Universiteit Leiden

Master Computer Science

On adaptive multi-modal trajectory prediction

Name:	Tobias Florin Oberkofler
Student ID:	s2965003
Date:	30/10/2023
Specialisation:	Data Science
1st supervisor:	Dr. Mitra Baratchi
2nd supervisor:	Dr. Javier Alonso-Mora, Prof. dr. Holger H. Hoos

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Prediction of human motions is key for a variety of direct applications, such as developing safe and intelligent autonomous systems and derivative applications, such as urban planning or sports analytics. Human behaviour is inherently stochastic, and at any given time point in any given scene, multiple valid motion hypotheses might exist. It remains an open challenge to accurately predict future human motion, incorporating this uncertainty. Furthermore is human motion context and state-dependent. Hence, different motion models must exist for different scenarios, like driving a car or walking. In this thesis, we present AutoTraj, a flexible AutoML approach for short-term trajectory prediction. Contrary to previous work, we propose to use a combined neural architecture search and hyperparameter optimisation instead of designing domain-specific hand-crafted architectures. We achieve this by introducing a novel search space inspired by the recent successes of Conditional Variational Autoencoders in short-term trajectory prediction. The search space builds upon a wide range of operations for social-interaction encoding, intention encoding and generative procedures.

Our results show that our AutoTraj significantly outperforms current state-of-the-art short-term trajectory prediction methods in two out of three diverse real-world datasets. Our approach also achieves on-par results on the popular ETH/UCY benchmark, outlining its ability to generalise to new scenes within a fixed mode of movement. AutoTraj outperforms a simpler vanilla deep neural network-based AutoML approach every time, underscoring the merit of a customised search space for AutoML in the domain of short-term trajectory prediction. Our approach is easily extendable and allows researchers from any adjacent domain, even without a computer science background, to deploy powerful computational motion models for their work. We outline the potential for interdisciplinary research by developing a physics-inspired simulation-based decoder, which allows us to combine interpretable motion models with the power of per-case parameter estimation. By making our code publicly available, we hope to inspire more research into understanding human motion and facilitating safe and risk-aware autonomous systems.

Contents

AutoTraj: An adaptive multi-modal short-term trajectory prediction framework

1 Introduction

The ability to predict human movement in complex environments is crucial in developing safe and intelligent autonomous systems (AS) that directly interact with humans within a shared cyber-physical environment. Applications like self-driving cars and social service robots hold immense promises for saving and bettering the lives of millions that can only be redeemed by acquiring a deep understanding of human movement. Besides the use in AS, accurately predicting human movement is critical in a number of derivative applications such as urban planning, crowd flow management, evacuation situation analysis and sports analytics.

In this work, we focus on the challenging task of predicting trajectories from short-term historical observations with multiple moving agents. Short-term trajectory prediction is essential in domains such as transportation, robotics, and sports, as it can improve safety and efficiency in traffic, enable autonomous vehicles to avoid collisions and enhance athletes' performance.

Humans have an innate capacity to navigate social scenarios. We naturally reason and make predictions about other objects' movements, usually without ever having to think about it consciously. However, accurately predicting human behaviour is algorithmically very challenging:

(i) Movement patterns can be highly complex with many aleatory influences.

(ii) We need to find good representations for the spatial, temporal and social dimensions to be able to model the interaction effects at play.

(iii) Furthermore, we are often confronted with an epistemic lack of an agent's goals in the real world, forcing us to formulate the task as a multi-modal problem. Hence, we need a model that can make probabilistic estimates about the future and predict multiple possible future paths.

(iv) Moreover, patterns can be highly context-dependent. A model optimised for one scenario might completely fail in a different one. For example, the movements learned from a crowd scenario will only partially translate to the ones observed in a basketball game and will completely fail to apply when predicting car movements.

(v) Lastly, movements can be susceptible to influence from exogenous variables. For example, pedestrians might adapt their movements during heavy wind, but the wind might not be recorded in the recorded data. Similarly, semantic clues like the street layout heavily influence behaviour but might be unknown to us at the time of prediction.

Various methods have been proposed to address the challenges of multi-agent short-term trajectory prediction, ranging from deterministic, physical law or social rules-based models to deep generative models [32, 62, 82, 43, 16, 1, 30, 47, 3, 19, 96, 94]. Although tremendous progress has been made in the last decade, most of this progress can be attributed to combining ever-larger models with domain-specific insights. However, no model has yet been proposed to unify the various trajectory prediction problems into one framework. Such a unified framework can significantly speed up the development process of new models for different domains by introducing a shared vocabulary and operating process between them. For instance, a company developing software for autonomous vehicles has to incorporate various mechanisms to model the behaviour of car traffic, pedestrians, cyclists, etc. By treating these tasks as instances of one united problem, significant engineering effort can be saved that would have gone into hand-crafting each task individually from scratch. Moreover, can a unified framework help us uncover new insights and place data in context, as it provides us with the tools to compare data from different sources.

We propose an automated machine learning (AutoML) based approach to trajectory prediction by using building blocks from various state-of-the-art methods and combining them via an efficient macro neural architecture search. We introduce a novel, domain-inspired search space that allows us to effectively pick data representations, pooling methods and the recurrent structure of the network based on data properties. By keeping the architecture general and easy to adapt, we allow practitioners to adjust the model to their needs and reduce the search space by using their domain knowledge as inductive bias, for example, through the use of a physics-based forward module. Within this framework, we address challenges (i) to (iv) by proposing a stochastic network which can meaningfully deal with the presented uncertainty and a robust feature- and representation-search mechanism that can adapt to different physical and social settings. This work does not address the challenge of missing semantic knowledge. In line with previous work [1, 30, 73, 60, 49, 19, 94, 96] we operate under the "open world hypothesis", assuming a flat open-space environment. We furthermore present a newly curated diverse benchmark for trajectory prediction in the form of "diversity bench" to illustrate the ability of different state-of-the-art methods to provide short-term trajectory predictions in different contexts.

The main contributions of our work are thereby the following:

- We introduce an openly available, flexible AutoML framework for multi-modal trajectory prediction problems - the first of its kind.
- We propose a novel search space based on extensive analysis of prior work and investigate the effects of the tailored search space on the algorithm's performance.
- We provide interpretations for the found architectures and their respective interaction with problem domains, outlining a potential pathway to foster more interpretable deep learning models and inspire future lines of research.
- We show how our approach can be incorporated into various domains, achieving better or on-bar results with state-of-the-art methods.

In the following, we first formalise the problem of short-term trajectory prediction and introduce the required notation and background in Sections 2 and 3. Section 4 provides an overview of existing related literature divided by model complexity and considered problem dimensions. Next, a detailed outline of our methods and models is given in Section 5. A comprehensive analysis of our data and the proposed curated benchmark, as well as our experimental setup, is provided in section 6. The results and discussion are found in Section 7 and 8. Concluding, we summarise our findings and put them in perspective to current state-of-the-art methods and real-world practices in section 9. We give examples of possible practical applications and outline pathways to potential fruitful future applications.

2 Problem statement

Socio-spatio-temproal data

In contrast to most prominent applications of deep learning, which either focus on the spatial (e.g. image classification, segmentation, object detection) or the temporal dimension (e.g. time-series-forecasting, time-series-classification), trajectory prediction requires modelling dependencies on both dimensions. Furthermore, there is a social aspect to short-term trajectory prediction. Besides physical feasibility (e.g., collision avoidance), we also have to consider the social comfort of a given agent (e.g., avoiding direct eye contact) that will lead to changes in his short-term behaviour [3, 22, 32, 45]. Understanding these interaction effects and their drivers is of particular importance to facilitate safe and meaningful interactions based on our predictions.

Problem formulation

Formally we can represent the past multi-agent trajectories X of length h in the form of

$$\mathbf{X}^{t-h:t} = (\mathbf{X}^{t-h}, \mathbf{X}^{t-(h-1)}, ..., \mathbf{X}^{t})$$

Each X^t represents the state of an environment E at a given time t filled with a variable number of N actors, hence $X^t = (X_1^t, X_2^t, ..., X_n^t)$. We refer to the collection of all agent states, past and future, as a scene S. Disregarding exogenous factors such as the layout of the environment and social-visual cues such as head position, we can represent each agent's past trajectory by its coordinates $x_i^t \in R^d$ and their derivative attributes such as speed, acceleration, heading angle, or relative position between two agents. For the short-term trajectory prediction task, our goal is to predict k future values

$$\mathbf{X}_i^{t+1:t+k} = (\mathbf{X}_i^{t+1}, \mathbf{X}_i^{t+2}, ..., \mathbf{X}_i^{t+k})$$

based on h past observations $\mathbf{X}^{t-h:t}$.

Predictions can be performed jointly for all actors at once [1, 30, 49] or iteratively [74, 19, 94] for a given primary agent i, $X_i^{t+1:t+k} = (X_i^{t+1}, X_i^{t+2}, ..., X_i^{t+k})$ which is often referred to as ego-agent as predictions are performed from his respective point of view.

As it is often impossible to know the exact intent of an agent from the given observations, trajectory prediction is usually posed as a multi-modal prediction problem where instead of the deterministic future $(X_i^{t+1}, X_i^{t+2}, ..., X_i^{t+k})$ we are trying to predict the conditional distribution $p_\theta(X_i^{t+1:t+k}|\mathbf{X}^{t-h:t})$, where $X_i^{t+1:t+k}$ is one of the possible future trajectories. We can translate this formulation back into the single-modal version by ranking the predictions and prompting the most likely of the predicted futures at each time-point as output.

In line with the majority of recent work, we facilitate this prediction through a neural network with trainable parameters θ. In contrast to other work, however, we strive to optimize the architecture parameters ω and hyperparameters λ, in addition to the network parameters. For reasons of simplicity, we will refer to architectural parameters and hyperparameters jointly as $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_n$ for the rest of this work. Based on the presented data, our network not only optimises its weights towards the approximated optimum θ^* but also the underlying structure of the network itself Λ^*. This makes our work the first proposal of AutoML for multi-modal trajectory prediction.

In alignment with the existing related literature, we measure our performance by using the minimal average displacement and minimal final displacement error [1, 30, 96, 94, 63, 3, 21].

3 Background and Definitions

This section is dedicated to providing readers unfamiliar with the domain with the required vocabulary and notation to more easily follow along with the main propositions of this work. It is directed towards people with a basic understanding of prevalent computer science concepts such as deep learning but does not assume familiarity with more advanced or specialised topics such as AutoML or socio-spatio-temporal data tasks.

3.1 Trajectory data

According to the definition by Zheng [99], Trajectory data can be understood as a trace generated by a moving object in space consisting of a series of points represented by a time stamp and a spatial location. Hence, a point $p \in R^2$ is represented by $p = (x, y)$ and a trajectory would be a chronologically ordered collection of points connected to a given moving object $X = p_1, p_2, .., p_n$. For our purpose, we define trajectory data as spatio-temporal data tied to a uniquely identifiable moving agent. Particularly, in the context of human agents, there is a significant social component at play. In many scenarios, human movement is governed by a plethora of social norms and conventions [32, 65, 30]. In this scenario, it is vital to consider the social interactions within a group of agents to better understand the resulting movement patterns. It can, therefore, be useful to add relative positions of coexisting agents to the individual trajectory data.

3.2 Automated Machine Learning

Automated machine learning can broadly be understood as the task of automating deep learning pipelines [40, 81]. Deep learning has demonstrated incredible success in recent years. However, it has also become increasingly more complex, time- and resource-consuming. To address these demands and lower the entry barrier established by the cost of obtaining the required expert knowledge, the field of automated machine learning established itself with the broader goal of automating the derivation of insight from raw data and, in particular, to automate model-building. Automated machine learning combines various sub-fields such as Neural Architecture Search (NAS), Hyperparameter optimisation (HPO), or joint tasks such as Combined Algorithm Selection and Hyperparameter optimisation (CASH). An extended summary of AutoML can be found in the work of Hutter et al. [40] and Baratchi et al. [6]. A schematic of the AutoML paradigm can be seen in Figure 1.

Hyperparameter optimization: HPO [25] is a broad subfield of AutoML, focusing on tuning the meta parameters that define the methods learning process. Such parameters include, for example, the learning rate or weight regularisation. However, the HPO framework can also be applied to NAS [40]. Through the lens of HPO, we look at each architecture as a (discrete) parameter choice. NAS, thereby, may be viewed as a special case of HPO. A classical HPO approach towards NAS requires sequential evaluation of each possible parameter space and is, therefore, very costly in practice. One-shot techniques, such as Liu et al. [56] or Pham et al. [68], that perform neural architecture search on a super-network can get around this sequential performance evaluation, drastically speeding up the search process at the cost of extended memory usage.

Neural Architecture Search: NAS can be understood as the subfield of AutoML that focuses on the design of the network architecture. In most current deep learning projects, this is by far the most costly and expensive process. It requires considerable thought and manual effort by human experts to decide on the best architecture for a given task. NAS already has been shown to outperform state-of-the-art manually designed architectures in various tasks such as Image classification [103, 56], object detection [103], semantic segmentation [18], time series prediction [41, 66] or natural language processing [56, 68] among various more [23]. This is achieved by a combination of creating insightful search spaces as well as utilising efficient search methods. So far, no NAS solutions have been provided for trajectory prediction. For a more complete overview of NAS, we refer the interested reader to the works of Elksen et al. [23], White et al. [89] and Ren et al. [70].

Search Space: Loosely speaking, we can define the search space as the range of design choices of a given neural architecture. It defines the operations and connections that the network can perform. More precisely, it is the set of Operations O, such as convolutions, linear transformations or activation functions, that together with their connection to each other $\alpha_{(i,j)}$ and to the input data x define the network N_θ. A common categorisation in regards to search space is in macro- and micro-architecture. The micro- or cell-based search space is motivated by the omnipresence of repeating structures (often called cells or blocks) in state-of-the-art neural architectures [23]. The main idea is to use a limited set of handcrafted operations to form a basic unit, such as a convolutional or recurrent block, which then can be stacked to obtain a deep network. Successful early examples of such attempts include Zoph et al. [103], Liu et al. [56], and Pham et al. [68]. This approach has the advantage of drastically reducing the search space and increasing search speed. However, it does not answer the question of how the final architecture should interconnect but rather just what pieces can be used to assemble it. The question of the overall structure is addressed in the macro-search. Unrestricted macro-search, however, can be prohibitively expensive. In recent years, a trend toward a mix of both approaches can be witnessed. An example that has proven very useful, particularly in parameter-sensitive domains such as network compression, is block-based or chain-based architecture search, where a predefined macro-architecture is used, and promising operations are sampled for each layer of the network [92, 84]. For our work, we utilise a block-based approach building on the strong baseline performance of a variational auto-encoder backbone architecture, which has been consistently used in recent state-of-the-art methods. We thereby limit our search space to embedding and pooling operations inspired by the work of Kothari et al. [49] as well as various different decoder network typologies inspired by the work of Helbing and Molnar [33], Alahi et al. [1], Gupta et al. [30], Mersch et al. [60], Cheng et al. [19] and Xu et al. [94].

At first glance, it might appear beneficial to have a vast search space and, therefore, a very expressive network; however, the practical drawbacks of such a vast exploration space almost always outweigh its benefits and make such an approach infeasible in reality. Consequently, incorporating prior knowledge in the form of human priors is essential to simplifying the search space and thereby improving the chances of obtaining good results [23, 54, 66]. Nonetheless, the reduced search space often comes at the cost of preventing the discovery of entirely novel architectures. It is, therefore, a balancing act to define the search space to be expressive enough to find close to optimal solutions but small enough to be computationally feasible. Defining an effective search space, therefore, is a crucial part of Neural Architecture Search,

which requires both domains as well as machine learning knowledge. We address this issue by identifying critical differences and overlap between state-of-the-art methods through an extensive review of domain literature which inspire our operators. In particular, we restrict our search space by recognising the importance of recurrent cell structures across different baseline approaches to model the sequential nature of trajectories in the temporal domain as well as the common embedding and pooling operations which take place in the socio-spatial domain to model interaction effects between agents.

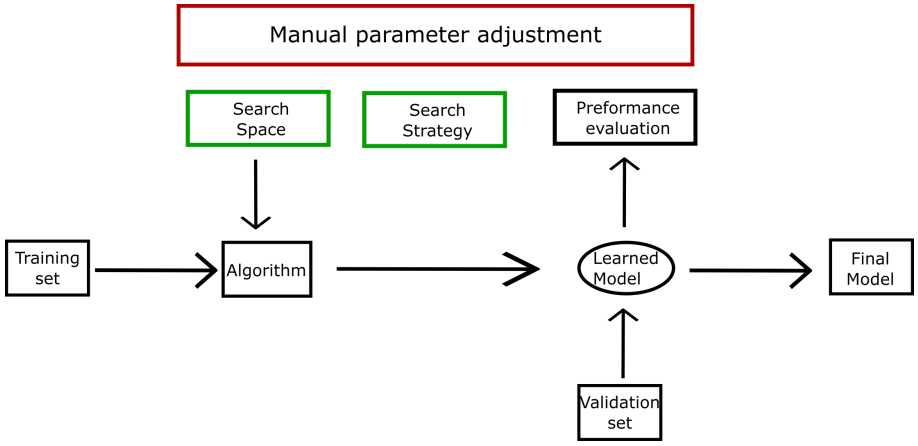

Figure 1: Schemata of the Neural Architecture search process when carried out automatically versus when performed via manual iterations. Inspired by a sketch from Song et al. [77]

.

Search Strategy: The search strategy formalises the optimisation technique, which is used to efficiently traverse the search space by selecting promising candidate solutions. In the early days, search spaces were often small enough to be explored exhaustively. However, this has become increasingly infeasible in practical modern applications with search spaces reaching 10^{20} possible configurations. In its simplest form, the search space might just be traversed randomly in a process accordingly known as "random search".

For higher-dimensional search spaces (more than five dimensions), random search often becomes too inefficient to achieve satisfactory results [6]. Bayesian optimisation (BO) or Sequential Model-Based Optimisation (SMBO) seeks to reduce the number of necessary function evaluations by incorporating acquired knowledge about the search space. The key components of BO are the surrogate model and the acquisition function. The surrogate model is used to approximate the objective function f. The acquisition function is used to determine the following configuration to be sampled, controlling the trade-off between expected performance and information gain. The most widely used surrogate models are Gaussiaun processes [91], which are most suitable for continuous search spaces. However, adaptions have been proposed to extend the method to integer and categorical hyperparameters [27]. Other popular surrogate models include random forest-based methods such as SMAC [39] or tree-based models such as TPE [10]. The acquisition function, in turn, controls the balance between the surrogates model prediction and the reaming posterior uncertainty to balance exploration and exploitation. The most popular example is Expected improvement (EI) [42], which combines the mean value and variance of the estimated objective function to perform the exploration-exploitation trade-off.

For even higher dimensional search spaces(more than 100 dimensions), the performance of BO methods tends to deteriorate [6]. Gradient-based or differentiable methods provide a promising paradigm for these hyperparameter spaces. Differentiable architecture search, such as proposed by Liu et al. [56], addresses the challenge of high dimensional hyperparameter spaces by relaxing the bi-level optimisation problem into a tandem problem where we do not optimise each architecture to the optimum but instead iteratively switch between network-weight and architecture updates. This approach increases search efficiency at the cost of increased memory requirements to store all intermediate variables. Provided the substantial memory costs this carries for today's large SOTA neural networks, this approach is not yet as widely utilised as BO.

3.3 Generative Artificial Intelligence and Stochastic networks

The origins of generative modelling can be traced back way into the 1950s with the emergence of Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) for sequential data such as speech and time series problems [17]. It has since experienced periodic revivals, such as in the 1980s and especially recently with the advent of large language models (LLMs) and large image diffusion models [14, 17]. Generative methods are usually trained self-supervised or unsupervised, making them distinctly different from the classical, well-explored, supervised machine learning paradigm. This makes them particularly powerful and flexible but also hard to train. As generative models are often used in tasks that deal with inherent uncertainty, such as predicting human utterance or predicting a person's future movements, methods have been proposed to incorporate stochastic elements into the model. This methods provide a way to deal with aleatoric uncertainty, which is inevitable in trajectory prediction. Over the past decade, especially two types of generative models have been pursued to achieve SOTA performance in the short-term trajectory prediction task: generative adversarial networks and variational autoencoders. In recent years, the focus has increasingly shifted towards variational autoencoders, which have taken the top spots in various widely adapted benchmarks [19, 94].

4 Related Work

Trajectory forecasting problems come in various forms, each demanding a unique set of solutions. Historically, a shift from theory-driven to data-driven methods can be observed. Furthermore has the multi-modal nature of trajectory prediction tasks lately received increasing attention. The following provides a short overview outlining the key avenues leading to the latest advancements in short-term trajectory prediction and AutoML techniques.

4.1 Trajectory modeling

Explicit motion modelling

The arguably oldest form of trajectory prediction is by incorporating knowledge of the physical dynamics which govern the process. Often, such models can be based on classical mechanics, assuming the laws of motion to be the central driving forces of action. Examples of such kinematic models can be found in the work of Miller et al. [62], Hillenbrand et al. [35], or Bektache et al. [9], expiring a height of research interest in the early 2000s. A more abstract spin to the rule-based modelling can be found in the social forces model by Helbing [32], which assumes that pedestrian motion can be approximated as the sum of a limited number of forces, each capturing a distinct attracting or repulsing interaction effect. Enforcing physical limitations into our model often yields more realistic patterns; however, it is very restrictive in the number of manoeuvres it can express. Recently, Helbling's method has found a revival in its fusion with data-driven methods, such as in the work of Sven Kreiss [51].

Data-driven modeling

Even before the arrival of the "Big Data Era" the focus of a significant portion of the research community can be observed shifting from rule-based toward data-driven systems. In such a framework, we do not make explicit assumptions about the governing dynamics of the interactions but instead strive to extract these effects from the data itself. For example, by exploiting the prevalent structure of street layouts and approximating the underlying probability distributions of the traffic process to create prototype trajectories. Examples of motion pattern-based prototype sampling approaches can be found in the work of Vasquez et al. [82] or Hermes et al. [34]. This prototype-based approach is often limited by its demand for very large training sets to accurately learn to represent the vast variation in existing movement patterns. Gaussian Processes, on the other hand, allow for a time-independent representation of motion patterns, which reduces the need for training examples but drastically increases the computational demands of the model. Examples of the Bayesian learning approach can be found in the work of Joseph et al. [43], Tran et al. [80], and Hu et al. [37]. When considering data-driven models, the separation between single and multi-modal is important. As the deterministic case is conceptually simpler and usually cheaper to compute, past research often focused on the single trajectory case. However, in recent years, the focus has shifted more and more towards multi-modal prediction to better incorporate the inherent uncertainty of the problem and minimise the risk of catastrophic failure. In contrast to rule-based models, data-driven models are not able to restrain physically impossible trajectories from being sampled and, therefore, sometimes might create unrealistic or infeasible predictions.

Sequence modelling (Recurrent Neural Networks)

One of the most common ways of modelling trajectories today is by representing them as a sequence. An agent motion is then predicted step by step. Traditionally, Kalman filters, Markov decision processing, or Gaussian processes have been used to that extent. Due to the recent deep learning revolution, more and more models are based on recurrent neural networks (RNN) and their derivatives, such as the Long-Short-Term Memory module (LSTM). A popular example of the use of RNNs for trajectory forecasting is the Social-LSTM [1], which uses multiple LSTM modules to jointly predict the paths of multiple pedestrians in crowded scenes. Another example is the Sr-LSTM by Zhang et al. [98], which uses a states refinement module to jointly refine the hidden states of all observed agents through iterative message passing, considering their intentions and social interactions, as well as the MX-LSTM by Hasan et al. [31], which uses a combination of tracklets and vislets exploiting head pose information of agents to improve estimations of its moving intention. Recently, take-ups of the Variational Recurrent Neural Network architecture [20] (VRNN) have been deployed for short-term trajectory prediction to better model the inherent multi-modality [16, 94].

Spatial modelling (Convolutional Neural Networks)

Next to the sequential representation, it is common to model movement patterns conditioned on their spatial location. Suggestions in traditional manoeuvre-based motion models go as far as to eliminate the temporal dependencies completely by mapping locations to velocities [43]. Mersch et al. [60] take a similar approach by using a Fully Convolutional Network to model trajectory offset and manoeuvre intentions. They, however, include not only velocities as input to their network but also acceleration and position values.

Latent modelling (Autoencoders and Transformers)

With the success of transformer models in various domains, such as natural language processing (NLP) and computer vision, researchers have begun to adapt transformer models for trajectory forecasting. These models have shown impressive performance leaves, among other things, due to their strong ability to capture long-range dependencies[88]. Classical transformer models require sequential data. To circumvent this limitation and allow joint modelling of temporal and spatial dependencies, data is often flattened into a single dimension and the transformer is equipped with costume time encoders and spatial attention modules, such as in the case of the Agentformer model by Yaun et al. [96] or costume temporal attention as in the case of Cheng et al. [19].

Autoencoders work by learning mappings of high dimensional input data to low dimensional manifolds and back to high dimensional outputs. The classical architecture consists of an encoder module, often a simple convolutional network, and a decoder network, often described as a "deconvolutional" network. This architecture type can create a latent representation of the temporal and spatial dimension and then learn to create a joint trajectory prediction. Examples include the Social-LSTM [1] and Convolutional social pooling [21]. It can be argued that such a dense representation furthermore improves the reasoning quality of the model, as it is closer aligned with the way humans tend to think about their movements, rather in low dimensional vague ideas instead of exact coordinate sequences [61].

Generative modelling (Variational Autoencoders and Generative Adversarial Network)

Trajectory forecasting is a problem of commonly high uncertainty. This is in stark contrast to the deterministic nature of classical rule-based, prototype, and neural network-based approaches, which are all deterministic in their outcome. In order to adapt to this specific challenge, recent approaches have focused increasingly on modelling not only a single trajectory but providing multi-modal predictions for any given agent, allowing accurate modelling of trajectories for different unknown intentions (and updating those beliefs based on new information). By using generative models, we can better model this uncertainty. Particularly by using (Conditional) Variational Autoencoders (CVAE), we can draw samples from a continuous latent space, which we can explore via a latent vector. Variational Autoencoders share a lot of similarities with the deterministic counterpart of a simple Autoencoder. However, they differ significantly in the decoder part of the architecture, which is used as a generative model in the Variational Autoencoder. This enables us to create various prototype trajectories from the same latent space, which might be conceptualised as representing a continuous intention/behaviour space. Many current state-of-the-art architectures such as [11, 19, 96, 94] rely on CVAEs as generative models.

Generative Adversarial Networks (GANs), on the other hand, work by using a competing generator and discriminator module where the generator draws samples from a random latent space, and the discriminator has to learn between samples produced by the generator module and real-world samples. Conceptually, one might picture this process as the discriminator supervising the generator to produce as realistic data samples as possible. Again, we can sample the latent space to achieve different outputs in the real space. This approach has also been used in various state-of-the-art models, including [30, 47, 3, 73].

4.2 Automated Machine Learning

Automated Machine Learning can be traced as far back as 1976 and the work of Rice on Algorithm selection [71]. The field, since then, has made incremental progress throughout the years. Besides its focus on the algorithm selection problem and hyperparameter optimisation, the challenge of Neural Architecture Search, the search for optimal network topology, has gained increasing importance within the field in recent years. Although Neural Architecture Search ideas can be traced back to as early as 2002 [78], it was not until very recently that the field took off, exemplified by successes in the fields of image processing and natural language processing. The pioneering work of Zoph and Le [102] and Baker et al. [5] has given an immense boost to the field and reinforced beliefs in the potential it holds. In NAS, a common distinction is between Macro- and Micro search spaces, based on whether we optimise macro hyperparameters of the entire network or the singular cell design, forming part of a larger network. Micro search space-based methods have the advantage of allowing to find very fine-grained and detailed novel cell structures but come at the drawback of being restricted to a singular repeated cell design, while macro search spaces allow for a diversity of cell architectures that can differ at each level of the network. Given that most current SOTA work in trajectory predictions relies on an array of different and non-repeating building blocks for encoding and generating trajectories, our work focuses on Macro search spaces such as found in the work of Tan et al. [79] and Zimmer et al. [101] to facilitate this qualification.

Automated Machine Learning for socio-spatio-temporal data

Li et al. [54] and Pan et al. [66] propose a domain-guided neural architecture search for a convolutional and a graph-convolutional network for the traffic forecasting task. They show that a restricted and domain-adapted search space can greatly improve the efficiency of the neural architecture search and lead to better prediction results than various baseline models. Furthermore, various approaches for neural architecture search on video data have natural parallels to the type of spatio-temoral data investigated in our work [83, 100, 87]. Critically, however, video data usually relies on the presence of dense grid representations, whereas in trajectory prediction, we are usually dealing with very sparse grids, which are often better represented using graph-like structures rather than grid-based approaches.

To the best of our knowledge, no work has yet been carried out in the direction of neural architecture search for multi-modal trajectory prediction. A neural-architecture-search-based approach has the advantage of being adaptive to the underlying problem, allowing a generalised framework for problems that would otherwise require considerable individual network design efforts.

4.3 Summary and evaluation of related literature

There can be learned a lot from existing rule-based and data-driven methods. Each approach has its strengths, such as the effective interaction pooling mechanism of Kothari et al. [49] to capture pedestrian interaction in narrow spaces. The flexible time-wise latent variables introduced in the work of Xu et al. [94] to facilitated fast-evolving latent intentions in dynamic environments, or the restrictive model of Helbing et al. [33] which offers great interpretability and is well-suited for applications in crowd flow management. However, all these methods struggle when brought into new contexts where the adequacy of this mechanism needs to be assessed and often restructured. Based on the existing body of work, we can derive an evident need for a new, adaptable and robust method for multi-modal short-term trajectory prediction. Although not without merit, current SOTA methods for trajectory prediction require intensive domain-specific architecture structuring, which is tedious and costly. Old rule-based and manoeuvre-based methods are very robust and interpretable but unflexible and often subpar performant due to the inherent complexity of trajectory data, which can not well be represented in a number of rules without them growing exponentially large with the number of movement options and interacting agents. Current state-of-the-art deep learning methods lack the direct physical interpretability of rule-based methods and the data insights gained from manoeuvre-based methods. Within our framework, we address this by proposing a combination of deep learning encoder methods with interpretable physical simulation methods. We, moreover, derive valuable insights into training data by using statistical methods to analyse the results of architecture searches across different datasets. This provides a new paradigm of data-guided methods in which our learning process is not only driven by data but also enables us to uncover new insights from data through its interaction with the search space. This process can steer further refinements and again lead to the discovery of new data and network properties.

5 Methodology

Our AutoTraj approach builds upon four main building blocks: (i) a strong backbone architecture, introduced in Section 5.2, (ii) data-guided feature selection presented in Section 5.3, (iii) an expressive domain-inspired search space, outlined in Section 5.4 and (iv) an efficient search strategy summarised in Section 5.5. In this following subsections, we introduce and motivate our reasoning behind the choices for each of these building blocks. We follow a "programming by optimisation" [36] approach, which makes our method extendable to almost any type of trajectory prediction problem. Given a dataset D consisting of multiple scenes S, our method approximates an optimal model to estimate the next k positions of all fully observed agents based on their last h observations. Concretely, for each scene of N agents moving in R^d we take their past observations $\mathbf{X}^{t-h:t} = X_0^{t-h:t}, X_1^{t-h:t}, ..., X_n^{t-h:t}$ where $X_i^{t-h:t} = (X_i^{t-h}, X_i^{t-(h-1)}, ..., X_i^t)$ and predict $p(X_i^{t:t+k}|\mathbf{X}^{t-h:t})$. $X_i^{t-h:t}$ thereby has to be understood not only as a representation of the trajectory coordinates but also of derivative and social features such as relative position to other agents or acceleration. Selecting the right combination of these representation states is a vital component of our method. We estimate the future positions of each agent individually and iteratively, making the process independent of the total number of agents present in a scene.

In this section, we first outline the overall design of the AutoTraj framework. We then outline the backbone macro structure of the model. Subsequently, we discuss the importance of various feature types for the trajectory prediction problem. Then, our search space is introduced, and lastly, we motivate our search strategy.

5.1 AutoTraj framework

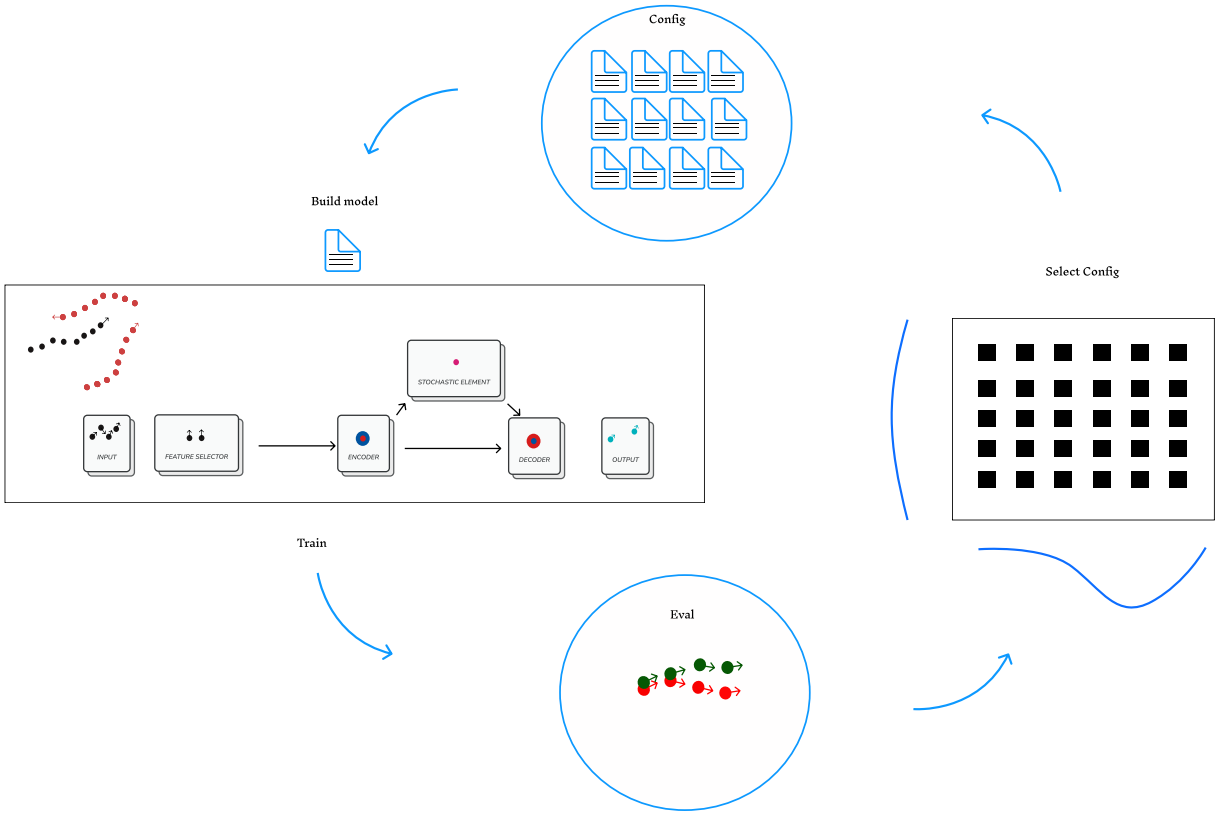

Figure 2: Outline of the AutoTraj framework. We search through a domain-inspired search space of operations to fill predefined blocks of our network design. By evaluating against the validation set, we can score each network configuration and use the obtained insight to pick the next promising configuration.

The framework consists of a unified data pre-processing module, which converts any given trajectory data consisting of at least an agent identifier, a timestamp and a position value per agent-timestamp combination. Besides the pre-processing module, a macro skeleton network architecture is provided. This skeleton defines the overall structure of the network. Based on the successes of previous work [94, 96, 19, 74], we use a conditional Variational Autoencoder as backbone architecture; however, our framework allows that choice to be easily adjusted. A search space is provided that defines the possible configuration choices of the network, such as the number of layers, the selected features, the pooling mechanism or the type of recurrent operations used. Given a training dataset D, the data is split into a train set D_T and a validation set D_V. The train set is used to optimise the trainable network parameters θ, while the validation set is used to optimise the hyperparameters λ. The possible hyperparameter choices can be summarized by $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_n$. We then aim to find the optimal parameter configuration $\lambda^* \in \arg\min_{\boldsymbol{\lambda} \in \mathbf{\Lambda}} \frac{1}{k} \Sigma_{i=1}^{k} \mathcal{L}\left(M_{\boldsymbol{\lambda}}, \mathcal{D}_T^i, \mathcal{D}_V^i\right)$ where our network M is trained on D_T and evaluated, using the ADE performance metric, on D_V. A combination of Bayesian Optimisation (BO) and hyperband (HB) [24] is used to select candidate configurations of the network architecture. After a fixed amount of time or number of runs m, the best-performing

network is chosen, and the topology parameters are fixed. Optionally, a second search through network hyperparameters such as learning rate and batch size can be performed with the architecture parameters frozen. The result is a unique per-dataset adapted model. An overview of the schematics is presented in Figure 2. A listing of all the hyperparameter choices is given in Section 10.3.

5.2 Network architecture

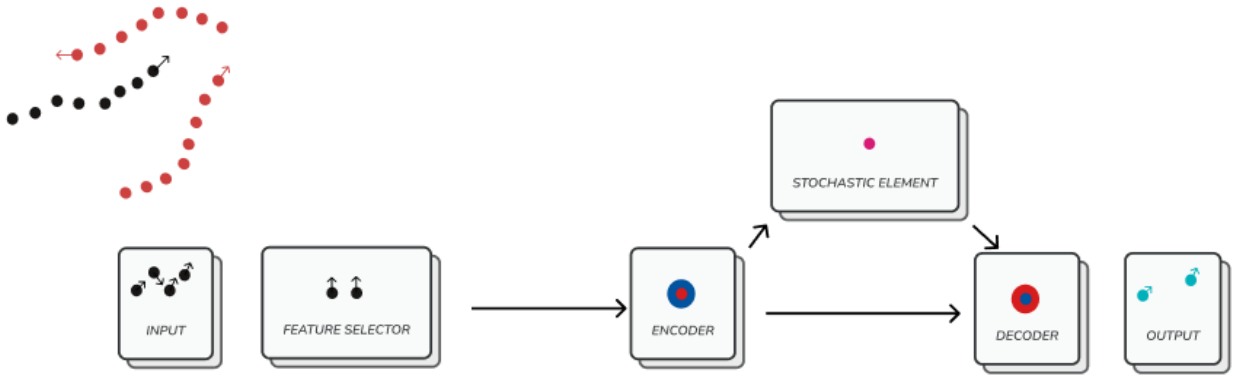

Figure 3: The overall outline of the AutoTraj macro architecture. The input data is processed by a feature selector and passed to an Encoder module. The encoder module creates a dense latent representation, which is passed in combination with a stochastic element to the decoder. The decoder generates possible future trajectories.

As sketched, we use a CVAE as our backbone architecture, building upon the positive recent results in the domain of short-term trajectory prediction [19, 96, 94, 74]. The CVAE has the decisive advantage of being able to explicitly handle the multi-modality of the problem by leveraging a latent variable z. The dimensionality of this latent variable can be adapted to the complexity of the irreducible aleatory variability in the data. The observation that the generative mechanisms of the VAE outperform the ones of a GAN may find motivation in the recent work of Bardes et al. [7] and LeCun [52] outlining relatively bad generalisation qualities and high computational demands of contrastive methods such as GANs over information maximising methods such as VAEs.

The network consists of a feature processor, an encoder, and a decoder module. Their interaction is illustrated in Figure 3.

Feature Processor: In the feature processor, various physical and social features are computed, such as an agent's velocity and relative velocity to other agents. Drawing on existing literature, we base our selection on well-established features for multi-modal short-term trajectory prediction, as found in the work of Xu et al. [94], Kothari et al. [49] and Amirian et al. [3]. An overview of them can be found in Section 10.1 of the appendix. In each network initialisation, only a subset of the possible features is selected as input to the encoder.

Encoder Module: The encoder module, in turn, consists of three components. A social interaction, an individual representation and a social representation module. The social interaction module pools all agents' information n_t and relates it to the primary agent x_t. The individual and social representation creates a dense representation of the trajectory data via recurrent operations performed on a per-agent basis. Similar to previous work [1, 30, 74, 49, 19, 94], the individual representation module is applied to the primary agent, while other agents are passed through the social representation module. As we only predict future timesteps for the primary agent, this separation aids the network in extracting the maximum amount of direct movement information from his trajectory data and the maximum amount of relative movement information from his surrounding agents. The representation states from the encoder module are concatenated at each timestep t to create a joint representation state s_t, which is passed through a recurrent cell together with the previous hidden state to create the next hidden state.

$$s_t = \phi_\theta(x_t); \phi_\theta(n_t)$$

$$h_t = \mathrm{RNN}_\theta(s_t; h_{t-1})$$

During training time, we similarly encode the information from future timepoints. The final hidden representation of the past and future data is passed through dense layers to obtain the parameters of a normal distribution μ_f, σ_f^2. In parallel, parameters μ_k, σ_k^2 are estimated using only the past information. The difference between the two distributions is calculated using the Kullback-Leibler information divergence (KL-Loss) as described in the work of Xu et al. [94]. During inference time, only the information from the past timepoints is given, and the latent variable z is drawn from the conditioned prior distribution p_k. Using the latent variable, we can facilitate a one-to-many mapping via repeated sampling.

Decoder Modules: The decoder module takes the final dense hidden representation h of the past and the drawn latent variable z as input. The concatenated information is passed through either a dense layer, a recurrent network or a simulation process based on the chosen architecture parameter. Hence the output can be either a direct nonlinear transformation of $X_i^{t+1:t+k} = \phi_\theta(h; z)$, a recurrence relation with a single latent $h_{t+1} = \mathrm{RNN}_\theta(z; h_t)$ or multiple time-wise latent variables $h_{t+1} = \mathrm{RNN}_\theta(z_t; h_t)$ where $z_t = \phi_\theta(h_{t-1})$ and $X_i^t = \phi_\theta(h_t; z_t)$. Lastly, a simulation-based approach, similar to the social force model [33], is possible where a desired destination $r_\alpha = \phi_\theta(h; z)$ is estimated from the latent information together with a desired velocity v_α and estimations for the repulsion and attraction forces $F_{\alpha B}$ of other agents B towards the primary agent α. The desired location is assumed to have an attraction force G_α and the desired velocity a modulating force M_α. The trajectory can then be estimated by iteratively calculating the next velocity values $v_{t+1} = v_t + G_\alpha + M_\alpha + \sum_B F_{\alpha B}$. All four of these approaches have their merits in different circumstances. So might, a direct decoder lead to more stable predictions, while a timewise-latent decoder might be preferred in fast-changing dynamic environments. On the other hand, there might exist situations in which more control over the model is required, which can best be achieved via the simulation-based approach. Making these different generating networks part of our search space allows our method to pick the best choice for a given context adaptively.

Depending on whether the network configuration has set position, velocity or acceleration values for the output and the metric calculation, the respective mean squared difference is taken as loss and added to the weighted KL-loss. Formally, the loss term may then be defined by $L_{CVAE} = \sum_{t=1}^{k} ||x_i^t - \tilde{x}_i^t||_2^2 + \lambda D_{KL}(p_k(z|X^{t-h:t})||p_f(z|X^{t-h:t}, X^{t+1:t+k}))$, where x_i^t is the

true and $\tilde{x}_i^t$ the estimated state of agent i at time t and p_k, p_f are the parameterised Gaussian distributions outlined in the Encoder section.

5.3 Feature selection

To be able to model the variety of complex social interactions and physical realities of short-term trajectory prediction, we found that it is necessary to compute a large array of representative features such as acceleration values, heading angles, relative distances, minimum predicted relative distances, etc. The relevance of those features might vary per dataset. To address this challenge, we leave the decision for or against a given feature group open in our search space and use the search procedure to find the optimal input as well as target representation. As the number of observable actors within a scene can change over time, a number of difficulties are introduced for various machine-learning methods. To combat this challenge, we implement the common approach to focus only on the k-closest agents or only consider agents within a fixed radius r and apply pooling or padding strategies accordingly to ensure all vectors have the same dimension [19, 49, 96]. A graphical intuition of movement features such as velocity, relative velocity and relative distance is given in Figure 4.

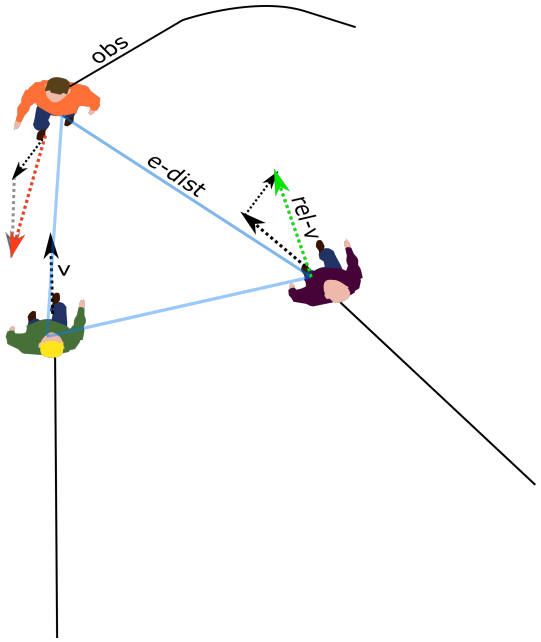

Figure 4: An overview of some of the possible features that can be extracted from scene context. The past movements and multi-agent relations are used to inform the future movement prediction. The black arrows represent the agent's velocity, and the coloured arrows represent the relative speed towards the agent opposing him. The blue line indicates the Euclidean distance between the agents.

Besides input values, the target value representation can greatly impact the learning process as the optimisation against velocity or position values leads to largely varying loss landscapes. Although we expect velocities to be the preferred mode of optimisation as they are independent of an agent's initial position, heavily reducing the amount of required training data to model the movement process, there are some scenarios in which it might be an advantage to use

positional or acceleration information as state representation. This might, for example, be true for an aeroplane performing a landing manoeuvre or a basketball player trying to score on a standard court, where knowledge of the initial location is critical in assessing the likely future positions. By leaving the framework to pick the features per dataset, we avoid making assumptions about the data and instead let the data guide the process towards an optimal bias.

5.4 Search space

Our AutoTraj framework introduces a novel domain-inspired search space that builds upon a long history of research into short-term trajectory prediction [1, 30, 73, 3, 74, 49, 19, 94]. Attempting to summarise previous literature from various contexts we find that despite big differences in implementation, they overwhelmingly draw from the same set of ideas. They propose task-specific features, social interaction mechanisms mostly based around attention and pooling, information encoders to create a dense representation of the trajectory data and generative mechanisms to create a probability distribution for future movements. In the face of these similarities, the detailed implementation between the different models, however, deviates greatly. Moreover, there does not seem to be a consistent best operation over the different contexts. We address this challenge by keeping our base design flexible and using the possible reasonable choices as building blocks of our search space. Concretely, our search space encompasses the following architectural choices: choice of input features, target and metric representation, the depth, width, activation function and dropout value of dense layers, the choice between a gated recurrent unit (GRU) and long short term memory (LSTM) for the recurrent operations, the preference to encode the Time Series forward or in reverse, the pooling operations and its settings and the decoder type. In addition, hyperparameters such as learning rate, batch size, missing value imputation method, KL loss weighting, etc, are included in the search space, and optimised jointly with the architectural parameters. A full listing of the search space can be found in the appendix. These design choices find their motivation in related literature and practical reasoning. For example, the choice between a time-wise latent or a single latent value can be understood as a trade-off between consistency and dynamic abilities. In the case of predicting car coordinates on a highway, it can be more beneficial to use a single latent variable representing basic manoeuvres such as accelerating, deaccelerating, left-turn and right-turn; While to predict the movements during a basketball game, it will likely be more beneficial to use time-wise latent variables to capture the fast-evolving dynamics and intends of the game. Similarly, some contexts might require interpretable models excluding all but simulation scenarios as valid decoders. Kothari et al. [49] have carried out large-scale experiments on determining good social pooling mechanisms and found that simple concatenation was most effective in dense crowd scenarios; however, in different contexts, different pooling mechanisms performed better [60, 96, 94, 19]. In a similar line of argumentation, Xu et al. [94] found that using reverse time-coded trajectory data works best with their recurrent network, while most other work uses simple forward time-coded input. Similar differences can be observed for the type of recurrent operator, such as whether to use an LSTM or GRU. By making our search space span these differences, we hypothesise that our framework can better adapt to new contexts than current stationary state-of-the-art models.

It is also important to note that although we strive for the search space to be as expressive as possible, we also want the resulting model to be robust and function on all data within a certain context distribution. Hence, we want to avoid "optimising" the model so far to a given

training set that it no longer generalises to other data from the same context. For example, a model trained on pedestrian data from one street should also work with pedestrian data from a different street. For that, it is important to use human prior in designing a restrictive search space and validate that no configuration can perfectly map all training data for a reasonable-sized training set. We validate the robustness of our model by running it on the ETH/UCY benchmark, which follows a leave-one-out approach for training and testing on different scenes.

To gain insights into the impact of the search space on the performance, we compare our base approach to a

(i) Basic AutoML approach, where we simply search through the width and depth of fully connected layers that transform the input features into the target predictions.

(ii) We furthermore introduce a minimalistic setting, which we shall refer to as Minimal Auo-Traj, of our full search space, which also builds upon the conditional variational autoencoder as backbone architecture; however, it does not contain social-pooling or decoding mechanisms. Hence, the minimalistic search space is limited to the input and target feature selector, embedding sizes, network width and depths and activation functions, and the same training hyperparameters, such as learning rate and batch size, as used for the full search space. All the included parameters are also listed in Section 10.3.

5.5 Search Strategy

As search strategy, we utilise Bayesian optimisation in combination with hyperband. Concretely, we use the BOHB [24] implementation of the Weights and Biases (wandb) python library [12]. We choose BOHB for its efficiency, enabling us to search through the vast parameter search space, allocating computational resources judiciously, and ensuring a more rapid convergence. It provides a good balance between observed any-time performance as well as final performance. Particularly through the inclusion of the feature selection, we found that an aggressive early stopping mechanism such as successive halving is necessary to not overload the framework with unpromising runs taking up a large amount of computation while not leading to much information gain in the regions around the optimal result.

It is worth pointing out that among close inspection of the source code, we found some notable differences to the original proposal of Falkner et al. [24]. In this implementation, a Gaussian Process is used as the surrogate model for BO. In particular, we find that the Scikit-Learn implementation for Gaussian processes is used[1], which builds upon the work of Rasmussen et al. [69]. Runtime is determined via an online version of the successive halving method. There is no initial space budget for a run other than the maximum run time b_{max}. However, runs are checked every b^{η}_{min} epochs, with η being a hyperparameter controlling the aggressiveness of the pruning. If a run falls beyond the $1/\eta$ fraction, it is stopped.

[1] https://scikit-learn.org/stable/modules/gaussian_process.html

6 Experiments

We carry out two sets of experiments:

(i) To demonstrate the robustness of our approach against overfitting of the architecture, we illustrate results on the widely used ETH/UCY benchmark. Using a leave-one-out strategy to train on four scenarios and make predictions for a fifth one.

(ii) To illustrate the adaptiveness of trajectory prediction models in different scenarios, we perform tests on three datasets of the curated diversity benchmark.

We investigate the case-by-case prediction quality of the methods for each scenario and rank the methods' overall ability to adapt to the different scenarios. We thereby focus on the following questions:

- Can AutoTraj achieve lower prediction errors compared to baseline methods?
- How well can these models generalize across different scenarios?
- What conclusions can we derive on the importance of different parameters of the neural network architecture?

6.1 ETH/UCY benchmark

To directly compare related literature, we use the well-established [1, 30, 73, 3, 74, 94, 16, 96, 95, 93, 85, 61], public ETH/UCY benchmark [53, 67]. The benchmark is widely used with a standard protocol for which dozens of papers have reported results throughout the years, permitting large-scale comparisons. We obtained the dataset directly from the official GitHub repository of the work of Mohamed et al. [64][2]. A general risk of optimising a network architecture based on a subset of data is to overfit and no longer generalize to other data, even when the data stems from the same context. As the ETH/UCY dataset consists of five individual sets, we can test the generalizability of our produced models by optimising on four sets and testing the performance on the fifth one, using the standard leave-one-out cross-validation approach.

The ETH/UCY contains data from $1,536$ pedestrians recorded at four locations and five different time points, with a frame rate of 2.5 Hertz. A more detailed exploration of the datasets' properties can, for example, be found in the work of Becker et al. [8] and Amirian et al. [3].

We forecast trajectories of 12 timesteps (4.8s) based on observations from 8 timesteps (3.2s). In line with previous work, we do not use any semantic information to aid predictions.

6.2 Diversity benchmark

To test the ability of the different methods to adapt to different contexts, we curated an expressive and diverse real-life benchmark, which we refer to as *diversity benchmark*. The benchmark consists of data from three datasets: the HighD [50], InD [13] and the SportsVU

[2] https://github.com/abduallahmohamed/Social-Implicit/tree/main/datasets

dataset [48]. Each dataset aims to represent a different real-world scenario, each with varying properties (such as linearity, speed, goal positions, interaction effects, etc.) of the underlying trajectories. The scenarios span from basketball gameplay to car and mixed traffic data. To enable as many researchers as possible to benchmark their performance with this dataset, we randomly selected $2,500$ representative trajectories per dataset to limit the computational burden while training to facilitate a 5-fold cross-validation within feasible time constraints. Typical methods can be trained and benchmarked on the dataset within less than a week of GPU runtime.

To provide a deeper understanding of the benchmark dataset, we briefly lay out the fundamental properties and background for each candidate set. A short summary is also given in table 1 and Figure 5.

Datasets

HighD: With the rise of autonomous driving applications, traffic datasets have become an increasingly important field of research for the trajectory prediction community. The HighD [50] provides a high-quality, large-scale naturalistic vehicle trajectory dataset with trajectories from over $110,000$ drivers (including cars and trucks) taken on six different locations of the German highway. Observations were taken by drones flying stable over a fixed piece of the highway for a total of 16.5 hours. The original data was recorded at 25 frames per second. We down-sample the data by a factor of 10 to 2.5 Hz to be more comparable to the ETH/UCY benchmark. Visual tracking in combination with Rauch-Tung-Striebel (RTS) Smoothing was applied to obtain the coordinate data.

InD: The InD [13] dataset provides drone-captured trajectories for four German intersections from $11,500$ traffic participants (including pedestrians, cyclists, cars, trucks and buses). The original data was recorded at 25 frames per second. We again down-sample the data by a factor of 10 to 2.5 Hz. Visual tracking in combination with Bayesian Smoothing and a constant acceleration model was applied to obtain refined coordinate data.

SportVU - NBA: SportVU is a commercial provider of motion tracking in professional sports. In the 2015-2016 NBA season, the complete tracking information of each game of the year was provided online for public access. Recently an active field of research around the dataset has been established, including studies into various trajectory prediction tasks [94, 2, 57, 97]. It provides data for over 500 basketball games with two teams of 5 players and the ball. For our use case, we limit the focus to interesting gameplay moments in the form of scoring events similar to related work [2, 94]. Details on the data extraction process can be found in the appendix 10.2.

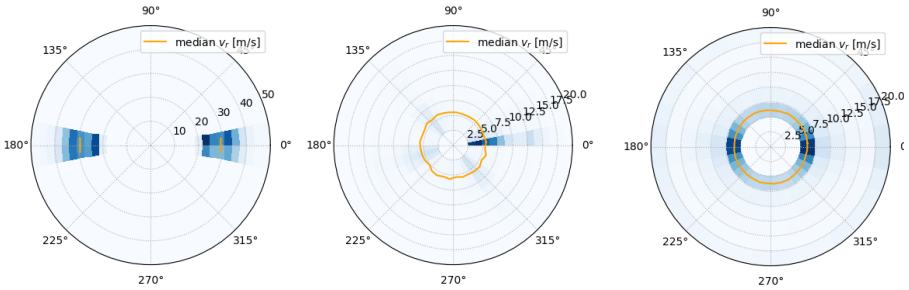

Figure 5: Illustration of the speed and directional distribution of the datasets. From left to right: HighD, InD, SportVU-NBA. The yellow line indicates the median speed per heading angle, and the underlying blue colouring indicates the density (dividing the angle into 64 and the speeds into 16 bins). Note how the car-centric data consists mostly out of fast linear movement, while the mixed data is very centred around movement within the field of view of a given agent, and the basketball data shows a lot more range, almost to the extent of seemingly erratic movement.

Dataset	Properties	Source	Applications
InD	non-homogenous intersection data, mostly non-linear	Bock et al. [13]	[13, 19, 29]
HighD	highway car data, mostly linear	Krajewski et al. [50]	[50, 60, 86]
SportVU - NBA	basketball game data, highly non-linear	Linou et al. [48]	[94, 2, 57, 97]

Table 1: A short summary of critical data properties.

Protocol We follow a 5-fold cross-validation procedure to validate our results for each dataset. This means we split each dataset into 5 equal parts. For each run, we use 4 non-overlapping parts, hence 80% of the data, for training and the remaining 20% for testing. Each trajectory consists of 20 frames, of which 8 are past observations and 12 future observations, which we attempt to predict.

Each fold from each dataset constitutes a separate experiment. We run our AutoML approach from scratch for each experiment and perform a separate search to avoid data leakage. Due to the time-intensive nature of these runs, we are only able to test with one random seeding per fold and dataset.

6.3 Data properties

Having a solid understanding of the data is critical in understanding the trajectory prediction task and its difficulties, and it is worth looking into the unique properties of the trajectory data in more detail. The attributes outlined in the following section are common to all trajectory data; however, they are here, mainly analysed in the context of our datasets. They all represent different sources of real-world difficulties of the trajectory prediction task. Our selected datasets are also composed of the ambition to be a fair and diverse representation of those attributes.

Physics of movement: Although as Mangalam et al. [58] eloquently stated: "humans are not inanimate Newtonian entities, slave to predetermined physical laws & forces", it is evident that humans are, at least on a macro scale, also no quantum particles, tunnelling between places. Large parts of naturalistic trajectories do in fact follow simple mechanics. We

can use these simple mechanics to generate hypotheses about important aspects of human motion modelling. A critical variable is the field-of-view (FOV) of an agent, which (in the absence of auditory stimuli) restricts the range of its reactive space. Following, most of the social interaction effects occur when neighbouring agents appear within the cone of the FOV of the agent. Apart from interactions, human motion tends to be mostly linear. This can be explained by the body mechanics that make it far less convenient for humans to walk backwards or sidewards than straight. Without interference, most pedestrian and traffic trajectories will follow a more or less linear path. A clear exception to this can be found in the basketball data, where it is crucial for the players of each team to "surprise" their opponent with fast, unexpected movement to get past them or anticipate the opponents' movement to hinder them from passing. This leads to a much broader distribution of heading angles and speeds in the sports data than the commonly analyzed pedestrian datasets.

Sources of randomness: The provided data tries to capture the socio-spatio reasoning of human agents in different environments. We dissect this short-term reasoning into three main drivers: spatial attention, social attention, and goal-directed attention. The first incorporates the physical possibilities and limitations of a given environment. The second includes all social cues given from one agent to another reflected in the agent's interpretation of the clues. Lastly, goal-directed attention drives the agent's more long-term behaviour. It is important to note that of those three drivers, we can only accurately reconstruct the spatial and social attention given an agent's trajectory without further information. Mangalam et al. [58] categorize these sources of randomness into epistemic uncertainty (e.g., uncertainty caused by hidden variables like long-term goals) and aleatoric variability (e.g., random decision variables such as environmental factors). Andel et al. [4] provide a methodological evaluation of the impact of missing goal information on the example of the Stanford drone dataset. Hence, no prediction can be expected to be perfect as the true generating process can never be fully estimated from the given data. However, it should also be clear that a dataset representing cars on a highway will show far less short-term divergence than, for example, a dataset of crowded pedestrian scenes.

6.4 Metrics

We report the minimum average-displacement-error (mADE) and the minimum final-displacement-error (mFDE), which are the standard evaluation metrics for the trajectory prediction task [1, 30, 96, 94, 63, 3, 21]. In particular we use the best-of-20 ADE/FDE. The ADE defines the average aligned euclidean distance between the ground truth y and the $\hat{y}$ or in the case of multi-modal predictions the minimum of such distances[1, 30]

$$\mathrm{ADE}_K = \frac{1}{T} \min_{k=1}^{K} \sum_{t=T_{\text{obs}}+1}^{T_{\text{pred}}} \left\| \hat{\mathbf{y}}_n^{t,(k)} - \mathbf{y}_n^t \right\|^2$$

The FDE measures the Euclidean distance between the final time-step of the ground truth and prediction data.

$$\mathrm{FDE}_K = \min_{k=1}^{K} \left\| \hat{\mathbf{y}}_n^{T,(k)} - \mathbf{y}_n^T \right\|^2$$

Here and throughout, $\hat{\mathbf{y}}_n^{t,(k)}$ denotes the estimated future position of agent n at time t in the k-th sample of a multi-sample prediction and $\mathbf{y}_n^T$ is the corresponding ground truth.

Baselines and Methods

For the diversity benchmark, we compare against three popular state-of-the-art baseline methods for short-term trajectory prediction, which we outline briefly:

- Linear: simple baseline based on a continuous and steady movement assumption.
- AMENet [19]: A conditional variational auto-encoder based on a mixture of dynamic convolutional and recurrent operations. SOTA model for the Trajnet dataset. The source code can be found on github.
- SocialVAE [94]: A timewise variational autoencoder combined with social attention for multi-modal predictions. Claimed SOTA for various trajectory prediction settings, including SportVU-NBA. For our experiments, we use the SVAE version without post-processing. The source code can be found on github.
- Minimal Autoraj: To further investigate the importance of the search space design, we provide a stripped-down version of our approach. The model uses the same feature selection and backbone model as the full AutoTraj method; however, it does not include social pooling operations or decoder mechanisms and instead just includes fully connected neural networks and simple recurrent networks of variable width and depth in the search space.

We chose the AMENet and SocialVAE as they are SOTA models for at least one dataset and were already trained and tested for different scenarios in their original papers. We, therefore, expect them to perform better than methods just trained for pedestrian prediction on our diverse benchmark. We conduct 100 runs of hyperparameter optimisation for each baseline method, tuning the key parameters of a given method using Bayesian Optimisation. These include learning rate, batch size and latent dimensions. The search space is based on established reasonable intervals from previous work and the original hyperparameter settings reported in the respective papers. The exact search intervals per method are reported in section 10.3 of the appendix.

For the ETH/UCY experiments, we use the results as reported by Xu et al. [94], Lian et al. [55] or self-reported from a wide range of methods to present a detailed picture of how our method compares to different current state-of-the-art architectures. For baseline methods for which no results had been published prior, namely Kalman [44] and Social Force [33], we ran the experiments ourselves, using the same standard evaluation protocol as previous work [1, 30, 73, 3, 74, 94, 16, 96, 95, 93]. We do not consider methods such as AgentFormer[96], PECNet[59] and MemoNet[93] who depend on costly post-processing steps. For the Trajectron++[74], SGNet-ED[85] and BiTraP[95], where the original papers have known bugs in their implementation, we took over the adjusted values from the work of Xu et al. [94] who recomputed them with the fixed code provided by the respective authors.

Implementation Details

During training, we use the mean squared error as the learning criterion for the network, which enforces the minimization of the average displacement error. We utilize a Xavier initialisation strategy [28] to set the initial network weights. We use rotation and flipping augmentations

to regularize the models. Instead of creating new data samples, we randomly replace existing samples with transformed versions, keeping the amount of data constant. We deal with the variable number of neighbouring agents by only considering the 16 closest agents (based on Euclidean distance) and padding with a large value when less than 16 agents are observed. We use an Adam optimizer[46]. Hyperparameters such as learning rate, drop out, batch size, missing value fill strategy, etc., along with architectural parameters such as the dimensions of the hidden state of encoder and decoder, latent dimension or pooling method, are set per run configuration. We use the top-20 minimal average displacement error over the validation set as the criterion for the meta-optimisation on the hyperparameter level. For the diversity benchmark, we run AutoTraj for each experiment for a maximum of 500 runs or five days, followed by 100 runs of fine-tuning, optimising only hyperparameters but keeping architectural parameters frozen. In a second set of experiments for the ETH/UCY benchmark with a reduced search space, we run AutoTraj for 100 runs without fine-tuning.

The experiments were carried out on the Data Science Lab cluster of Leiden University with a 48-core machine using the Intel(R) Xeon(R) Silver 4214R CPU clocked at 2.40GHz with a total of 252GB of RAM and 2 NVIDIA GeForce-RTX 3090 with 24 GB of graphic-ram each.

7 Results

7.1 ETH/UCY benchmark

For the ETH/UCY benchmark, we find that our method performs on par or better than any non-postprocessing method in terms of ADE. Regarding FDE, several baselines outperform our approach. However, this is expected as we explicitly optimise for minimal ADE in our framework during the architecture selection process.

ADE/FDE top20	ETH	Hotel	Univ	Zara01	Zara02	Average
Linear[1]	1.07/2.28	0.31/0.61	0.52/1.16	0.42/0.95	0.32/0.72	0.53/1.14
Kalman[1]	1.20/2.41	0.26/0.48	0.75/1.44	0.61/1.19	0.46/0.90	0.66/1.28
Social Force[1]	1.32/2.28	0.65/1.16	0.90/1.63	1.38/2.53	0.82/1.50	1.01/1.82
SocialGAN	0.64/1.09	0.46/0.98	0.56/1.18	0.33/0.67	0.31/0.64	0.46/0.91
SoPhie	0.70/1.43	0.76/1.67	0.54/1.24	0.30/0.63	0.38/0.78	0.54/1.15
SocialWays	0.39/0.64	0.39/0.66	0.55/1.31	0.44/0.64	0.51/0.92	0.46/0.83
Social-STGCNN	0.64/1.11	0.49/0.8	0.56/0.98	0.44/0.79	0.34/0.53	0.49/0.84
PTP-STGCN	0.63/1.04	0.34/0.45	0.48/0.87	0.37/0.61	0.30/0.46	0.42/0.69
STAR	**0.36**/0.65	0.17/0.36	0.31/0.62	0.29/0.52	0.22/0.46	0.27/0.52
TransformerTF	0.61/1.12	0.18/0.30	0.35/0.65	0.22/0.38	0.17/0.32	0.31/0.55
MANTRA	0.48/0.88	0.17/0.33	0.37/0.81	0.22/0.38	0.17/0.32	0.28/0.54
Social-LSTM[1]	1.09/2.35	0.79/1.76	0.67/1.40	0.47/1.00	0.56/1.17	0.72/1.54
SR-LSTM-2[1]	0.63/1.25	0.37/0.74	0.51/1.10	0.41/0.90	0.32/0.70	0.45/0.94
Trajectron ++	0.54/0.94	0.16/0.28	0.28/0.55	0.21/0.42	0.16/0.32	0.27/0.5
BiTraP	0.56/0.98	0.17/0.28	0.25/0.47	0.23/0.45	0.16/0.33	0.27/0.5
SGNet-ED	0.47/0.77	0.21/0.44	0.33/0.62	**0.18**/0.32	0.15/0.28	0.27/0.49
SocialVAE	0.47/0.76	**0.14**/0.22	0.25/0.47	0.20/0.37	**0.14**/0.28	**0.24**/0.42
Basic AutoML	2.90/3.71	2.75/3.72	2.37/2.49	2.23/4.32	1.96/3.02	2.44/3.46
Minimal AutoTraj	0.64/1.20	0.20/0.37	0.29/0.58	0.38/0.76	0.20/0.41	0.34/0.66
AutoTraj	0.46/0.82	**0.14**/0.24	**0.24**/0.49	0.20/0.42	0.17/0.32	**0.24**/0.46

Table 2: Results for the ETH/UCY benchmark. As per convention, we report the best-of-20 ADE/FDE. We divide the results into five main sections: classical baselines, GAN-based methods, spatio-temporal-graph-based methods, transformer and memory-based methods, methods that are potentially included in our search space and AutoML-based methods. We observe that CVAE-based methods significantly outperform GAN-based methods. Our method performs on par or better than any baseline method in terms of ADE. Methods marked with [1] are non-probabilistic methods.

We likewise see a dominance of CVAE-based approaches over GAN-based approaches. Spatio-temporal-graph-based methods tend to outperform classical and GAN-based methods but stay behind other architecture types. Moreover, we find that the basic AutoML approach with a simple feed-forward network-based search space cannot catch on to the complex dynamics required to create accurate trajectory representations. Figure 6 provides a visual overview of the relevant methods that fall within our search space as outlined in section 5. We find that on average, over all the datasets, we perform on par with the SocialVAE model. On two datasets,

namely the Eth and Univ dataset, we performed slightly better; on two datasets, the Zara1 and hotel, we achieved the same results, and on Zara2 we performed worse. The higher error on the Zara2 dataset significantly impacts our overall result.

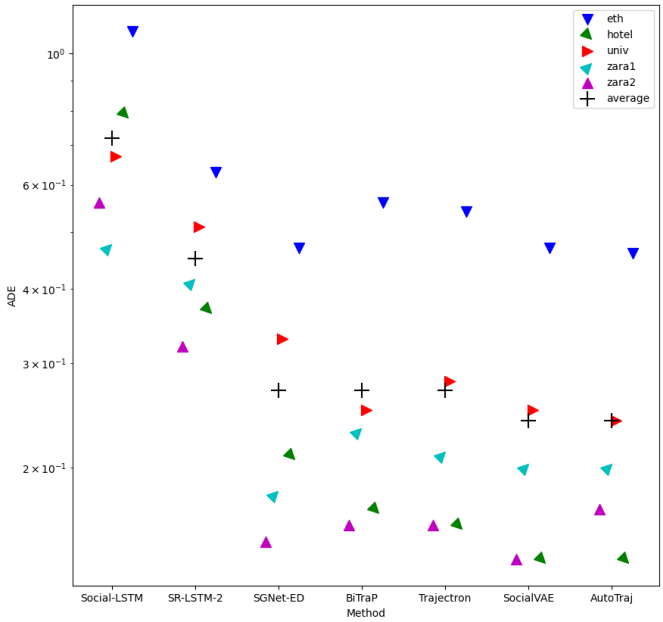

Figure 6: Minimal average displacement error for the ETH/UCY benchmark. Visualizing the most relevant methods for direct multi-modal short-term trajectory prediction. Each triangle symbol represents a dataset, and the black + the average over the datasets. On the x-axis, we see the different methods and on the y-axis, the ADE on a log-scale.

7.2 Diversity benchmark

For the diversity benchmark we trained the SocialVAE, AMENet, a basic AutoML approach in the form of Minimal AutoTraj and the full-scale AutoTraj model on the sampled version of the HighD, InD and SportVU NBA dataset. The SocialVAE is a state-of-the-art model for the ETH/UCY and SportVU NBA dataset, the AMENET is state-of-the-art for the Trajnet [72] and the InD dataset. Hence, other than most short-term trajectory prediction work, they were already explicitly trained on multiple datasets and furthermore have an overlap with the datasets used in our experiments. We, therefore, believe that they can provide a strong baseline comparison. We furthermore include the classical constant velocity model as a base measurement from a deterministic method to provide further context on the magnitude of performance achievements.

On the diversity benchmark, we find that our method can adapt well between highly linear car trajectories and non-linear basketball-player trajectories. On the HighD dataset, AutoTraj outperforms the second-best method by nearly 25% and 26% in terms of ADE and FDE. Our approach outperforms the next best baseline by 4% and 21% on the InD dataset and is outperformed by 3% and 32% on the Sport-VU NBA dataset. The AMENET performs

relatively well in mixed traffic scenarios for which it was optimised but does not generalize as well as other methods. Figure 7 shows box plots of the results with the mean signalised by the dotted green line and the median by the continuous orange line. The average score over the 5-folds per dataset and method can be found in Table 3.

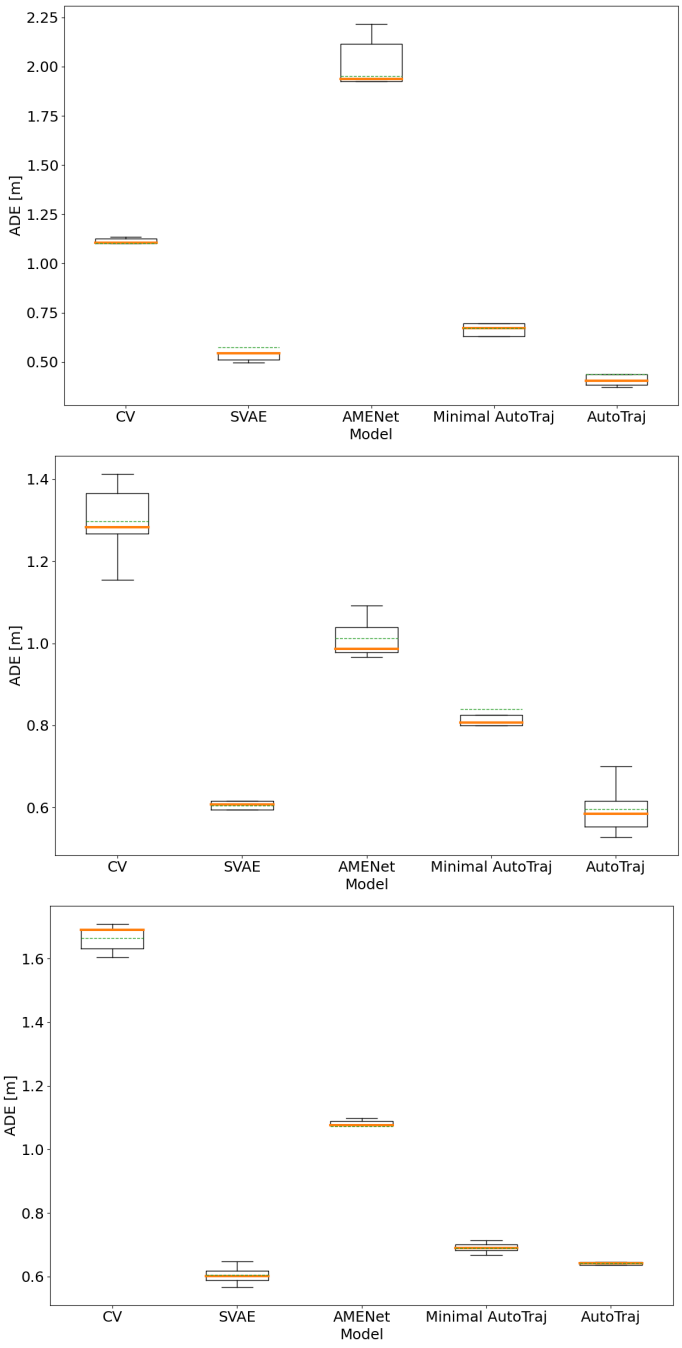

Figure 7: A boxplot for the performance on the HighD, InD and NBA dataset respectively, from left to right. The mean performance in terms of ADE is signalised by the dotted green line and the median by the continuous orange line.

Table 3: Results for HighD, InD, and NBA datasets. We report the mean best-of-20 ADE/FDE. We observe that our method outperforms all other baselines in two datasets and performs second best on the third one.

ADE/FDE top20	HighD	InD	NBA
Linear	1.10/2.67	1.28/2.46	1.48/3.35
SVAE	0.57/1.38	0.60/2.08	**0.61**/0.96
AMENET	1.98/3.68	0.98/1.87	1.08/2.29
Minimal AutoTraj	0.63/1.60	0.84/1.81	0.69/1.17
AutoTraj	**0.43**/1.02	**0.58**/1.65	0.63/1.12

As the error values from the different datasets are not on the same scale, averaging them can not give us a good estimate of the actual preference between methods. To find the best overall method, we need to perform a ranking between the candidate methods. Comparing the different methods by bootstrapping the results with 1000 samples of size three and performing a Wilcoxon signed-rank test[90] for non-normally distributed samples to award rankings based on significantly better performance, we find the average ranking outlined in Table 4. With an average rank of 1.33 AutoTraj achieves the lowest average ranking, followed by the Social Variational Autoencoder. In third place, we find the minimal version of AutoTraj with a simple search space. This is followed by the AMENet. As expected, in the last place, we find the classical rule-based approach of constant velocity.

Table 4: Average ranking across the benchmark

Model	Average Ranking
Linear	4.66
SVAE	1.67
AMENET	4.33
Minimal AutoTraj	3.00
AutoTraj	**1.33**

7.3 Parameter importance

Gaining insights into the performance of different network components can help us develop a further understanding of the data as well as guide us towards more efficient search spaces by outlining which parts of the search space underperformed on a given dataset. Such insights may then, in turn, be used to create a more restrictive arrangement that can be searched quicker or more precisely, with longer training times per configuration. Studies into the marginal performance and influence of the parameter choices can additionally aid interpretability by being able to pinpoint the strengths and weaknesses of different settings. To obtain insights into the importance of various parts of our architecture, we perform a functional ANOVA [76] employing the efficient random forest-based procedure of Hutter et al. [38]. Leveraging this framework, we obtain estimates of the marginal effect of single hyperparameters and interactions between hyperparameters by analysing the performance of different architectures on the validation set. We observe that, in particular, hyperparameters such as learning rate but also feature selection settings have a significant contribution to a model's performance. We find that to be true in terms of input feature selection as well as target feature selection. We often achieve better results when using velocity or acceleration values as prediction targets

than when using position directly. This indicates that the behavioural change can encode information more universal than the direct positional state. This can be expected as it imposes a form of normalization on the training samples, making them independent from their starting position and likely aiding a better gradient flow. This indicates the importance of utilizing the correct coordinate system even when dealing with universal function approximators such as neural networks. It should be noted that the stark differences in performance might also be a consequence of the limited training time and that with longer training runs, the network could learn better mappings of the positional information. Further investigations into the interactions between the hyperparameter choices under different computational budgets open up a promising avenue for further research.

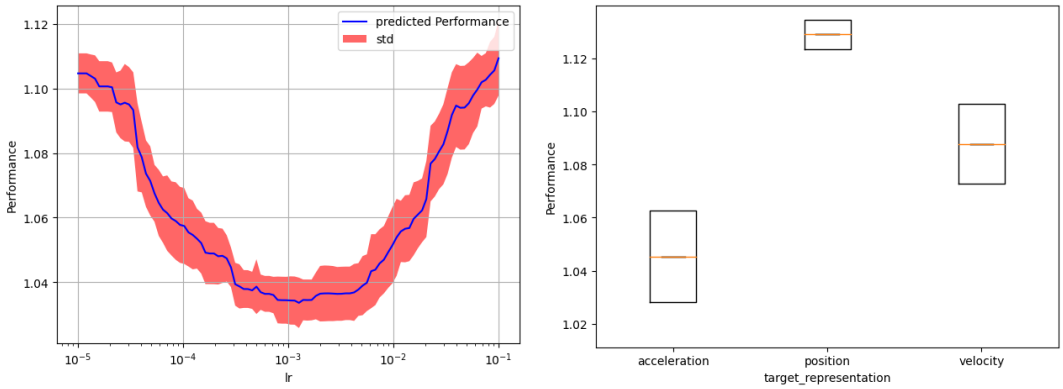

Figure 8: Marginal importance of hyperparameters on the InD dataset. On the left: Marginal performance achieved when varying the learning rate over the x-axes. The solid blue line is the mean, and the shaded red area indicates one standard deviation. On the right: Marginal performance for different target representation features. The boxplot represents their range and mean of observed results. To avoid outliers distorting the image, we resort to visualizing results from the top-50% quantile.

Besides hyperparameters, we also find functional choices such as the type of network activation and the hidden and latent dimensions of the network of central importance to a given network performance. As the performance of a network architecture is highly sensitive to the hyperparameter settings chosen for training, the joint optimisation of hyperparameters and architectural parameters is important to obtain an accurate proxy of the true optimal network performance. An illustration of the effects at play between learning rate and decoder type is shown in Figure 9. Our findings suggest that particularly for our highway data, feature options using pure position information could be pruned from the search space as velocity or acceleration information was shown to be always preferred by our feature selection. Furthermore, can the width of some of the parameter ranges regarding the dimensions of fully connected networks potentially be reduced without leading to a significant performance decrease. Further investigations are warranted to delve deeper into these findings.

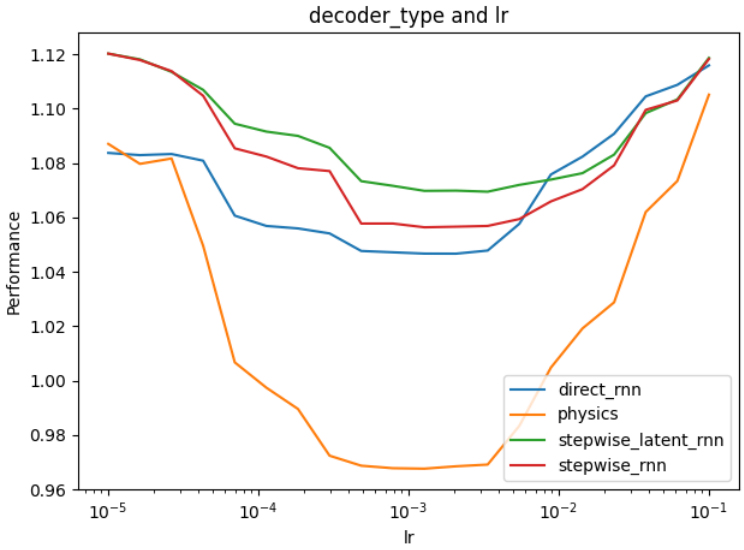

Figure 9: An illustration of the possible interactions between different parameter choices. Here, we show the pairwise marginal computed on the validation set of the InD subset. Different architecture types might have different optimal learning rates. To get a good estimation of the true optimal network performance, we need to find the right combination.

8 Discussion

We find that AutoTraj provides a convenient framework for multi-modal short-term trajectory prediction. AutoTraj easily adapts to different scenarios while still not overfitting the provided training data, as shown by the robust performance on the ETH/UCY benchmark and the best score on the diversity benchmark. Our experiments on the diversity benchmark demonstrate that the model can make accurate predictions for car-centric as well as human-centric data. This illustrates the advantage of an adaptive AutoML approach versus a static domain-specific design. Analysing the performance of different parameter settings provides us with a valuable tool for examining the impact of various design choices, reflecting the interplay between data properties and model properties. We observe that the physics-inspired decoder module is chosen in many configurations for pedestrian data. This outlines the importance of solid priors in human-centric trajectory prediction. This is especially significant when considering that the social-force model, which inspires the decoder, achieves very poor performance. We conjecture this is primarily related to three properties: i) Better goal condition. In contrast to the deterministic and linear goal prediction of the original social force model, the destination encodings extracted from our encoder provide a much stronger approximation of the actual agents' desired location. ii) Sample-specific estimation. Other than the social force model, we estimate the parameters governing the attracting and repelling forces on a per-case basis and infer them from the observed past trajectory, while the original model keeps the parameters fixed for all samples. iii) Better noise handling. Our model is better able to handle noise, as instead of using the raw past trajectory to estimate parameters like the agents' destination, we use the compressed, dense latent representation to reason. This makes the model less susceptible to single outliers in the raw trajectory data.

It should be noted that within our work, we did not compare against methods that employ post-processing techniques. Post-processing methods [59, 93, 96] have the significant drawback of needing to carry out vastly more computations at inference time, hence introducing further complexity and latency into the system. The main improvement of such methods is often achieved by producing more diverse prediction anchors and, therefore, optimizing for variability at the cost of confidence. This might reflect in better best-of-20 metrics; however, it will often lead to undesired effects in real-world deployment, for example, in planning applications, as it may increase the number of trajectories required to be considered. In general, we found that the standard metrics in the form of ADE and FDE tend to favour methods with more diverse predictions, disregarding their actual plausibility. However, as short-term trajectory prediction is inherently multi-modal, there is a difficult trade-off between diverse sampling and concise directed predictions as we want to achieve both, namely high sample efficiency and high sample diversity. Inherently, these two goals are often opposing. Low sample efficiency would lead to increased latency, while low sample diversity would lead to unsafe planning in a real-world environment. Further research into this area appears highly worthwhile.

Training a generative information maximizing method such as the conditional variational autoencoder is challenging. The model performance relies on the right balance between reconstruction loss and the Kullback–Leibler divergence term, which is notoriously hard to set in auto-regressive settings[26]. In addition, our non-normalized mean-squared error term leads to a significant imbalance between the KL and the reconstruction error term. The terms must be weighted to ensure the model optimizes for both aspects. If the weighting is off, KL-vanishing, in which the model completely ignores the latent dimensions and only optimizes for the prior, or KL-term explosion, in which the model does not optimize the KL-term at all, leading to an

unrepresentative and meaningless latent space, can appear. Various techniques, such as Monotonic scheduling [15] or cyclic annealing [26] have been proposed. For our work, we found that cyclic annealing provides a good mode for balancing KL-loss and reconstruction loss. The precise trade-off between the two terms can then be optimised via parameter optimisation within the AutoTraj framework.

Qualitatively looking at the results, we found that for the physics-inspired and direct decoder module, our method tends to produce more coherent but less varied predictions than approaches like the SVAE. We notably observed this pattern on the HighD dataset, where our AutoML approach preferred a direct decoder method over a VRNN approach. This holds less so for the stepwise-latent-based decoders, where the predictions are much more varied. We hypothesise that the direct decoding mechanism leads to more stable predictions at the cost of dynamic range. On the NBA dataset, our models' predictions are more varied, as expected by the fast and high-powered movements. However, the predictions still do not manage to cover all movement modes, making it perform slightly worse than the SVAE. For predictions from AMENet, on the other hand, we often observe a mode collapse ending up with many similar predictions covering only one movement option. Figure 10 shows examples of the predicted trajectories.

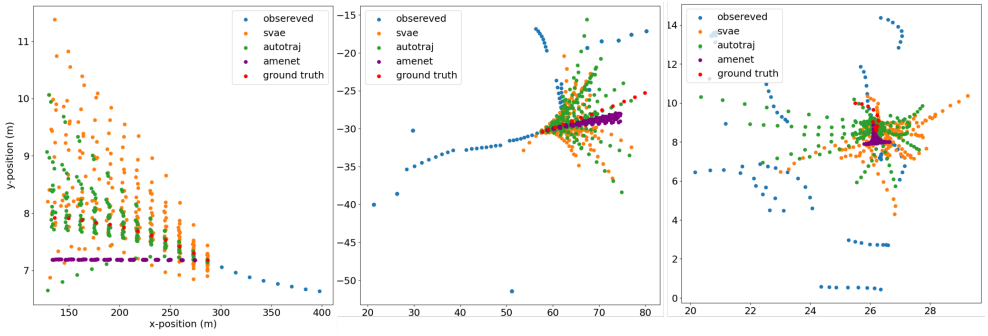

Figure 10: Example of the predictions from various models. In blue, the observed input data. In red is the ground truth. In purple, the AMENet; in orange, the SVAE; and in green, the AutoTraj predictions. Scenes from left to right taken from the HighD, InD and NBA datasets.

This qualitative observation best reflects quantitively in the score for the HighD dataset, where the AutoTraj performs significantly better by having the predictions focused on the most realistic path while other models spread around unrealistic scenarios. Conversely, in the UCY Zara scenario, where pedestrians walk around the corner of a shopping street, this likely leads to worse performance as the probability of pedestrians taking a turn is often underestimated. The inclusion of semantic information could significantly improve prediction quality. An example of the predictions overlayed on the video of the Zara scene can be seen in Figure 11. From the Figure, we can see various insightful scenarios. In the second image, we can notice an attempt at collision avoidance of our model. In the third image, we can observe an attraction force towards a near pedestrian, likely wrongly attributed to enacting a group cohesive force on the other pedestrian as one of the possible movement scenarios. Lastly, we can see the model's ability to predict the turn based on limited observation data.

Figure 11: Example of the predictions on Zara1. For all pedestrians at once, and for a clearer view for some individual examples. In blue, the observed input data. In red is the ground truth. In orange the AutoTraj predictions.

To investigate our models' ability to represent the actual underlying motion hypothesis of a given scenario, we prepared a synthetic dataset similar to the one outlined in the work of Mészáros et al. [61]. We created the synthetic data by using a shared sequence of 8 timesteps for all trajectories followed by 12 timesteps of logarithmically either moving left or right, creating a bi-module distribution. To prevent perfect memorization and mode collapse, we add slight random variations to the x and y coordinates. The resulting synthetic dataset can be seen on the left in Figure 12. In the same fashion, we also create a synthetic validation and test set with the same underlying distribution. From the results, we can observe that our method can capture the underlying distribution and truthfully reproduce the principal movement modes. Looking at the KDE of our test predictions in the rightmost part of Figure 12, we can see that, indeed, the most likely path for our predictions lies along the bi-modal distribution of the generated synthetic data. It should be noted that this movement can not have been extracted from the observed past trajectories as it is shared for both movement modes, however, it is "stored" in the learned latent distribution of our model.

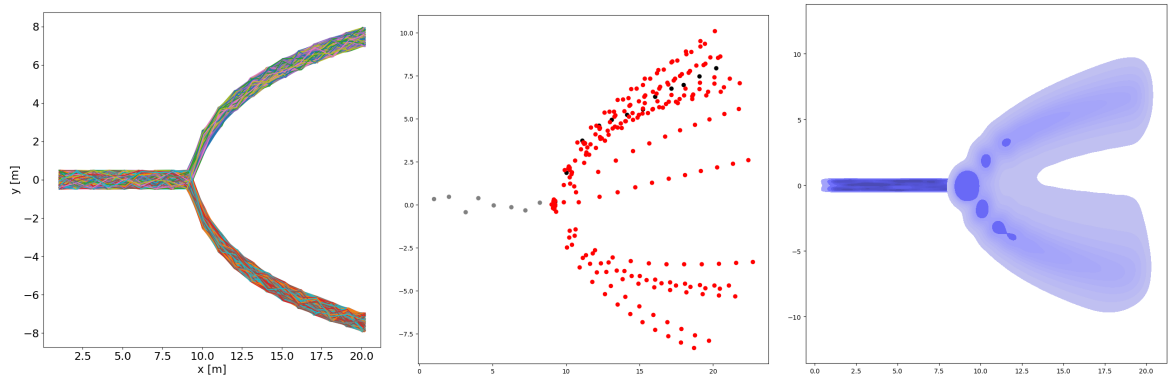

Figure 12: Left: he generated synthetic dataset. We can clearly observe the bi-modal distribution of the prediction targets that need to be learned by the model via its conditioning. Middle: an example prediction from our model on the synthetic test set. Right: The estimated Kernel density for all our test data predictions.

8.1 Limitations

As our approach currently does not consider semantic information, the prediction quality is limited in environments with many natural obstacles unknown to the model. However, this could easily be addressed by merging the current trajectory encoder with an environment encoder to create a shared representation of the agent and surrounding state. Furthermore,

the stepwise decoder modules are limited by their auto-regressive nature, resulting in a linear increase of inference time as the number of predicted time steps increases. However, this challenge is by no means unique to our approach, and previous work has demonstrated that this does not need to be detrimental to the use of the model in a real-world application [61, 94]. Lastly, although not very memory intensive, our approach is very time intensive and, therefore, still requires significant computational resources. Our experiments were, therefore, limited to relatively small datasets of about $2,500$ scenes per dataset. Large-scale experiments investigating the various models' performance given more training data pose an interesting question for future research.

9 Conclusion

In this thesis, we have investigated AutoTraj, an AutoML approach for short-term trajectory prediction. We considered the restricted case, where no semantic scene information is given, and no post-processing is applied to the predictions. Our experiments on the ETH/UCY and the diversity benchmark, consisting of subsets from the InD, HighD and Sports-VU NBA datasets, demonstrated that our approach can learn complex behaviour mechanisms from real-world datasets. We verified that it can adapt to various scenarios such as predicting car trajectories, mixed traffic data, basketball player or pedestrian movement. We found that it can adapt better to different scenarios than any baseline method, achieving the highest ranking in two of the three diversity benchmark datasets and on-par results on ETH/UCY while providing a vastly more controllable and often more interpretable model for this dataset than comparable state-of-the-art methods. Through the combination of data-guided feature selection, an adaptive variety loss and a well-conditioned prior distribution, our model can effectively pick up on modes of movement, providing a good balance between exploring a variety of possible actions and exploiting knowledge of high-probability regions. We found strong evidence that different scenarios necessitate different encoding and decoding mechanisms. In particular, we have found that car data can best be directly predicted, while the NBA data required a more dynamic time-wise latent-based approach. For pedestrian data, we found that a physics-inspired framework can achieve on-par results to state-of-the-art while allowing insights into the model's inner workings. We believe that the data-guided "optimisation by design" approach of AutoTraj will provide many practitioners from adjacent fields an accessible way to deploy powerful computational models without the necessity of a deep computer science background. By making our code publicly available, we hope to inspire more researchers to experiment with additional trajectory datasets, further extending the diversity benchmark.

9.1 Future research

In this work, we introduced a flexible approach toward short-term trajectory prediction. Our approach was focused on extracting information from agent behaviour and did not consider semantic information such as street layouts. A combination of agent trajectory and semantic scene information could allow us to model agent-space and agent-agent interaction. We believe a fusion of multiple systems, such as video and navigation data, where available, with agent information can lead to significant performance improvements. Particularly for heterogeneous data and scenes with many physical barriers considered in an agent's movement planning. Further investigation into how to include AutoML capabilities into this process lays out an especially promising pathway for future research. In our experiments, we furthermore focused on optimising the average displacement error. However, optimising other criteria, such as the final displacement error or negative log-likelihood, provides an exciting area of further study to generalise the outlined approach to different tasks, such as goal point estimation. Additionally, our experiments have so far been limited by tight computational constraints. Investigating the model's performance when given more training data as well as longer training times per instance could provide even better insights into a model's true underlying optimal predictive power. Given the complexity of our models, it is very possible that the current performance underestimates the true optimal performance. Running experiments, increasing a single model's runtime up to 500 epochs will likely lead to substantial performance improvements. To investigate the reliability of models for real-world use cases, it would be beneficial to employ

a team of human annotators to predict the future trajectory of an agent provided the same information as our algorithm. Such an experiment could provide valuable perspective into the difference between current human-level and machine-level performance. Studying trajectory prediction in entirely new contexts, such as ship or aeroplane traffic, even further removed from common pedestrian trajectory prediction, could give valuable further insights into the transferability of interaction-aware approaches to interaction-free scenarios. We hypothesise that the ability of automated feature selection, along with a joint hyperparameter and architectural optimisation, will provide even greater advantages over traditional baselines in such applications. In this context, we also believe that developing new automated feature extraction methods provides a promising track towards end-to-end learning. Towards the end of such a goal, investigations into using data properties, such as linearity, path length or prediction horizon, as initial priors or heuristics for the search space design and model selection appear to be a promising track to improve the efficiency of the AutoML framework further. Further exploration of the necessity, sufficiency, and influence of different network architecture components with extensive ablation studies could give valuable insights into the potential for network compression. Along the lines of such research, examining multi-objective optimisation for improving latency alongside the prediction error does outline an interesting path for further improving the usability of such methods in real-world systems. Lastly, subsequent research might include the integration of AutoTraj within a motion planning framework and, thereupon, into a full perception-prediction-planning pipeline.

9.2 Braoder applicability

Our work provides a first study of AutoML for multi-modal short-term trajectory prediction, introducing a unified framework which can be adapted to different contexts. In this thesis, we have shown the potential for such an approach on various real-world trajectory prediction datasets. For it to be employed in practice, further research is needed to integrate it with common motion planning and perception systems. Identifying the models' ability to handle measurement errors and other imperfections introduced by the perception pipeline will be of crucial importance for its application in practice.

A well-integrated prediction module can help facilitate critical applications such as autonomous driving, potentially helping to save numerous lives. Another promising area of application lies in social service robots. In the near future, these robots could assist with tasks such as food delivery and elderly care, reducing operational costs and improving living standards worldwide. Lastly, we hope that our work may also be a stepping stone for the further development of autonomous agents. Such agents may recognise by themselves when their current motion model fails in a given environment or for a given entity and adapt accordingly by learning a new motion model from the observation of the given context or entity. Such an online learning approach could make agents more resilient and reliable.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[2] M. A. Alcorn and A. Nguyen. baller2vec: A multi-entity transformer for multi-agent spatiotemporal modeling. *arXiv preprint arXiv:2102.03291*, 2021.

[3] J. Amirian, J.-B. Hayet, and J. 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*, pages 0–0, 2019.

[4] J. J. Andle, N. Soucy, S. Socolow, and S. Y. Sekeh. The stanford drone dataset is more complex than we think: An analysis of key characteristics. *IEEE Transactions on Intelligent Vehicles*, 2022.

[5] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[6] M. Baratchi. Automated machine learning: Past, present and future, under review.

[7] A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[8] S. Becker, R. Hug, W. Hübner, and M. Arens. An evaluation of trajectory prediction approaches and notes on the trajnet benchmark. *arXiv preprint arXiv:1805.07663*, 2018.

[9] D. Bektache, C. Tolba, and N. Ghoualmi-Zine. Forecasting approach in vanet based on vehicle kinematics for road safety. *International Journal of Vehicle Safety*, 7(2):147–167, 2014.

[10] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[11] A. Bhattacharyya, B. Schiele, and M. Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8485–8493, 2018.

[12] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[13] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934. IEEE, 2020.

[14] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[15] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[16] B. Brito, H. Zhu, W. Pan, and J. Alonso-Mora. Social-vrnn: one-shot multi-modal trajectory prediction for interacting pedestrians. *arXiv preprint arXiv:2010.09056*, 2020.

[17] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *arXiv preprint arXiv:2303.04226*, 2023.

[18] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. *Advances in neural information processing systems*, 31, 2018.

[19] H. Cheng, W. Liao, M. Y. Yang, B. Rosenhahn, and M. Sester. Amenet: Attentive maps encoder network for trajectory prediction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 172:253–266, 2021.

[20] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

[21] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018.

[22] C. Dias, O. Ejtemai, M. Sarvi, and N. Shiwakoti. Pedestrian walking characteristics through angled corridors: An experimental study. *Transportation research record*, 2421(1):41–50, 2014.

[23] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[24] S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR, 2018.

[25] M. Feurer and F. Hutter. Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham, 2019.

[26] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*, 2019.

[27] E. C. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[28] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[29] C. Graber and A. Schwing. Dynamic neural relational inference for forecasting trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1018–1019, 2020.

[30] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[31] I. Hasan, F. Setti, T. Tsesmelis, A. Del Bue, F. Galasso, and M. Cristani. 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*, pages 6067–6076, 2018.

[32] D. Helbing. Physikalische modellierung des dynamischen verhaltens von fußgängern (physical modeling of the dynamic behavior of pedestrians). *Available at SSRN 2413177*, 1990.

[33] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[34] C. Hermes, C. Wohler, K. Schenk, and F. Kummert. Long-term vehicle motion prediction. In *2009 IEEE intelligent vehicles symposium*, pages 652–657. IEEE, 2009.

[35] J. Hillenbrand, A. M. Spieker, and K. Kroschel. A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs. *IEEE Transactions on intelligent transportation systems*, 7(4):528–540, 2006.

[36] H. H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, 2012.

[37] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1450–1464, 2006.

[38] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR, 2014.

[39] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer, 2011.

[40] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[41] R. Jie and J. Gao. Differentiable neural architecture search for high-dimensional time series forecasting. *IEEE Access*, 9:20922–20932, 2021.

[42] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.

[43] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011.

[44] R. E. Kalman et al. A new approach to linear filtering and prediction problems [j]. *Journal of basic Engineering*, 82(1):35–45, 1960.

[45] O. Khatib, E. Demircan, V. De Sapio, L. Sentis, T. Besier, and S. Delp. Robotics-based synthesis of human motion. *Journal of physiology-Paris*, 103(3-5):211–219, 2009.

[46] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. H. Rezatofighi, and S. Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *arXiv preprint arXiv:1907.03395*, 2019.

[48] M. d. B. Kostya Linou, Dzmitryi Linou. Nba player movements. https://github.com/linouk23/NBA-Player-Movements, 2016.

[49] P. Kothari, S. Kreiss, and A. Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[50] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018.

[51] S. Kreiss. Deep social force. *arXiv preprint arXiv:2109.12081*, 2021.

[52] Y. LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. 2022.

[53] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[54] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, and Y. Zheng. Autost: Efficient neural architecture search for spatio-temporal prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 794–802, 2020.

[55] J. Lian, W. Ren, L. Li, Y. Zhou, and B. Zhou. Ptp-stgcn: pedestrian trajectory prediction based on a spatio-temporal graph convolutional neural network. *Applied Intelligence*, 53(3):2862–2878, 2023.

[56] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[57] O. Makansi, J. Von Kügelgen, F. Locatello, P. Gehler, D. Janzing, T. Brox, and B. Schölkopf. You mostly walk alone: Analyzing feature attribution in trajectory prediction. *arXiv preprint arXiv:2110.05304*, 2021.

[58] K. Mangalam, Y. An, H. Girase, and J. Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15233–15242, 2021.

[59] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 759–776. Springer, 2020.

[60] B. Mersch, T. Höllen, K. Zhao, C. Stachniss, and R. Roscher. Maneuver-based trajectory prediction for self-driving cars using spatio-temporal convolutional networks. *arXiv preprint arXiv:2109.07365*, 2021.

[61] A. Mészáros, J. Alonso-Mora, and J. Kober. Trajflow: Learning the distribution over trajectories. *arXiv preprint arXiv:2304.05166*, 2023.

[62] R. Miller and Q. Huang. An adaptive peer-to-peer collision warning system. In *Vehicular Technology Conference. IEEE 55th Vehicular Technology Conference. VTC Spring 2002 (Cat. No. 02CH37367)*, volume 1, pages 317–321. IEEE, 2002.

[63] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel. 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*, pages 14424–14432, 2020.

[64] A. Mohamed, D. Zhu, W. Vu, M. Elhoseiny, and C. Claudel. Social-implicit: Rethinking trajectory prediction evaluation and the effectiveness of implicit maximum likelihood estimation. In *European Conference on Computer Vision*, pages 463–479. Springer, 2022.

[65] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz. The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PloS one*, 5(4):e10047, 2010.

[66] Z. Pan, S. Ke, X. Yang, Y. Liang, Y. Yu, J. Zhang, and Y. Zheng. Autostg: Neural architecture search for predictions of spatio-temporal graph. In *Proceedings of the Web Conference 2021*, pages 1846–1855, 2021.

[67] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pages 261–268. IEEE, 2009.

[68] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[69] C. E. Rasmussen, C. K. Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

[70] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020.

[71] J. R. Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

[72] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi. Trajnet: Towards a benchmark for human trajectory prediction. *arXiv preprint*, 2018.

[73] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1349–1358, 2019.

[74] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020.

[75] N. Seward. Nba player movements. https://github.com/sealneaward/nba-movement-data, 2018.

[76] I. Soboĺ. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp.*, 1:407, 1993.

[77] Q. Song, H. Jin, and X. Hu. *Automated Machine Learning in Action*. Manning, 2022.

[78] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[79] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[80] Q. Tran and J. Firl. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression. In *2014 ieee intelligent vehicles symposium proceedings*, pages 918–923. IEEE, 2014.

[81] L. Tuggener, M. Amirian, K. Rombach, S. Lörwald, A. Varlet, C. Westermann, and T. Stadelmann. Automated machine learning in practice: state of the art and recent results. In *2019 6th Swiss Conference on Data Science (SDS)*, pages 31–36. IEEE, 2019.

[82] D. Vasquez and T. Fraichard. Motion prediction for moving objects: a statistical approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3931–3936. IEEE, 2004.

[83] M. Verma, M. S. K. Reddy, Y. R. Meedimale, M. Mandal, and S. K. Vipparthi. Automer: Spatiotemporal neural architecture search for microexpression recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[84] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020.

[85] C. Wang, Y. Wang, M. Xu, and D. J. Crandall. Stepwise goal-driven networks for trajectory prediction. *IEEE Robotics and Automation Letters*, 7(2):2716–2723, 2022.

[86] X. Wang, J. Alonso-Mora, and M. Wang. Probabilistic risk metric for highway driving leveraging multi-modal trajectory predictions. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[87] Z. Wang, C. Lin, L. Sheng, J. Yan, and J. Shao. Pv-nas: Practical neural architecture search for video recognition. *arXiv preprint arXiv:2011.00826*, 2020.

[88] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[89] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter. Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*, 2023.

[90] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[91] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[92] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[93] C. Xu, W. Mao, W. Zhang, and S. Chen. Remember intentions: Retrospective-memory-based trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6488–6497, 2022.

[94] P. Xu, J.-B. Hayet, and I. Karamouzas. Socialvae: Human trajectory prediction using timewise latents. *arXiv preprint arXiv:2203.08207*, 2022.

[95] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du. Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation. *IEEE Robotics and Automation Letters*, 6(2):1463–1470, 2021.

[96] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. *arXiv preprint arXiv:2103.14023*, 2021.

[97] Y. Yue, P. Lucey, P. Carr, A. Bialkowski, and I. Matthews. Learning fine-grained spatial models for dynamic sports play prediction. In *2014 IEEE international conference on data mining*, pages 670–679. IEEE, 2014.

[98] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12085–12094, 2019.

[99] Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):1–41, 2015.

[100] Y. Zhou, B. Li, Z. Wang, and H. Li. Video action recognition with neural architecture search. In *Asian Conference on Machine Learning*, pages 1675–1690. PMLR, 2021.

[101] L. Zimmer, M. Lindauer, and F. Hutter. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090, 2021.

[102] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[103] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

ACKNOWLEDGMENTS

I very deeply want to thank Mitra for her continuous support during this incredible long, hard and inspiring research period. I furthermore want to thank Javier for his valuable insights and suggestions during our common meetings as well as Holger for his support and for welcoming me as a part of the ADA group.

10 Appendix

10.1 Features

For the feature selection of our approach, we build on established choices from the literature. In particular, we orient our choice on features that have been used for multi-modal short-term trajectory prediction, such as in the work of Xu et al. [94], Kothari et al. [49] and Amirian et al. [3]. As individual features per agent, we use the position and approximated velocity and acceleration information extracted from the recorded positional displacements. For the social inter-agent features, we consider the relative position, velocity and acceleration defined by $x_i^t - x_j^t$, as well as the Euclidean distance between agents, the bearing angle between them and the distance of the closest approach (e.g. the projected minimal distance between two agents, assuming constant velocity).

10.2 Additional notes on data extraction

Additional notes on the HighD and InD data extraction: The HighD[50] and InD[13] data is openly available for research purposes from leveLXData [3] and the Institute for Automotive Engineering (ika) of RWTH Aachen University[4]. Request for the data can be made via email. We processed the data by concatenating all recordings together and downsample them by a factor of 10 from 25 to 2.5 Hertz as used in the ETH/UCY benchmark. We divide the data into scenes of 8 seconds coinciding with 20 frames and randomly select 2.500 scenes from those.

Additional notes on the NBA-SportVU data extraction: We use the publicly available GitHub repository of Neil Seward [75][5] to obtain the raw movement data. Following we take the event log provided to identify moments labeled as "score" events. We divide these events up into scenes of constant 4s (20 frames) length and randomly select scenes from those, excluding scenes without movement (eg. where the primary agent moves less than a threshold of 1m). In total, we collected 2.500 unique scenes.

10.3 Search Space

The following section outlines the possible parameter choices for each method where parameter optimisation was applied.

AutoTraj Search Space A listing of the parameters used in the AutoTraj search space:

```
batch_size:
  values: [16, 32, 64, 128, 256, 512]
lr:
  distribution: "log_uniform_values"
  min: 0.00001
```

[3] https://levelxdata.com/
[4] https://www.ika.rwth-aachen.de/de/
[5] https://github.com/sealneaward/nba-movement-data/tree/master/data/events

```
  max: 0.1
grad_clip:
  distribution: "uniform"
  min: 0.1
  max: 5
beta:
  distribution: "uniform"
  min: 0.1
  max: 100
input_representation_state:
  values: ["v", "va", "p", "pv", "pa", "pva", "a"]
input_representation_rel_state:
  values: ["d", "db", "dbm", "b", "bm", "m", "v", "va", "p", "pv", "pa",
      "pva", "a"]
num_rel_state_embedding:
  values: [16, 32, 64, 128, 256, 512]
num_state_embedding:
  values: [16, 32, 64, 128, 256, 512]
layer_attributes_embedding:
  values: ["custom_linear"]
num_layers_attributes_embedding:
  values: [1, 2, 3, 4, 5]
num_units_attributes_embedding:
  values: [16, 32, 64, 128, 256, 512]
num_units_attributes_embedding:
  values: [16, 32, 64, 128, 256, 512]
dropout_attributes_embedding:
  values: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
activation_attributes_embedding:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu", "elu"]
num_hidden_embedding:
  values: [16, 32, 64, 128, 256, 512]
num_hidden_future_embedding:
  values: [16, 32, 64, 128, 256, 512]
layer_hidden_init:
  values: ["linear", "deep_linear"]
layer_hidden_step:
  values: ["gru", "lstm"]
future_use_neighbour:
  values: [0, 1]
future_direction:
  values: ["forwards", "backwards"]
neighbourhood_width:
  distribution: "uniform"
  min: 1
  max: 32
neighbourhood_height_front:
```

```
  distribution: "uniform"
  min: 1
  max: 16
neighbourhood_height_back:
  distribution: "uniform"
  min: 1
  max: 16
dd_dim:
  values: [1,2]
gridding_divisions:
  values: [4, 16, 32, 64]
layer_embedding_querry:
  values: ["linear", "deep_linear"]
target_representation:
  values: ["position", "velocity", "acceleration"]
metric_representation:
  values: ["position", "velocity", "acceleration"]
not_present_fill:
  values: [-1000000, 0, 1000000]
lambda_importance:
  distribution: "uniform"
  min: 0.1
  max: 10
gamma:
  distribution: "uniform"
  min: 0.1
  max: 1
decoder_type:
  values: ["stepwise_rnn", "stepwise_latent_rnn", "direct_rnn", "physics
     "]
z_dim:
  values: [16, 32, 64, 128, 256, 512]
hidden_dim:
  values: [16, 32, 64, 128, 256, 512]
hidden_dim_future:
  values: [16, 32, 64, 128, 256, 512]
embed_dim:
  values: [16, 32, 64, 128, 256, 512]
layer_step_decoder:
  values: ["lstm", "gru"]
pooling_attribute:
  values: ["hidden", "direct", "proxy", "rel_dist", "rel_vel", "rel_acc"
     ]
pooling_mechanism:
  values: ["attention", "gridding", "concat", "maxpool", "sumpool"]
explicit_predictions:
  values: [0, 1]
```

```
n_train_predictions:
  distribution: "q_uniform"
  min: 1
  max: 20
  q: 1
layer_interaction_embedding:
  values: ["custom_linear"]
num_interaction_embedding:
  values: [16, 32, 64, 128, 256, 512]
num_layers_interaction_embedding:
  values: [1, 2, 3, 4, 5]
dropout_interaction_embedding:
  values: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
activation_interaction_embedding:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu", "elu"]
num_units_interaction_embedding:
  values: [16, 32, 64, 128, 256, 512]
```

Refined AutoTraj Search Space for ETH/UCY A listing of the parameters used for the refined search space for the AutoTraj framework as applied on the ETH/UCY benchmark:

```
batch_size:
  values: [64, 128, 256, 512]
lr:
  distribution: "log_uniform_values"
  min: 0.0001
  max: 0.01
grad_clip:
  distribution: "uniform"
  min: 2
  max: 4
beta:
  distribution: "uniform"
  min: 10
  max: 50
input_representation_state:
  values: ["v", "a", "pa", "pva"]
input_representation_rel_state:
  values: ["d", "db", "dbm", "b", "bm", "m", "v", "va", "p", "pv", "pa"]
num_rel_state_embedding:
  values: [16, 32, 64, 128, 256, 512]
num_state_embedding:
  values: [16, 32, 64, 128, 256, 512]
layer_attributes_embedding:
  values: ["custom_linear"]
num_layers_attributes_embedding:
  values: [1, 2, 3, 4, 5]
```

```
num_units_attributes_embedding:
  values: [16, 32, 64, 128, 256, 512]
num_units_attributes_embedding:
  values: [16, 32, 64, 128, 256, 512]
dropout_attributes_embedding:
  values: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
activation_attributes_embedding:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu"]
num_hidden_embedding:
  values: [16, 32, 64, 128, 256]
num_hidden_future_embedding:
  values: [16, 32, 64, 128, 256]
layer_hidden_init:
  values: ["linear", "deep_linear"]
layer_hidden_step:
  values: ["gru", "lstm"]
future_use_neighbour:
  values: [0, 1]
future_direction:
  values: ["forwards", "backwards"]
neighbourhood_width:
  distribution: "uniform"
  min: 1
  max: 32
neighbourhood_height_front:
  distribution: "uniform"
  min: 1
  max: 12
neighbourhood_height_back:
  distribution: "uniform"
  min: 1
  max: 16
dd_dim:
  values: [1,2]
gridding_divisions:
  values: [16, 32]
layer_embedding_querry:
  values: ["linear", "deep_linear"]
target_representation:
  values: ["velocity", "acceleration"]
metric_representation:
  values: ["position", "velocity"]
not_present_fill:
  values: [-1000000, 0, 1000000]
lambda_importance:
  distribution: "uniform"
  min: 1
```

```
    max: 9
  gamma:
    distribution: "uniform"
    min: 0.2
    max: 0.8
  decoder_type:
    values: ["stepwise_rnn", "stepwise_latent_rnn", "physics"]
  z_dim:
    values: [16, 32, 64, 128, 256, 512]
  hidden_dim:
    values: [16, 32, 64, 128, 256, 512]
  hidden_dim_future:
    values: [16, 32, 64, 128, 256, 512]
  embed_dim:
    values: [16, 32, 64, 128, 256, 512]
  layer_step_decoder:
    values: ["lstm", "gru"]
  pooling_attribute:
    values: ["direct", "proxy", "rel_vel"]
  pooling_mechanism:
    values: ["attention", "gridding", "concat", "maxpool", "sumpool"]
  explicit_predictions:
    values: [0, 1]
  n_train_predictions:
    distribution: "q_uniform"
    min: 5
    max: 20
    q: 1
  layer_interaction_embedding:
    values: ["custom_linear"]
  num_interaction_embedding:
    values: [16, 32, 64, 128]
  num_layers_interaction_embedding:
    values: [1, 2, 3, 4, 5]
  dropout_interaction_embedding:
    values: [0.0, 0.1, 0.2, 0.3, 0.4]
  activation_interaction_embedding:
    values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu"]
  num_units_interaction_embedding:
    values: [16, 32, 64, 128, 256, 512]
```

Hyperparameter optimisation search space for AutoTraj A listing of the parameters used for the hyperparameter optimisation phase of the AutoTraj framework:

```
  batch_size:
      values: [16, 32, 64, 128, 256, 512, 1024]
  lr:
```

```
      distribution: "log_uniform_values"
      min: 0.00001
      max: 0.1
  grad_clip:
      distribution: "uniform"
      min: 0.1
      max: 10
  beta:
      distribution: "uniform"
      min: 0.01
      max: 10
  lambda_importance:
      distribution: "uniform"
      min: 0.01
      max: 10
  gamma:
      distribution: "uniform"
      min: 0.01
      max: 10
  n_cycles:
      values: [1, 2, 4, 6, 8, 16]
  cycle_ratio:
      distribution: "uniform"
      min: 0.1
      max: 1
  cycle_min:
      distribution: "uniform"
      min: 0
      max: 0.5
  cycle_max:
      distribution: "uniform"
      min: 0.5
      max: 1
  n_train_predictions:
      distribution: "q_uniform"
      q: 2
      min: 2
      max: 40
```

Minimal AutoTraj Search Space A listing of the parameters for the simpler AutoML search space:

```
encoder_embed_dim:
  values: [32, 64, 128, 256]
encoder_num_layers:
  values: [2, 3, 4, 5]
encoder_num_units:
```

```
  values: [64, 128, 256, 512]
encoder_activation:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu", "elu"]
encoder_dropout:
  values: [0.0, 0.1, 0.2]
encoder_use_batchnorm:
  values: [True, False]
encoder_future_embed_dim:
  values: [32, 64, 128, 256]
encoder_future_num_layers:
  values: [2, 3, 4, 5]
encoder_future_num_units:
  values: [64, 128, 256, 512]
encoder_future_activation:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu", "elu"]
encoder_future_dropout:
  values: [0.0, 0.1, 0.2]
encoder_future_use_batchnorm:
  values: [True, False]
distribution_num_layers:
  values: [2, 3, 4, 5]
distribution_num_units:
  values: [64, 128, 256, 512]
distribution_activation:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu", "elu"]
distribution_dropout:
  values: [0.0, 0.1, 0.2]
distribution_use_batchnorm:
  values: [True, False]
decoder_num_layers:
  values: [2, 3, 4, 5]
decoder_num_units:
  values: [64, 128, 256, 512]
decoder_activation:
  values: ["relu", "tanh", "sigmoid", "relu6", "leaky_relu", "elu"]
decoder_dropout:
  values: [0.0, 0.1, 0.2]
not_present_fill:
  values: [-1000000, 0, 1000000]
decoder_use_batchnorm:
  values: [True, False]
lr:
  distribution: "log_uniform_values"
  min: 0.00001
  max: 0.1
beta:
  distribution: "uniform"
```

```
  min: 0.1
  max: 100
grad_clip:
  distribution: "uniform"
  min: 0.1
  max: 5
n_train_predictions:
  distribution: "q_uniform"
  min: 1
  max: 20
  q: 1
lambda_importance:
  distribution: "uniform"
  min: 0.1
  max: 10
gamma:
  distribution: "uniform"
  min: 0.1
  max: 1
batch_size:
  values: [32, 64, 128, 256]
ar:
  values: [True, False]
input_representation_state:
  values: ["v", "va", "p", "pv", "pa", "pva", "a"]
input_representation_rel_state:
  values: ["d", "db", "dbm", "b", "bm", "m", "v", "va", "p", "pv", "pa",
      "pva", "a"]
target_representation:
  values: ["position", "velocity", "acceleration"]
metric_representation:
  values: ["position", "velocity", "acceleration"]
z_dim:
  values: [16, 32, 64, 128, 256]
num_hidden_embedding:
  values: [32, 64, 128 ,256]
embed_dim:
  values: [32, 64, 128, 256]
layer_step_decoder:
  values: ["gru", "lstm"]
rnn_decoder:
  values: [True,False]
neighbourhood_width:
  distribution: "uniform"
  min: 1
  max: 16
neighbourhood_height_front:
```

```
    distribution: "uniform"
    min: 1
    max: 16
  neighbourhood_height_back:
    distribution: "uniform"
    min: 1
    max: 16
```

Hyperparameter optimisation search space for SVAE The following parameters were used:

```
    batch_size:
    values: [8, 16, 32, 64, 128, 256]
  lr:
    distribution: "log_uniform_values"
    min: 0.00001
    max: 0.1
  epochs:
    distribution: "int_uniform"
    min: 10
    max: 200
  rnn_hidden_dim:
    values: [8, 16, 32, 64, 128, 256, 512]
```

Hyperparameter optimisation search space for AMENet The following parameters were used:

```
    batch_size:
    values:
    - 16
    - 32
    - 64
    - 128
    - 256
    - 512
    - 1024
  epochs:
    distribution: int_uniform
    max: 200
    min: 10
  hidden_size:
    values:
    - 4
    - 8
    - 16
    - 32
    - 64
```

```
    - 128
    - 256
    - 512
    - 1024
  lr:
    distribution: log_uniform_values
    max: 0.1
    min: 1.0e-05
```

10.4 Optimal selected architecture and parameter settings

To give an intuition of the type of architectures selected by our AutoML procedure we hereunder list the selection with minimal validation error for each dataset and each of the five folds.

HighD :


```
{'lr': 0.001871284874821101, 'beta': 10.629943437807665, 'gamma':
   0.6949478174063634, 'z_dim': 16.0, 'dd_dim': 2.0, 'embed_dim': 128.0,
   'grad_clip': 1.2189471957365376, 'batch_size': 512.0, 'hidden_dim':
   128.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_hidden', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'forwards', '
   not_present_fill': 1000000.0, 'hidden_dim_future': 64.0, '
   lambda_importance': 3.62889353970355, 'layer_hidden_init': 'linear', '
   layer_hidden_step': 'lstm', 'pooling_attribute': 'hidden', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 64.0, '
   layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
   n_train_predictions': 8.0, 'neighbourhood_width': 17.075128893810508,
   'num_state_embedding': 64.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 32.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 6.0, 'num_rel_state_embedding
   ': 512.0, 'neighbourhood_height_back': 9.471169204745804, '
   num_interaction_embedding': 512.0, 'input_representation_state': 'a',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 10.456941587630393, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
   0.1, 'dropout_interaction_embedding': 0.2, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pva', 'num_units_attributes_embedding': 16.0, '
   activation_attributes_embedding': 'elu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 2.0, '
   num_units_interaction_embedding': 16.0, '
```

```
   activation_interaction_embedding': 'relu6', '
   num_layers_interaction_embedding': 4.0}
{'lr': 0.0005485335156951346, 'beta': 5.250345525933247, 'gamma':
   0.5070119446339979, 'z_dim': 256.0, 'dd_dim': 1.0, 'embed_dim': 256.0,
    'grad_clip': 1.3590890123589998, 'batch_size': 64.0, 'hidden_dim':
   256.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_rel_acc', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 1000000.0, 'hidden_dim_future': 512.0, '
   lambda_importance': 8.79145098470304, 'layer_hidden_init': 'linear', '
   layer_hidden_step': 'lstm', 'pooling_attribute': 'rel_acc', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 16.0, '
   layer_step_decoder': 'gru', 'num_state_features': 4.0, '
   n_train_predictions': 16.0, 'neighbourhood_width': 9.191567277177944,
   'num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding
   ': 128.0, 'neighbourhood_height_back': 15.59359098885803, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'pv',
    'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 11.381529560252728, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
   0.0, 'dropout_interaction_embedding': 0.3, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pv', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 3.0, '
   num_units_interaction_embedding': 256.0, '
   activation_interaction_embedding': 'sigmoid', '
   num_layers_interaction_embedding': 4.0}
{'lr': 0.001733731734610129, 'beta': 5.6934429843431245, 'gamma':
   0.15411676672344485, 'z_dim': 128.0, 'dd_dim': 1.0, 'embed_dim':
   256.0, 'grad_clip': 3.526728887140914, 'batch_size': 64.0, 'hidden_dim
   ': 64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'gridding_direct', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 0.0, 'hidden_dim_future': 32.0, 'lambda_importance
   ': 1.7295968180764674, 'layer_hidden_init': 'deep_linear', '
   layer_hidden_step': 'gru', 'pooling_attribute': 'direct', '
   pooling_mechanism': 'gridding', 'gridding_divisions': 64.0, '
   layer_step_decoder': 'gru', 'num_state_features': 2.0, '
   n_train_predictions': 12.0, 'neighbourhood_width': 21.150008752776, '
   num_state_embedding': 64.0, 'explicit_predictions': 0.0, '
```

```
    future_use_neighbour': 0.0, 'num_hidden_embedding': 32.0, '
    seperate_interaction': 0.0, 'metric_representation': 'acceleration', '
    target_representation': 'acceleration', 'layer_embedding_querry': '
    deep_linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding
    ': 512.0, 'neighbourhood_height_back': 13.89266510897011, '
    num_interaction_embedding': 256.0, 'input_representation_state': 'a',
    'layer_attributes_embedding': 'custom_linear', '
    neighbourhood_height_front': 9.792813381627402, '
    layer_interaction_embedding': 'custom_linear', '
    num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
    0.1, 'dropout_interaction_embedding': 0.0, '
    batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
    ': 'va', 'num_units_attributes_embedding': 64.0, '
    activation_attributes_embedding': 'elu', '
    batchnorm_interaction_embedding': 0.0, '
    num_layers_attributes_embedding': 5.0, '
    num_units_interaction_embedding': 32.0, '
    activation_interaction_embedding': 'sigmoid', '
    num_layers_interaction_embedding': 1.0}
{'lr': 0.0007243026341478893, 'beta': 49.51575987055742, 'gamma':
    0.4122696621369472, 'z_dim': 128.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
     'grad_clip': 1.6695702218093782, 'batch_size': 32.0, 'hidden_dim':
    64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
    pooling_method': 'maxpool_direct', 'metric_is_polar': 0.0, '
    target_is_polar': 0.0, 'future_direction': 'backwards', '
    not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
    lambda_importance': 9.198549771058037, 'layer_hidden_init': '
    deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': 'direct
    ', 'pooling_mechanism': 'maxpool', 'gridding_divisions': 32.0, '
    layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
    n_train_predictions': 13.0, 'neighbourhood_width': 26.27988407920095,
    'num_state_embedding': 256.0, 'explicit_predictions': 0.0, '
    future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
    seperate_interaction': 0.0, 'metric_representation': 'velocity', '
    target_representation': 'acceleration', 'layer_embedding_querry': '
    linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding':
    512.0, 'neighbourhood_height_back': 6.286021204608768, '
    num_interaction_embedding': 256.0, 'input_representation_state': 'a',
    'layer_attributes_embedding': 'custom_linear', '
    neighbourhood_height_front': 8.890878310063822, '
    layer_interaction_embedding': 'custom_linear', '
    num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
    0.5, 'dropout_interaction_embedding': 0.3, '
    batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
    ': 'v', 'num_units_attributes_embedding': 256.0, '
    activation_attributes_embedding': 'leaky_relu', '
    batchnorm_interaction_embedding': 0.0, '
```

num_layers_attributes_embedding': 2.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 4.0}
{'lr': 0.00012564682513588392, 'beta': 50.07895194925274, 'gamma':
0.16488176095282453, 'z_dim': 256.0, 'dd_dim': 2.0, 'embed_dim': 32.0,
'grad_clip': 0.1613073103362389, 'batch_size': 128.0, 'hidden_dim':
32.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'concat_rel_acc', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
lambda_importance': 3.0623938423960624, 'layer_hidden_init': 'linear',
'layer_hidden_step': 'gru', 'pooling_attribute': 'rel_acc', '
pooling_mechanism': 'concat', 'gridding_divisions': 64.0, '
layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 7.0, 'neighbourhood_width': 25.65988347121772, '
num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 1.0, 'num_hidden_embedding': 64.0, '
seperate_interaction': 0.0, 'metric_representation': 'position', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 1.0, 'num_rel_state_embedding':
256.0, 'neighbourhood_height_back': 15.33109788084968, '
num_interaction_embedding': 128.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 14.420131737031737, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
0.0, 'dropout_interaction_embedding': 0.0, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'b', 'num_units_attributes_embedding': 32.0, '
activation_attributes_embedding': 'relu6', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 5.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 2.0}

InD :

{'lr': 0.001871284874821101, 'beta': 10.629943437807665, 'gamma':
0.6949478174063634, 'z_dim': 16.0, 'dd_dim': 2.0, 'embed_dim': 128.0,
'grad_clip': 1.2189471957365376, 'batch_size': 512.0, 'hidden_dim':
128.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'sumpool_hidden', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'forwards', '
not_present_fill': 1000000.0, 'hidden_dim_future': 64.0, '
lambda_importance': 3.62889353970355, 'layer_hidden_init': 'linear', '

```
   layer_hidden_step': 'lstm', 'pooling_attribute': 'hidden', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 64.0, '
   layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
   n_train_predictions': 8.0, 'neighbourhood_width': 17.075128893810508,
   'num_state_embedding': 64.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 32.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 6.0, 'num_rel_state_embedding
   ': 512.0, 'neighbourhood_height_back': 9.471169204745804, '
   num_interaction_embedding': 512.0, 'input_representation_state': 'a',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 10.456941587630393, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
   0.1, 'dropout_interaction_embedding': 0.2, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pva', 'num_units_attributes_embedding': 16.0, '
   activation_attributes_embedding': 'elu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 2.0, '
   num_units_interaction_embedding': 16.0, '
   activation_interaction_embedding': 'relu6', '
   num_layers_interaction_embedding': 4.0}
{'lr': 0.0005485335156951346, 'beta': 5.250345525933247, 'gamma':
   0.5070119446339979, 'z_dim': 256.0, 'dd_dim': 1.0, 'embed_dim': 256.0,
    'grad_clip': 1.3590890123589998, 'batch_size': 64.0, 'hidden_dim':
   256.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_rel_acc', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 1000000.0, 'hidden_dim_future': 512.0, '
   lambda_importance': 8.79145098470304, 'layer_hidden_init': 'linear', '
   layer_hidden_step': 'lstm', 'pooling_attribute': 'rel_acc', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 16.0, '
   layer_step_decoder': 'gru', 'num_state_features': 4.0, '
   n_train_predictions': 16.0, 'neighbourhood_width': 9.191567277177944,
   'num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding
   ': 128.0, 'neighbourhood_height_back': 15.59359098885803, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'pv',
    'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 11.381529560252728, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
```

```
   0.0, 'dropout_interaction_embedding': 0.3, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pv', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 3.0, '
   num_units_interaction_embedding': 256.0, '
   activation_interaction_embedding': 'sigmoid', '
   num_layers_interaction_embedding': 4.0}
{'lr': 0.001733731734610129, 'beta': 5.6934429843431245, 'gamma':
   0.15411676672344485, 'z_dim': 128.0, 'dd_dim': 1.0, 'embed_dim':
   256.0, 'grad_clip': 3.526728887140914, 'batch_size': 64.0, 'hidden_dim
   ': 64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'gridding_direct', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 0.0, 'hidden_dim_future': 32.0, 'lambda_importance
   ': 1.7295968180764674, 'layer_hidden_init': 'deep_linear', '
   layer_hidden_step': 'gru', 'pooling_attribute': 'direct', '
   pooling_mechanism': 'gridding', 'gridding_divisions': 64.0, '
   layer_step_decoder': 'gru', 'num_state_features': 2.0, '
   n_train_predictions': 12.0, 'neighbourhood_width': 21.150008752776, '
   num_state_embedding': 64.0, 'explicit_predictions': 0.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 32.0, '
   seperate_interaction': 0.0, 'metric_representation': 'acceleration', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding
   ': 512.0, 'neighbourhood_height_back': 13.89266510897011, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'a',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 9.792813381627402, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
   0.1, 'dropout_interaction_embedding': 0.0, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'va', 'num_units_attributes_embedding': 64.0, '
   activation_attributes_embedding': 'elu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 5.0, '
   num_units_interaction_embedding': 32.0, '
   activation_interaction_embedding': 'sigmoid', '
   num_layers_interaction_embedding': 1.0}
{'lr': 0.0007243026341478893, 'beta': 49.51575987055742, 'gamma':
   0.4122696621369472, 'z_dim': 128.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
    'grad_clip': 1.6695702218093782, 'batch_size': 32.0, 'hidden_dim':
   64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'maxpool_direct', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
```

not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
lambda_importance': 9.198549771058037, 'layer_hidden_init': '
deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': 'direct
', 'pooling_mechanism': 'maxpool', 'gridding_divisions': 32.0, '
layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
n_train_predictions': 13.0, 'neighbourhood_width': 26.27988407920095,
'num_state_embedding': 256.0, 'explicit_predictions': 0.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
seperate_interaction': 0.0, 'metric_representation': 'velocity', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding':
512.0, 'neighbourhood_height_back': 6.286021204608768, '
num_interaction_embedding': 256.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 8.890878310063822, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
0.5, 'dropout_interaction_embedding': 0.3, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'v', 'num_units_attributes_embedding': 256.0, '
activation_attributes_embedding': 'leaky_relu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 2.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 4.0}
{'lr': 0.00012564682513588392, 'beta': 50.07895194925274, 'gamma':
0.16488176095282453, 'z_dim': 256.0, 'dd_dim': 2.0, 'embed_dim': 32.0,
'grad_clip': 0.1613073103362389, 'batch_size': 128.0, 'hidden_dim':
32.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'concat_rel_acc', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
lambda_importance': 3.0623938423960624, 'layer_hidden_init': 'linear',
'layer_hidden_step': 'gru', 'pooling_attribute': 'rel_acc', '
pooling_mechanism': 'concat', 'gridding_divisions': 64.0, '
layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 7.0, 'neighbourhood_width': 25.65988347121772, '
num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 1.0, 'num_hidden_embedding': 64.0, '
seperate_interaction': 0.0, 'metric_representation': 'position', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 1.0, 'num_rel_state_embedding':
256.0, 'neighbourhood_height_back': 15.33109788084968, '
num_interaction_embedding': 128.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 14.420131737031737, '

```
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
   0.0, 'dropout_interaction_embedding': 0.0, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'b', 'num_units_attributes_embedding': 32.0, '
   activation_attributes_embedding': 'relu6', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 5.0, '
   num_units_interaction_embedding': 256.0, '
   activation_interaction_embedding': 'sigmoid', '
   num_layers_interaction_embedding': 2.0}
{'lr': 0.0009849023776411182, 'beta': 30.0385402041938, 'gamma':
   0.28075874198252565, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
    'grad_clip': 3.643101627463447, 'batch_size': 128.0, 'hidden_dim':
   256.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_hidden', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
   lambda_importance': 7.564493274359233, 'layer_hidden_init': 'linear',
   'layer_hidden_step': 'gru', 'pooling_attribute': 'hidden', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 64.0, '
   layer_step_decoder': 'gru', 'num_state_features': 2.0, '
   n_train_predictions': 5.0, 'neighbourhood_width': 6.009149142999294, '
   num_state_embedding': 64.0, 'explicit_predictions': 0.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 256.0, '
   seperate_interaction': 0.0, 'metric_representation': 'velocity', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   linear', 'num_rel_state_features': 3.0, 'num_rel_state_embedding':
   32.0, 'neighbourhood_height_back': 11.462160531997329, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'a',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 4.614985495199063, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 128.0, 'dropout_attributes_embedding':
   0.5, 'dropout_interaction_embedding': 0.4, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'dbm', 'num_units_attributes_embedding': 256.0, '
   activation_attributes_embedding': 'leaky_relu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 2.0, '
   num_units_interaction_embedding': 128.0, '
   activation_interaction_embedding': 'relu', '
   num_layers_interaction_embedding': 2.0}
{'lr': 0.0012383128479368905, 'beta': 2.462578522241241, 'gamma':
   0.8825587980631394, 'z_dim': 128.0, 'dd_dim': 1.0, 'embed_dim': 256.0,
    'grad_clip': 1.6841935506475347, 'batch_size': 256.0, 'hidden_dim':
   16.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
```

```
   pooling_method': 'maxpool_direct', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
   lambda_importance': 6.563668830665317, 'layer_hidden_init': 'linear',
   'layer_hidden_step': 'lstm', 'pooling_attribute': 'direct', '
   pooling_mechanism': 'maxpool', 'gridding_divisions': 32.0, '
   layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
   n_train_predictions': 8.0, 'neighbourhood_width': 2.1138183092631464,
   'num_state_embedding': 128.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 16.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 3.0, 'num_rel_state_embedding
   ': 32.0, 'neighbourhood_height_back': 2.2229321214993933, '
   num_interaction_embedding': 64.0, 'input_representation_state': 'a', '
   layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 13.300122026542429, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 128.0, 'dropout_attributes_embedding':
   0.4, 'dropout_interaction_embedding': 0.1, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'dbm', 'num_units_attributes_embedding': 32.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 2.0, '
   num_units_interaction_embedding': 256.0, '
   activation_interaction_embedding': 'sigmoid', '
   num_layers_interaction_embedding': 5.0}
{'lr': 0.0006912167047430998, 'beta': 13.870549759519514, 'gamma':
   0.7349153630288328, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 512.0,
   'grad_clip': 1.6372606478740337, 'batch_size': 128.0, 'hidden_dim':
   16.0, 'normalizer': 'none', 'decoder_type': 'stepwise_rnn', '
   pooling_method': 'sumpool_direct', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
   lambda_importance': 4.815990416640407, 'layer_hidden_init': '
   deep_linear', 'layer_hidden_step': 'lstm', 'pooling_attribute': '
   direct', 'pooling_mechanism': 'sumpool', 'gridding_divisions': 32.0, '
   layer_step_decoder': 'gru', 'num_state_features': 4.0, '
   n_train_predictions': 4.0, 'neighbourhood_width': 9.022686481496356, '
   num_state_embedding': 128.0, 'explicit_predictions': 0.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 128.0, '
   seperate_interaction': 0.0, 'metric_representation': 'acceleration', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 1.0, 'num_rel_state_embedding
   ': 512.0, 'neighbourhood_height_back': 7.960384888347766, '
   num_interaction_embedding': 128.0, 'input_representation_state': 'va',
```

```
 'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 7.057854177205689, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 512.0, 'dropout_attributes_embedding':
0.3, 'dropout_interaction_embedding': 0.4, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'm', 'num_units_attributes_embedding': 512.0, '
activation_attributes_embedding': 'tanh', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 3.0, '
num_units_interaction_embedding': 64.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 2.0}
{'lr': 0.00282162953278939, 'beta': 46.13907961289852, 'gamma':
0.5265736972326339, 'z_dim': 16.0, 'dd_dim': 1.0, 'embed_dim': 64.0, '
grad_clip': 4.299662885523541, 'batch_size': 32.0, 'hidden_dim':
128.0, 'normalizer': 'none', 'decoder_type': 'stepwise_latent_rnn', '
pooling_method': 'gridding_rel_dist', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': 1000000.0, 'hidden_dim_future': 128.0, '
lambda_importance': 1.1376385441900638, 'layer_hidden_init': '
deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': '
rel_dist', 'pooling_mechanism': 'gridding', 'gridding_divisions': 4.0,
 'layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 9.0, 'neighbourhood_width': 7.692591420821943, '
num_state_embedding': 16.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 16.0, '
seperate_interaction': 0.0, 'metric_representation': 'velocity', '
target_representation': 'acceleration', 'layer_embedding_querry': '
deep_linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding
': 64.0, 'neighbourhood_height_back': 15.767476341792984, '
num_interaction_embedding': 32.0, 'input_representation_state': 'a', '
layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 13.637952338768024, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
0.2, 'dropout_interaction_embedding': 0.1, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'db', 'num_units_attributes_embedding': 128.0, '
activation_attributes_embedding': 'relu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 5.0, '
num_units_interaction_embedding': 128.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 1.0}
{'lr': 0.0003350377920122411, 'beta': 24.87004153547444, 'gamma':
0.7232060810622909, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 128.0,
```

'grad_clip': 4.449465749614174, 'batch_size': 256.0, 'hidden_dim':
32.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'attention_rel_acc', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'forwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
lambda_importance': 9.25266580391018, 'layer_hidden_init': '
deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': '
rel_acc', 'pooling_mechanism': 'attention', 'gridding_divisions':
16.0, 'layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 19.0, 'neighbourhood_width': 6.23914563724863, '
num_state_embedding': 64.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
seperate_interaction': 0.0, 'metric_representation': 'velocity', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding':
32.0, 'neighbourhood_height_back': 12.6992986473658, '
num_interaction_embedding': 256.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 1.7393222681545977, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
0.2, 'dropout_interaction_embedding': 0.2, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'a', 'num_units_attributes_embedding': 512.0, '
activation_attributes_embedding': 'elu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 3.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'elu', '
num_layers_interaction_embedding': 5.0}

NBA :

{'lr': 0.001871284874821101, 'beta': 10.629943437807665, 'gamma':
0.6949478174063634, 'z_dim': 16.0, 'dd_dim': 2.0, 'embed_dim': 128.0,
'grad_clip': 1.2189471957365376, 'batch_size': 512.0, 'hidden_dim':
128.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'sumpool_hidden', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'forwards', '
not_present_fill': 1000000.0, 'hidden_dim_future': 64.0, '
lambda_importance': 3.62889353970355, 'layer_hidden_init': 'linear', '
layer_hidden_step': 'lstm', 'pooling_attribute': 'hidden', '
pooling_mechanism': 'sumpool', 'gridding_divisions': 64.0, '
layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
n_train_predictions': 8.0, 'neighbourhood_width': 17.075128893810508,
'num_state_embedding': 64.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 1.0, 'num_hidden_embedding': 32.0, '

```
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 6.0, 'num_rel_state_embedding
   ': 512.0, 'neighbourhood_height_back': 9.471169204745804, '
   num_interaction_embedding': 512.0, 'input_representation_state': 'a',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 10.456941587630393, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
   0.1, 'dropout_interaction_embedding': 0.2, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pva', 'num_units_attributes_embedding': 16.0, '
   activation_attributes_embedding': 'elu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 2.0, '
   num_units_interaction_embedding': 16.0, '
   activation_interaction_embedding': 'relu6', '
   num_layers_interaction_embedding': 4.0}
{'lr': 0.0005485335156951346, 'beta': 5.250345525933247, 'gamma':
   0.5070119446339979, 'z_dim': 256.0, 'dd_dim': 1.0, 'embed_dim': 256.0,
    'grad_clip': 1.3590890123589998, 'batch_size': 64.0, 'hidden_dim':
   256.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_rel_acc', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 1000000.0, 'hidden_dim_future': 512.0, '
   lambda_importance': 8.79145098470304, 'layer_hidden_init': 'linear', '
   layer_hidden_step': 'lstm', 'pooling_attribute': 'rel_acc', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 16.0, '
   layer_step_decoder': 'gru', 'num_state_features': 4.0, '
   n_train_predictions': 16.0, 'neighbourhood_width': 9.191567277177944,
   'num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding
   ': 128.0, 'neighbourhood_height_back': 15.59359098885803, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'pv',
    'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 11.381529560252728, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
   0.0, 'dropout_interaction_embedding': 0.3, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pv', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 3.0, '
```

num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 4.0}
{'lr': 0.001733731734610129, 'beta': 5.6934429843431245, 'gamma':
0.15411676672344485, 'z_dim': 128.0, 'dd_dim': 1.0, 'embed_dim':
256.0, 'grad_clip': 3.526728887140914, 'batch_size': 64.0, 'hidden_dim
': 64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'gridding_direct', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': 0.0, 'hidden_dim_future': 32.0, 'lambda_importance
': 1.7295968180764674, 'layer_hidden_init': 'deep_linear', '
layer_hidden_step': 'gru', 'pooling_attribute': 'direct', '
pooling_mechanism': 'gridding', 'gridding_divisions': 64.0, '
layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 12.0, 'neighbourhood_width': 21.150008752776, '
num_state_embedding': 64.0, 'explicit_predictions': 0.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 32.0, '
seperate_interaction': 0.0, 'metric_representation': 'acceleration', '
target_representation': 'acceleration', 'layer_embedding_querry': '
deep_linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding
': 512.0, 'neighbourhood_height_back': 13.89266510897011, '
num_interaction_embedding': 256.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 9.792813381627402, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
0.1, 'dropout_interaction_embedding': 0.0, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'va', 'num_units_attributes_embedding': 64.0, '
activation_attributes_embedding': 'elu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 5.0, '
num_units_interaction_embedding': 32.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 1.0}
{'lr': 0.0007243026341478893, 'beta': 49.51575987055742, 'gamma':
0.4122696621369472, 'z_dim': 128.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
 'grad_clip': 1.6695702218093782, 'batch_size': 32.0, 'hidden_dim':
64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'maxpool_direct', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
lambda_importance': 9.198549771058037, 'layer_hidden_init': '
deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': 'direct
', 'pooling_mechanism': 'maxpool', 'gridding_divisions': 32.0, '
layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
n_train_predictions': 13.0, 'neighbourhood_width': 26.27988407920095,

'num_state_embedding': 256.0, 'explicit_predictions': 0.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
seperate_interaction': 0.0, 'metric_representation': 'velocity', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding':
512.0, 'neighbourhood_height_back': 6.286021204608768, '
num_interaction_embedding': 256.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 8.890878310063822, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
0.5, 'dropout_interaction_embedding': 0.3, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'v', 'num_units_attributes_embedding': 256.0, '
activation_attributes_embedding': 'leaky_relu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 2.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 4.0}
{'lr': 0.00012564682513588392, 'beta': 50.07895194925274, 'gamma':
0.16488176095282453, 'z_dim': 256.0, 'dd_dim': 2.0, 'embed_dim': 32.0,
'grad_clip': 0.1613073103362389, 'batch_size': 128.0, 'hidden_dim':
32.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'concat_rel_acc', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
lambda_importance': 3.0623938423960624, 'layer_hidden_init': 'linear',
'layer_hidden_step': 'gru', 'pooling_attribute': 'rel_acc', '
pooling_mechanism': 'concat', 'gridding_divisions': 64.0, '
layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 7.0, 'neighbourhood_width': 25.65988347121772, '
num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 1.0, 'num_hidden_embedding': 64.0, '
seperate_interaction': 0.0, 'metric_representation': 'position', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 1.0, 'num_rel_state_embedding':
256.0, 'neighbourhood_height_back': 15.33109788084968, '
num_interaction_embedding': 128.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 14.420131737031737, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
0.0, 'dropout_interaction_embedding': 0.0, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'b', 'num_units_attributes_embedding': 32.0, '
activation_attributes_embedding': 'relu6', '

batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 5.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'sigmoid', '
num_layers_interaction_embedding': 2.0}
{'lr': 0.0009849023776411182, 'beta': 30.0385402041938, 'gamma':
0.28075874198252565, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
'grad_clip': 3.643101627463447, 'batch_size': 128.0, 'hidden_dim':
256.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'sumpool_hidden', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
lambda_importance': 7.564493274359233, 'layer_hidden_init': 'linear',
'layer_hidden_step': 'gru', 'pooling_attribute': 'hidden', '
pooling_mechanism': 'sumpool', 'gridding_divisions': 64.0, '
layer_step_decoder': 'gru', 'num_state_features': 2.0, '
n_train_predictions': 5.0, 'neighbourhood_width': 6.009149142999294, '
num_state_embedding': 64.0, 'explicit_predictions': 0.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 256.0, '
seperate_interaction': 0.0, 'metric_representation': 'velocity', '
target_representation': 'acceleration', 'layer_embedding_querry': '
linear', 'num_rel_state_features': 3.0, 'num_rel_state_embedding':
32.0, 'neighbourhood_height_back': 11.462160531997329, '
num_interaction_embedding': 256.0, 'input_representation_state': 'a',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 4.614985495199063, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 128.0, 'dropout_attributes_embedding':
0.5, 'dropout_interaction_embedding': 0.4, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'dbm', 'num_units_attributes_embedding': 256.0, '
activation_attributes_embedding': 'leaky_relu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 2.0, '
num_units_interaction_embedding': 128.0, '
activation_interaction_embedding': 'relu', '
num_layers_interaction_embedding': 2.0}
{'lr': 0.0012383128479368905, 'beta': 2.462578522241241, 'gamma':
0.8825587980631394, 'z_dim': 128.0, 'dd_dim': 1.0, 'embed_dim': 256.0,
'grad_clip': 1.6841935506475347, 'batch_size': 256.0, 'hidden_dim':
16.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
pooling_method': 'maxpool_direct', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'backwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
lambda_importance': 6.563668830665317, 'layer_hidden_init': 'linear',
'layer_hidden_step': 'lstm', 'pooling_attribute': 'direct', '
pooling_mechanism': 'maxpool', 'gridding_divisions': 32.0, '

```
    layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
    n_train_predictions': 8.0, 'neighbourhood_width': 2.1138183092631464,
    'num_state_embedding': 128.0, 'explicit_predictions': 1.0, '
    future_use_neighbour': 1.0, 'num_hidden_embedding': 16.0, '
    seperate_interaction': 0.0, 'metric_representation': 'position', '
    target_representation': 'acceleration', 'layer_embedding_querry': '
    deep_linear', 'num_rel_state_features': 3.0, 'num_rel_state_embedding
    ': 32.0, 'neighbourhood_height_back': 2.2229321214993933, '
    num_interaction_embedding': 64.0, 'input_representation_state': 'a', '
    layer_attributes_embedding': 'custom_linear', '
    neighbourhood_height_front': 13.300122026542429, '
    layer_interaction_embedding': 'custom_linear', '
    num_hidden_future_embedding': 128.0, 'dropout_attributes_embedding':
    0.4, 'dropout_interaction_embedding': 0.1, '
    batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
    ': 'dbm', 'num_units_attributes_embedding': 32.0, '
    activation_attributes_embedding': 'tanh', '
    batchnorm_interaction_embedding': 0.0, '
    num_layers_attributes_embedding': 2.0, '
    num_units_interaction_embedding': 256.0, '
    activation_interaction_embedding': 'sigmoid', '
    num_layers_interaction_embedding': 5.0}
{'lr': 0.0006912167047430998, 'beta': 13.870549759519514, 'gamma':
    0.7349153630288328, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 512.0,
    'grad_clip': 1.6372606478740337, 'batch_size': 128.0, 'hidden_dim':
    16.0, 'normalizer': 'none', 'decoder_type': 'stepwise_rnn', '
    pooling_method': 'sumpool_direct', 'metric_is_polar': 0.0, '
    target_is_polar': 0.0, 'future_direction': 'backwards', '
    not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
    lambda_importance': 4.815990416640407, 'layer_hidden_init': '
    deep_linear', 'layer_hidden_step': 'lstm', 'pooling_attribute': '
    direct', 'pooling_mechanism': 'sumpool', 'gridding_divisions': 32.0, '
    layer_step_decoder': 'gru', 'num_state_features': 4.0, '
    n_train_predictions': 4.0, 'neighbourhood_width': 9.022686481496356, '
    num_state_embedding': 128.0, 'explicit_predictions': 0.0, '
    future_use_neighbour': 1.0, 'num_hidden_embedding': 128.0, '
    seperate_interaction': 0.0, 'metric_representation': 'acceleration', '
    target_representation': 'acceleration', 'layer_embedding_querry': '
    deep_linear', 'num_rel_state_features': 1.0, 'num_rel_state_embedding
    ': 512.0, 'neighbourhood_height_back': 7.960384888347766, '
    num_interaction_embedding': 128.0, 'input_representation_state': 'va',
     'layer_attributes_embedding': 'custom_linear', '
    neighbourhood_height_front': 7.057854177205689, '
    layer_interaction_embedding': 'custom_linear', '
    num_hidden_future_embedding': 512.0, 'dropout_attributes_embedding':
    0.3, 'dropout_interaction_embedding': 0.4, '
    batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
```

```
    ': 'm', 'num_units_attributes_embedding': 512.0, '
    activation_attributes_embedding': 'tanh', '
    batchnorm_interaction_embedding': 0.0, '
    num_layers_attributes_embedding': 3.0, '
    num_units_interaction_embedding': 64.0, '
    activation_interaction_embedding': 'sigmoid', '
    num_layers_interaction_embedding': 2.0}
{'lr': 0.00282162953278939, 'beta': 46.13907961289852, 'gamma':
    0.5265736972326339, 'z_dim': 16.0, 'dd_dim': 1.0, 'embed_dim': 64.0, '
    grad_clip': 4.299662885523541, 'batch_size': 32.0, 'hidden_dim':
    128.0, 'normalizer': 'none', 'decoder_type': 'stepwise_latent_rnn', '
    pooling_method': 'gridding_rel_dist', 'metric_is_polar': 0.0, '
    target_is_polar': 0.0, 'future_direction': 'backwards', '
    not_present_fill': 1000000.0, 'hidden_dim_future': 128.0, '
    lambda_importance': 1.1376385441900638, 'layer_hidden_init': '
    deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': '
    rel_dist', 'pooling_mechanism': 'gridding', 'gridding_divisions': 4.0,
     'layer_step_decoder': 'gru', 'num_state_features': 2.0, '
    n_train_predictions': 9.0, 'neighbourhood_width': 7.692591420821943, '
    num_state_embedding': 16.0, 'explicit_predictions': 1.0, '
    future_use_neighbour': 0.0, 'num_hidden_embedding': 16.0, '
    seperate_interaction': 0.0, 'metric_representation': 'velocity', '
    target_representation': 'acceleration', 'layer_embedding_querry': '
    deep_linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding
    ': 64.0, 'neighbourhood_height_back': 15.767476341792984, '
    num_interaction_embedding': 32.0, 'input_representation_state': 'a', '
    layer_attributes_embedding': 'custom_linear', '
    neighbourhood_height_front': 13.637952338768024, '
    layer_interaction_embedding': 'custom_linear', '
    num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
    0.2, 'dropout_interaction_embedding': 0.1, '
    batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
    ': 'db', 'num_units_attributes_embedding': 128.0, '
    activation_attributes_embedding': 'relu', '
    batchnorm_interaction_embedding': 0.0, '
    num_layers_attributes_embedding': 5.0, '
    num_units_interaction_embedding': 128.0, '
    activation_interaction_embedding': 'sigmoid', '
    num_layers_interaction_embedding': 1.0}
{'lr': 0.0003350377920122411, 'beta': 24.87004153547444, 'gamma':
    0.7232060810622909, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 128.0,
    'grad_clip': 4.449465749614174, 'batch_size': 256.0, 'hidden_dim':
    32.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
    pooling_method': 'attention_rel_acc', 'metric_is_polar': 0.0, '
    target_is_polar': 0.0, 'future_direction': 'forwards', '
    not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
    lambda_importance': 9.25266580391018, 'layer_hidden_init': '
```

```
   deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': '
   rel_acc', 'pooling_mechanism': 'attention', 'gridding_divisions':
   16.0, 'layer_step_decoder': 'gru', 'num_state_features': 2.0, '
   n_train_predictions': 19.0, 'neighbourhood_width': 6.23914563724863, '
   num_state_embedding': 64.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 128.0, '
   seperate_interaction': 0.0, 'metric_representation': 'velocity', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding':
   32.0, 'neighbourhood_height_back': 12.6992986473658, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'a',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 1.7393222681545977, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
   0.2, 'dropout_interaction_embedding': 0.2, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'a', 'num_units_attributes_embedding': 512.0, '
   activation_attributes_embedding': 'elu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 3.0, '
   num_units_interaction_embedding': 256.0, '
   activation_interaction_embedding': 'elu', '
   num_layers_interaction_embedding': 5.0}
{'lr': 5.003116235175852e-05, 'beta': 76.46229511412517, 'gamma':
   0.29966434764739824, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 64.0,
   'grad_clip': 1.8812745476626016, 'batch_size': 32.0, 'hidden_dim':
   128.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'maxpool_proxy', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'forwards', '
   not_present_fill': -1000000.0, 'hidden_dim_future': 64.0, '
   lambda_importance': 5.773556207324463, 'layer_hidden_init': '
   deep_linear', 'layer_hidden_step': 'lstm', 'pooling_attribute': 'proxy
   ', 'pooling_mechanism': 'maxpool', 'gridding_divisions': 64.0, '
   layer_step_decoder': 'lstm', 'num_state_features': 6.0, '
   n_train_predictions': 6.0, 'neighbourhood_width': 16.78066998369117, '
   num_state_embedding': 256.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 0.0, 'num_hidden_embedding': 256.0, '
   seperate_interaction': 0.0, 'metric_representation': 'velocity', '
   target_representation': 'velocity', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 6.0, 'num_rel_state_embedding
   ': 16.0, 'neighbourhood_height_back': 5.450585946506681, '
   num_interaction_embedding': 512.0, 'input_representation_state': 'pva
   ', 'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 1.260724107286125, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 512.0, 'dropout_attributes_embedding':
```

```
   0.2, 'dropout_interaction_embedding': 0.5, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pva', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 5.0, '
   num_units_interaction_embedding': 32.0, '
   activation_interaction_embedding': 'relu', '
   num_layers_interaction_embedding': 3.0}
{'lr': 0.0003701595829194 56, 'beta': 10.823624849500655, 'gamma':
   0.3827151945980989, 'z_dim': 16.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
   'grad_clip': 4.454783812664381, 'batch_size': 64.0, 'hidden_dim':
   64.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_proxy', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'forwards', '
   not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
   lambda_importance': 1.2017653635234251, 'layer_hidden_init': 'linear',
    'layer_hidden_step': 'lstm', 'pooling_attribute': 'proxy', '
   pooling_mechanism': 'sumpool', 'gridding_divisions': 16.0, '
   layer_step_decoder': 'lstm', 'num_state_features': 4.0, '
   n_train_predictions': 9.0, 'neighbourhood_width': 28.85926079161925, '
   num_state_embedding': 512.0, 'explicit_predictions': 0.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 32.0, '
   seperate_interaction': 0.0, 'metric_representation': 'position', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding
   ': 32.0, 'neighbourhood_height_back': 12.354547294915404, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'va',
    'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 13.638442092641183, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 32.0, 'dropout_attributes_embedding':
   0.5, 'dropout_interaction_embedding': 0.4, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'a', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'relu', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 4.0, '
   num_units_interaction_embedding': 256.0, '
   activation_interaction_embedding': 'relu', '
   num_layers_interaction_embedding': 2.0}
{'lr': 0.0028974507054689 12, 'beta': 75.17653774541392, 'gamma':
   0.6126137887436076, 'z_dim': 256.0, 'dd_dim': 2.0, 'embed_dim': 16.0,
   'grad_clip': 1.1791225757286417, 'batch_size': 16.0, 'hidden_dim':
   32.0, 'normalizer': 'none', 'decoder_type': 'physics', 'pooling_method
   ': 'sumpool_direct', 'metric_is_polar': 0.0, 'target_is_polar': 0.0, '
   future_direction': 'backwards', 'not_present_fill': -1000000.0, '
```

```
   hidden_dim_future': 512.0, 'lambda_importance': 4.013541394899489, '
   layer_hidden_init': 'deep_linear', 'layer_hidden_step': 'gru', '
   pooling_attribute': 'direct', 'pooling_mechanism': 'sumpool', '
   gridding_divisions': 64.0, 'layer_step_decoder': 'gru', '
   num_state_features': 4.0, 'n_train_predictions': 9.0, '
   neighbourhood_width': 19.29697963820832, 'num_state_embedding': 128.0,
    'explicit_predictions': 0.0, 'future_use_neighbour': 0.0, '
   num_hidden_embedding': 256.0, 'seperate_interaction': 0.0, '
   metric_representation': 'position', 'target_representation': 'velocity
   ', 'layer_embedding_querry': 'deep_linear', 'num_rel_state_features':
   2.0, 'num_rel_state_embedding': 64.0, 'neighbourhood_height_back':
   9.641398755622053, 'num_interaction_embedding': 32.0, '
   input_representation_state': 'pv', 'layer_attributes_embedding': '
   custom_linear', 'neighbourhood_height_front': 14.7672981412385, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
   0.4, 'dropout_interaction_embedding': 0.1, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'v', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 2.0, '
   num_units_interaction_embedding': 32.0, '
   activation_interaction_embedding': 'elu', '
   num_layers_interaction_embedding': 4.0}
{'lr': 0.00011006232009122806, 'beta': 84.3090602459455, 'gamma':
   0.828503296745276, 'z_dim': 256.0, 'dd_dim': 2.0, 'embed_dim': 512.0,
   'grad_clip': 1.24774261067126, 'batch_size': 16.0, 'hidden_dim':
   128.0, 'normalizer': 'none', 'decoder_type': 'direct_rnn', '
   pooling_method': 'sumpool_rel_acc', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 1000000.0, 'hidden_dim_future': 256.0, '
   lambda_importance': 8.433254408816271, 'layer_hidden_init': '
   deep_linear', 'layer_hidden_step': 'gru', 'pooling_attribute': '
   rel_acc', 'pooling_mechanism': 'sumpool', 'gridding_divisions': 32.0,
   'layer_step_decoder': 'gru', 'num_state_features': 2.0, '
   n_train_predictions': 17.0, 'neighbourhood_width': 3.061300813200284,
   'num_state_embedding': 512.0, 'explicit_predictions': 1.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 128.0, '
   seperate_interaction': 0.0, 'metric_representation': 'velocity', '
   target_representation': 'acceleration', 'layer_embedding_querry': '
   linear', 'num_rel_state_features': 4.0, 'num_rel_state_embedding':
   256.0, 'neighbourhood_height_back': 9.863578946786957, '
   num_interaction_embedding': 256.0, 'input_representation_state': 'p',
   'layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 13.30340746965446, '
   layer_interaction_embedding': 'custom_linear', '
```

num_hidden_future_embedding': 256.0, 'dropout_attributes_embedding':
0.5, 'dropout_interaction_embedding': 0.0, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'pv', 'num_units_attributes_embedding': 128.0, '
activation_attributes_embedding': 'sigmoid', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 4.0, '
num_units_interaction_embedding': 64.0, '
activation_interaction_embedding': 'relu6', '
num_layers_interaction_embedding': 2.0}
{'lr': 0.00031955143583573106, 'beta': 55.7764372356832, 'gamma':
0.973939605868919, 'z_dim': 64.0, 'dd_dim': 2.0, 'embed_dim': 64.0, '
grad_clip': 2.378743716578445, 'batch_size': 16.0, 'hidden_dim':
128.0, 'normalizer': 'none', 'decoder_type': 'stepwise_rnn', '
pooling_method': 'sumpool_rel_acc', 'metric_is_polar': 0.0, '
target_is_polar': 0.0, 'future_direction': 'forwards', '
not_present_fill': -1000000.0, 'hidden_dim_future': 16.0, '
lambda_importance': 5.135268935204726, 'layer_hidden_init': 'linear',
'layer_hidden_step': 'gru', 'pooling_attribute': 'rel_acc', '
pooling_mechanism': 'sumpool', 'gridding_divisions': 4.0, '
layer_step_decoder': 'gru', 'num_state_features': 4.0, '
n_train_predictions': 17.0, 'neighbourhood_width': 15.876558845674875,
'num_state_embedding': 256.0, 'explicit_predictions': 1.0, '
future_use_neighbour': 0.0, 'num_hidden_embedding': 256.0, '
seperate_interaction': 0.0, 'metric_representation': 'velocity', '
target_representation': 'velocity', 'layer_embedding_querry': '
deep_linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding
': 256.0, 'neighbourhood_height_back': 5.401240257359211, '
num_interaction_embedding': 128.0, 'input_representation_state': 'va',
'layer_attributes_embedding': 'custom_linear', '
neighbourhood_height_front': 11.833233487627249, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 128.0, 'dropout_attributes_embedding':
0.3, 'dropout_interaction_embedding': 0.4, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'a', 'num_units_attributes_embedding': 256.0, '
activation_attributes_embedding': 'relu6', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 4.0, '
num_units_interaction_embedding': 256.0, '
activation_interaction_embedding': 'relu', '
num_layers_interaction_embedding': 4.0}

ETH,Hotel,Zara1,Zara2,UNIV :

{'lr': 0.0001800228519791787, 'beta': 54.12278285548539, 'gamma':
0.9964492297093844, 'z_dim': 512.0, 'dd_dim': 2.0, 'embed_dim':

```
      128.0, 'grad_clip': 1.725972658708054, 'batch_size': 32.0, '
      hidden_dim': 16.0, 'normalizer': 'none', 'decoder_type': 'physics
      ', 'pooling_method': 'gridding_direct', 'metric_is_polar': 0.0, '
      target_is_polar': 0.0, 'future_direction': 'backwards', '
      not_present_fill': -1000000.0, 'hidden_dim_future': 256.0, '
      lambda_importance': 9.231358185121133, 'layer_hidden_init': '
      linear', 'layer_hidden_step': 'gru', 'pooling_attribute': 'direct
      ', 'pooling_mechanism': 'gridding', 'gridding_divisions': 32.0, '
      layer_step_decoder': 'lstm', 'num_state_features': 4.0, '
      n_train_predictions': 10.0, 'neighbourhood_width':
      23.247259254450473, 'num_state_embedding': 16.0, '
      explicit_predictions': 1.0, 'future_use_neighbour': 1.0, '
      num_hidden_embedding': 64.0, 'seperate_interaction': 0.0, '
      metric_representation': 'position', 'target_representation': '
      velocity', 'layer_embedding_querry': 'linear', '
      num_rel_state_features': 2.0, 'num_rel_state_embedding': 16.0, '
      neighbourhood_height_back': 12.423163843924597, '
      num_interaction_embedding': 256.0, 'input_representation_state': '
      pv', 'layer_attributes_embedding': 'custom_linear', '
      neighbourhood_height_front': 1.9072409754882504, '
      layer_interaction_embedding': 'custom_linear', '
      num_hidden_future_embedding': 256.0, 'dropout_attributes_embedding
      ': 0.1, 'dropout_interaction_embedding': 0.1, '
      batchnorm_attributes_embedding': 0.0, '
      input_representation_rel_state': 'p', '
      num_units_attributes_embedding': 32.0, '
      activation_attributes_embedding': 'tanh', '
      batchnorm_interaction_embedding': 0.0, '
      num_layers_attributes_embedding': 5.0, '
      num_units_interaction_embedding': 512.0, '
      activation_interaction_embedding': 'relu6', '
      num_layers_interaction_embedding': 2.0}
{'lr': 0.0009032429519430018, 'beta': 57.5818234268588, 'gamma':
   0.4672615133854059, 'z_dim': 512.0, 'dd_dim': 1.0, 'embed_dim': 128.0,
    'grad_clip': 3.992368789542757, 'batch_size': 256.0, 'hidden_dim':
   16.0, 'normalizer': 'none', 'decoder_type': 'physics', 'pooling_method
   ': 'gridding_rel_acc', 'metric_is_polar': 0.0, 'target_is_polar': 0.0,
    'future_direction': 'backwards', 'not_present_fill': 1000000.0, '
   hidden_dim_future': 32.0, 'lambda_importance': 3.673604292838199, '
   layer_hidden_init': 'deep_linear', 'layer_hidden_step': 'gru', '
   pooling_attribute': 'rel_acc', 'pooling_mechanism': 'gridding', '
   gridding_divisions': 16.0, 'layer_step_decoder': 'lstm', '
   num_state_features': 4.0, 'n_train_predictions': 17.0, '
   neighbourhood_width': 3.61981864251481, 'num_state_embedding': 512.0,
   'explicit_predictions': 1.0, 'future_use_neighbour': 1.0, '
   num_hidden_embedding': 32.0, 'seperate_interaction': 0.0, '
   metric_representation': 'velocity', 'target_representation': 'velocity
```

```
   ', 'layer_embedding_querry': 'deep_linear', 'num_rel_state_features':
   4.0, 'num_rel_state_embedding': 128.0, 'neighbourhood_height_back':
   6.809267306781644, 'num_interaction_embedding': 256.0, '
   input_representation_state': 'pv', 'layer_attributes_embedding': '
   custom_linear', 'neighbourhood_height_front': 1.2786971914559184, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
   0.4, 'dropout_interaction_embedding': 0.0, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'pa', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'tanh', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 3.0, '
   num_units_interaction_embedding': 64.0, '
   activation_interaction_embedding': 'tanh', '
   num_layers_interaction_embedding': 1.0}
{'lr': 0.00012836192304400713, 'beta': 24.22825654979371, 'gamma':
   0.2911909856469384, 'z_dim': 256.0, 'dd_dim': 2.0, 'embed_dim': 32.0,
   'grad_clip': 0.9757009902026637, 'batch_size': 64.0, 'hidden_dim':
   128.0, 'normalizer': 'none', 'decoder_type': 'physics', '
   pooling_method': 'maxpool_rel_dist', 'metric_is_polar': 0.0, '
   target_is_polar': 0.0, 'future_direction': 'backwards', '
   not_present_fill': 1000000.0, 'hidden_dim_future': 16.0, '
   lambda_importance': 0.21853923525714797, 'layer_hidden_init': 'linear
   ', 'layer_hidden_step': 'gru', 'pooling_attribute': 'rel_dist', '
   pooling_mechanism': 'maxpool', 'gridding_divisions': 32.0, '
   layer_step_decoder': 'lstm', 'num_state_features': 2.0, '
   n_train_predictions': 13.0, 'neighbourhood_width': 15.262069789237444,
    'num_state_embedding': 128.0, 'explicit_predictions': 0.0, '
   future_use_neighbour': 1.0, 'num_hidden_embedding': 16.0, '
   seperate_interaction': 0.0, 'metric_representation': 'velocity', '
   target_representation': 'velocity', 'layer_embedding_querry': '
   deep_linear', 'num_rel_state_features': 2.0, 'num_rel_state_embedding
   ': 64.0, 'neighbourhood_height_back': 3.358019053938518, '
   num_interaction_embedding': 16.0, 'input_representation_state': 'v', '
   layer_attributes_embedding': 'custom_linear', '
   neighbourhood_height_front': 10.633442693407336, '
   layer_interaction_embedding': 'custom_linear', '
   num_hidden_future_embedding': 16.0, 'dropout_attributes_embedding':
   0.4, 'dropout_interaction_embedding': 0.0, '
   batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
   ': 'db', 'num_units_attributes_embedding': 128.0, '
   activation_attributes_embedding': 'sigmoid', '
   batchnorm_interaction_embedding': 0.0, '
   num_layers_attributes_embedding': 1.0, '
   num_units_interaction_embedding': 512.0, '
   activation_interaction_embedding': 'elu', '
```

num_layers_interaction_embedding': 3.0}
{'lr': 0.0008407870842467528, 'beta': 62.0771117466555, 'gamma':
0.1286660076081022, 'z_dim': 16.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
'grad_clip': 3.090252383329071, 'batch_size': 256.0, 'hidden_dim':
16.0, 'normalizer': 'none', 'decoder_type': 'physics', 'pooling_method
': 'maxpool_rel_vel', 'metric_is_polar': 0.0, 'target_is_polar': 0.0,
'future_direction': 'forwards', 'not_present_fill': 0.0, '
hidden_dim_future': 512.0, 'lambda_importance': 9.964529731773396, '
layer_hidden_init': 'deep_linear', 'layer_hidden_step': 'gru', '
pooling_attribute': 'rel_vel', 'pooling_mechanism': 'maxpool', '
gridding_divisions': 4.0, 'layer_step_decoder': 'lstm', '
num_state_features': 4.0, 'n_train_predictions': 20.0, '
neighbourhood_width': 19.94206767269679, 'num_state_embedding': 16.0,
'explicit_predictions': 0.0, 'future_use_neighbour': 0.0, '
num_hidden_embedding': 128.0, 'seperate_interaction': 0.0, '
metric_representation': 'velocity', 'target_representation': 'velocity
', 'layer_embedding_querry': 'deep_linear', 'num_rel_state_features':
1.0, 'num_rel_state_embedding': 16.0, 'neighbourhood_height_back':
5.332332023991855, 'num_interaction_embedding': 256.0, '
input_representation_state': 'va', 'layer_attributes_embedding': '
custom_linear', 'neighbourhood_height_front': 4.691796266029817, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 128.0, 'dropout_attributes_embedding':
0.3, 'dropout_interaction_embedding': 0.2, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'b', 'num_units_attributes_embedding': 128.0, '
activation_attributes_embedding': 'sigmoid', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 5.0, '
num_units_interaction_embedding': 32.0, '
activation_interaction_embedding': 'elu', '
num_layers_interaction_embedding': 4.0}
{'lr': 0.0014228671772788368, 'beta': 54.09287170526349, 'gamma':
0.9191417073768892, 'z_dim': 32.0, 'dd_dim': 2.0, 'embed_dim': 256.0,
'grad_clip': 2.3660076254032028, 'batch_size': 256.0, 'hidden_dim':
64.0, 'normalizer': 'none', 'decoder_type': 'physics', 'pooling_method
': 'gridding_direct', 'metric_is_polar': 0.0, 'target_is_polar': 0.0,
'future_direction': 'forwards', 'not_present_fill': -1000000.0, '
hidden_dim_future': 16.0, 'lambda_importance': 2.6042251797828273, '
layer_hidden_init': 'deep_linear', 'layer_hidden_step': 'gru', '
pooling_attribute': 'direct', 'pooling_mechanism': 'gridding', '
gridding_divisions': 32.0, 'layer_step_decoder': 'gru', '
num_state_features': 2.0, 'n_train_predictions': 9.0, '
neighbourhood_width': 6.413262767737724, 'num_state_embedding': 32.0,
'explicit_predictions': 1.0, 'future_use_neighbour': 1.0, '
num_hidden_embedding': 64.0, 'seperate_interaction': 0.0, '
metric_representation': 'velocity', 'target_representation': 'velocity

', 'layer_embedding_querry': 'linear', 'num_rel_state_features': 1.0,
'num_rel_state_embedding': 256.0, 'neighbourhood_height_back':
3.2497057832937424, 'num_interaction_embedding': 32.0, '
input_representation_state': 'a', 'layer_attributes_embedding': '
custom_linear', 'neighbourhood_height_front': 3.0128549014073203, '
layer_interaction_embedding': 'custom_linear', '
num_hidden_future_embedding': 64.0, 'dropout_attributes_embedding':
0.0, 'dropout_interaction_embedding': 0.4, '
batchnorm_attributes_embedding': 0.0, 'input_representation_rel_state
': 'd', 'num_units_attributes_embedding': 64.0, '
activation_attributes_embedding': 'leaky_relu', '
batchnorm_interaction_embedding': 0.0, '
num_layers_attributes_embedding': 2.0, '
num_units_interaction_embedding': 128.0, '
activation_interaction_embedding': 'tanh', '
num_layers_interaction_embedding': 2.0}