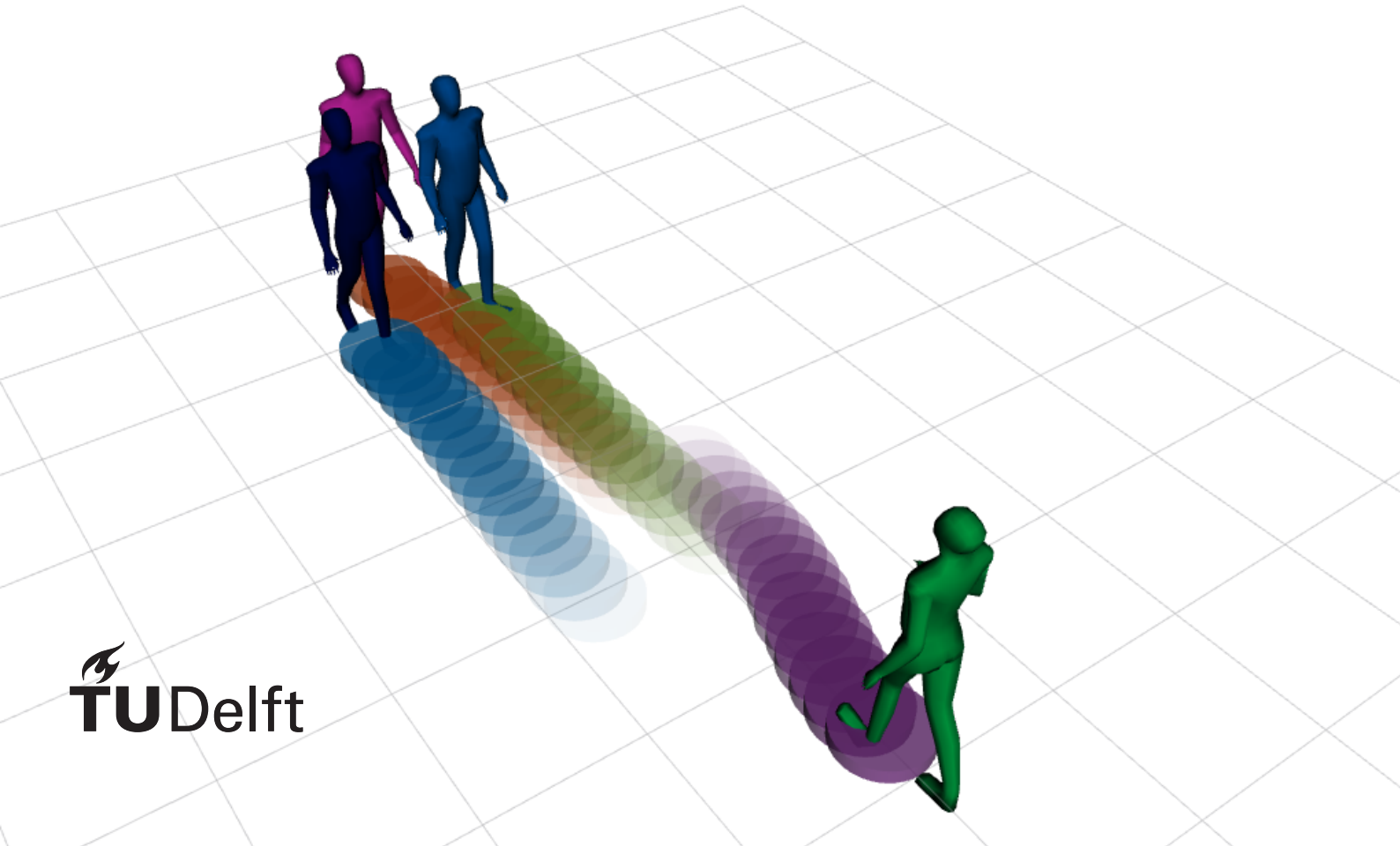
Self-Supervised Continual Learning for Interaction-Aware Pedestrian Prediction Models

C. Salmi

Technical University of Delft



TU Delft

Self-Supervised Continual Learning for Interaction-Aware Pedestrian Prediction Models

by

C. Salmi

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday April 22, 2021 at 15:00 PM.

Student number:	4374533	
Project duration:	March 1, 2020 – April 22, 2021	
Thesis committee:	Dr. J. Alonso-Mora,	TU Delft, supervisor
	Dr. J. F. P. Kooij,	TU Delft
	Ir. B.F. Ferreira de Brito,	TU Delft, supervisor

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft

Preface

Before you lies the thesis work ”Self-Supervised Continual Learning for Interaction-Aware Pedestrian Prediction Models”, the basis of which is the exploration of a novel framework to train a neural network online to improve a pedestrian trajectory prediction model, using the onboard sensors of a mobile robot platform.

In March 2020, I embarked upon this journey after receiving the opportunity to pursue my Masters of Science in the exciting field of robotics, at the Cognitive Robotics Department of the Technical University of Delft. The basis for this research aligned perfectly with my growing passion for autonomous vehicle and mobile robot technologies. Neural networks are becoming an integral part of modern robotics, but how can we utilize them in dynamic unknown environments in our ever changing society? To find out I have spend the last year developing a framework to continually train a neural network for pedestrian trajectory prediction.

I would like to thank my supervisors, Ir. Bruno de Brito and Dr. Javier Alonso-Mora, for their guidance during this process. I also wish to thank my fellow students in the Cognitive Robotics Lab, especially Niels Marcelis, for helping with experiments and making my time spent there a fun experience. Finally I would like to thank my family and friends for supporting me throughout this sometimes hectic and stressful, but fun period in my life.

C. Salmi
Delft, April 2021

Contents

1 Introduction **1**
1-1 Research Objective 2

2 Scientific Paper **5**

3 Discussion **19**
3-1 Data Aggregation 19
3-2 Elastic Weight Consolidation 20
3-3 Prediction Network Architecture 20
3-4 Motion Planner Integration 20

4 Conclusion **23**

5 Recommendations For Future Work **25**

A Mobile Robot Platform **27**
A-1 Overview 27
A-2 Design Choices 28
A-2-1 Dedicated Laptop For Heavy Workloads 28
A-2-2 LiDAR And Cameras Setup 29

B Additional Experimental Results **31**
B-1 Optitrack Experiments 31
B-2 Real World Experiments 31
B-2-1 Static Robot 31
B-2-2 Dynamic Robot 32

C Prediction Network Implementation Details **37**
C-1 Prediction Network Details 37
C-2 Pre-training Data Details 38

D Additional Integrated Motion Planner Experiments **45**

1 Introduction

Fuelled by the potentially large commercial and societal benefits, many research institutions and companies are working on autonomous vehicle and mobile robot technologies [1]. Autonomous vehicles are expected to improve safety, reduce traffic congestion [2], and allow people to do productive or leisure activities while driving. One of the major challenges still facing autonomous vehicles is driving in urban environments [3]. In order to safely and efficiently drive in dynamic urban environments, autonomous vehicles need to be able to account for the intentions of surrounding pedestrians.

Mobile robots, on the other hand, have been used for some time in constraint environments e.g. warehouses and production lines. For example, Amazon Robotics already deployed 15,000 mobile robots in their warehouses in 2014 [4]. But when moving to dynamic environments, mobile robots face similar challenges as autonomous vehicles. It is, however, expected that mobile robots will also have a significant impact on our society in applications like service robots, parcel delivery robots, nursing robots, and guidance robots. In a large part of these applications, mobile robots will interact with humans in dynamic and unstructured environments e.g. hospitals, airports, or sidewalks, which brings with it several challenges. Robots should:

- Navigate safely and conform to the established social rules and norms
- Act predictably to humans to make the interaction pleasant
- Adapt to new and unseen situations

By navigation, we mean the ability of the robot to plan a safe path or trajectory towards a specified goal location. These challenges require the robot to take interactions with humans into account while navigating, which will be referred to as socially aware motion planning in this thesis. Additionally, the social acceptance of robots in the public domain remains an active area of research. To improve the social acceptance of robots, Kruse et al. [5] state three explicit goals. Naturalness, which has to do with the similarity between humans and the robot in their low-level behavior patterns. Sociability, which is the adherence to explicit high-level cultural conventions. And comfort, the absence of annoyance and stress for humans in the interaction with robots.

Many State-of-The-Art (SoTA) socially-aware motion planning algorithms rely on predictions of the future trajectory of humans [6]. If a robot has a sense of what surrounding pedestrians are going to do, it can take this into account when planning its movement. Moreover, to act predictably towards humans requires knowledge of the intention and implicit social behavior of humans.

However, explicit prediction of the trajectory of humans is not the only way of including social awareness in motion planning algorithms. Another option is learning the planning policy directly from sensor data, and thus implicitly learn social awareness. Approaches like this are referred to as End-to-End learning approaches [7] and they have become an active area of research because of the breakthroughs in artificial intelligence and deep neural networks of the last decade [8]. However, in practice, the latter option is often currently not feasible or accepted. Autonomous motion planning algorithms in the real world are inherently high risk and often explainable solutions are preferred over black-box deep neural networks. Furthermore, End-to-End approaches still face essential challenges with regards to verifiability, safety, and explainability and have only been shown to work reliably in simplified settings [6].

Keeping this in mind, the short-term more feasible solutions towards socially aware motion planning rely on explicit human motion prediction. A large body of research is currently done in the direction of human motion trajectory prediction models [9]. Traditional physically inspired models [10, 11] have for a large part been replaced by neural network-based models [12] because they allow for better prediction in more complex scenarios based on learned features. Neural network models are not constrained by manual feature extractions, which allows them to exploit highly dimensional sources of context information like obstacle grids, satellite images, semantic maps or onboard cameras [13, 14, 15, 16]. However human motion is inherently multi-modal, which means there are multiple plausible trajectories a human might take in a given scenario. Generative neural networks are therefore being used to capture this multi-modality [17, 18, 19].

In many of the earlier mentioned applications, mobile robots operate in dynamic, changing environments, and consequently the behavior of surrounding pedestrians subject to change. However human motion prediction models are currently trained offline on beforehand available datasets [20, 21] and they don't adapt to new and unseen behaviors or environments. To improve pedestrian prediction models online, we explore multiple solutions from the recently emerged research paradigm of continual learning [22, 23], which is concerned with training neural networks online from streams of data. Using the online observations of an existing detection and tracking pipeline we aim to investigate the possibility of training a neural network-based prediction model continually.

1-1 Research Objective

Recent years have shown tremendous progress in architecture designs for neural network-based pedestrian prediction models. However previous works focus solely on improving prediction network architectures, whilst training them offline. Ultimately

it is still unknown if and how much continual learning of neural networks can contribute to socially aware motion planning. Therefore, I pose the following research questions:

1. Can we employ continual learning techniques making use of online observations to improve a prediction model?
2. Which continual learning techniques can be applied to develop a stable learning algorithm that consolidates knowledge over time?
3. Can we use the learned model for socially aware motion planning?

2

Scientific Paper

Self-Supervised Continual Learning for Interaction-Aware Pedestrian Prediction Models

Chadi Salmi
Cognitive Robotics
Technical University Delft
Delft, Netherlands
salmi.chadi@gmail.com

***Abstract*—Learning human motion prediction models online is key for autonomous navigation in unknown dynamic scenarios. Previous works focus solely on improving prediction network architectures, whilst training them offline. This paper introduces a self-supervised continual learning framework that continuously improves data-driven pedestrian trajectory prediction models online across various environments. We propose to use online streams of pedestrian data, normally available from detection and tracking pipelines. Examples are autonomously extracted from this data stream and aggregated in temporally bounded episodes, where the data of each episode is discarded as soon as the model has been adapted to it. Our framework overcomes the problem of catastrophic forgetting across episodes by selectively slowing down learning of important neurons and by rehearsing a small set of examples of constant length. Our approach is shown to significantly improve prediction performance in new and unseen environments compared standard gradient descent approaches. Finally, we present qualitative experimental results in simulation and in real environments.**

I. Introduction

Autonomous mobile robots are being deployed in a variety of application domains [1], such as service robots in hospitals or guidance robots in airports. In these environments, robots will often have to navigate in a joint space with pedestrians. Like humans, autonomous robots need to have a sense of what the people around them are going to do to navigate efficiently and safely. Prediction models that capture the intent of a pedestrian, enable robots to anticipate behaviors [2], avoid other pedestrians, and comply with underlying social rules without direct communication [3].

In dense pedestrian-rich spaces, pedestrians adjust their path to comply with implicit social rules and norms [4], see Fig. 1. At the same time, pedestrians are constrained by the static environment, such as obstacles or walls [5], and there are often multiple plausible ways a pedestrian can reach his goal [6]. Furthermore, the presence of a robot in the environment can change the behavior of pedestrians [7]. Hence, pedestrian trajectory prediction is a complex task.

A large amount of research was done on prediction models that focus on the influence of implicit social rules and norms on human motion [8]. Additionally, much progress has been made in the direction of stochastic prediction models that account for the multi-model nature of humans [6]. And recently, many works combine social compliance with surrounding semantic features in stochastic models [9, 10]. However, these models are trained offline and don't adapt to new and unseen behaviors or environments.

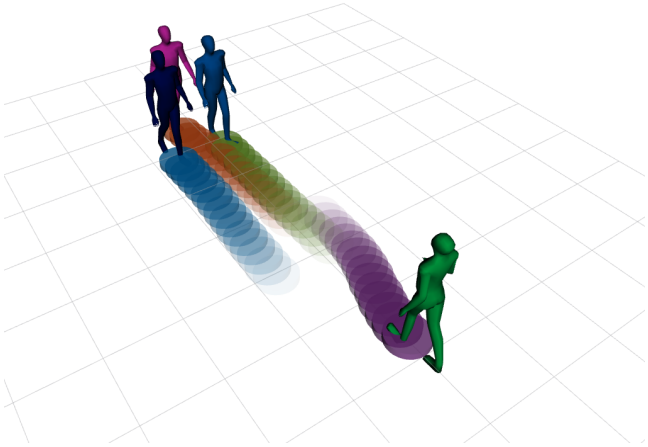


Fig. 1: Example of interaction aware predictions by a continually learned prediction model.

In the field of robotics, observational data usually comes in sequential never-ending streams, which makes it is possible to take advantage of the incoming data to continuously gather training examples. However, as stated in [11], creating labeled data is probably the slowest and the most expensive step in most machine learning systems. This is one of the reasons continually learning neural networks is often infeasible. Our key insight is that the robots operating in the same environment as pedestrians can autonomously collect training examples based on the robot's observations. If a robot can efficiently and autonomously collect examples, its internal prediction models can be updated on the fly and the robot effectively adapts its behavior.

Another reason continually learning neural networks is difficult is because previously learned behavior, in the form of weight values, can be overwritten by new update steps. Especially when the distribution of the stream of training examples changes, knowledge of previous training examples can be quickly forgotten. For example, when a mobile robot moves to a different environment. In literature, this is referred

to as the problem of *catastrophic forgetting* in neural networks. To overcome catastrophic forgetting, we use a *regularization* strategy to selectively slow down learning on important neurons and we *rehearse* a small set of previous examples.

In this paper, we employ the online observations of a mobile robot to adapt a pre-trained prediction model on the fly. Our contributions are the following:

1) We formulate the online optimization of pedestrian prediction models from observations as a self-supervised online learning problem and propose an online learning framework allowing to continuously improve a prediction model.
2) We present simulation results against baseline methods demonstrating that our framework can improve prediction performance and avoid catastrophic forgetting.
3) We demonstrate through extensive experiments on a mobile robot platform, that our framework can continuously improve a prediction model in the real world.

To our knowledge, this is the first application of online learning to improve a pedestrian prediction model on the fly.

II. Related Work

A. Pedestrian Trajectory Prediction

There is a vast amount of work devoted to pedestrian trajectory prediction models [8]. Early on, physically inspired models were most prevalent. In particular, the social forces model was pioneering for pedestrian trajectory prediction and crowd flow modeling [12]. The social forces model uses attracting and repulsive potentials to model the social behavior of pedestrians. In [7], a data-driven approach based on Gaussian processes (GPs) was proposed. Every pedestrian trajectory is modeled as a Gaussian process, where the kernel is learned from data, and an interaction potential is used to model social interactions. However, both of these methods utilize handcrafted features, limiting their ability to capture complex interactions. Recently, recurrent neural networks (RNNs) have been used for trajectory prediction because of their success in sequence-sequence modeling tasks. Instead of using handcrafted features, RNN-based pedestrian prediction models can learn to extract features based on a set of example trajectories (supervised-learning). [4] proposed to model every pedestrian as a separate Long Short Term Memory unit (LSTM) and propagate information about surrounding pedestrians using a social pooling grid. Other works instead use attention mechanisms to selectively propagate features from surrounding pedestrians [13]. Following the trend towards data-driven prediction models, many works explore the use of more context information to improve prediction accuracy. Different birds-eye-view (BEV) 2D maps of the environment have been used to extract context features, like obstacle grids [5], satellite images of the environment [14, 10], and semantic maps [15]. While other works focus on extracting cues of surrounding agents, like head pose [16], observed actions and awareness of the robot's presence [17]. Visual cues of agents can also be learned directly from on-board camera images [18].

Concurrently, recent works explore stochastic data-driven prediction models to capture the inherent multi-model nature of human motion. Generative Adversarial Networks (GANs) have been used to generate multiple socially plausible futures, by randomly sampling from the latent space [6]. However, GANs are typically very difficult to train and can suffer from mode collapse causing the models to predict very similar trajectories. A recent extension of GAN called info-GAN was used to prevent mode collapse [19]. In contrast to GANs, many works explore the use of Conditional Variational Autoencoders (CVAE) to explicitly model the variation [14]. Using variational learning, [20] propose a new Variational Recurrent Neural Network (VRNN) architecture to generate more diverse socially-plausible trajectories from fewer examples.

At the same time, there have been many works that explore new ways of modeling interactions. Graph neural networks (GNN) were used as attention mechanisms to propagate the hidden states of LSTMs [21], variational learning was combined with GNNs [15], and a Spatio-temporal GNN architecture was proposed that directly models pedestrian trajectories as a graph [22]. Another model architecture that is considered state-of-the-art (SoTA) in neighboring sequence-sequence learning domains, is the Transformer network [23]. In [24], a model architecture is proposed that models complex Spatio-temporal interactions by interleaving between spatial and temporal Transformers.

In the body of literature on prediction models, normally performance is evaluated and compared using (offline) benchmarking datasets [25, 26, 27, 28]. And often these prediction models are deployed in AVs or autonomous mobile robots, to anticipate the behavior of surrounding agents. However, [29] noted that a learning-based state-action policy should be learned online since the distribution of input data is dependent on the policy. And this can be extended to learning a prediction model because it ultimately affects the policy of the AV or mobile robot.

B. Online Learning

Regret-based online convex optimization (OCO) has been extensively studied [30]. The goal of OCO, when applied to our problem setting, is to accurately predict pedestrian trajectories, given knowledge of previously observed trajectories. Follow the Leader (FTL), is the most straightforward OCO method that works by optimizing across all previous knowledge every step. Inspired by FTL, [29] introduced DAgger, an online learning algorithm to learn a state-action policy. DAgger aggregates all previous observations online and queries an expert to get the optimal action in hindsight. However, DAgger and FTL both save all previous examples and periodically retrain a network on the entire aggregate, which quickly becomes very computationally expensive.

Therefore, a large body of research is done in the direction of learning deep NN-based models online, without saving all previous examples [11, 31]. The biggest challenge of training neural networks online is to retain good performance on examples that are not shown anymore for a long time, or

as [32] states it, to prevent *catastrophic forgetting*. In their pioneering work, [33] introduced *Elastic Weight Consolidation* (EWC), a regularization approach that limits the plasticity of specific neurons based on their importance. However, EWC assumes that the input distribution can be split into different tasks and separate importances can be calculated for each task. Other regularization approaches have focused on relaxing this assumption by automatically inferring task-boundaries [34], or by calculating the importance in an online fashion over the entire learning trajectory in parameter space [35]. The downside of regularization-based techniques is that an additional loss term is added which can lead to a trade-off between knowledge consolidation and performance on novel tasks.

Inspired by the biological interplay between the hippocampus and neocortex, many works explored the use of an episodic memory that is rehearsed periodically. There are two types of episodic memory methods that differ in the way they model memory, *rehearsal* approaches explicitly save examples [36] and *pseudo-rehearsal* approaches save a generative model instead [37]. A straightforward rehearsal approach can be achieved by randomly selecting examples to add to the episodic memory [38]. However, [36] proposed to select a curated set of examples instead, such that the average feature vector of the set best matches the average feature vector of the entire task. Pseudo-rehearsal methods can use several different generative models (e.g. GANs, CVAE) [37]. However, replay using standard generative models was shown to break down for complex tasks by [39] and they proposed a modified CVAE that serves as both generator and solver by equipping it with generative feedback connections. Additionally, they propose to replay representations internally at the 'hidden level'.

Instead of saving information through examples or generative models, many works focus on expanding the model architecture [40, 41, 42]. Intuitively, these approaches prevent forgetting by populating new untouched weights instead of overwriting existing ones. However, the model complexity grows with the number of tasks.

Importantly, most of the time combining continual learning strategies allows finding the best solutions [11].

III. Problem definition

The goal of our framework is to **adapt** a trajectory prediction model from the streamed positions $\mathbf{p} = \{p_x, p_y\}$ and velocities $\mathbf{v} = \{v_x, v_y\}$ of all surrounding agents and a map $\mathcal{S} \in \mathbb{R}^2$ of the static environment. We assume a detection and tracking pipeline exists that streams the states and provides the map.

The trajectory prediction problem can be formally defined as follows: Estimate the future velocities of all tracked pedestrians from their past states including information about their environment.

For pedestrian i at time $t = 0$ his surrounding static environment is denoted as $\mathbf{s_0^i} \in \mathcal{S}$. We now refer to the past t_{obs} states of pedestrian i as:

$$\mathbf{x}_0^i = \{(\mathbf{p}_\tau^i, \mathbf{v}_\tau^i, \mathbf{s}_0^i) | \tau = -t_{obs}, \dots 0\}$$

and the future t_{pred} velocities of pedestrian i as:

$$\mathbf{y}_0^i = \mathbf{v}_{1:t_{pred}}^i = \{\mathbf{v}_\tau^i | \tau = 1, \dots t_{pred}\}$$

We want to find a data-driven prediction model $f(\theta)$ that best approximates f^*, where $f^*(\mathbf{x}_t^i) = \mathbf{y}_t^i$ across the entire previous stream of states for every tracked pedestrian $i \in [1 \dots N]$. Here θ refers to the weights of the model and N refers to the total number of tracked pedestrians. Throughout this paper the subscript i denotes the *ego pedestrian*, i.e. the pedestrian whose trajectory we want to predict.

In literature, online learning problems are often formulated as a game where the answer becomes available in hindsight [30]. The goal of online learning problems is to minimize the loss on all past rounds, often referred to as regret. We propose to view the problem of continually learning a data-driven prediction model from observations as an online learning problem:

for $t = 1, 2, ...$
- observe $\mathbf{x}_t^{1:N}$
- predict $\hat{\mathbf{y}}_t^{1:N}$
- receive true answer $\mathbf{y}_t^{1:N}$
- update $\mathscr{L}(\hat{\mathbf{y}}_t^{1:N}, \mathbf{y}_t^{1:N})$

Here $\hat{\mathbf{y}}_t$ refers to the trajectory prediction of our data-driven network ($f(\theta)$):

$$f(\mathbf{x}_t; \theta) = \hat{\mathbf{y}}_t = \hat{\mathbf{v}}_{t+1:t+t_{pred}}$$

In our setting the true answer $\hat{\mathbf{y}}_t$ will not directly become available after round t, but only after observing the ground truth trajectory that a pedestrian ends up taking for t_{pred} rounds. So the update can be done earliest after round $t+t_{pred}$.

The goal of our framework is to minimize regret for all past rounds by updating the weights θ of a data-driven prediction network:

$$\min_\theta \sum_{j=0}^{t-(t_{pred}+1)} \mathscr{L}(\hat{\mathbf{y}}_j, \mathbf{y}_j)$$

IV. Approach

In this section, we introduce Self-supervised Continual RNN Learning (SCRNN-L) an online learning framework to continually improve pedestrian prediction models. We explain how the combination of a *rehearsal* and *regularization* strategy allows SCRNN-L to overcome the problem of catastrophic forgetting. We also briefly describe the pedestrian prediction model architecture considered. SCRNN-L is designed to be used in conjunction with a pedestrian detection and tracking pipeline on a mobile robot.

A. SCRNN-L overview

SCRNN-L consists of two stages, a model adaptation stage, and a task aggregation stage, see Fig. 2. The framework proceeds as follows: Initially, we use a pre-trained prediction model on the robot and aggregate new training examples using the surrounding pedestrians as experts (task aggregation).

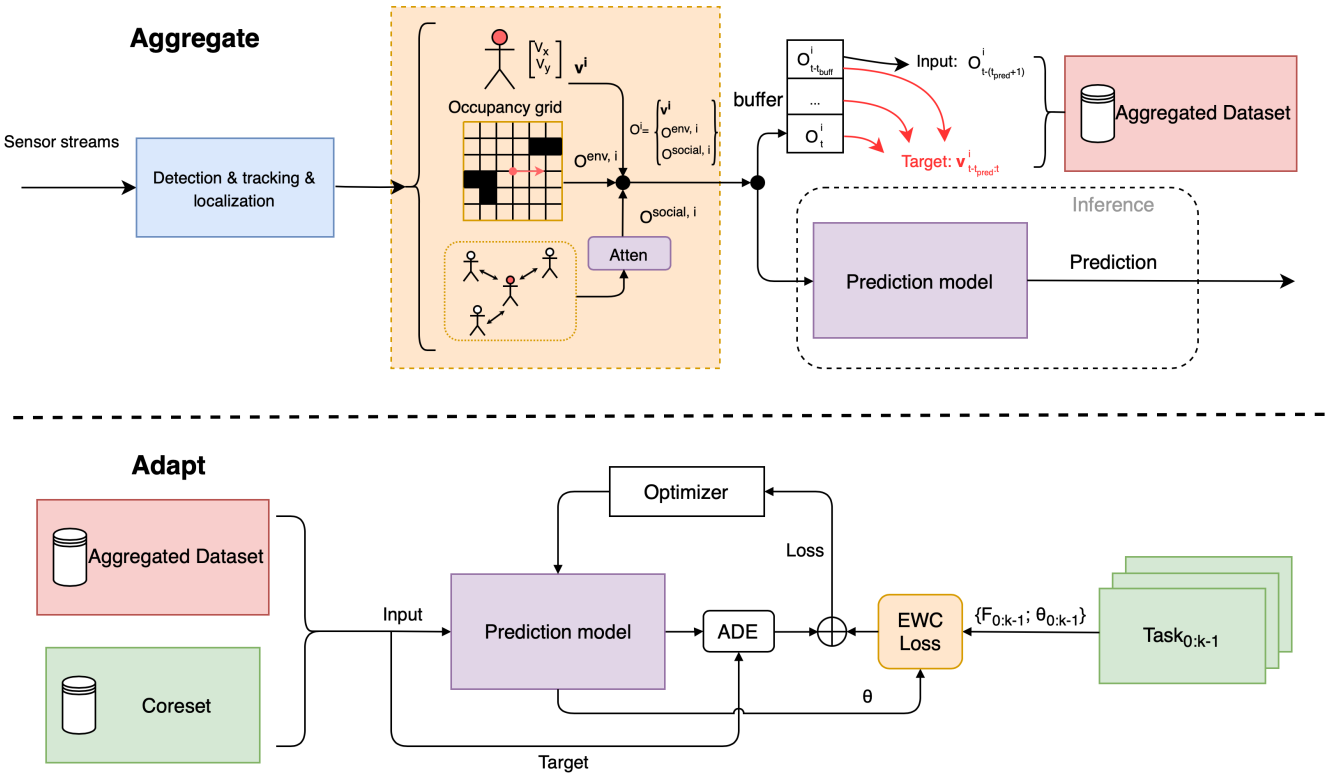


Fig. 2: Schematics of the two stages of SCRNN-L. The aggregation stage (top) extracts examples from the stream of tracked surrounding pedestrians and stores them in a temporary dataset. The adaptation stage (bottom) trains the prediction model using the aggregated dataset and a separately saved coreset. An additional EWC regularization term is added to the loss to prevent catastrophic forgetting when training online.

After T seconds the prediction model is updated using the aggregated task (model adaptation). The two stages are run alternately over time to create a lifelong learning autonomous agent.

SCRNN-L aims to learn from a time-varying continuous data stream to have the least regret across the entire stream. At the same time, SCRNN-L has to be computationally and memory-wise feasible to run on a mobile robot platform. Therefore our framework does not store all examples of the input stream. Instead, we split the input stream into temporally bounded episodes and separate our framework in an aggregation and adaption stage. The aggregated dataset is then discarded after training the prediction model in the adaptation stage. This ensures the training dataset doesn't grow over time and remains feasible to train on. The consequence of discarding the aggregated datasets is that training the prediction network becomes susceptible to catastrophic forgetting since data from previous episodes is not being trained on anymore. To address this problem, we use a regularization strategy that consolidates knowledge on an episode-by-episode basis. The optimal episode length T depends on the environment and how many pedestrians the robot is able to observe and learn from. At the same time setting T too large can cause the aggregated datasets to be too big.

Our framework uses EWC as the regularization strategy. EWC consolidates knowledge on a task-by-task basis by selectively slowing down the learning of neurons that are deemed important for previous tasks. Throughout this paper, the aggregated dataset of a single episode is referred to as a task. EWC assumes that task boundaries are given and that every task contains new feature patterns to remember. EWC is a suitable choice for our problem setting, because the distribution of the continuous data stream gradually changes across tasks, for example when the robot moves to a new environment. So examples within a single task likely contain new input and behavior patterns that EWC can consolidate. A drawback of regularization-based strategies is that they can lead to a trade-off between knowledge consolidation and performance on novel tasks. Therefore SCRNN-L also rehearses a set of examples of constant length, referred to as the *coreset*. Coreset rehearsal allows better knowledge consolidation while performing comparably on novel tasks. The downside of coreset rehearsal is that it adds a memory overhead to the framework. However, when the coreset size is small compared to the size of the aggregated task, the memory overhead on the overall framework is comparatively minimal.

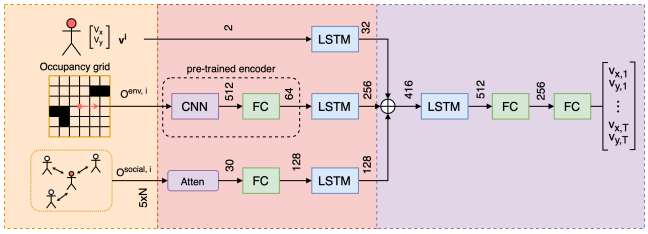


Fig. 3: Prediction Model architecture.

B. Prediction network architecture

To test our online learning framework we use a data-driven pedestrian trajectory prediction model that builds on [5]. The model architecture is shown in Fig. 3. Three streams of information are used as inputs. First, the velocity of the ego pedestrian ($\mathbf{v}_t^i$). Secondly, an occupancy grid around the ego pedestrian containing information about the static obstacles in the surrounding ($O_t^{env,i}$). For the final input, [5] use an angular pedestrian grid (APG) that encodes the relative positions of surrounding pedestrians in polar coordinates and discretizes the angle. However, during experiments, we found the model has difficulties learning social interactions. The APG lacks explicit information about relative pedestrian velocities. Instead, the model has to learn to integrate across the discretized bins of the APG using its internal LSTM state, which makes learning social interactions more difficult.

To address this limitation, we use a different input entirely. Similar to [20], we use the relative velocity, position, and heading of surrounding pedestrians and combine them into a vector:

$$O_t^{social,i} = [\mathbf{p}_t^j - \mathbf{p}_t^i, \mathbf{v}_t^j - \mathbf{v}_t^i, \phi_t^j - \phi_t^i] \quad \forall j \in [1 \dots N]$$

We use the permutation invariant $sort$ function as an attention mechanism by sorting the relative vectors of surrounding agents by euclidean distance [10]. To handle a variable number of pedestrians, only the closest 5 pedestrians relative vectors are used. For situations when there are fewer than 5 surrounding pedestrians, the relative vector of the closest pedestrian is repeated.

C. Task aggregation

During the task aggregation stage, SCRNN-L saves the inputs of the prediction model $O_t^i = \{\mathbf{v}_t^i, O_t^{env,i}, O_t^{social,i}\}$ (see Fig. 2) for every tracked pedestrian i in a buffer of the last t_{buff} previous time steps. Our key observation is that, for a model with a prediction horizon of t_{pred}, the ground truth velocity vectors $\mathbf{v}_{t:t+t_{pred}}^i$ can be extracted from the buffer after t_{pred} timesteps (red arrows in Fig. 2). We then aggregate the velocity vectors (target) together with the corresponding model inputs (input) as an example and store them in the aggregated dataset. Because our prediction model is recurrent (meaning it depends on the previous model state), we aggregate sequences of examples. Collecting structured sequences of examples, enables the model to be trained through truncated backpropagation through time (tbptt) and learn the temporal dependencies of the model state. The aggregated sequences of examples are of length t_{tbptt}, which makes the total buffer length t_{buff} equal to $t_{pred} + t_{tbptt}$.

D. Model adaptation

In the model adaptation stage, SCRNN-L uses the latest aggregated task and trains the network for Q epochs. To distinguish between tasks we will refer to a task as $task_k$, where $k = 0$ for the initial task and k increments per task.

EWC is applied during training on $task_k$ to preserve prediction performance on the previous tasks ($task_{0:k-1}$). For each previous task, we saved an importance measure F together with the network weights θ directly after training. Based on $F_{0:k-1}$ and $\theta_{0:k-1}$ the following regularization term is added to the loss function:

$$\mathscr{L}_{ewc}(\theta) = \sum_{j=0}^{k-1} \frac{\lambda}{2} F_j(\theta - \theta_j)^2$$

Where θ is the current set of weights, k refers to the index of the current task, and λ is a hyperparameter that dictates how important not forgetting the old tasks is compared to learning the new one. The complete loss function then looks as follows:

$$\mathscr{L} = \mathscr{L}_{ADE} + \mathscr{L}_{ewc}$$

$$\mathscr{L}_{ADE}(\hat{\mathbf{v}}_{1:t_{pred}}, \mathbf{v}_{1:t_{pred}}) = \frac{1}{t_{pred}} \sum_{j=1}^{t_{pred}} |\hat{\mathbf{v}}_j - \mathbf{v}_j|^2$$

$\mathscr{L}_{ADE}$ refers to the average displacement error (ADE) of the predicted trajectory, often used as a loss metric for trajectory prediction models [8].

Coreset Rehearsal is used in addition to EWC to mitigate forgetting. A set of examples (coreset) of constant length N is saved alongside the model. The goal is for the coreset to contain representative examples of all previous tasks. During the model adaptation stage, the coreset is combined with the aggregated task set to rehearse the examples. The total dataset looks as follows:

$$\hat{\mathcal{D}} = \mathcal{D}_k \bigcup \mathcal{D}_{coreset}$$

Where $\mathcal{D}_k$ contains all examples of $task_k$ and $\mathcal{D}_{coreset}$ contains all examples of the coreset. After the model adaptation stage is completed, we update the coreset with M examples of the latest task $\mathcal{D}_k$. Importantly the new examples replace existing ones to ensure the coreset remains of constant length N. We use a random selection of examples to update our coreset.

E. SCRNN-L

In summary, SCRNN-L continually adapts the pedestrian prediction model of a mobile robot using surrounding pedestrians as experts. In its simplest form, the algorithm proceeds as follows (algorithm 1), every T seconds a dataset $\mathcal{D}_t$ is aggregated. Using $\mathcal{D}_t$ and the saved $\mathcal{D}_{coreset}$ the model is adapted. During training, learning is selectively slowed down on important weights to preserve knowledge about previous environments.

Algorithm 1: SCRNN-L

Load pretrained model: $f(\theta)$
Load map: $\mathcal{S}$
Initialize coreset: $\mathcal{D}_{coreset} \leftarrow \emptyset$
for *i = 0 to* ∞ **do**
  Initilize empty task dataset: $\mathcal{D}_k \leftarrow \emptyset$
  Aggregate examples for T seconds as follows:
  for *t = 0 to T* **do**
    Process pedestrian positions $\mathbf{p}_t$, velocities $\mathbf{v}_t$, and map state $\mathcal{S}$ to model inputs O_t and save them to a buffer
    Execute prediction network $f(\theta)$ on model inputs
    Get examples from buffer: $\mathcal{D}_t = \{(O_{t-(t_{pred}+1)}, \mathbf{v}_{t-t_{pred}:t})\}$
    Update task dataset: $\mathcal{D}_k \leftarrow \mathcal{D}_k \bigcup \mathcal{D}_t$
  end
  Combine coreset and task: $\hat{\mathcal{D}} \leftarrow \mathcal{D}_k \bigcup \mathcal{D}_{coreset}$
  Train prediction model $f(\theta)$ on $\hat{\mathcal{D}}$ using EWC.
  Add EWC importances for task dataset $\mathcal{D}_k$
  Update coreset $\mathcal{D}_{coreset}$ with random examples from $\mathcal{D}_k$
  Clear $\mathcal{D}_k$ from memory
end

V. Experiments

A. Implementation Details

The model parameters of our prediction network are kept as shown in Fig. 3. We predict 15 steps with a time-step of 0.2 seconds resulting in a 3.0 seconds prediction horizon at a frequency of 5 Hz. Note that inference on the prediction model can be run at higher frequencies, only the aggregation of examples is constrained to 5 Hz. All models used in our experiments are pre-trained on the ETH and UCY pedestrian prediction datasets [25, 26], for 60 epochs with a learning rate of 2e-3 and L2 regularization of 5e-4. The total training time was 20 minutes on an Intel Core i7.

Our online learning framework will improve this pre-trained model based on the behavior of surrounding pedestrians. We use a task aggregation buffer length $t_{buff} = 30$, consisting of the 15 step prediction horizon and a 15 step truncated backpropagation through time. We switch stage after $T = 200$ seconds and train the network for $Q = 10$ epochs. For our model that includes EWC, we use $\lambda = 1e6$, and for our model that includes coreset rehearsal, we use a coreset size of $N = 100$ and an update size of $M = 20$.

B. Evaluation Details

We include two versions of our SCRNN-L method in experiments:

1) **EWC:** A prediction model that is trained using the same aggregated tasks as SCRNN-L, while also using EWC regularization, but without coreset rehearsal.
2) **EWC+Coreset:** A prediction model that is trained using SCRNN-L with both EWC regularization and Coreset rehearsal.

Furthermore, we compare against the following baselines:

1) **Vanilla:** A prediction model that is trained using the same aggregated tasks as SCRNN-L, while using standard gradient descent without any continual learning methods.
2) **Offline:** A prediction model that is trained offline on all aggregated tasks simultaneously.

The focus of this paper is on applying continual learning strategies to improve a prediction model on the fly without forgetting. Therefore we only change the learning strategy across baselines and keep the prediction network architecture the same. Similarly, we don't directly compare against the SoTA in pedestrian prediction models because our focus is on the learning strategy and not on the model architecture. Similar to other works on pedestrian prediction models, we use the average (ADE) and final displacement error (FDE) as performance metrics [10, 20].

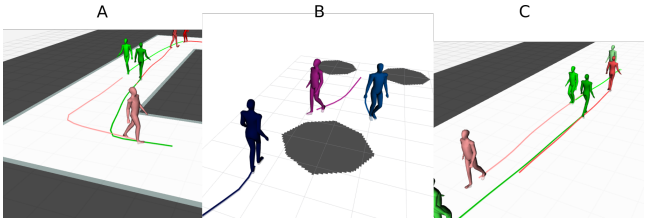


Fig. 4: The incremental learning scenario consists of (A) Square, (B) Obstacles, and (C) Hallway environments.

C. Incremental Learning Benchmark

To effectively compare learning strategies, we propose a simulation-based incremental learning benchmark for pedestrian trajectory prediction. Incremental learning scenarios are often used as continual learning benchmarks. At the same time, incremental learning benchmarks allow to easily evaluate how much a model forgets by continually reevaluating the model's performance on previous tasks. Popular continual learning benchmarks, like split MNIST [39], focus on classification problems. In classification problems, the task boundaries can simply be defined by grouping class labels, so each new task consists of new class labels. In contrast, the prediction of human motion is a regression problem hence defining task boundaries is more difficult. Since human motion is heavily influenced by environmental context, we propose to define task boundaries in our benchmark by the environment.

We include three environments that contain distinctly different motion behavior based on environmental as well as social context. We train prediction models incrementally on arbitrary orders of the environments which enables us to evaluate different online learning methods based on their ADE/FDE performance on test sets of all environments. The following three environments are considered, see Fig. 4:

1) **Square:** An infinite corridor setting with three pedestrians walking clockwise and three anticlockwise.
2) **Obstacles:** Pedestrians walking towards each other in an obstacle filled space.
3) **Hallway:** Pedestrians walking towards each other in a hallway while behaving cooperatively.

The pedestrian behavior is simulated in ROS using the open-source pedsim simulation framework [1] employing the social forces model [12].

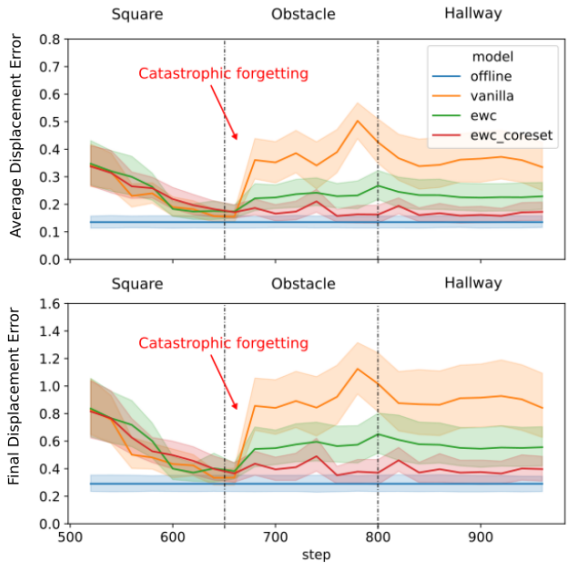


Fig. 5: Prediction performance on **only** the square scenario. Shows ADE (top) and FDE (bottom) of all models on the test set of the square task, while learning new tasks. Note that the offline model is used as an upperbound on prediction performance for comparison purposes only. Since in the real world, data of the scenarios will not be known beforehand and can be time-varying.

D. Quantitative results in simulation

We compare our methods against baselines on various sequence orders of our proposed incremental learning benchmark. Table I reports the average ADE and FDE evaluated at the end of the sequence on all three scenarios, under the columns denoted *average*. Additionally, Table I reports the average ADE and FDE increase of previous environments across the incremental learning sequence, under the columns denoted *avg. forgotten*. As shown, EWC+Coreset significantly outperforms Vanilla. Our method is also independent of sequence order, arriving at within +/- 0.02 the same average ADE/FDE for all sequence orders. The significant increase in average forgotten ADE and FDE indicates that the Vanilla baseline suffered from catastrophic forgetting across the incremental learning sequence.

To gain insight into where catastrophic forgetting takes place in the Vanilla model, we plotted the ADE/FDE of the test set of **only** the initial square scenario across the entire learning trajectory, using the sequence square $\rightarrow$ obstacle $\rightarrow$ hallway (Fig. 5). By evaluating a single environment over time, we can clearly visualize when and how much the models degraded in prediction performance in the respective environment. For ease of comparison, the offline trained model is also plotted as a constant line. In the first section, all models are trained on the aggregated dataset of the square scenario and, as expected, all online learning methods similarly converge towards the offline trained prediction model. However, when changing from the square environment to the obstacle environment, the ADE/FDE performance quickly and drastically degrades for the Vanilla baseline (red arrow). The green line in Fig. 5 depicts that using EWC to selectively slow down learning on important neurons helps to significantly mitigate the magnitude of the loss in ADE/FDE. Nevertheless, after two subsequent tasks, the EWC baseline performed $\sim 30\%$ worse on FDE and $\sim 20\%$ worse on ADE. The red line in Fig. 5 shows that combining EWC with the rehearsal of a small Coreset of examples helps to further mitigate forgetting. Using EWC and Coreset rehearsal SCRNN-L was able to train in two subsequent scenarios while retaining knowledge of the initially experienced scenario.

E. Qualitative analysis in simulation

To further inspect the benefit of EWC and Coreset rehearsal we compared predicted trajectories of all models **after** online training in the three tasks of our proposed incremental learning benchmark, depicted in Fig. 6. In Fig. 6A our EWC+Coreset method learned from observations that pedestrians are likely to follow a corner in a corridor. Furthermore, it shows that our method can maintain this prediction behavior after learning on two subsequent tasks. Note how near corners our method that solely uses EWC outperforms the Vanilla benchmark but was not able to fully consolidate it's prediction performance (green line), indicating the need for an additional method of consolidation like Coreset rehearsal. In Fig. 6B our model adapted to an environment with different static obstacle configurations and learned obstacle avoidance behavior from observations. Finally, Fig. 6C demonstrates that our model can also adapt to take social interactions into account.

F. Real-world validation with perfect observations

While the simulation results indicate EWC and Coreset rehearsal, significantly help maintain performance on previous tasks, this is not necessarily true in a real-world situation because the policy of a real pedestrian can differ from the social forces model. To validate whether it is possible to improve a prediction model with SCRNN-L from observed pedestrians in the real world, we ran a number of controlled experiments. To eliminate the perception-related errors as

[1]https://github.com/srl-freiburg/pedsim_ros

Method	square → obstacle → hallway		obstacle → square → hallway		hallway → obstacle → square		obstacle → hallway → square	
	avg. forgotten	average	avg. forgotten	average	avg. forgotten	average	avg. forgotten	average
Vanilla	+0.15 / +0.41	0.24 / 0.55	+0.15 / +0.37	0.24 / 0.52	+0.16 / +0.41	0.24 / 0.56	+0.25 / +0.63	0.30 / 0.71
Ours (EWC)	+0.04 / +0.12	0.16 / 0.37	+0.06 / +0.17	0.18 / 0.40	+0.09 / +0.22	0.21 / 0.47	+0.07 / +0.19	0.19 / 0.42
Ours (EWC+Coreset)	**+0.01 / +0.04**	**0.15 / 0.33**	**+0.03 / +0.07**	**0.17 / 0.34**	**+0.02 / +0.06**	**0.16 / 0.35**	**+0.02 / +0.05**	**0.16 / 0.35**

TABLE I: Quantitative results of EWC + Coreset vs. Vanilla on various orders of our incremental learning scenarios. The table lists the average ADE / FDE on all environments at the end of the sequence. We additionally list the average forgotten ADE / FDE, which refers to the average increase in ADE / FDE on previous environments across the learning sequence.

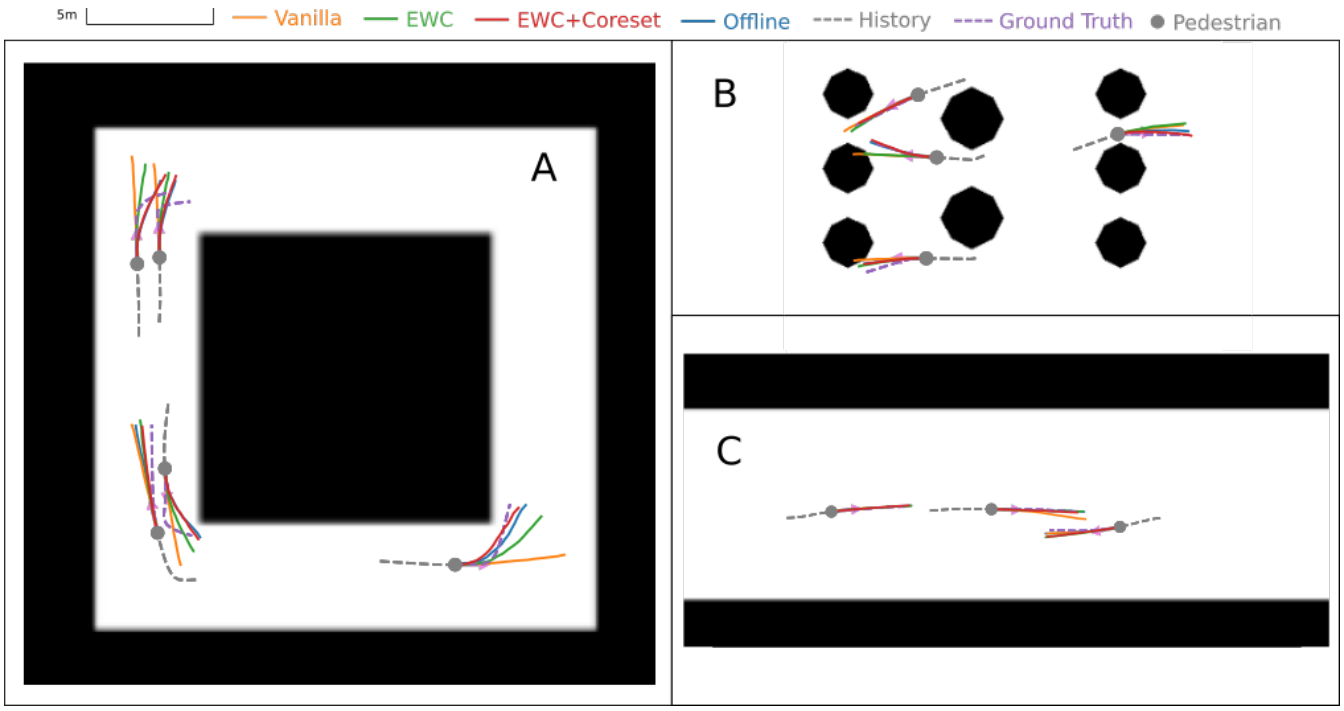


Fig. 6: Qualitative evaluation of all prediction models. The BEV of all three evaluation scenarios are depicted. The view contains obstacles (black) and pedestrians with their ground truth paths (dashed grey). Predictions are color coded by model (solid lines). The three scenarios consist of: (A) Square scenario, (B) Obstacles scenario, and (C) Hallway scenario.

Method	square → obstacle → cooperative		obstacle → square → cooperative	
	avg. forgotten	average	avg. forgotten	average
vanilla	+0.24 / +0.58	0.46 / 0.97	+0.21 / +0.50	0.45 / 0.94
Ours(EWC)	+0.19 / +0.42	0.43 / 0.86	+0.12 / +0.31	0.41 / 0.87
Ours(EWC+Coreset)	**+0.04 / +0.13**	**0.36 / 0.73**	**+0.05 / +0.11**	**0.40 / 0.80**

TABLE II: Quantitative results of EWC + Coreset vs. Vanilla on real world data collected using an optical tracking system. The table lists the average ADE / FDE on all environments at the end of the sequence. We additionally list the average forgotten ADE / FDE, which refers to the average increase in ADE / FDE on previous environments across the learning sequence.

much as possible, we first tested with an optical tracking system (optitrack) that provides pose information for all tracked pedestrians. We again started with a model pre-trained on ETH and UCY. Furthermore, we set up three scenarios similar to our simulation benchmark (see Appendix B). Because of Covid related regulations, we weren't able to test with more than two people. Nevertheless, our model was still able to improve prediction performance and learn certain concepts, such as avoiding crashing into walls, pillars, or fences. Table II reports quantitative results on two different orders of our experiment. Similar to Table I, the average ADE and FDE evaluated at the end of the sequence is shown, in addition to the average ADE and FDE increase of previous environments across the incremental learning sequence. Our framework again significantly outperformed the Vanilla baseline on both metrics. Indicating that we can not only learn a prediction model from real human motion but also consolidate the learned knowledge. Figure 7 shows a qualitative example of the experiment, where our framework learns to avoid both static obstacles and pedestrians.

G. Real-world results with integrated perception pipeline

Finally, we evaluated our framework in an uncontrolled environment using the robot's detection and tracking pipeline instead of the optitrack system. In Fig. 8 we show that our framework improves a prediction model by observing

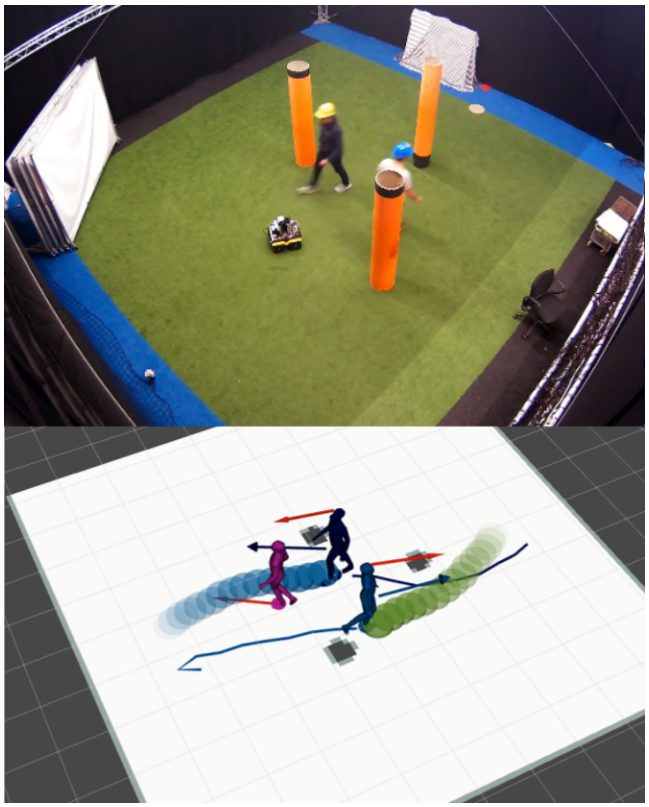


Fig. 7: Real world validation using an optical tracking system that streams tracked pedestrian states.

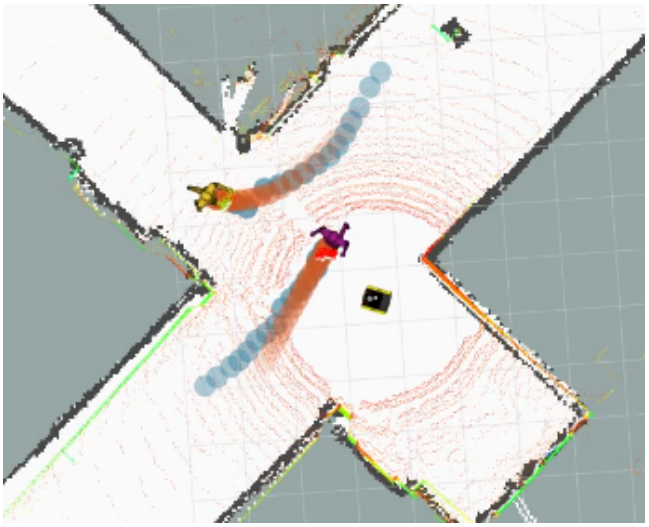


Fig. 8: Learned predictions on a real world mobile robot platform. Pedestrians are tracked using only the robots own sensors (LiDAR and cameras). The figure shows predictions (orange) and ground truths (blue).

surrounding pedestrians in a hallway. The orange markers represent the predictions and the blue markers the ground truth positions detected by the perception pipeline. The prediction model learned online when pedestrians are likely to take corners, by observing how real pedestrians walk in the environment. Note that the ETH and UCY datasets, on which our model was pre-trained, contain almost no interactions with static obstacles, yet our framework autonomously learns complex obstacle interactions. Furthermore, the occupancy map shown in Fig. 8 is generated by the robot itself using depth information from its LiDAR. So our framework can continuously learn in new and unseen environments entirely autonomously. However, in Fig. 8, the mobile robot was stationary while observing pedestrians, making it easier for the detection and tracking algorithms of our perception pipeline. In other experiments, with a moving robot, the tracked pedestrian states contained too much noise, prohibiting our framework from improving the prediction model online. Appendix B elaborates on the experiments.

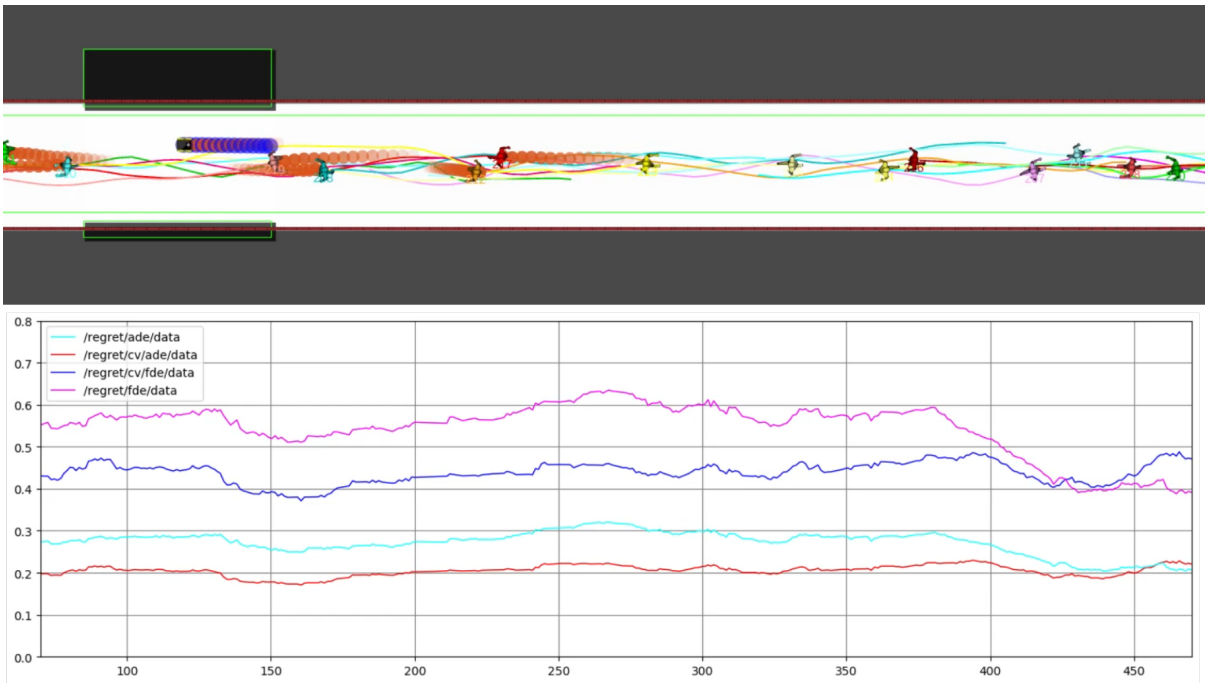
H. Integrated motion planner

To summarize, we've shown through extensive experiments that it is possible to improve a prediction model online by observing how surrounding pedestrians behave. Additionally, we've shown that using our online learning framework (SCRNN-L) allows a mobile robot platform to autonomously improve its prediction model while overcoming the problem of catastrophic forgetting.

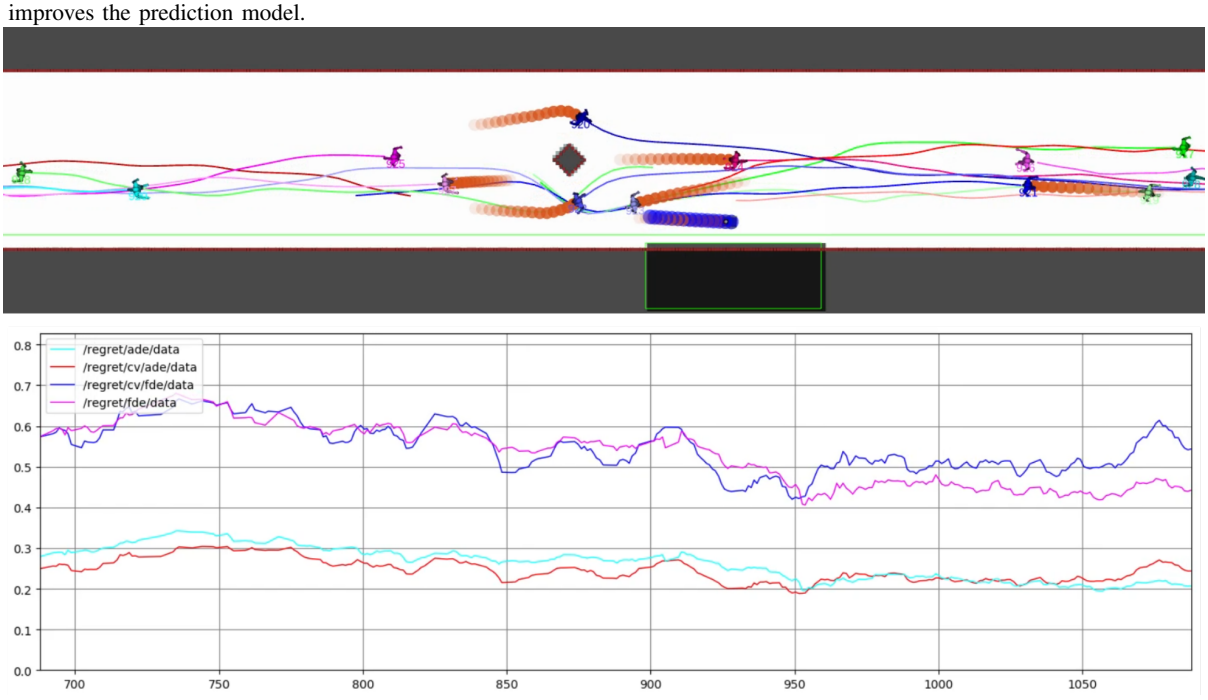
Ultimately the predictions of surrounding pedestrians allow a mobile robot platform to anticipate the intentions of pedestrians and adjust its movements accordingly. To gain insight into the integration of a motion planning algorithm with SCRNN-L we've done several experiments. Since we've found earlier that our detection and tracking pipeline on the real mobile robot platform is not yet robust enough during movement, these experiments are performed in simulation.

A model predictive control (MPC) algorithm based on [9], is used as the motion planner. MPC algorithms work by optimizing (predicting) a fixed number of time steps ahead, subject to several costs and constraints. To account for surrounding static obstacles first the obstacles are parametrized by fitting polygonal shapes around them. Using these polygonal shapes a free space area is calculated that contains no obstacle polygons. The MPC then adds as a constraint to the optimization that the predictions have to be inside the free space area. Similarly, surrounding pedestrians are accounted for, the main difference is that the location of the pedestrian can change over the prediction horizon. Therefore, using the predictions of our prediction model, a dynamic constraint is added to the optimization formulation. A circle with a fixed radius of 0.3 meters around the location of the pedestrians is used in the constraint formulation.

Initially, the above-mentioned MPC is used in conjunction with our pedestrian prediction model pre-trained on ETH and UCY. We use SCRNN-L to improve the prediction model every 200 seconds. To quantitatively evaluate the prediction



(a) Crowded hallway scenario where the robot continually moves from one side of the corridor to the other while SCRNN-L improves the prediction model.



(b) Hallway scenario with an obstacle in the middle. The robot continually moves from one side of the hallway to the other while SCRNN-L improves the prediction model.

Fig. 9: Results of various simulation scenarios combining an MPC motion planner with SCRNN-L. The running mean of the ADE and FDE of last 10 seconds is plotted. The running means can be viewed as regret values. The robot only has access to pedestrian locations within 10 meters, simulating the sensor range of a detection and tracking pipeline. Additionally, the figures show the current predictions of SCRNN-L (orange) and the current prediction horizon of the MPC (blue).

performance over time, we plot in real-time the running mean of the ADE and FDE of the past 10 seconds. Furthermore, we compare this running mean to the constant velocity (CV) predictions. CV is often used as a baseline for pedestrian prediction model [8].

Firstly a crowded hallway scenario is considered, where we instruct the mobile robot to move towards a goal position at the opposite end of the hallway, see Fig. 9a. Initially, the pre-trained prediction model performed worse than CV on ADE and FDE regret. We attribute this to the fact that the simulation scenario differs in observed behavior from the behaviors shown in the ETH and UCY dataset. After the first update stage of SCRNN-L, at approximately t = 400, our prediction model closes the gap and performs comparably to CV in ADE and FDE regret. Qualitatively the improved model predicts more cooperative trajectories. As a second scenario we placed an obstacle in the middle of the hallway, to test if, with SCRNN-L, we can take advantage of the local environmental context of pedestrians to improve predictions, see Fig. 9b. Again CV outperformed our pre-trained prediction model initially. However, after the second update stage of SCRNN-L, at approximatley t = 950, our improved prediction model now significantly outperforms CV in both ADE and FDE regret. We refer the reader to the supplementary video accompanying this paper, for a more in depth look at the results of the integrated motion planning experiments. [2]

How to appropriately quantify the performance of the motion planning algorithm is an unsolved problem. Multiple factors are of importance for instance: number of collision, time to goal and how the movement is perceived by other pedestrians. Improving the performance of the prediction model in terms of regret is the main goal of this paper, therefore we leave the further quantification of the integration with a motion planning algorithm to future work. Appendix D elaborate on our attempt at quantifying the motion planners performance using the number of collision, however we did not observe a significant improvement. We encourage future work to incorporate additional performance metrics.

VI. Conclusion

In this paper, we posed the problem of learning a pedestrian trajectory prediction model as a self-supervised continual learning problem. We proposed a continual learning framework that includes EWC and coreset rehearsal to overcome catastrophic forgetting. We show through extensive experiments that it significantly outperforms vanilla gradient descent and performs similarly to offline trained models with full access to data of the environments. To the best of our knowledge, we present the first approach that combines recent advances in the field of continual learning with a data-driven pedestrian trajectory prediction model. Additionally, we show using real-world experiments that our pedestrian prediction model can learn to generalize to new and unseen environments over time. Finally, we demonstrated the integration of SCRNN-L with an MPC-based motion planner in simulation.

[2]Video is attached to email.

References

[1] Wei He, Zhijun Li, and CL Philip Chen. "A survey of human-centered intelligent robots: issues and challenges". In: *IEEE/CAA Journal of Automatica Sinica* 4.4 (2017), pp. 602–609.

[2] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018).

[3] SF Chik et al. "A review of social-aware navigation frameworks for service robot in dynamic human environments". In: *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 8.11 (2016), pp. 41–50.

[4] Alexandre Alahi et al. "Social lstm: Human trajectory prediction in crowded spaces". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 961–971.

[5] Mark Pfeiffer et al. "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[6] Agrim Gupta et al. "Social gan: Socially acceptable trajectories with generative adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2255–2264.

[7] Peter Trautman and Andreas Krause. "Unfreezing the robot: Navigation in dense, interacting crowds". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 797–803.

[8] Andrey Rudenko et al. "Human motion trajectory prediction: A survey". In: *arXiv preprint arXiv:1905.06113* (2019).

[9] Bruno Brito et al. "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4459–4466.

[10] Amir Sadeghian et al. "Sophie: An attentive gan for predicting paths compliant to social and physical constraints". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1349–1358.

[11] Timothée Lesort et al. "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges". In: *Information Fusion* 58 (2020), pp. 52–68.

[12] Dirk Helbing and Peter Molnar. "Social force model for pedestrian dynamics". In: *Physical review E* 51.5 (1995), p. 4282.

[13] Tharindu Fernando et al. “Soft+ hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection”. In: *Neural networks* 108 (2018), pp. 466–478.

[14] Namhoon Lee et al. “Desire: Distant future prediction in dynamic scenes with interacting agents”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 336–345.

[15] Tim Salzmann et al. “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data”. In: *arXiv preprint arXiv:2001.03093* (2020).

[16] Irtiza Hasan et al. “Mx-lstm: mixing tracklets and vislets to jointly forecast trajectories and head poses”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6067–6076.

[17] Julian FP Kooij et al. “Context-based path prediction for targets with switching dynamics”. In: *International Journal of Computer Vision* 127.3 (2019), pp. 239–262.

[18] Ewoud AI Pool, Julian FP Kooij, and Dariu M Gavrila. “Context-based cyclist path prediction using Recurrent Neural Networks”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 824–830.

[19] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. “Social ways: Learning multi-modal distributions of pedestrian trajectories with gans”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[20] Bruno Brito et al. “Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians”. In: *arXiv preprint arXiv:2010.09056* (2020).

[21] Vineet Kosaraju et al. “Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks”. In: *arXiv preprint arXiv:1907.03395* (2019).

[22] Abduallah Mohamed et al. “Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14424–14432.

[23] Ashish Vaswani et al. “Attention is all you need”. In: *arXiv preprint arXiv:1706.03762* (2017).

[24] Cunjun Yu et al. “Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 507–523.

[25] Stefano Pellegrini et al. “You’ll never walk alone: Modeling social behavior for multi-target tracking”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 261–268.

[26] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. “Crowds by example”. In: *Computer graphics forum*. Vol. 26. 3. Wiley Online Library. 2007, pp. 655–664.

[27] Holger Caesar et al. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11621–11631.

[28] Ming-Fang Chang et al. “Argoverse: 3d tracking and forecasting with rich maps”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8748–8757.

[29] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 627–635.

[30] Shai Shalev-Shwartz et al. “Online learning and online convex optimization”. In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[31] German I Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural Networks* (2019).

[32] Robert M French. “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[33] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[34] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. “Task-free continual learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11254–11263.

[35] Friedemann Zenke, Ben Poole, and Surya Ganguli. “Continual learning through synaptic intelligence”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3987–3995.

[36] Sylvestre-Alvise Rebuffi et al. “icarl: Incremental classifier and representation learning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 2001–2010.

[37] Hanul Shin et al. “Continual learning with deep generative replay”. In: *Advances in neural information processing systems*. 2017, pp. 2990–2999.

[38] Timothée Lesort et al. “Generative models from the perspective of continual learning”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[39] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. “Brain-inspired replay for continual learning with artificial neural networks”. In: *Nature communications* 11.1 (2020), pp. 1–14.

[40] Jaehong Yoon et al. “Lifelong learning with dynamically expandable networks”. In: *arXiv preprint arXiv:1708.01547* (2017).

[41] Steven CY Hung et al. “Compacting, picking and growing for unforgetting continual learning”. In: *arXiv preprint arXiv:1910.06562* (2019).

[42] Xilai Li et al. “Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3925–3934.

3

Discussion

In general, the proposed self-supervised continual learning framework is a feasible solution to train a pedestrian prediction model on an online stream of data. Although research in the field of continual learning shows the online training of neural networks can be applied to robotics, to the best of our knowledge, we are the first to apply continual learning methods to the pedestrian prediction problem. The division of the online learning procedure in dedicated aggregation and adaptation stages allowed us to incrementally learn and consolidate new behaviors experienced in the surrounding. Additionally, the incremental learning of temporally bounded aggregated datasets, allowed us to run the framework using the onboard perception and computational resources of the mobile robot platform. The regularization-based approach towards mitigating catastrophic forgetting met the expectation that it significantly reduced the magnitude of the loss in prediction performance over time. The essential trade-off between stability and plasticity of regularization-based approaches encouraged us to incorporate a secondary means of consolidation. Through the rehearsal of a small persistent set of examples, the magnitude of loss in performance over time is further mitigated.

3-1 Data Aggregation

We have presented a self-supervised approach towards the aggregation of pedestrian trajectory examples, using surrounding pedestrians as experts. Combining the aggregation procedure with the inference of the prediction model allowed to store the already pre-processed examples, circumventing a significant portion of computation time in the following adaptation stage. The results of our integrated motion planner experiment showed that the temporally bounded data aggregation procedure is feasible in real-time in a crowded area with a limited sensing range. An observed drawback of this approach however is the fixed aggregation time. In the simulation, the density of surrounding pedestrians was relatively constant over time, which resulted in equally sized aggregated datasets. Nevertheless, in real-world experiments, there were few pedestrians around, because of the covid regulations on campus. This resulted in relatively few periods of dense crowds and large periods without any observations. We

encourage future work to apply different methods of aggregation e.g. by having a fixed minimum number of aggregated examples, or by separating the aggregation spatially instead of temporally. Finally, the performance of the data aggregation procedure is tightly coupled to the accuracy of the detection and tracking pipeline. In real-world experiments, the frequency of observed tracks of our detection and tracking pipeline was inconsistent and the tracks themselves contained too many false positives, resulting in inappropriate ground truth values for many aggregated examples. Additionally, it is not yet well understood if occlusions in the view of the robot's sensors have an impact on the aggregated examples and subsequently on the learned prediction model.

3-2 Elastic Weight Consolidation

The task-based EWC regularization method was shown to significantly mitigate the magnitude of the loss in performance in average and final displacement error in our pedestrian prediction incremental learning benchmark. Furthermore, the performance on novel tasks remained comparative to vanilla gradient descent. By using EWC we essentially save the information contained in the latest aggregated dataset, in a more compressed form as Fisher importances on the neurons of the prediction network. The inherent limitation of the current implementation of EWC, in the long run, is that the number of saved parameters still grows with the number of previous tasks, albeit not as rapidly as saving the dataset directly.

3-3 Prediction Network Architecture

The proposed obstacle-aware prediction model was shown to be able to use the generated occupancy grid map of the environment. The occupancy grid is generated using the onboard 3D LiDAR, allowing the online learning framework to take advantage of changing environmental context. Explicitly using a vector of relative position, velocity, and heading enabled the prediction model to take social interactions into account. Overall, the purpose of our proposed prediction model was to show through a relatively straightforward prediction model architecture that the online learning framework can learn social as well as environmental interactions. Future research is needed to verify the proposed online learning approach is feasible for SoTA multi-modal pedestrian prediction models.

3-4 Motion Planner Integration

By modeling the predicted trajectories of pedestrians as dynamic obstacles in the constraint formulation of a Model Predictive Controller (MPC), we demonstrated it is possible to integrate the presented online learning framework with a socially aware motion planning algorithm. Through the simulation of a dense crowd in a hallway, we demonstrated the prediction error goes down over time yet we did not observe a significant decrease in the number of collisions. Further research is needed to evaluate if the improvement of the prediction model translates into quantifiable improvements

of the socially aware motion planner.

4

Conclusion

The field of pedestrian trajectory prediction is moving towards interaction-aware neural network models. An inherent limitation of the way neural networks are currently used is that they are trained on a constant dataset of examples that are known and available beforehand. However, in the field of pedestrian prediction for autonomous navigation, data comes in sequential never-ending streams. This thesis work focussed on contributing to the field of pedestrian trajectory prediction by taking advantage of the online data streams of a detection and tracking pipeline, to continually train a neural network. The first research question is directed towards the feasibility of learning from online observations to improve a neural network-based prediction model, since to our knowledge this concept is not explored in previous works. With the second research question, we aimed to find a stable learning algorithm that consolidates knowledge over time. Finally, the last research question seeks to find out whether the concept can be applied in combination with a socially aware motion planning algorithm. The presented self-supervised continual learning framework, including extensive experimental evaluation, provides an answer to the research questions. In both simulation and real-world experiments, we've shown that it is possible to learn from an online stream of tracked pedestrians. By designing an incremental learning benchmark for pedestrian trajectory prediction, we demonstrated the framework can consolidate prediction performance over time. Lastly, we showed in simulation that it's possible to integrate the framework with an MPC socially-aware motion planning algorithm. Due to problems with our detection and tracking pipeline we were not able to perform real-world experiments with an integrated motion planner. Further improvements on the proposed framework can be made by exploring different aggregation strategies, using SoTA prediction models, and incorporating recent more complex methods from the field of continual learning.

5

Recommendations For Future Work

1. **Investigate different methods of defining aggregated tasks.** The proposed online learning framework works with fix temporally bounded aggregation intervals. During real-world experiments, this form aggregation proved sub-optimal because the aggregated dataset size is determined by the number of surrounding pedestrians, which is not necessarily constant over time. Recent work on continual learning proposed to automate the process of determining when to update the model, by monitoring the loss over time [24]. Additionally bounding the aggregation intervals spatially instead of temporally is worth investigating.

2. **Integrate SoTA multi-modal pedestrian prediction model.** Recent pedestrian prediction models aim to capture the multi-modality of human motion by using generative neural network models like GANs or CVAEs. The currently proposed framework is only tested using a deterministic prediction model. Further research is required to make sure the proposed online learning framework allows for training generative models.

3. **Incorporate recent innovations in the field of continual learning.** The field of continual learning has recently seen an increasing number of research papers. Promising proposed solutions to the problem of catastrophic forgetting include: Generative Replay methods [25, 26], Network Expansion methods [27, 28, 29].

4. **Thorough evaluation of integrated motion planner with additional evaluation metrics.** In our attempt at quantifying the performance of our proposed online learning framework integrated with a socially aware MPC, we looked at the number of collisions, travel time, average prediction error, and final prediction error. Further evaluation is required, with additional metrics to indicate the social acceptance of the motion planner.

5. **Investigate different approaches to integrate the proposed online learning framework with a motion planning algorithm.** In the demonstrated integration of the proposed online learning framework with a socially aware MPC, the predictions were only used in the constraint formulation to set the location of dynamic obstacles. In future research it could be promising to use the pedes-

trian prediction model on the robot itself to use as a guide, possibly making the movement of the robot more predictable towards humans.

6. **Incorporate onboard camera view around detected pedestrian as additional input for improved feature extraction.** Incorporating high-dimensional context information for pedestrian trajectory prediction is a promising area of research. In the proposed online learning framework an occupancy grid of the environment is used, however, it is worth investigating if it is possible to leverage the onboard cameras of our mobile robot to extract additional context features from surrounding pedestrians and improve the prediction accuracy online.
7. **Use pedestrian goal estimator instead of trajectory predictor.** The prediction of a fixed horizon trajectory is but one way to gain insight into the intention of surrounding pedestrians. Another option to predict a pedestrian's goal location in the environment. Incorporating a pedestrian goal estimator in the proposed online learning framework could be a promising area of research.

A

Mobile Robot Platform

This research was done as part of an MSc thesis. The appendixes, therefore, contain additional information on related subjects that were explored during the thesis. In this section, we give an overview of our in-house designed and built mobile robot platform Appendix A-1. In, Appendix A-2 we explain unique design choices.

A-1 Overview

During my thesis, I helped design, build and integrate a mobile robot platform from scratch. We started with a bare-bones Jackal UAGV mobile robot unit [1], without any sensors except for the built-in imu and GPS. The Jackal is a small robotics research platform with an onboard computer, which runs Ubuntu Linux, see Fig. A.1. The jackal is a differential wheeled robot, which means the two wheels on each side of the robot are driven independently. This removes the need for an explicit steering mechanism.

Figure A.1: Jackal Unmanned Autonomous Ground Vehicle (UAGV)

[1] `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/` (accessed April 10, 2021)

The onboard computer has an Intel Core i5 4570T Dual-core processor at 2.9 GHz and is only meant to run the low-level controllers for wheels and other non computationally heavy workloads. To run a detection and tracking pipeline we, therefore, chose to add a computer in the form of a laptop on top of the Jackal. Furthermore, we use the combination of an ouster 64-layer LiDAR and five intel stereo cameras to accurately build occupancy maps of the environment, localize the robot, and detect and track surrounding pedestrians. The cameras are mounted in a pentagonal pattern to give full 360° vision. All devices on the Jackal are on the same local area network (LAN) by connecting them through an ethernet switch. To allow for remote access we extended the network using a Deco m9 router that includes a wireless access point. This makes it possible to wirelessly connect to the ROS master of the Jackal and visualize all internal messages of the network. Our fully assembled mobile robot platform is shown in Fig. A.2.

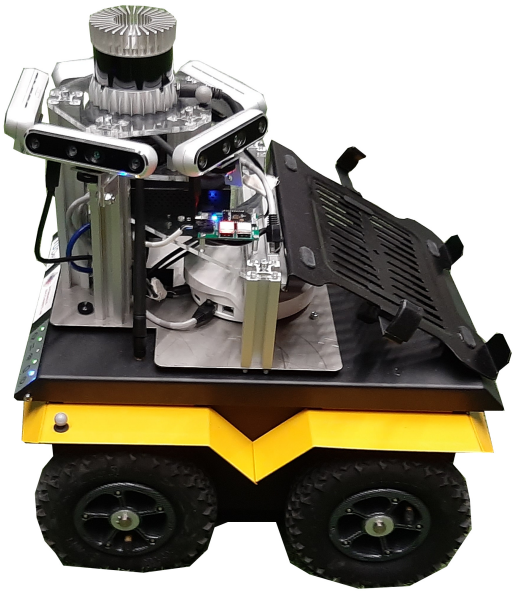


Figure A.2: Mobile robot development platform

A-2 Design Choices

A-2-1 Dedicated Laptop For Heavy Workloads

As mentioned the onboard computer of the Jackal alone is not computationally powerful enough for our purposes. Therefore, we use a Dell G7 laptop as the second computer for heavy computational workloads. The laptop includes an Intel Core i7-8th gen and an Nvidia GTX 2070. We chose a laptop over a small form-factor computer like an intel NUC and/or Nvidia Xavier because a laptop is more convenient as a research and development platform. We can use the laptop independently from the mobile robot platform, which allows remote working. Additionally, the Dell G7 outperforms most small form-factor computers both in CPU and GPU performance. With development convenience and flexibility in mind, we designed a mounting mechanism to

house the laptop on top of the Jackal, see Fig. A.2. The laptop can be entirely disconnected from the Jackal by unclipping the clamps and disconnecting the cables without requiring any tools.

A-2-2 LiDAR And Cameras Setup

One of the key factors when designing our sensor setup was the detection and tracking pipeline. Although the detection and tracking pipeline is not part of this thesis, I'll shortly explain how it works in simplified terms to justify our design choices. First, the 3D point cloud of the LiDAR sensor is fused with depth information of the five stereo cameras. Using the fused point cloud a NN is used to detect pedestrians. Additionally, a Yolo pedestrian detector runs directly on all five camera images. The Yolo detections are used to validate the detections made by the point cloud detector. Since all 3D detections are to be verified by a corresponding Yolo detection, we require full 360° camera vision. The additional benefit of being able to detect pedestrians in all directions is that our mobile robot platform can observe more pedestrians and possibly for longer trajectories. This benefits our proposed online learning method since we essentially observe and learn from more pedestrians.

B

Additional Experimental Results

B-1 Optitrack Experiments

Our simulation experiments were useful, but ultimately we wanted to know if it also works on real pedestrians. An autonomous pipeline of a robot is a complex system and there are multiple possible sources of errors or inaccuracies, such as the robot's sensing, localization, detection, and tracking. To get rid of as many additional sources of error as possible we used an optical tracking system, that we refer to as the Cyberzoo. By placing reflective markers on the robot and helmets, the overhead camera array can accurately track the poses. A minimum of three markers is required per tracked rigid body, to estimate full pose information. The pose information is streamed through ROS onto the LAN of the robot. In total we did experiments in three different scenarios, similar to our simulation scenarios, see Fig. B.1.

B-2 Real World Experiments

The logical next step after our Cyberzoo experiments was to test our framework on the robot using its localization, detection, and tracking pipelines. We tested our network throughout the TU Delft 3ME faculty building. Every 200 seconds we aggregated a dataset using the people walking by as examples and updated our prediction model.

B-2-1 Static Robot

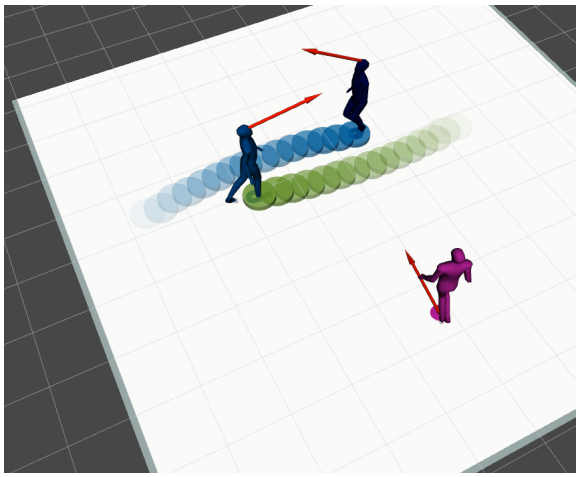
In the first two experiments, the robot remained stationary while observing pedestrians. The first test scenario consisted of an intersection in a hallway, see Fig. B.2(a). The robot remained stationary at the intersection to observe people from all directions. As shown in Fig. B.2(a), after observing pedestrians for 200 seconds the prediction model learned to infer from the surrounding local static environment of pedestrians that they are going to take a corner. The second test scenario consisted of a more open hallway at the main entrance of the building, see Fig. B.2(b). Again the robot remained stationary in the environment while observing. This time the predictions were

less affected by static obstacles since there was more open space.

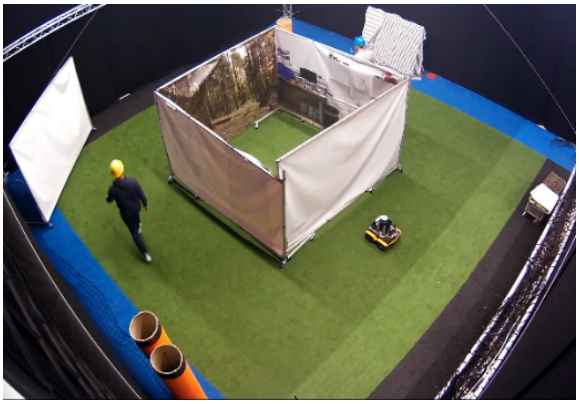
B-2-2 Dynamic Robot

In our final experiments, we drove the robot manually across the building. The goal of this experiment was to show the learned predictions generalize across environments. Fig. B.3, depicts a snapshot of one of the experiments. However, we did not observe a significant improvement of predictions by learning online while moving the robot. We hypothesize that, when moving the robot, the quality of the tracks of pedestrians was not high enough, and too many false positives are introduced in the aggregated dataset.

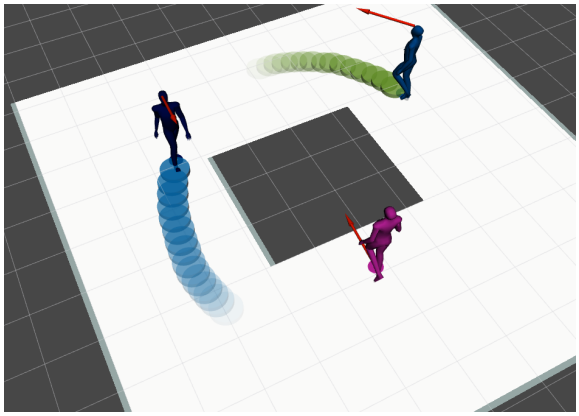
(a) Camera view of cooperative movement scenario.



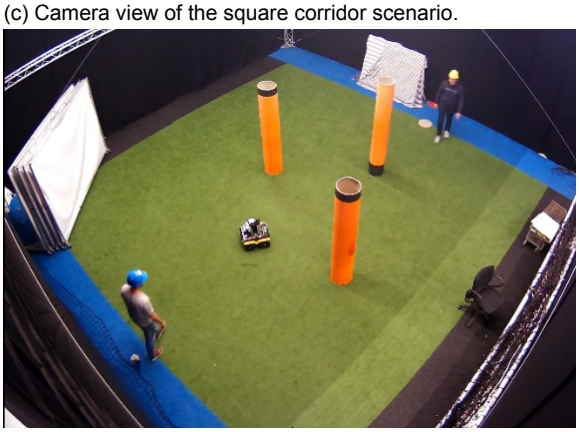
(b) Predictions after learning online.



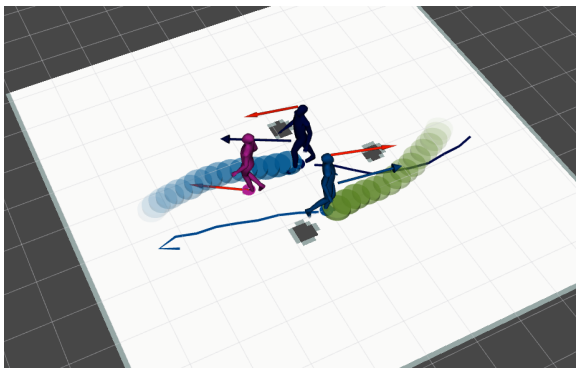
(c) Camera view of the square corridor scenario.



(d) Predictions after learning online.

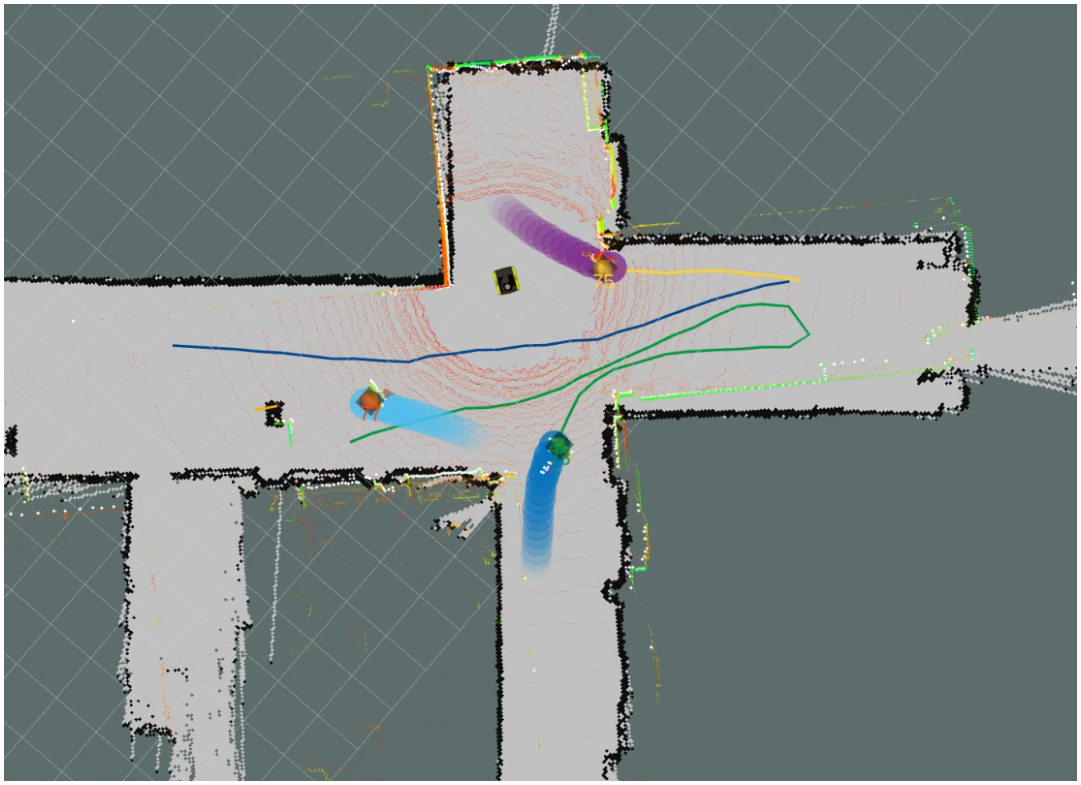


(e) Camera view of the obstacles scenario.

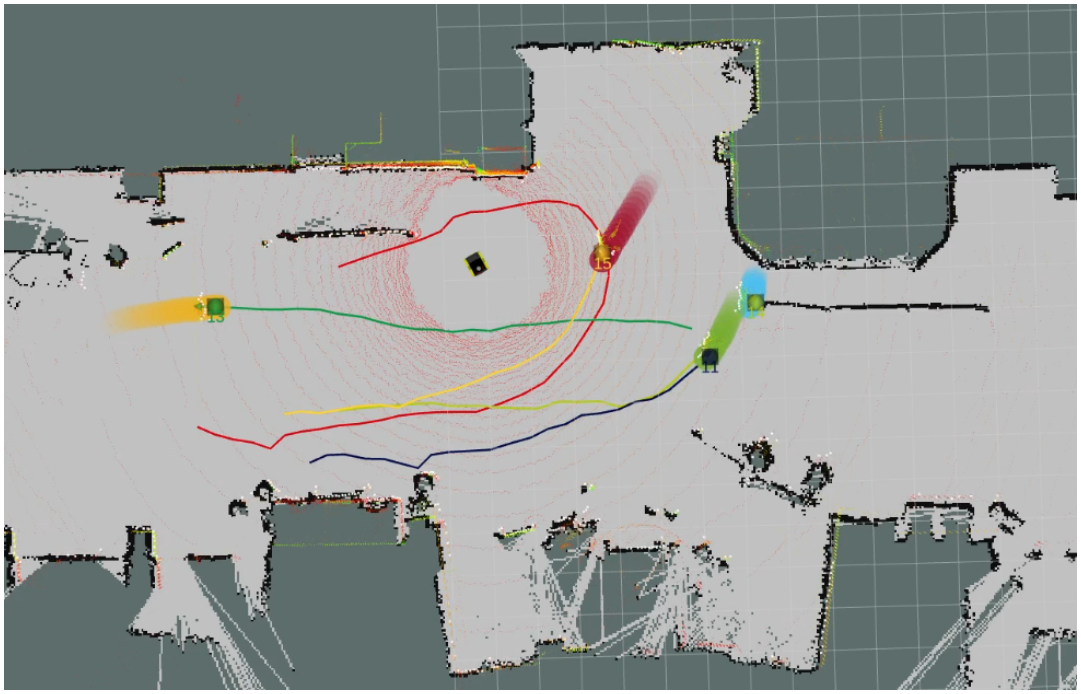


(f) Predictions after learning online.

Figure B.1: Examples of predictions after learning online in cyberzoo test scenarios.



(a) BEV of the intersection between hallways, including predictions after learning online.



(b) BEV of the mainentrance, including predictions after learning online.

Figure B.2: Real world tests. All agents are tracked using the robots detection and tracking pipeline.

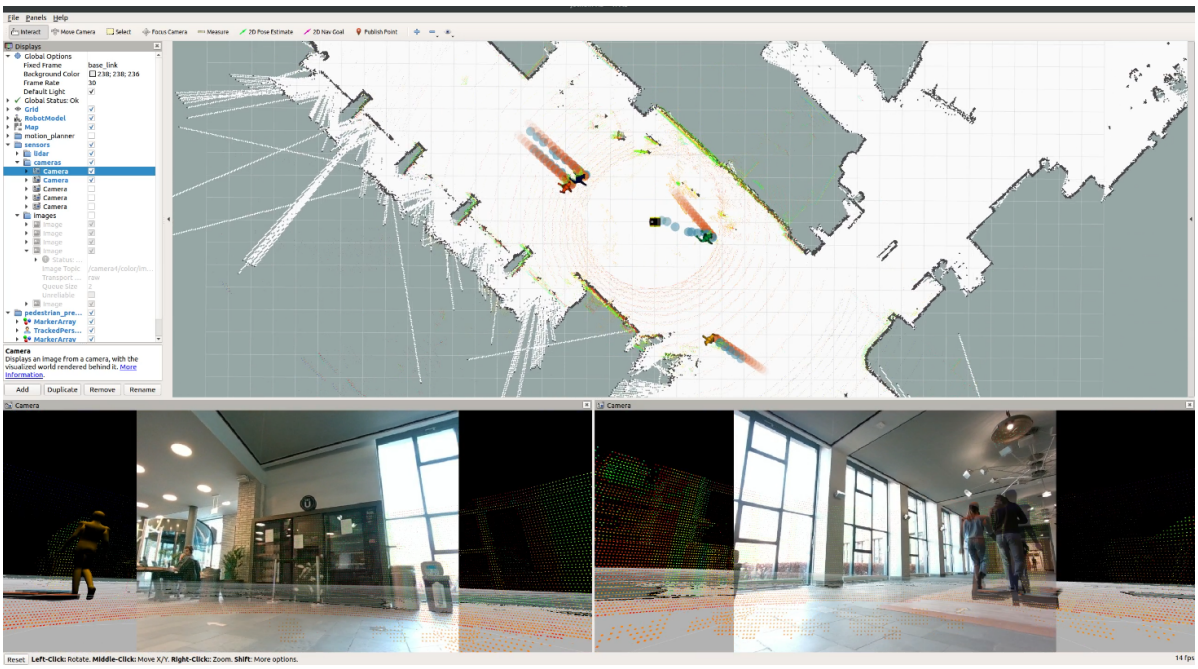


Figure B.3: Real world tests. All agents are tracked using the robots detection and tracking pipeline.

C

Prediction Network Implementation Details

C-1 Prediction Network Details

In this research, a recurrent neural network prediction model is used with model architecture as in Fig. C.1.

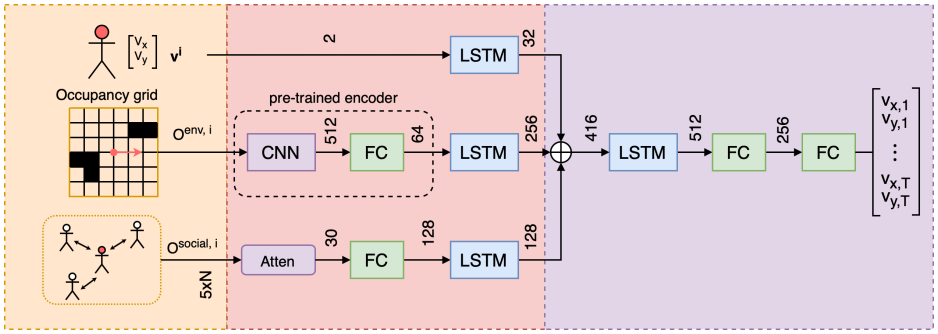


Figure C.1: Prediction Model architecture.

Encoder To extract environmental features from the local occupancy grids surrounding pedestrians, we use a multi-layer convolutional neural network (CNN). The output of the CNN is flattened and passed through a multi-layer perceptron (MLP). We first train this network separately from the rest of the prediction network and freeze the weights afterward. The purpose of the CNN is to extract descriptive features of the local occupancy grid and these features are subsequently used by an LSTM module. Because we only have access to a limited number of examples including obstacles, we did not train the CNN in conjunction with the rest of the network. Instead, the CNN was trained as the encoder part of an autoencoder. We used the MNIST dataset to pre-train the encoder (CNN), the MNIST dataset contains single layer (black and white) images of numbers. In our obstacle grids we also mostly deal with sharp edges and corners. In an obstacle grid, a cell is either an obstacle or empty space and in

the MNIST dataset, a cell is either a letter or not a letter. The similar binary nature of these tasks encouraged us to use MNIST to pre-train.

Other pedestrians Social interactions between pedestrians affect their motion plans, therefore the prediction network has a separate input for information on surrounding pedestrians. Like [19], we encode every surrounding pedestrian as a relative vector of $[\delta x, \delta y, \delta v_x, \delta v_y, \delta \phi]$. Next, the vectors are appended together to form the input of an LSTM. Since we use the LSTM as a sequence across time, the input has to be of fixed size each timestep. However, the number of surrounding pedestrians is variable. Therefore, we sort the pedestrian vectors by Euclidean distance, so that the furthest away pedestrian is appended last. Now we simply crop the input to only contain a fixed amount of pedestrian vectors. When there are fewer than expected surrounding pedestrians, we pad the input with the vector of the closest pedestrian until it's of a fixed size again.

C-2 Pre-training Data Details

For all our experiments we use the ETH and UCY datasets to pre-train the networks. The ETH dataset contains data collected from two locations:

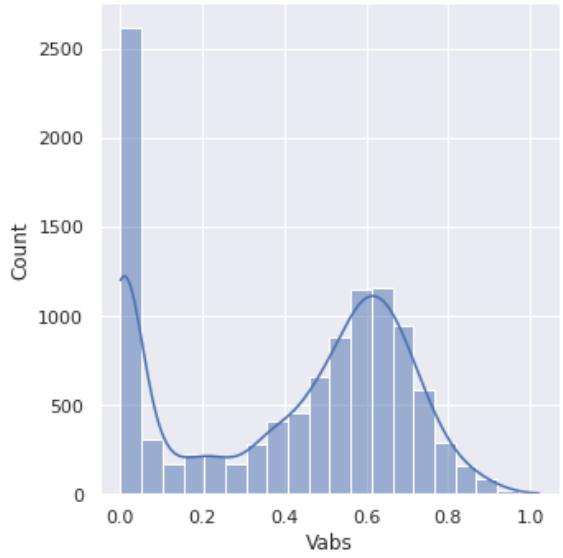
1. Hotel: A busy street outside the entrance of a hotel. There is also a tram present in the scenario (Fig. C.2).
2. Univ: The outside of the entrance of an ETH university building (Fig. C.3).

The UCY dataset contains data collected from two locations:

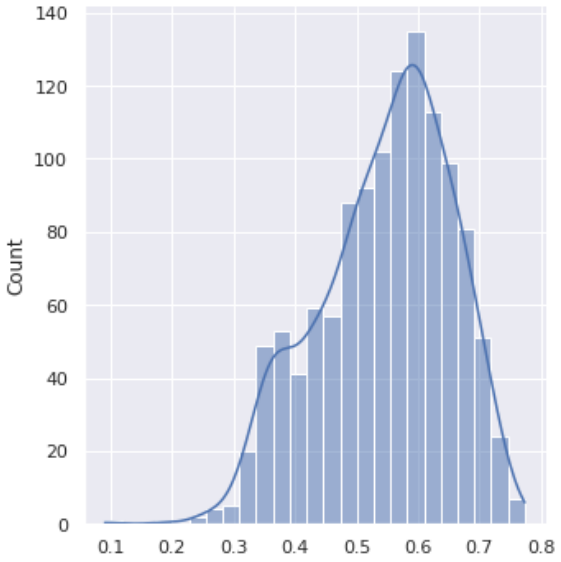
1. Zara: A busy street outside of a shop (Fig. C.4).
2. University: A very crowded park, including many groups of pedestrians (Fig. C.5).

To get a better understanding of our training data we visualized the follow:

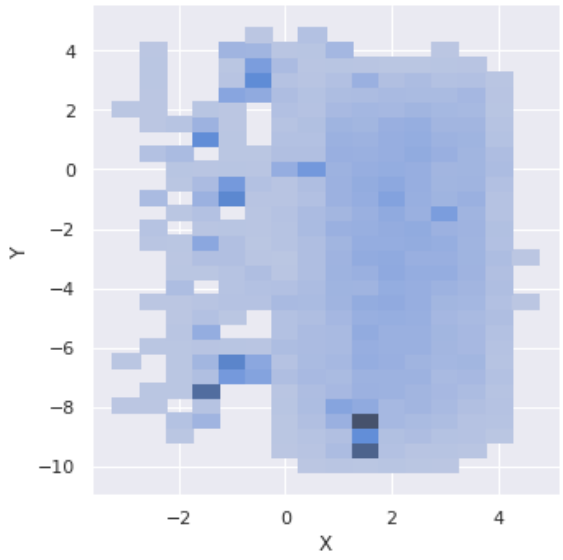
1. Velocity profile of all tracks in datasets.
2. Velocity profile of all training examples. The distribution of training examples is different because we filtered out examples of people that are standing still.
3. A BEV heatmap of the scenario, to get a better understanding of the densest areas in the dataset.
4. Ground truth prediction profile distribution of all training examples, relative to the pedestrians heading. As expected the mean of the distribution lies at an approximately constant velocity.



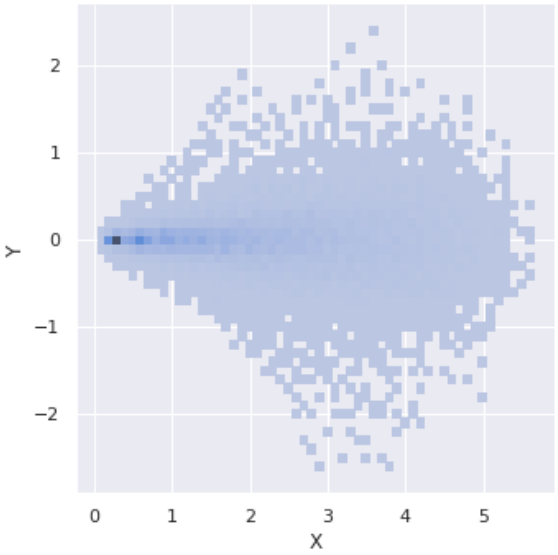
(a) Velocity profile of all tracks in dataset.



(b) Velocity profile of all training examples. We filtered out the undesired peak at velocity of zero.

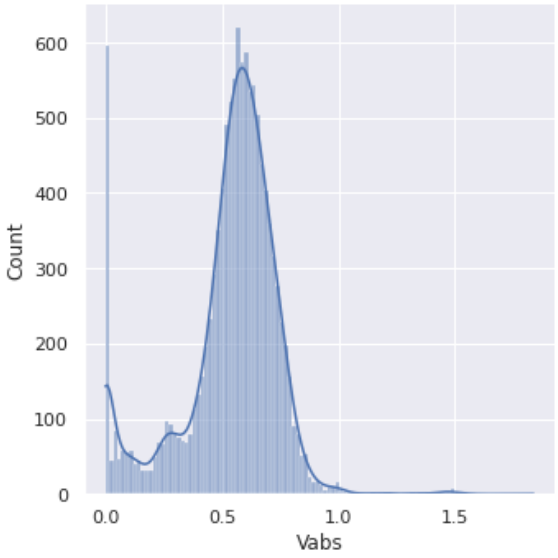


(c) Birds eye view heatmap of the scenario.

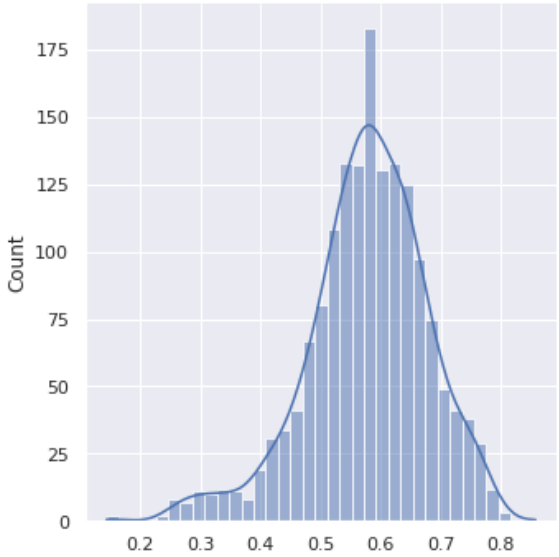


(d) Ground truth prediction profile heatmap, relative to the pedestrians heading. As expected the mean of the distribution lies at approximately constant velocity.

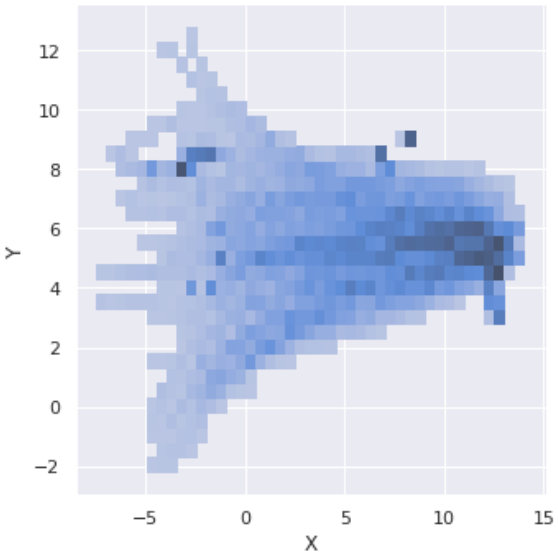
Figure C.2: ETH hotel dataset



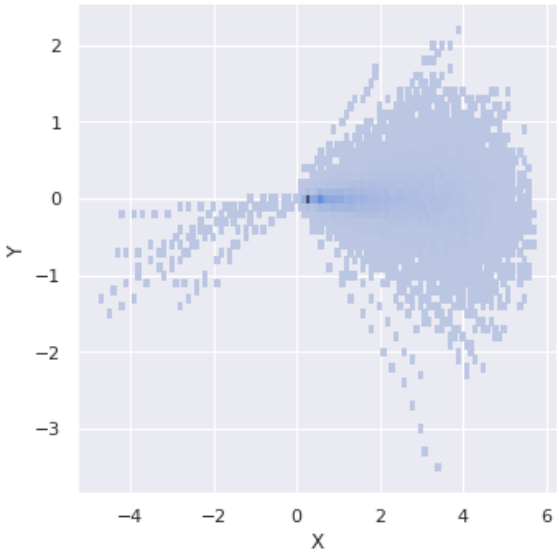
(a) Velocity profile of all tracks in dataset.



(b) Velocity profile of all training examples. We filtered out the undesired peak at velocity of zero.

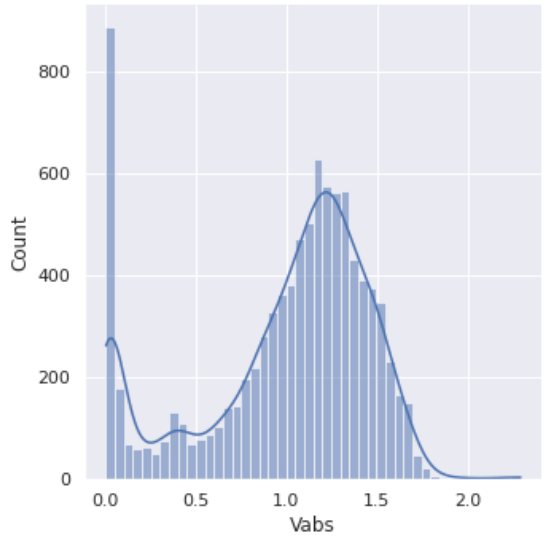


(c) Birds eye view heatmap of the scenario.

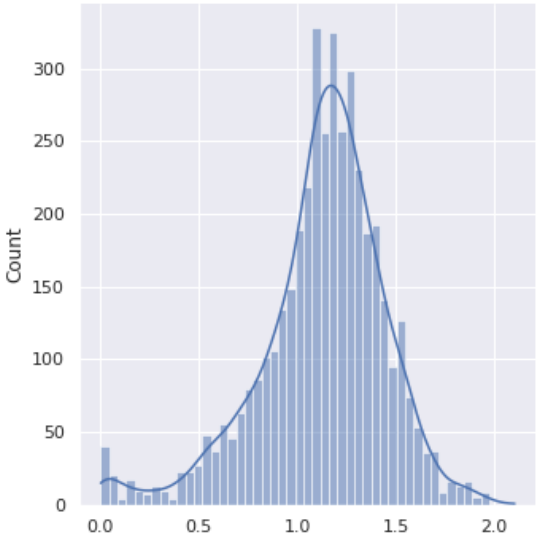


(d) Ground truth prediction profile heatmap, relative to the pedestrians heading. As expected the mean of the distribution lies at approximately constant velocity.

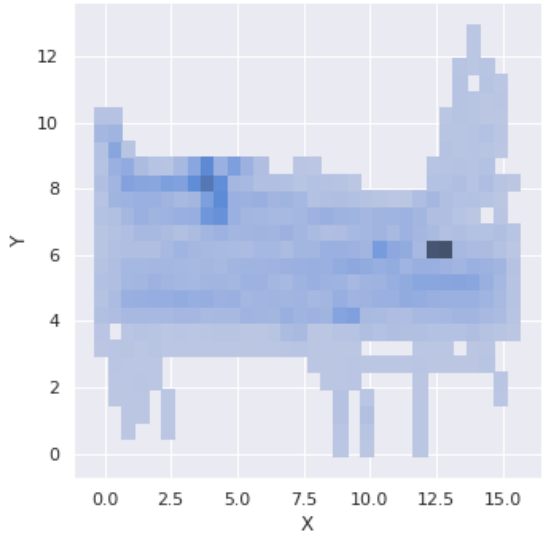
Figure C.3: ETH university dataset



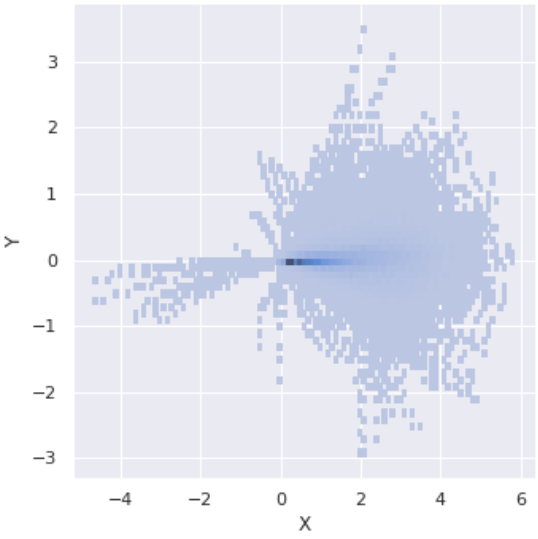
(a) Velocity profile of all tracks in dataset.



(b) Velocity profile of all training examples. We filtered out the undesired peak at velocity of zero.

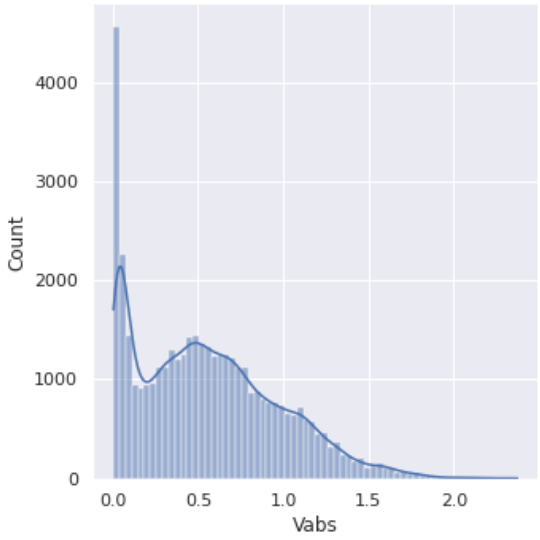


(c) Birds eye view heatmap of the scenario.

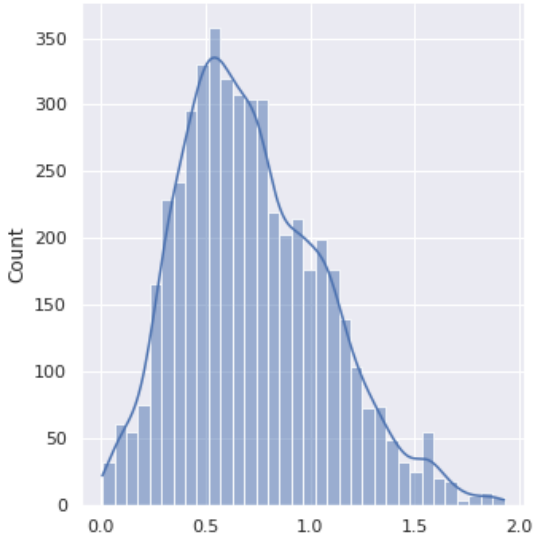


(d) Ground truth prediction profile heatmap, relative to the pedestrians heading. As expected the mean of the distribution lies at approximately constant velocity.

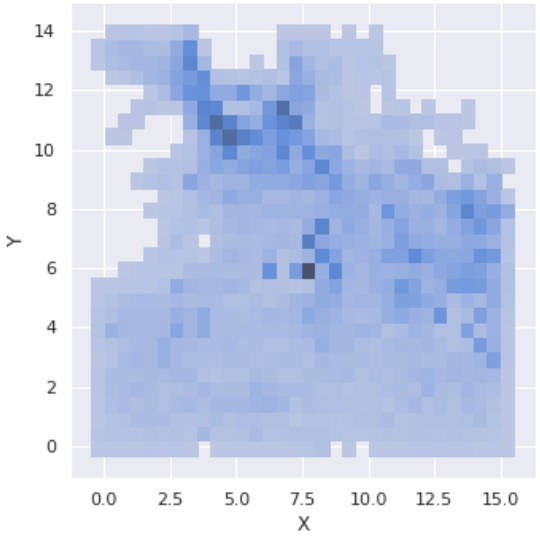
Figure C.4: Ucy Zara dataset



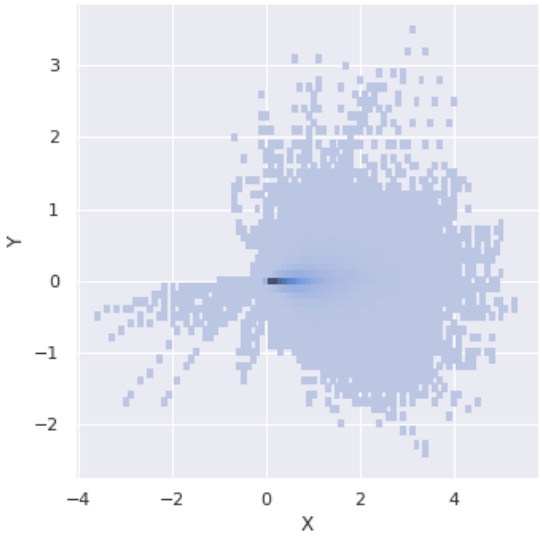
(a) Velocity profile of all tracks in dataset.



(b) Velocity profile of all training examples. We filtered out the undesired peak at velocity of zero.

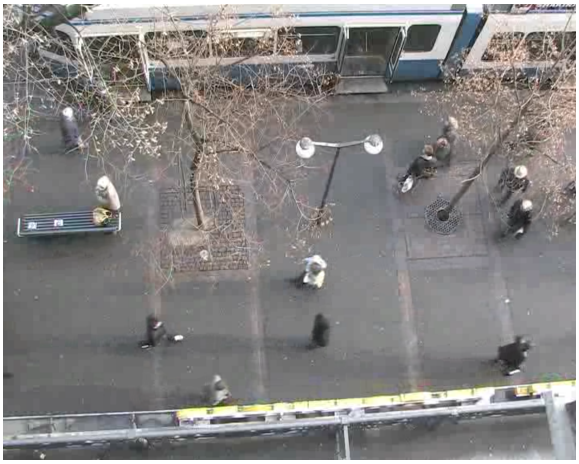


(c) Birds eye view heatmap of the scenario.



(d) Ground truth prediction profile heatmap, relative to the pedestrians heading. As expected the mean of the distribution lies at approximately constant velocity.

Figure C.5: Ucy students dataset



(a) BEV camera image of the Eth Hotel scenario.

(b) BEV camera image of the Eth University scenario.

(c) BEV camera image of the Ucy Zara scenario.

(d) BEV camera image of the Ucy Students scenario.

Figure C.6: Overhead images of all (offline) pre-training scenarios.

D

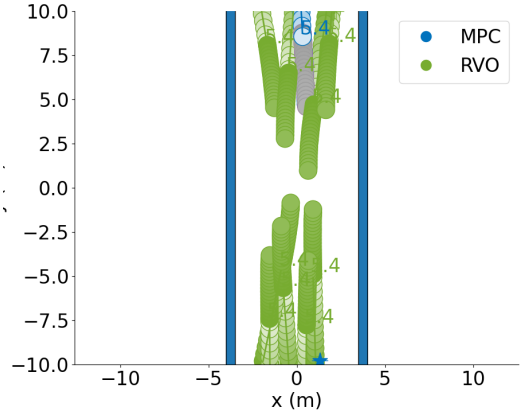
Additional Integrated Motion Planner Experiments

In this research, we investigate the possibility of improving pedestrian trajectory prediction models online. One of the main motivations is that mobile robots can take advantage of better predictions about surrounding agents, to improve their motion planning. However this assumption hasn't been thoroughly tested, and recent work suggests that anything better than a constant velocity model won't have much impact on motion planning performance. To put this to the test, we use an MPC motion planner that models the surrounding agents as dynamic obstacles. The positions of the dynamic obstacles over the prediction horizon are provided by our pedestrian prediction model. Although in our paper we've already shown the integration of the MPC algorithm with our SCRNN-L algorithms, we did not yet quantitatively evaluate the planner's performance over time.

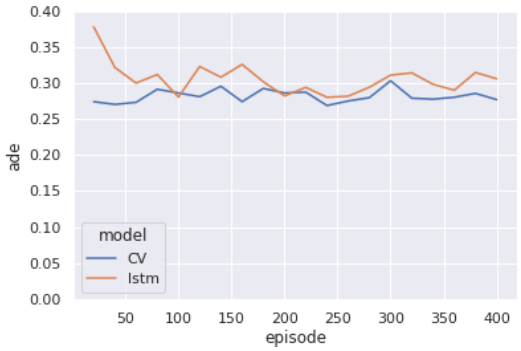
To do this we use a simulation environment adapted from the open-source gym collision environment[1]. Our test case scenario consists of a corridor of three meters wide that several agents have to cross. The number of agents present in the scene dictates the difficulty of the task. We found that using 10 agents in total provides a difficult yet feasible challenge. The behavior of the simulated agents depends on their policy, which in turn affects the prediction performance (since the policy was not provided to our prediction model beforehand). The initial prediction model has an architecture as in Fig. C.1 and was trained on the ETH and UCY datasets only. We tested with two different policies for other agents, first we used an RVO policy, which is a reactive policy based on velocity obstacles. Secondly, we used an MPC for every agent in the scene, which in contrast to RVO is a predictive policy. To quantify the motion planning performance we used the following metrics: Number of collisions, number of deadlocks, time to goal, total traveled distance. To link this to prediction model performance we used: Average displacement error, Final displacement error, Average squared displacement error, Final squared displacement error. We included the squared displacement error variants because large prediction errors should matter more than small prediction errors.

[1] https://github.com/mit-acl/gym-collision-avoidance

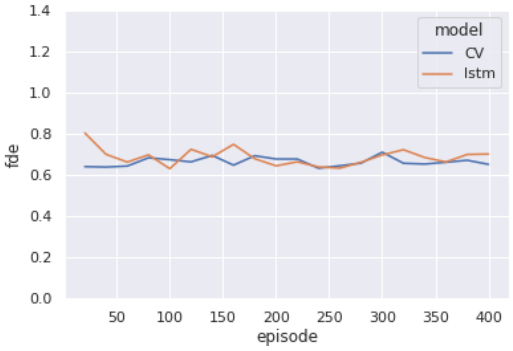
We ran the corridor scenario, see Fig. D.1(a), for 400 episodes, retraining the prediction model every 20 episodes. In Fig. D.1(b) and Fig. D.1(c) we show the ADE and FDE when using the multiagent MPC policy. Somewhat surprisingly the ADE and FDE show the same or worse performance compared to CV. In Fig. D.1(d) and Fig. D.1(e) we show the ASDE and FSDE of the same run. Using the squared displacement error, our prediction model consistently outperforms CV. This means our prediction model on average has a larger displacement error, but the peak errors (that could lead to collisions) are lower. However this theory doesn't translate to the actual number of collisions, see Fig. D.1(f). We think this could be because the collisions are caused by other factors not directly related to the predictions. Further research in this direction is necessary to improve the integration of motion planning and prediction.



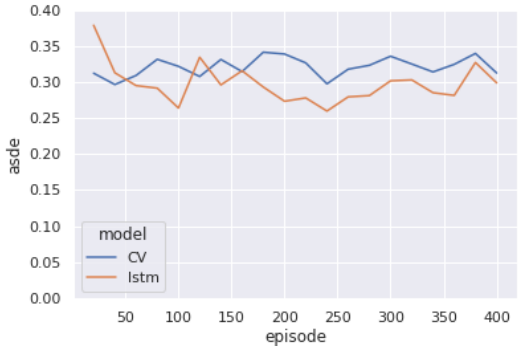
(a) Gym simulator corridor scenario



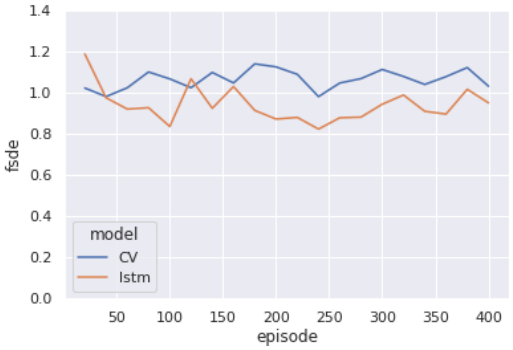
(b) Average displacement error compared to constant velocity



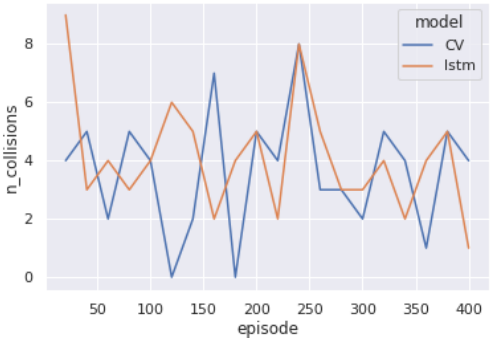
(c) Final displacement error compared to constant velocity



(d) Average squared displacement error compared to constant velocity



(e) Final squared displacement error compared to constant velocity



(f) Number of collisions

Figure D.1: Motion planning integration

Bibliography

[1] Mike Daily et al. “Self-driving cars”. In: *Computer* 50.12 (2017), pp. 18–23.

[2] Daniel J Fagnant and Kara Kockelman. “Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations”. In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 167–181.

[3] Amir Rasouli and John K Tsotsos. “Autonomous vehicles that interact with pedestrians: A survey of theory and practice”. In: *IEEE transactions on intelligent transportation systems* 21.3 (2019), pp. 900–918.

[4] Matt O’Brien. *As robots take over warehousing, workers pushed to adapt*. URL: `https://apnews.com/article/056b44f5bfff11208847aa9768f10757` (visited on 04/10/2021).

[5] Thibault Kruse et al. “Human-aware robot navigation: A survey”. In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1726–1743.

[6] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. “Planning and decision-making for autonomous vehicles”. In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018).

[7] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[9] Andrey Rudenko et al. “Human motion trajectory prediction: A survey”. In: *arXiv preprint arXiv:1905.06113* (2019).

[10] Dirk Helbing and Peter Molnar. “Social force model for pedestrian dynamics”. In: *Physical review E* 51.5 (1995), p. 4282.

[11] Jur Van den Berg, Ming Lin, and Dinesh Manocha. “Reciprocal velocity obstacles for real-time multi-agent navigation”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 1928–1935.

[12] Alexandre Alahi et al. “Social lstm: Human trajectory prediction in crowded spaces”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 961–971.

[13] Mark Pfeiffer et al. “A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[14] Amir Sadeghian et al. “Sophie: An attentive gan for predicting paths compliant to social and physical constraints”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1349–1358.

[15] Tim Salzmann et al. “Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data”. In: *arXiv preprint arXiv:2001.03093* (2020).

[16] Ewoud AI Pool, Julian FP Kooij, and Dariu M Gavrila. “Context-based cyclist path prediction using Recurrent Neural Networks”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 824–830.

[17] Agrim Gupta et al. “Social gan: Socially acceptable trajectories with generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2255–2264.

[18] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. “Social ways: Learning multi-modal distributions of pedestrian trajectories with gans”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[19] Bruno Brito et al. “Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians”. In: *arXiv preprint arXiv:2010.09056* (2020).

[20] Stefano Pellegrini et al. “You’ll never walk alone: Modeling social behavior for multi-target tracking”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE. 2009, pp. 261–268.

[21] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. “Crowds by example”. In: *Computer graphics forum*. Vol. 26. 3. Wiley Online Library. 2007, pp. 655–664.

[22] German I Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural Networks* (2019).

[23] Timothée Lesort et al. “Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges”. In: *Information Fusion* 58 (2020), pp. 52–68.

[24] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. “Task-free continual learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11254–11263.

[25] Hanul Shin et al. “Continual learning with deep generative replay”. In: *Advances in neural information processing systems*. 2017, pp. 2990–2999.

[26] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. “Brain-inspired replay for continual learning with artificial neural networks”. In: *Nature communications* 11.1 (2020), pp. 1–14.

[27] Xilai Li et al. “Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3925–3934.

[28] Jaehong Yoon et al. “Lifelong learning with dynamically expandable networks”. In: *arXiv preprint arXiv:1708.01547* (2017).

[29] Steven CY Hung et al. “Compacting, picking and growing for unforgetting continual learning”. In: *arXiv preprint arXiv:1910.06562* (2019).