This problem is suitable for regression methods in nature because it tries to extrapolate the vehicle history of steps to produce the future ones.

Another limitation in this field is that the self-driving vehicles (SDVs) are only trained and monitored within bounded areas and closed track, and we still do not get them in the open streets. Even if Yandex SDG launched the 1st prototype of their robotaxi project in 2017 in streets, and they started to provide delivery services in Russia and USA by 2020, they never published any information about the disengagement rate. As missing this important piece of information in their reports, we want to continue working on this problem to have a reliable SDV with clear reports. For more information about the definition of disengagement, refer to this article.

The problem of trajectory prediction has been extensively investigated and is still being studied, since it is a cornerstone in the industry of vehicular automation. In this chapter, we are going to briefly discuss some of the recent works that have had impactful achievements in the research field with a variety of techniques and topics such as Conditional Variational Auto-Encoder (CVAE) [2, 23], kinematic models [8] and Imitation Learning (IL) [1, 7, 30].

In 2014, the survey [24] was compiled as an overview of the current motion modeling approaches. Figure 1.2 summarizes the main 3 families which this survey discusses in details. In short, the 1st family, physics-based motion models, is the simplest one because it only applies the classical laws of physics, mathematics, and kinematics, using the low-level attributes of motion like velocity, weight, and friction. This, in turn, makes this motion model face a challenge to be used in practice for only short-term predictions because it does not take into account the other external factors. Here comes the 2nd family, Maneuver-based motion models. This family anticipates the future trajectories by analyzing the driving patterns from the past steps in order to match the upcoming movements. This can be done in several methods, for example agglomeration (clustering), or Gaussian Processes (GP) [33]. However, this family assumes that the vehicles are independent, so the interventions between different vehicles would not be considered. In addition to this, it is not generalizable enough to suit all road layouts because the model is trained on a specific intersection geometry. Finally, the 3rd family, Interaction-aware motion models, takes into account the interaction between entities as shown by the name. This family has two main approaches: prototype

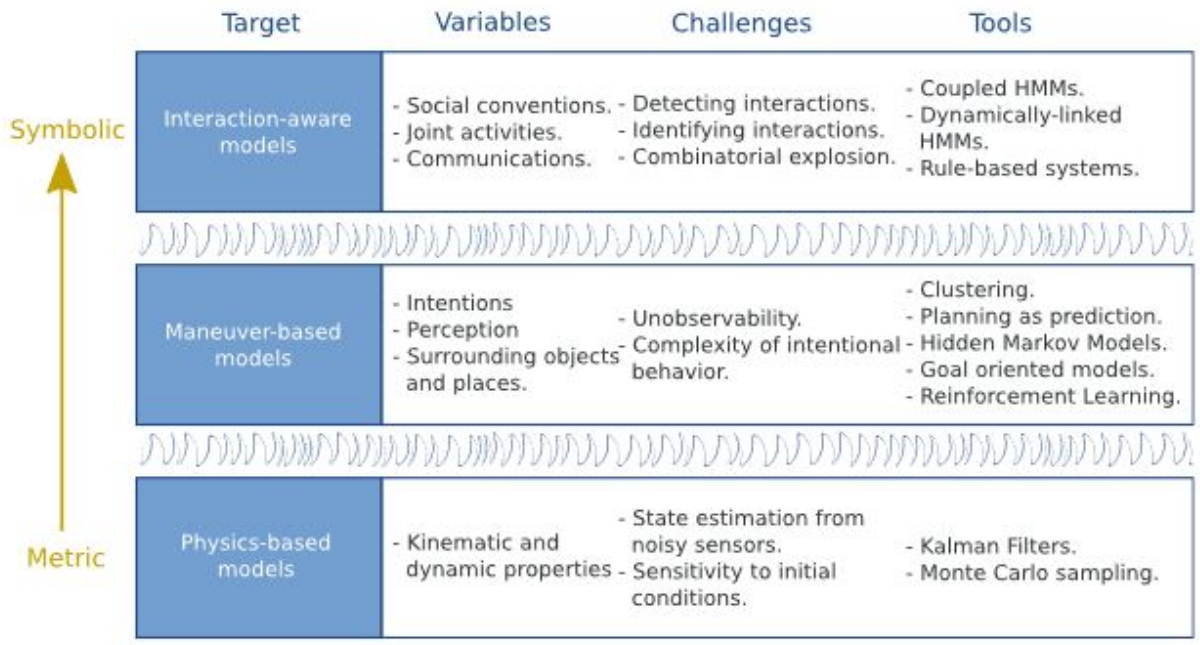

Figure 1.2: Main families of motion modeling. Figure from [24]

trajectories, and Dynamic Bayesian Networks (DBNs) [19]. Without going into details, both approaches search a pairwise combinations of all trajectories to filter the impossible ones. This may sound closer to [3], which presents various heuristic algorithms for path reduction. It evaluates a set of paths in order to maintain the diversity and robustness of a set of trajectories by pruning a large set of candidate paths to a more tractable set size. Although this family is more reliable than the previous two ones, it is computationally expensive because it tries to exhaustively search all the possible scenarios, which is not suitable for real-time tasks.

Many works, which tackled this problem, only focused on the dataset collection task to establish libraries of thousands of scenes, provide an API as a development kit for data manipulation and transformation, and to have some baseline solutions as a side work. ArgoVerse [6] is a collection of two datasets for motion forecasting and 3D tracking, gathered by a group of autonomous vehicles in two different cities. Motion Forecasting dataset contains more than 300,000 5-second scenes, side by side to high-definition (HD) maps with geometric and semantic metadata. Then comes NuScenes [4], the 1st dataset to have the full suite of sensors of cameras, radars, and LiDAR with a full view of 360 degrees. As an example, Figure 1.3 shows 6 different views of the same scene, annotated with scene description. This dataset includes approximately 20-second-long 1000 scenes, annotated with 3D ground-truth bounding boxes for almost 23 different classes and 8 attributes. In 2020, Lyft [15] was released on a large scale, containing more than 1,000 recorded hours of data using a fleet of 20 vehicles over 4 months. Furthermore, it contains the most detailed semantic map to date, having over 15,000 annotations, including lane segments and road polygons. Most importantly, the authors provided L5Kit, a public development kit in Python, for efficient data access and visualization. In addition to these contributions, they also trained two baseline models for two different tasks. Adopting ChaufferNet [1] helps solve the motion planning problem. It was about predicting the future poses of the SDV in a 5-second time horizon. This approach mainly relies on *IL* [17]. The following figure 1.4 shows the high-level diagram of the used model architecture. It is designated to receive 224×224 rasterized bird-eye-view (BEV) input images through a convolutional feature network called *FeatureNet*, to produce contextual features to be consumed by *AgentRNN*, a recurrent network, predicting successive points in the future, alongside the bounding box of the vehicle. The model also makes use of the concept of co-training with the other two networks. *PerceptionRNN* is another recurrent network which predicts the future locations of all the

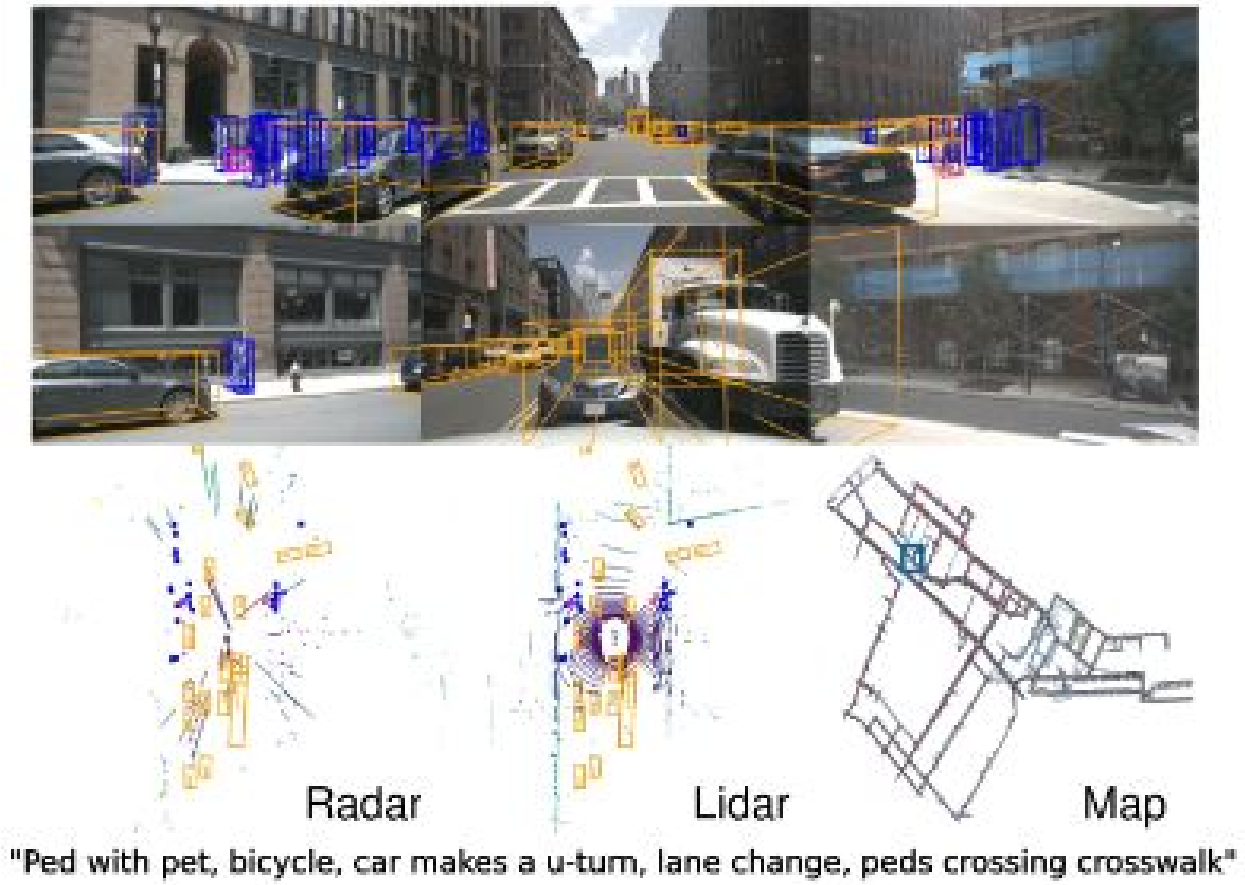

Figure 1.3: Example from NuScenes origina paper [4]

other surrounding objects, while the Road Mask Network (*RMN*) filtrates the feasible or available road areas. Applying *IL* to this approach, they used 3 imitation losses for the agent position, box prediction and heading respectively, as Figure 1.5 shows, where $\mathcal{H}(a, b)$ is the cross-entropy function, $P_k(x, y)$ is the probability distribution over the spatial coordinates for every iteration k, $B_k(x, y)$ is the predicted box after a per-pixel sigmoid activation function, θ_k is the box heading angle, and the superscript *gt* denotes the corresponding ground truth values. Recently, Shifts [25] dataset has been released as a group of datasets for 3 different tasks, including *vehicle motion prediction*, dedicated for a challenge, targeting the research of distributional shift and uncertainty estimation of the collected data, and how this affects the robustness and reliability of the trained models. This dataset contains almost 600,000 scenes span 6 locations in two countries, four weather conditions, and 3 seasons of the year, where each scene lasts for 10 seconds, and the last inclusive 5 seconds represent the ground truth predictions. In addition to providing a toolkit for data manipulation, the authors provide pretrained baseline models as examples for training. Those models are directly based on [7] and [30], which are variants of Robust Imitative Planning (RIP) [11]. The 1st variant, RIP-BC [7], is a simple behavioral cloning network with an RNN backbone as a conditional *IL* approach. On the other hand, the 2nd variant is an autoregressive flow-base deep imitative model (RIP-DIM) [30]. Briefly, it is a combination of *IL* and model-based Reinforcement Learning (RL) in order to combine the advantages of both of them additively, and to produce a model which is tolerant enough to wrong goals with negative rewards.

Another important approach proposed by [5] is a regression Gaussian Mixture (GM) model with RNN, which targets predicting the future states of the agent. Given the past trajectories of all agents in the scene and a detailed HD map, this work outputs a probability distribution of the future trajectories and a weighted set of trajectories from this distribution. Figure 1.7, as usual, shows that

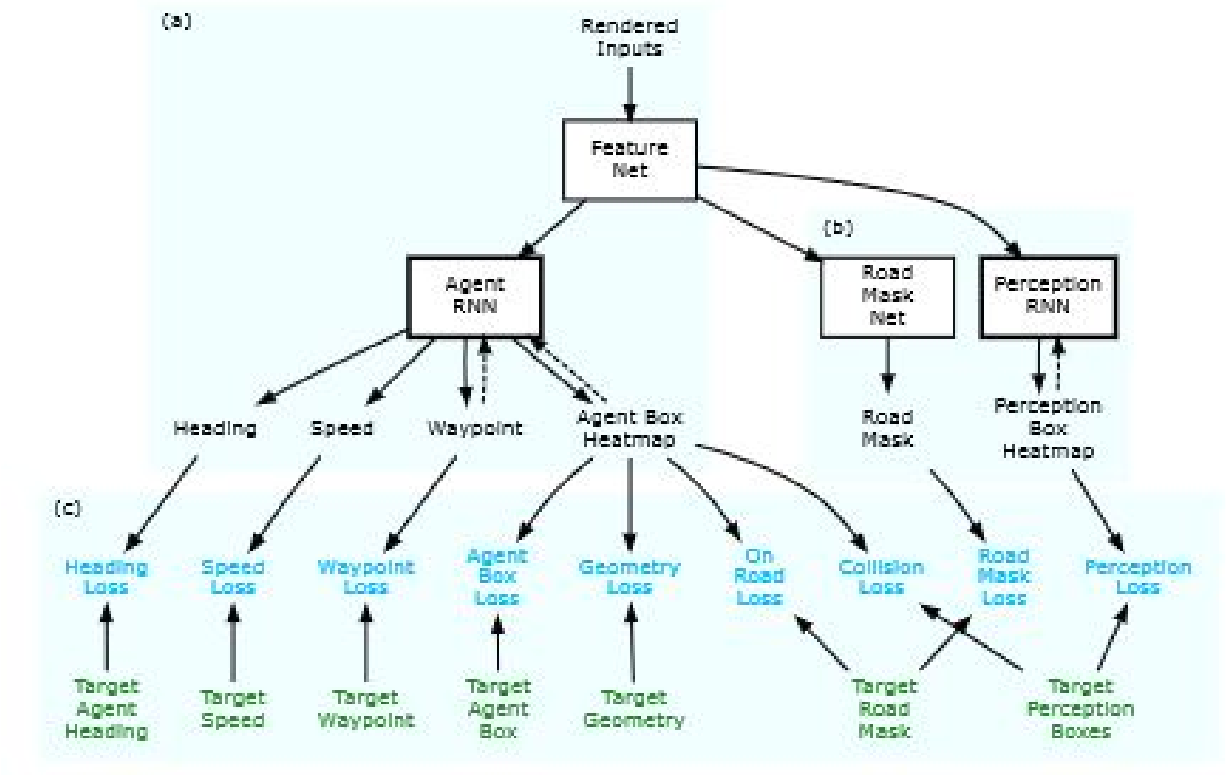

Figure 1.4: ChauffeurNet model architecture. Figure from [1]

$$\mathcal{L}_p = \mathcal{H}(P_k, P_k^{gt})$$
$$\mathcal{L}_B = \frac{1}{WH} \sum_x \sum_y \mathcal{H}(B_k(x,y), B_k^{gt}(x,y))$$
$$\mathcal{L}_\theta = \left\| \theta_k - \theta_k^{gt} \right\|_1$$

Figure 1.5: Imitation losses from [1]

MultiPath [5] model consumes the scene representation through a CNN feature extractor to have encoded mid-level features, which help in the next stage to regress over a fixed set of K anchor trajectories.

Moreover, a Long Short-Term Memory (LSTM) encoder-decoder model was introduced in the paper [10]. This approach was loosely based on maneuver-based models, which are previously mentioned in [24]. It is an interesting model, shown in Figure 1.8, which is divided into three main components with important responsibilities. The 1st module is the LSTM encoder with shared weights, which takes the past history of the agent to learn the dynamics of the object motion, where the weight sharing guarantees the correspondence among all the components in the scene over time. The in-between component, which is the main contribution of this work [10], is the convolutional social pooling layer. This component is responsible for capturing the interdependencies of all the surrounding vehicles motions in the scene by pooling the LSTM states into a *social tensor*, as they called it. The most important part in this component is that it ends with convolutional and pooling layers, *convolutional social pooling*, in lieu of fully-connected layers, in order to avoid unfolding the spatial configurations of the scene. Instead, CNN layers facilitate learning some local features

Dataset	Scene Length (s)	# Scenes Train	Dev	Eval	Total Size (h)	Avg. # Actors
Argoverse	5	205,942	39,472	78,143	320	50
Lyft	25	134,000	11,000	16,000	1,118	79
Waymo	20	72,347	15,503	15,503	574	-
Shifts	10	500,000	50,000	50,000	1,667	29

Figure 1.6: Datasets comparison. Table from [25]

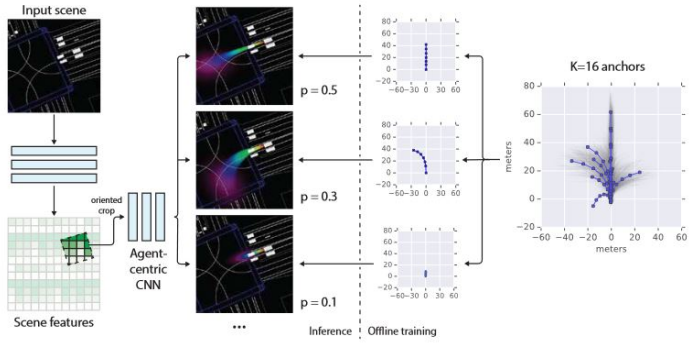

Figure 1.7: MultiPath Architecture from [5]

in this *social tensor*, while the max pooling layer adds the translational invariance property to the model. At last, the 3rd component is the LSTM decoder that outputs a multi-modal distribution of the future motion. In more details, it has two softmax layers to produce the lateral and longitudinal probabilities from a bivariate Gaussian distribution, whose parameters are resulted by the LSTM.

1.1 Baseline models

This work is a part of motion prediction project, conducted by a team from Mobile Robotics Lab. This section gives a brief about the other models used by team members, and they will be used as baseline models to compare the work with them.

The 1st model is RIP-BC [7] mentioned above, which is a simple RNN-based BC network. BC can consume large-scale multimodal data from great fleet of agents, which makes it able to successfully learn end-to-end policies. However, it still suffers from the unrealistic scenarios because it doesn't take the physical map into consideration.

The 2nd model is RIP-DM [30], which is an autoregressive flow-based DIM. It slightly has the same setup as BC, but it changed the GRU decoder to an autoregressive flow decoder to predict trajectories. Still, it has the same drawback, which it sometimes goes off the road.

The 3rd baseline is Set-Transformer [22], an attention-based permutation-invariant network to capture all the interaction and dependencies among the input elements (static and time-dependent features). It follows the encoder-decoder architecture with attention mechanism in both components.

Y-net [26] is the 4th baseline model in this work. Y-net is a trajectory forecasting network. The significant novelty of this model is that it uses two different decoders: one is responsible for the distribution of trajectory prediction, and the other is responsible for the distribution of waypoints and routes. Figure 1.9 shows the diagram of Y-net model.

Finally, Trajectron++ [31] is the last baseline model. It is a graph-structured recurrent

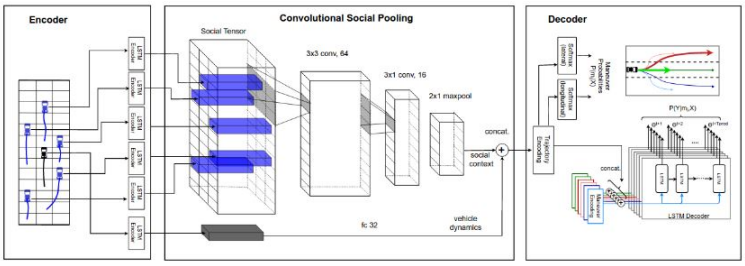

Figure 1.8: LSTM encoder-decoder model from [10]

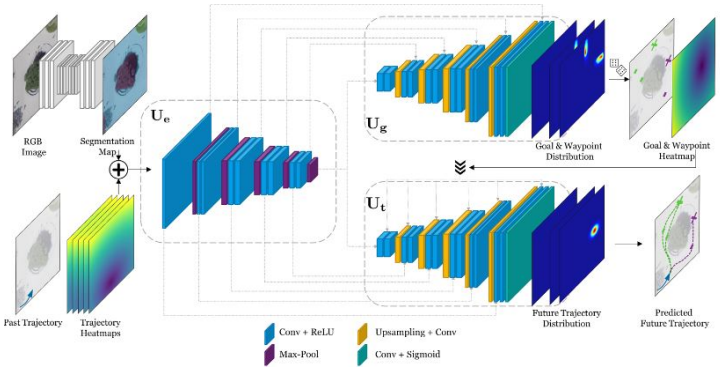

Figure 1.9: Y-net model architecture. Figure from [26]

model. As shown below in figure 1.10, it is a modular network that treats the input as a directed *spatio-temporal* graph, where the agents are vertices in this graph and any interaction adds an edge between the corresponding agents. The recurrent model shown on the left half of the figure is the corresponding network for node 1 for example.

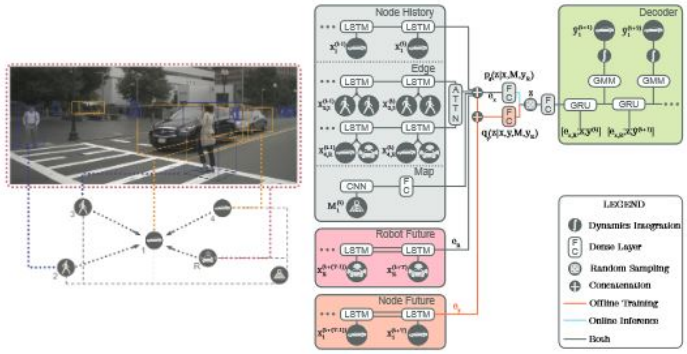

Figure 1.10: Trajectron++ model architecture. Figure from [31]

Chapter 2
Problem statement

2.1 Formulation

As discussed in details, literature review 1 shows the most common methodologies to approach the problem of trajectory prediction. Classification is recently, but rarely, used in this field, so this brief section shows how this problem under study is formulated as a classification problem.

2.1.1 Input format

The input per prediction request is supposed to be two main things. The 1st input is the HD map of the context, which is commonly a 3-channel RGB image. However, this may not be available in some datasets, like Shifts [25] for example, where those channels store different data about the environment, such as: road polygons, lane speed limit … etc. Regarding the 2nd input, it is the agent state vector (ASV) that contains 3 important pieces of information about the to-be-predicted agent: velocity, acceleration, and yaw rate.

2.1.2 Trajectory set generator

This is a separate component which generates a specific-length set of trajectories which covers the whole context as much as possible. This is the set which the model classifies over.

In this work, this approach is used and implemented using (2.1) as a metric to decide the closeness of trajectories to each other. More information will be provided later in chapter 3.

$$\begin{aligned} &\underset{\mathcal{K}}{\operatorname{argmin}} |\mathcal{K}| \text{ subject to } \mathcal{K} \subseteq \mathcal{K}', \\ &\forall k \in \mathcal{K}', \exists l \in \mathcal{K},\ \delta(k,l) \leq \epsilon, \\ &\text{where } \delta(s_{t:t+N}, \hat{s}^*_{t:t+N}) = \max_{\tau=t}^{t+N} \|s_\tau - \hat{s}^*_\tau\|_2 \end{aligned} \tag{2.1}$$

2.2 Notations (Symbols)

s	Ground truth trajectory
$\hat{s}$	Predicted trajectory
s^*	Most likely trajectory
k	Number of modes
$\mathcal{P}$	Set of k most likely trajectories
t	Current time step
N	Number of future steps
$\|.\|$ or $\|.\|_2$	2nd order (Euclidean) norm
ϵ	Error tolerance
$\mathcal{K}$	Trajectory set
$\mathcal{K}'$	Large subsample from the dataset
δ	Coverage metric

Chapter 3
Methodology

This chapter discusses the model architecture and the algorithms used in this research work in order to obtain the subsequent results. The 1st section is about the description of the DL network, while the main algorithm for generating the trajectory set is described in the 2nd section.

3.1 Model architecture

In this section, we will discuss everything about the proposed model, SkolNet. Figure 3.1 shows the high-level diagram of SkolNet. This design is mainly inspired by CoverNet [29], which in turn follows [8].

The network is expected to take an n-channel semantic map as an input. The shape of the map depends on the suggested resolution from the dataset authors or by the renderer used to preprocess the dataset. This map is consumed by the backbone, the responsible component for feature extraction. In this work, we followed the same choice from the original authors, which is ResNet50 [14], pretrained on ImageNet [9]. Although ResNet50 proves its efficacy in this research field, the provided implementation makes the model customizable enough to work with MobileNetV2 [32] (or any other feature extractor) as a backbone.

The prediction request must also provide the ASV for all steps of the history. this ASV, as mentioned before, includes the velocity, acceleration, and yaw rate of the agent over the time.

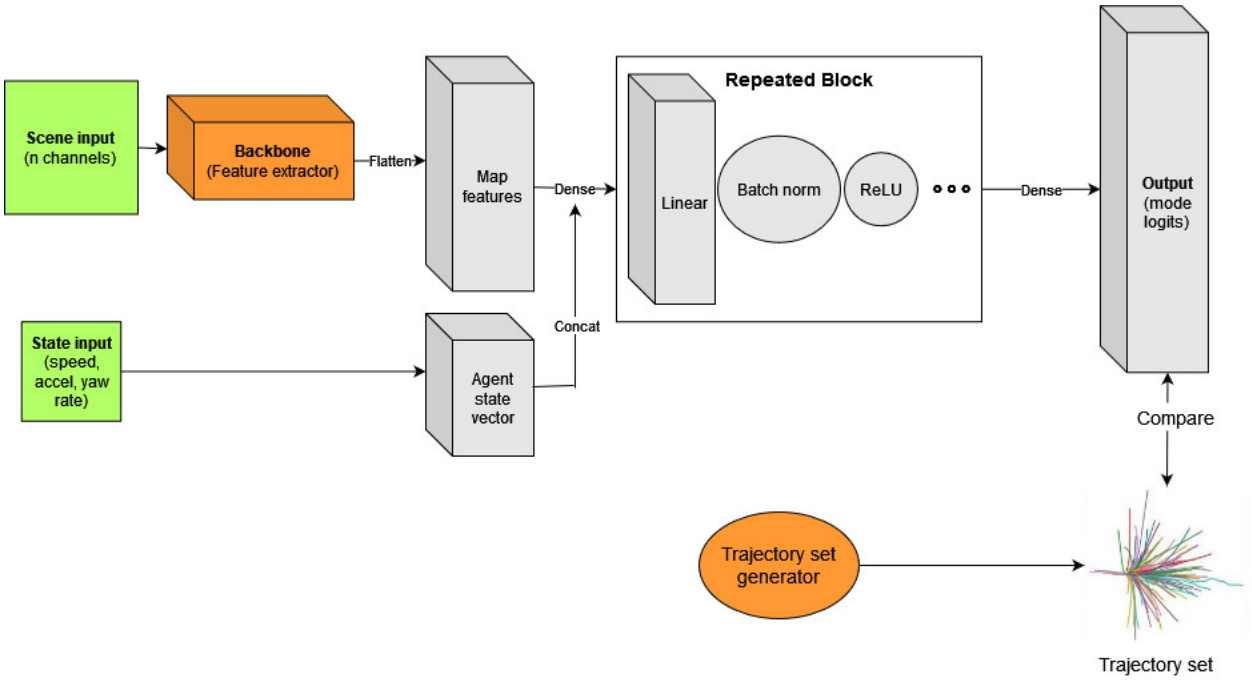

Figure 3.1: SkolNet overview. The architecure follows [29]

The features, extracted by the model backbone, are concatenated to the ASV, and the combined vector goes through a series of blocks decreasing in size, till it finishes at the final linear layer with size of k.

The aforementioned block contains a linear layer to consume the input features, followed by a BN layer for the purpose of stabilization, which is important in case of huge datasets as the one we used in this work, Shifts [25], which will be described later in chapter 4, and internally mitigates the problem of gradient explosion instead of adding a gradient clipping step to the main training pipeline. Finally, the block ends with a ReLU activation function, the most popular function for deep neural networks, to add non-linearity to the neural network. The choice of ReLU is encouraged by many factors, including its durability against gradient vanishing problem and efficient computation, by way of example and not as a limitation. Although there are currently multiple variants of ReLU, the original authors from Motional specifically recommends it.

At the bottom right side of the diagram above, it is shown that the TSG component is an indispensable part of the whole system. The next section shows how to build the trajectory set, so that the model can classify over it.

3.2 Trajectory set generator

This module is responsible for constructing the trajectory set in a way that the most possible space is covered according to the given coverage metric δ, which is maximum element-wise Euclidean distance in this work. In other words, the maximum difference between any two corresponding points in two trajectories doesn't exceed the error tolerance, ϵ. Besides δ and ϵ, the algorithm requires a subsample $\mathcal{K}'$, a subset of ground truth trajectories from the training dataset.

The following pseudocode 1 shows the ad-hoc algorithm followed to build the trajectory set. It starts by building an unweighted undirected graph, where every trajectory is a vertex in this graph, and there is an edge between two trajectories if and only if the coverage metric between them doesn't exceed ϵ. This graph contains an adjacency list, where each vertex keeps its neighbors, and a counting list, *degrees*, which stores the degree of the vertex.

The next part of the algorithm cherry-picks the most connected vertices as the representatives of their clusters, and then makes sure that those vertices and all of their neighbors are covered and will not be chosen again, which guarantees that the result trajectory set includes different trajectories, spanning the entire space as much as possible.

In this study, we limited the size of $\mathcal{K}'$ to 50000. However, its size is directly proportional to the coverage percentage of the trajectory set, so it is advisable to increase this number if there is enough computational power to run this $O(N^2)$ algorithm, where N is $|\mathcal{K}'|$

Algorithm 1 Trajectory Set Generator

Input: k, ϵ, $\mathcal{K}'$, δ
Output: $\mathcal{K}$
\\ Build undirected unweighted Graph
$n \leftarrow |\mathcal{K}'|$
$G \leftarrow Graph(n)$
for $i \leftarrow 0$ to $n-1$ **do**
    for $j \leftarrow i+1$ to n **do**
        if $\delta(\mathcal{K}'_i, \mathcal{K}'_j) \leq \epsilon$ **then**
            G.connect(i, j)
        end if
    end for
end for
\\ Generate trajectory set of size k
$\mathcal{K} \leftarrow []$
degrees $\leftarrow G.degrees$ sorted in ascending order
V $\leftarrow \{\}$
while $|\mathcal{K}| < k$ **do**
\\ This is the most connected trajectory (cluster representative)
    idx $\leftarrow$ pop last element from degrees list
    if idx $\notin$ V **then**
        add $\mathcal{K}'_{idx}$ to $\mathcal{K}$
\\ This is to ensure not to choose a covered trajectory later
        add idx and all its neighbors to V
    end if
end while
return $\mathcal{K}$

Chapter 4
Numerical experiments

In this chapter, we provide the conducted experiments preceded by all the important information to make it as reproducible as possible. This information includes: the experimental setup with programming languages and libraries used, the dataset used in this work with detailed description, the metrics used to train and evaluate the model, and finally the training parameters.

4.1 Experimental setup

- Languages: Python 3.9 [34] (Anaconda)
- Libraries: NumPy [13], Matplotlib [16]
- Frameworks/Platforms: PyTorch [28], Jupyter Notebook [20], Lechuga Server (Mobile Robotics Lab)

4.2 Dataset

We decided, in this work, to train and validate the proposed model, SkolNet, on Shifts [25] dataset. This dataset is collected and presented by Yandex Self-Driving Group (SDG) via Yandex Challenge. Working with this dataset is more challenging than the normal case because it is an in-the-wild dataset, as mentioned by the original authors. It is intended to provide the implications of uncertainty and distributional shifts because this is the common case in the field of autonomous driving.

To maintain the diversity of the scenes in this dataset, it spans six different locations in Russia and Israel. Moreover, it spans three seasons excluding Spring, three times of day, and four weather conditions, including rain and snow.

Each scene lasts for 10 seconds, where the 1^{st} half represents the history and the 2^{nd} half is the ground truth (to-be-predicted) future. In other words, the aim of the work is to predict the trajectories at time T $\in$ (0, 5], given all the history log of the agents and context at time T $\in$ [-5, 0]. In addition, it includes an HD map, which includes lane information such as road polygons, speed limit … etc. Each scene also includes the log of dynamic objects (i.e., cars, buses, pedestrians), where it keeps on each step the position, the velocity, the acceleration, and the yaw rate (orientation up to $\pm\pi$) of each object.

Figure 4.1 shows a scene from the dataset as an example. The authors provided a rendered dataset with maps of size 128×128 and resolution of 1, which means that each pixel represents 1 meter. Although the training dataset contains almost $388,406$ scenes with a total of approximately 5 million prediction requests, we decided to work with only $\%10$ of this number because of the time and power constraints.

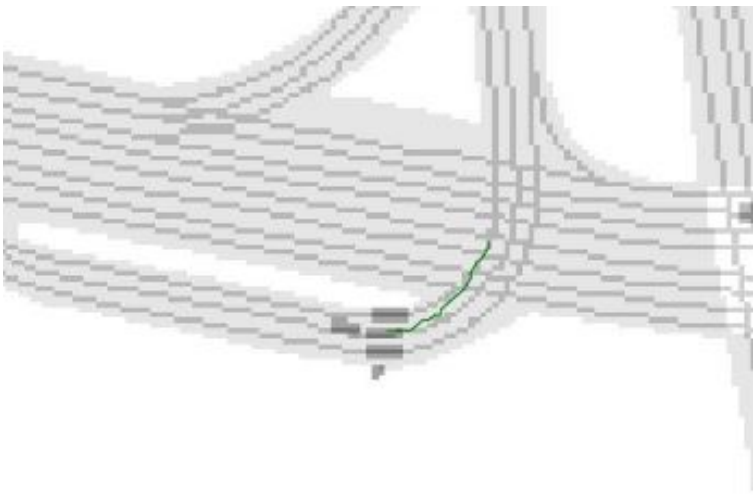

Figure 4.1: A trajectory example from Shifts dataset. It shows the map context, while the ground truth trajectory of the prediction request is colored in green

4.3 Metrics

Equations (4.1) and (4.2) show the most commonly used metrics in this research field. They are *average displacement error* (ADE) and *final displacement error* (FDE) respectively. The 1st metric, ADE, is, as the name indicates, the average point-wise norm between the predicted (most likely) trajectory and the ground truth trajectory, while FDE focuses only on the comparison between the last points in the trajectories.

$$ADE_k = \min_{\hat{s} \in \mathcal{P}} \frac{1}{N} \sum_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\| \tag{4.1}$$

$$FDE = \|s_{t+N} - \hat{s}^*_{t+N}\| \tag{4.2}$$

Regarding the loss function, I used the same loss function that is used in the original work [29], Constant Lattice Loss. This criterion computes the cross entropy loss between the input and target. In our case, the input is the k logits, produced by the model, and the target is the **approximate** ground truth trajectory, s^*

Indeed, it is unfair to compare the model outputs to the ground truth trajectory because the model only classifies over a finite set of trajectories. So, the target in this loss function is the closest trajectory from this set to the ground truth trajectory, that is why it is not called cross entropy loss from the start. Mathematically, equation (4.3) denotes the rough-estimated class $\mathcal{C}$ (i.e., index) of the ground truth trajectory, which is the target for computing the cross entropy loss. As usual, the closeness metric used here is l2 distance.

$$C = \underset{c}{\text{argmin}} \frac{1}{N} \sum_{\tau=t}^{t+N} \|s^c_\tau - s^*_\tau\| \tag{4.3}$$

4.4 Implementation details

The whole system has a tremendous number of hyperparameters on different levels. For the dataset, we have various hyperparameters to control the resolution and size of the map and the number of history steps to consider. Moreover, the number of modes, k, affects the model architecture and TSG, while the error tolerance, ϵ, affects TSG. The model itself can be customized to use one backbone from multiple options. Not only this, but even the training process has its own hyperparameters like the optimizer type and helper hyperparameters, and the batch size.

In order to avoid the full factorial design, we decided to tune only the primary factors of the system according to its uniqueness compared to the other systems and check the effect on the results while fixing the other secondary factors.

The following list shows the values of the fixed factors:

- batch size: 64
- model backbone: Pretrained ResNet50
- repeated blocks: only one with size 4096
- optimizer: Stochastic Gradient Descent (SGD)
  - learning rate α: $1\text{e}{-4}$
  - momentum μ: 0.9
  - weight decay λ: $5\text{e}{-4}$

Most of these values are decided by the original authors via this thread. As mentioned before, tuning these hyperparameters probably results in a more optimal functionality, but this is not the final aim of this study. The actual target is to study the distinctive hyperparameters, which makes this model distinguishable from the other ones. Thus, the prime factors of these experiments are decided to be the number of modes, k, and the error tolerance, ϵ that directly affects the coverage.

Moreover, while training the model for 10 epochs, it is validated every 2 epochs on almost $\%50$ of the validation set, which means $200,000$ prediction requests, according to the loss function mentioned in equation (4.3). However, the whole validation set is used for the final evaluation of the model, applying ADE and FDE from the equations (4.1) and (4.2).

4.5 Results

In this section, we give an adequate presentation to the results of all the conducted experiments, starting from generating the trajectory sets and training the model till ending with the model evaluation metrics and the comparison with the other models on the same dataset.

4.5.1 Trajectory set generator

In order to construct the trajectory set as explained in section 3.2 as fair as possible, a constant trajectory is added manually to cover all stationary vehicles, and then the algorithm subsamples from the moving trajectories to choose the remaining $k-1$ trajectories.

For building the trajectory set, we tried two different values for ϵ, 2 and 6, as shown in figure 4.2. It is clear that ϵ directly affects the smoothness and coverage of the trajectory set. The set on the right side is denser, and it is not enough to spread over the whole space as the other set with $\epsilon = 6$

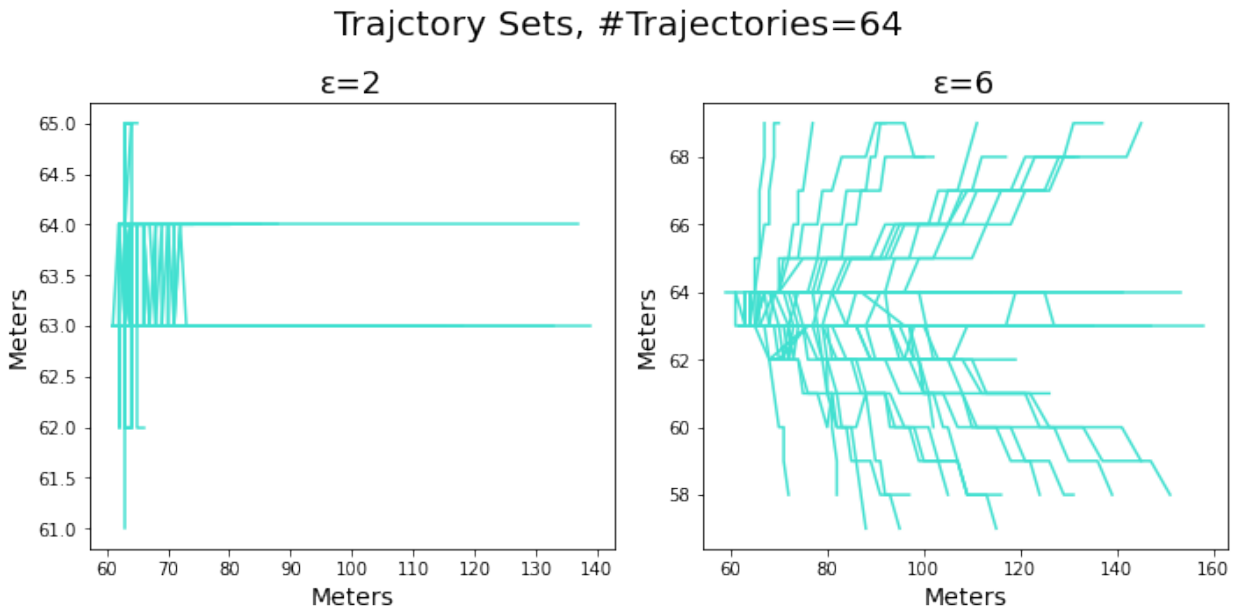

Figure 4.2: Trajectory sets

As per proving that the set with larger error tolerance is better in terms of coverage, we generated another larger set with this exact value to check the effect of k. It is also noticed in figure 4.3 that the trajectory set becomes smoother when the number of modes increased from 64 to 128.

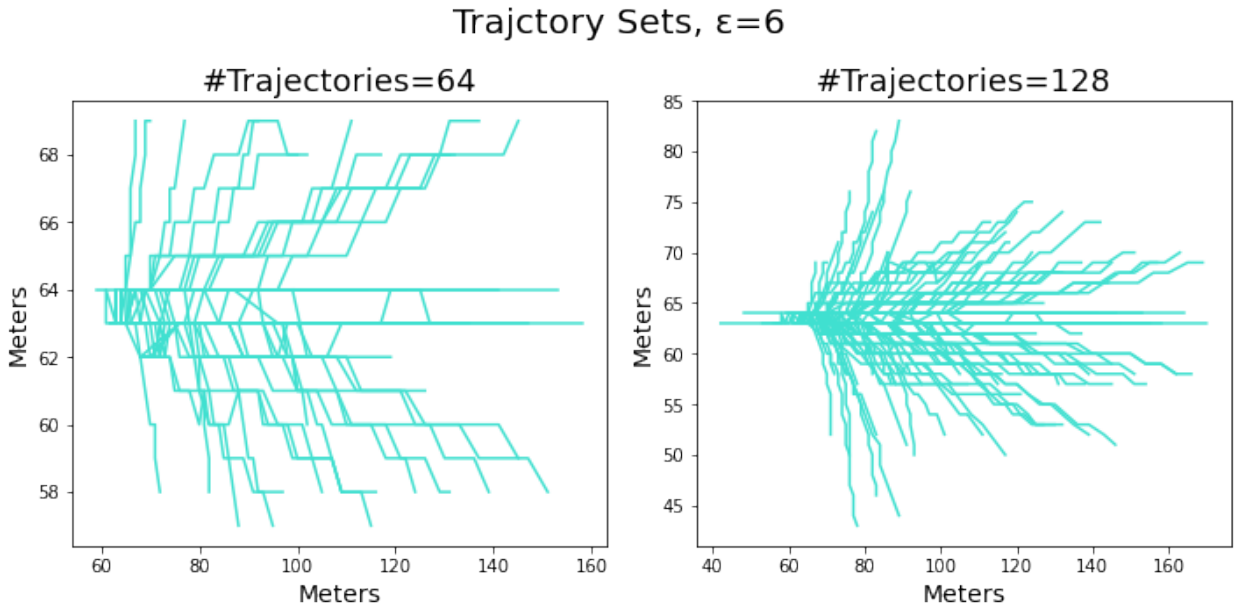

Figure 4.3: Trajectory sets

4.5.2 Training loss

Here, let us compare the constant lattice loss while training the models with different parameters. While fixing the error tolerance ϵ, the figure 4.4 shows the model performance using two different numbers of modes. It is supposed to have better results when the number of modes gets increasing because it. However, using the same error tolerance implies that the same graph will be

built and the same most covering trajectories will be chosen before adding the extra ones. In other words, the smaller trajectory set is actually the dominant subset of the larger one, which means that the model will not necessarily discover new horizons or clusters by adding more trajectories from the same dataset.

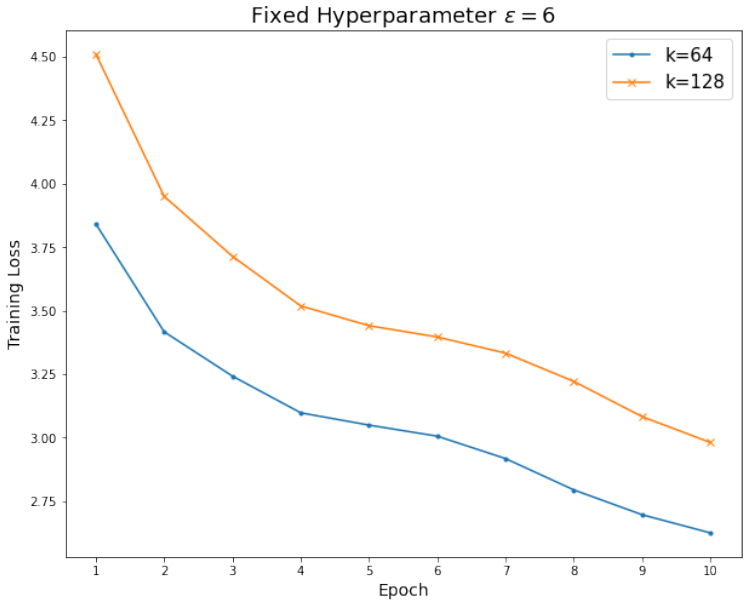

Figure 4.4: The effect of k on the training loss

However, this could be more useful in case of lower values of ϵ because, as shown before in figure 4.2, it is more compact and requires more trajectories to cover almost the same space covered by larger values, which is also proved by the inspiring work [29].

On the other hand, increasing the value of ϵ gives better loss according to the figure 4.5 when the number of modes k is fixed. This is already proven by the plot of trajectory sets in the figure 4.2 because the lower value of ϵ makes TSG fail to cover most of the cases if the number of modes is not sufficient.

4.5.3 Model evaluation

As mentioned earlier, the trained SkolNet models are evaluated on the full validation dataset against *ADE* and *FDE*. Table 4.1 summarizes the final results of all experiments. Beside SkolNet, it includes all the other techniques used in this research project by the motion prediction team in Mobile Robotics Lab using the same dataset, Yandex Shifts [25].

As being a type of time-series problems with infinite number of possible outputs, it is not a straightforward solution to reformulate the problem of trajectory prediction as a classification problem. Thus, it is clear from the results that this approach is still **experimental** and cannot outperform, or even match, the popular state-of-the-art models, on the contrary, it requires more investigation to analyze and improve its drawbacks, which will be discussed in the next chapter.

Although this recent approach is not yet established, compared to the other models, it is shown in figure 4.6 that it is able to predict some future trajectories because of the data distribution. This is more clear in figure 4.7 because it shows that the model decides to predict that the vehicle will be stationary in the near future. This is mainly because of the dataset distribution that the trajectory set is sampled from. The model failed to predict the vehicles which turned back because the majority of the trajectories are moving to the right. This made the generator biased more to this

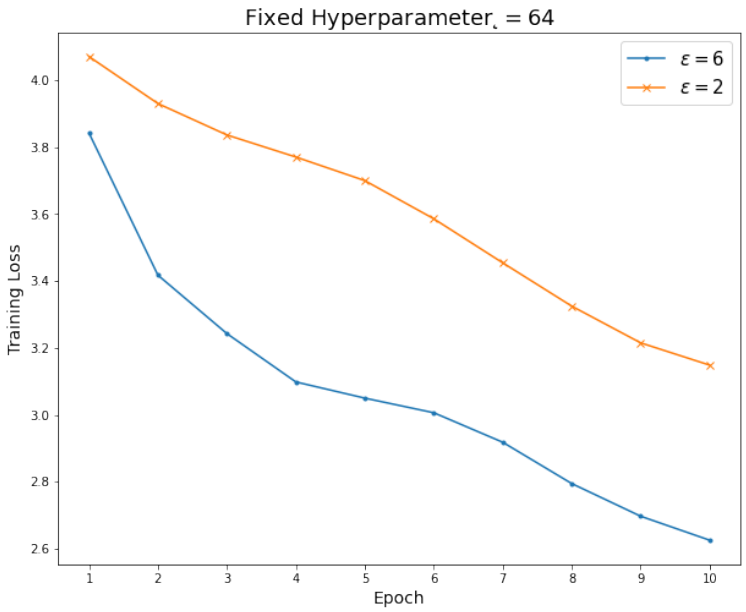

Figure 4.5: The effect of ϵ on the training loss

	ADE	FDE
DIM [30]	0.758	1.524
BC [7]	1.2	2.4
Set-Transformer [22]	**0.568**	1.132
Y-Net [26]	0.75	**1.18**
Trajectron++ [31]	0.8	1.47
SkolNet, ϵ=6, k=64	8.69	16.7
SkolNet, ϵ=6, k=128	9.02	17.4

Table 4.1: Collective table of ADE and FDE results for all experiments by team members from Mobile Robotics Lab. Smaller ADE and FDE are better

direction and neglect the other ones, so the model decides the fixed trajectory to minimize the error as much as possible (which is the right decision given the current circumstances).

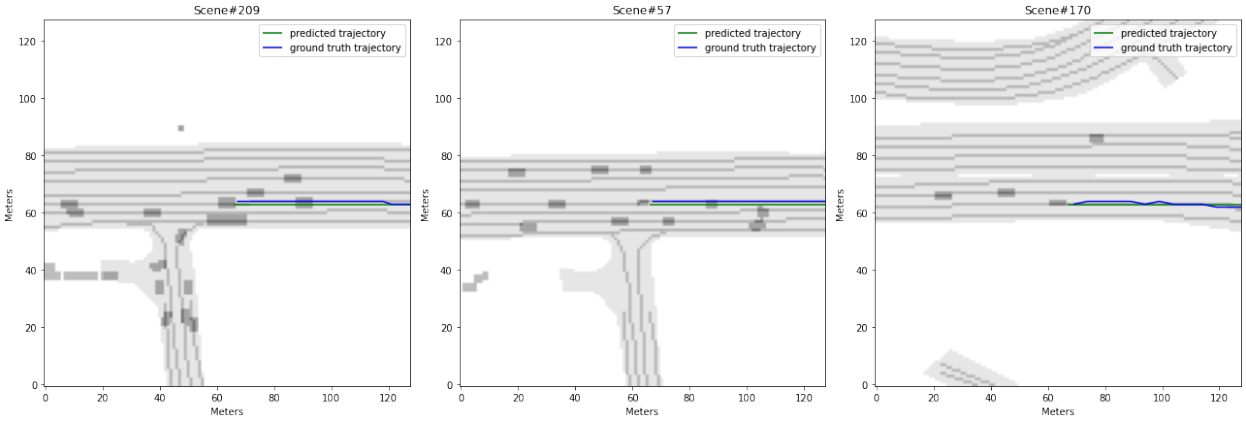

Figure 4.6: Correct predictions by SkolNet (ϵ=6, k=64)

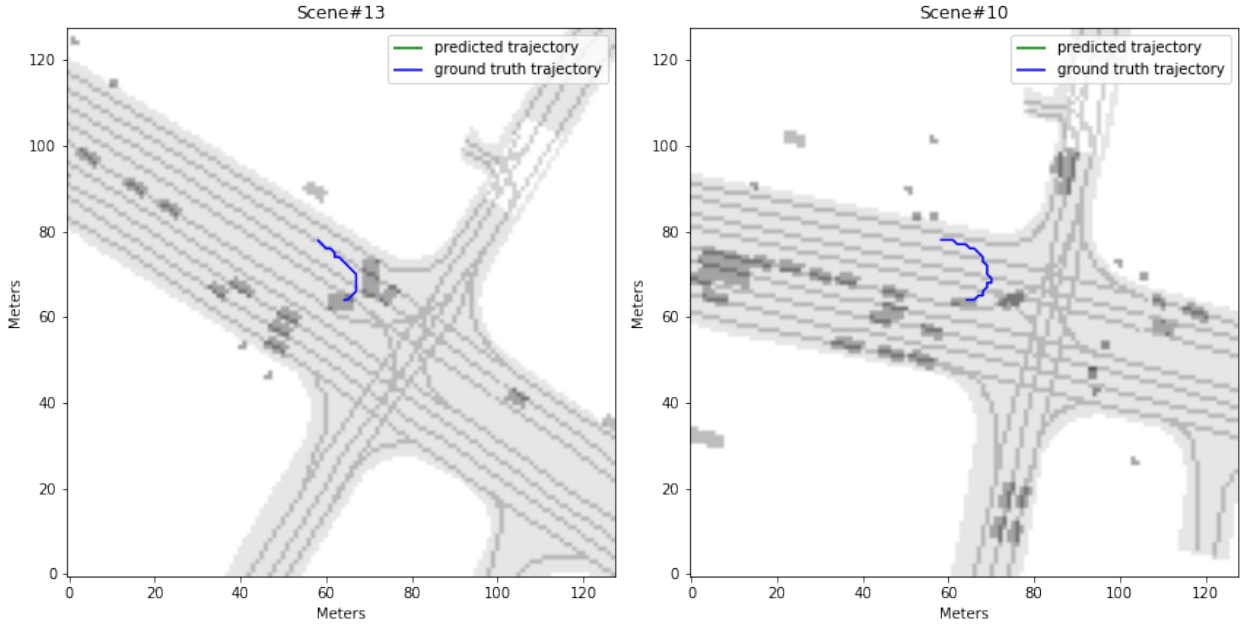

Figure 4.7: Wrong predictions by SkolNet (ϵ=6, k=64)

Chapter 5
Discussion and conclusion

5.1 Discussion of results

Looking at the results in the previous chapter shows that the model performs better using large error tolerance to cover wider spaces and a reasonable number of modes. This conclusion is not final because the comparison with the other robust models prove that the classification approach is still experiential and there is much room for more research and improvements, even if it follows the arms-twisting policy to reformulate the trajectory prediction problem to suite the classification approach.

In fact, generating the trajectory set to work as classes or labels did not work as it should because it totally depends on the distribution of the training dataset, and it is reasonable that the model will not succeed if evaluated on a different distribution, which is the case of Shifts dataset because it is divided into different canonical partitions. As it appears from the dataset name, it contains intentional distributional shifts. This definitely makes the model trains well on the training set and achieves good loss by time, but it fails to achieve acceptable metrics on the validation set.

Moreover, even if the training dataset itself suffers from data imbalance, this would result in a biased trajectory set because it is, after all, sampled from this set. This is also clear because all trajectory sets represent the motion to the right, so the model successfully predicted those cases, but failed to work on the remaining ones.

Although this approach is still not reliable to compete with the other proven approaches, it is challenging for research and investigations during the time of this work. It is a large system which combines different modules together, so it is worth time in the future to analyze all these modules and inspect the system bottlenecks.

5.2 Research limitations

Let us summarize the findings of the system investigation to have this fragile performance and how to mitigate these shortcomings in the future work.

Starting from the dataset, it will be an obstacle if the dataset is imbalanced because this will affect TSG.

Regarding TSG, it basically depends on the training dataset, and does not change over the lifetime of the training, and this is not reasonable because the pool of trajectories should also take into account the state of the agent. Another thing that it should not be deceived by the imbalanced training dataset to represent the trajectories as fair as possible.

Regarding the model itself, most of the well-known feature extractors, especially if they are pretrained, are more suitable for 3-channel images, so it would require more effort to choose a model backbone to consume this n-channel feature map. The owners and authors of Shifts work provided some baseline models via this link, and they used MobileNetV2 instead of ResNet50, so it should be more useful to use this backbone because it is pretrained on the same dataset to extract the map features. Kindly note that SkolNet is flexible enough to accept this pretrained network as a model backbone. It would also be beneficial to use a chain of repeated blocks to help the model

learn more complex scenarios. In this study, only one block is used for simplicity, but $2-4$ blocks may give better performance.

Talking about the classification as an approach to apply on this problem, it is clear that it is not advantageous if it is used as it is to tackle this kind of problems in the light of the existence of more reliable alternatives.

5.3 Future work

Finally, we aim to point out some tips or solutions to address the above-mentioned issues and limitations as follows:

- Instead of using the trajectories as labels, the model may classify the motion itself, not the trajectory. For example, the classes or modes could be "turn right", "move forward", and so on, instead of predicting the numerical trajectories.
- TSG should consider the weights of the different directions, and use a weighted random sampling from the training dataset. Additionally, it may generate a trajectory set per direction. In the later case, ensemble models [12] technique can be used.
- TSG should be dynamic enough to cope with the current state of the prediction request. This dynamic approach can make use of the velocity, acceleration, and rotation (yaw rate) can help to determine the possible trajectories by forward integration.
- Using pretrained networks has been proven effective for a long time. However, it may not add a big difference to the model if they were pretrained on a different kind of datasets of problems.
- The model can be trained using the past steps of the agents, not only the current state at time step 0. This would give more information about the agent behavior.

Acknowledgements

Here I would like to thank all the people in Mobile Robotics Lab for everything related to physical working space, servers and knowledge transfer. Especially, the following people for their help and contributions:

- Prof. Gonzalo Ferrer (Head of Mobile Robotics Lab, Assistant Professor)
- Mirfarid Musavian (Research Engineer)
- Aleksey Postnikov (PhD Student)
- Rahim Tariverdizadeh (Researcher)
- Aleksandr Gamayunov (PhD Student)
- Gabriel Rozzonelli (MS Student)

Bibliography

[1] Bansal, M., Krizhevsky, A., and Ogale, A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst, 2018.

[2] Bhattacharyya, A., Schiele, B., and Fritz, M. Accurate and diverse sampling of sequences based on a "best of many" sample objective, 2018.

[3] Branicky, M. S., Knepper, R. A., and Kuffner, J. J. Path and trajectory diversity: Theory and algorithms. In *2008 IEEE International Conference on Robotics and Automation* (2008), pp. 1359–1364.

[4] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027* (2019).

[5] Chai, Y., Sapp, B., Bansal, M., and Anguelov, D. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction, 2019.

[6] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. Argoverse: 3d tracking and forecasting with rich maps, 2019.

[7] Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. End-to-end driving via conditional imitation learning, 2017.

[8] Cui, H., Nguyen, T., Chou, F.-C., Lin, T.-H., Schneider, J., Bradley, D., and Djuric, N. Deep kinematic models for kinematically feasible vehicle trajectory predictions, 2019.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255.

[10] Deo, N., and Trivedi, M. M. Convolutional social pooling for vehicle trajectory prediction. *CoRR abs/1805.06771* (2018).

[11] Filos, A., Tigkas, P., Mcallister, R., Rhinehart, N., Levine, S., and Gal, Y. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 3145–3153.

[12] Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., and Suganthan, P. N. Ensemble deep learning: A review, 2021.

[13] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362.

[14] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

[15] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. One thousand and one hours: Self-driving motion prediction dataset, 2020.

[16] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering 9*, 3 (2007), 90–95.

[17] Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv. 50*, 2 (apr 2017).

[18] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR abs/1502.03167* (2015).

[19] Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. Hands-on bayesian neural networks – a tutorial for deep learning users, 2020.

[20] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C. Jupyter notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (2016), F. Loizides and B. Schmidt, Eds., IOS Press, pp. 87 – 90.

[21] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature 521* (05 2015), 436–44.

[22] Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning* (09–15 Jun 2019), K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 3744–3753.

[23] Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H. S., and Chandraker, M. Desire: Distant future prediction in dynamic scenes with interacting agents, 2017.

[24] Lefevre, S., Vasquez, D., and Laugier, C. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal 1* (07 2014).

[25] Malinin, A., Band, N., Ganshin, Alexander, Chesnokov, G., Gal, Y., Gales, M. J. F., Noskov, A., Ploskonosov, A., Prokhorenkova, L., Provilkov, I., Raina, V., Raina, V., Roginskiy, Denis, Shmatova, M., Tigas, P., and Yangel, B. Shifts: A dataset of real distributional shift across multiple large-scale tasks, 2021.

[26] Mangalam, K., An, Y., Girase, H., and Malik, J. From goals, waypoints & paths to long term human trajectory forecasting, 2020.

[27] Motional. nuscenes-devkit. `https://github.com/nutonomy/nuscenes-devkit`, 2018.

[28] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[29] Phan-Minh, T., Grigore, E. C., Boulton, F. A., Beijbom, O., and Wolff, E. M. Covernet: Multimodal behavior prediction using trajectory sets, 2019.

[30] Rhinehart, N., McAllister, R., and Levine, S. Deep imitative models for flexible inference, planning, and control, 2018.

[31] Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data, 2020.

[32] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[33] Tran, Q., and Firl, J. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. *2014 IEEE Intelligent Vehicles Symposium Proceedings* (2014), 918–923.

[34] Van Rossum, G., and Drake, F. L. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.