Multiple Trajectory Prediction of Moving Agents with Memory Augmented Networks

Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, Alberto Del Bimbo

Abstract—Pedestrians and drivers are expected to safely navigate complex urban environments along with several non cooperating agents. Autonomous vehicles will soon replicate this capability. Each agent acquires a representation of the world from an egocentric perspective and must make decisions ensuring safety for itself and others. This requires to predict motion patterns of observed agents for a far enough future. In this paper we propose MANTRA, a model that exploits memory augmented networks to effectively predict multiple trajectories of other agents, observed from an egocentric perspective. Our model stores observations in memory and uses trained controllers to write meaningful pattern encodings and read trajectories that are most likely to occur in future. We show that our method is able to natively perform multi-modal trajectory prediction obtaining state-of-the art results on four datasets. Moreover, thanks to the non-parametric nature of the memory module, we show how once trained our system can continuously improve by ingesting novel patterns.

Index Terms—Trajectory prediction, Memory Augmented Networks, Egocentric Perception, Autonomous Driving

1 Introduction

Sensing the surrounding environment is a key ability for reasoning. Humans are able to achieve this mostly through visual perception and adapting what they see to a representation of the world built through their own experience. Grounding what is perceived with experience, it is possible for humans to recall previously seen episodes that are likely to suggest state evolutions of other agents acting in the environment. A simple example can be found in a pedestrian about to cross the road: if a car is approaching, it might stop to let him pass, keep moving in front of him or even make a turn without crossing paths at all. Having observed similar behaviors in the past, the pedestrian will wait until a safe crossing scenario is foreseeable. Similarly, other moving agents, such as cyclists and car drivers, must apply this kind of predictive reasoning while driving.

When egocentric perception or the reasoning grounded on such evidence falls short, tasks such as interacting with other objects or people become difficult if not even dangerous, indoors and especially outdoors. In particular, to be able to safely navigate an outdoor space, such as an urban environment, it is necessary to sense its structure and understand and predict the motion of surrounding agents populating it. To this end, for visually impaired humans, wearable devices are becoming a possible aid to correctly perceive the surroundings and provide assistance for navigation. If the navigating entity is not human, but instead an intelligent agent such as a robot or an autonomous vehicle, egocentric perception covers a pivotal role since all its aspects must be explicitly modeled, from sensing to prediction. In general, a moving agent, let it be human or artificial, has to rely on egocentric perception to plan a safe navigation.

The problem of predicting trajectories of navigating agents is deeply entwined with egocentric perception and has a central importance in guaranteeing safety for both the observer and the observed. Moreover, predicting future trajectories is a problem with an inherently multimodal nature: the dynamics of a moving agent, observed from an external point of view, can yield a variety of similarly likely outcomes (Fig. 1). Being able to predict where others will go, allows the observer to take counter-actions or simply pay more attention to certain elements populating the environment. Regardless of the nature of the observer (a pedestrian, a cyclist, a person driving, an autonomous vehicle) and the means of perception (the human eye, smart helmets or wearable devices, on-vehicle ego cameras), the problem can be cast as sensing others, represent their motion in a reference system up to an instant identifiable as present and infer their position in future time-steps.

As for perception, considerable steps forward have been made in the past years for autonomous driving [1], [2], [3]. Vehicles are in fact equipped with a large array of sensors (e.g., GPS, LiDAR, RGB cameras, stereo camera rigs) to build a precise representation of what is observed from their point of view. Yet, when inferring future positions of others, current approaches still lack the ability of explicitly addressing specific occurrences from experience. This is particularly important to make accurate predictions and directly reflects onto the ability of planning actions for safe navigation.

Humans can address this task by implicit learning, i.e., exploiting procedural memory (knowing how to do things) from similar scenarios of previous experience, without explicit and conscious awareness. For machines, instead, this task is much harder. Common machine learning models, such as the LSTM variant of Recurrent Neural Networks, have been applied with some success to predict trajectories and produce probabilistic information about the future lo-

- *F. Marchetti, F.Becattini, L. Seidenari, A. Del Bimbo are with the Media Integration and Communication Center (MICC) of the University of Florence, Italy.*

- *Authors would like to thank the research team at IMRA Europe S.A.S. for the useful discussions and insights.*

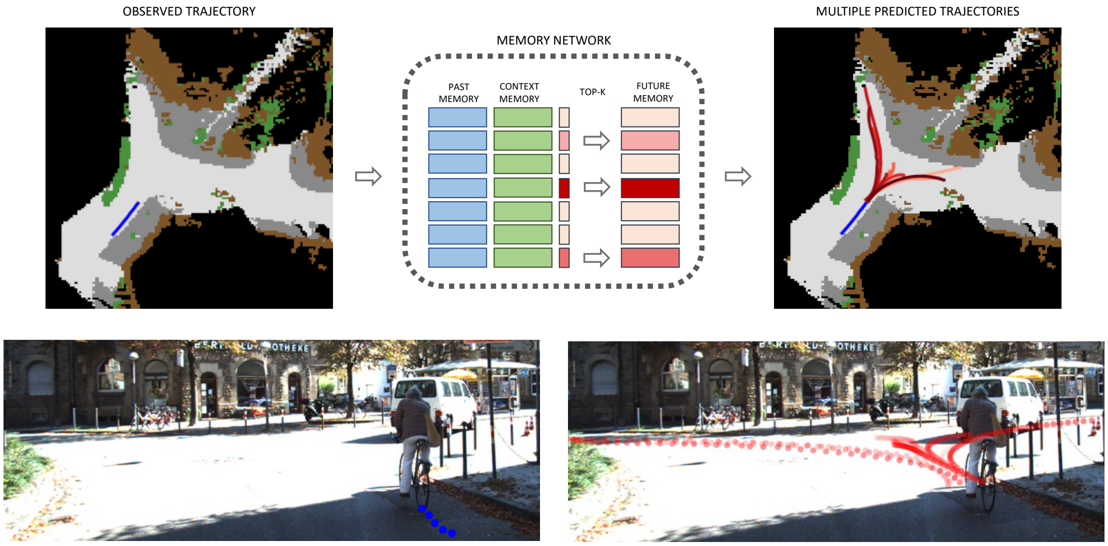


Fig. 1. MANTRA addresses multimodal trajectory prediction. We obtain multiple future predictions given an observed past and its context, relying on a Memory Augmented Neural Network.

cations of vehicles [4], [5] or pedestrians [6], [7], [8]. LSTMs have been capable of storing past information into a single hidden representation, updated at every time step, and make predictions based on long term patterns.

In this paper, we present MANTRA: a Memory Augmented Neural TRAjectory predictor. In contrast to the solutions referred above, MANTRA addresses vehicle trajectory prediction by following a novel approach and implementing a persistent Memory Augmented Neural Network (MANN) [9]. In our model, an external, associative memory is trained to write useful and non-redundant trajectories. Instead of a single hidden representation addressable as a whole, our memory is element-wise addressable and permits to selectively access only relevant pieces of information at runtime.

The model incrementally creates a knowledge base that is used as experience to perform meaningful predictions, combining information from the past dynamics of the vehicle and the environment in which it is moving. This mimics the way in which implicit human memory works. Since the knowledge base is built from trajectory samples, it can also include instances observed while the system is running, after it has been trained. In this way the model gains experience online, increasing its accuracy and capability to generalize at no training cost.

Samples are stored in memory by separating past, future and context information. In this way, at test time the actual coordinates are obtained by decoding a future read from memory, conditioned with the observed past and context. Therefore, the output is not a simple copy of previously seen examples, but is instead a newly generated trajectory obtained from both the system experience (i.e., its memory) and the observed instance. By reading multiple futures from memory, diverse meaningful predictions can be obtained. The main contributions of this paper are the following:

- We propose a novel architecture for multiple trajectory prediction of moving agents in urban environments based on Memory Augmented Neural Networks. The model is equipped with memory controllers for writing and reading only relevant samples. To the best of our knowledge we are the first to adopt MANNs for trajectory prediction.
- Our formulation, exploiting an encoder-decoder pipeline augmented with an associative memory, is easier to inspect and provides naturally multimodal predictions, obtaining state-of-the-art results on four traffic datasets.
- Our model is able to improve incrementally, after it has been trained, when observing new examples online. This trait is important for industrial automotive applications and is currently lacking in other state of the art predictors.

A preliminary version of our model was described in [10]. The model presented in this work differs substantially from [10] in several ways: (i) feature encoding now includes context instead of just trajectories; (ii) feature decoding is trained with a multi-task loss using a cross-entropy to reconstruct semantic maps in addition to the MSE to reconstruct future trajectories; (iii) memory keys are now made by tuples of past and context; (iv) memory access is done through a trainable reading controller. Furthermore, we use an additional metric, taking into account all predictions, without limiting the evaluation to the best-of-K. We also provide a more comprehensive review of related works and report an extended evaluation, including an additional new and challenging dataset [11].

2 Related Work

Egocentric Perception A large variety of applications has been studied in literature regarding egocentric perception: assistance for visually impaired people [12], [13], lifelogging [14], [15], [16], navigation [17], [18], [19], mixed reality [20], [21]. Although the nature of these applications is different, they all share the usage of a camera for capturing what an agent observes. Typically, said agent is human and uses a wearable device to get assistance in everyday life or experience an augmented or mixed reality as entertainment. Egocentric perception, though, is not limited to humans, and can focus on intelligent agents such as robots or autonomous cars. In particular, it has been playing a relevant role in aiding navigating agents, e.g. humans with visual impairment, robots or vehicles [22], [23], [24], [25]. This stems from the need to accurately perceive the surrounding environment and react accordingly. Two macro-areas can be distinguished: indoor and outdoor navigation. Indoor navigating agents must perceive objects and be able to interact with them [17], [19], [26], [27], [28], [29], [30], [31]. In fact, the final goal is usually to handle daily life activities, such as cooking or washing dishes [32], which require to correctly interact with several elements of the environment. Interaction is not limited to objects and may also involve other humans, especially during social [33], [34] or sport activities [35]. A notable subfield of research is egocentric perception for cultural heritage [36], [37], [38], where user interests and behaviors are tracked to offer enhanced museum visits.

When focus shifts to outdoor navigation, such as urban environments, egocentric perception has to be defined differently. Whereas navigation in indoor settings is a mean to an end (cooking, interacting with something/someone), the goal of outdoor agents is more focused on navigation itself, which needs to be performed safely without harming or being harmed by other navigating agents (e.g., avoiding collisions and complying with road traffic regulations) [18], [39]. Interaction is still a matter of primary importance but is now considered as the analysis of social patterns that determine how groups move together in a shared space [6], [7], [40]. At the same time, for urban navigation, agents need to infer the layout of the scene, recovering possible occlusions that they might have from their point of view [41], [42]. A correct understanding of both context and other agents is pivotal for safety, since it enables the anticipation of dangerous situations such as car accidents [43]. Several works have dealt with the problem of predicting future agent locations, often focusing on ego-motion [23], [44], [45]. In this paper, rather than focusing on ego-motion, we are interested in the ability to look at other agents from an egocentric point of view and forecast how they will act in the near future.

Trajectory Prediction Significant effort has been made in the past years regarding trajectory prediction. Several researchers have focused on trajectories of pedestrians [6], [7], [46], [47], [48], either regarded as individuals or crowds, also exploiting social behaviors and interactivity between individuals [6], [7], [46], [47], [49], [50].

For vehicle trajectory prediction, the focus shifts on the observation of motion of individual agents (their past trajectory) and the understanding of the surrounding environment [11], [51], [52]. Traffic dynamics likely reduce to simpler scenarios where movement is limited and constrained by the environment. Efforts have been made to understand and predict vehicle trajectories in urban scenarios [40], [51], [52], [53], [54], [55], [56], [57], [58], [59], also taking into account social interactions. Although, from the empirical evidence presented in [11], [51], the explicit modeling of social interactions for vehicles was shown not to provide valuable improvements in trajectory prediction. A notable exception is estimating lane changes on highways [60], [61].

Distinguished systems that provide multiple trajectory prediction in complex environments are DESIRE [51] and INFER [52]. DESIRE uses a Conditional Variational Autoencoder for estimating a distribution from which future trajectories can be sampled. A large number of predictions is needed to cover all the search space and Inverse Optimal Control is then used to extract a final ranked subset. INFER instead exploits a fully convolutional model that takes into account intermediate semantic representations and generates multimodal heatmaps of possible future locations, then looking for peaks of the distribution.

In our work, we address multiple trajectory prediction of agents navigating in an urban scenario. Examples of urban contexts where such multiple predictions may be necessary are roundabouts and crossroads where vehicles might take different, equally possible paths.

We train a Memory Augmented Neural Network model to generate multimodal trajectories, which to the best of our knowledge has never been used for this purpose. The usage of MANNs has two main advantages: (i) multiple futures can be read from memory for a given observation, making the model compliant to the multimodal nature of the problem; (ii) by retrieving a likely future from memory we can rely on an oracle that suggests what is going to happen in the near future. Differently from prior work, our trajectory prediction model is also capable of growing online, improving incrementally its performance from new observations after it has been trained.

A conceptually similar research direction to ours is the one of intention-based methods [54], [55], [56]. In these works, some representative anchor information (such as trajectories, actions or locations) are predefined and then used to guide predictions after estimating a probability distribution over each candidate. In [54], predictions on human agents are conditioned by the state of a robot agent, for which a goal is given or estimated. The authors of [55] propose a model specialized on intersections that generates a likelihood over 5 fixed map zones, which entail different motion patterns (go straight, turn left, turn right, stop and reach the middle of the intersection). These anchors though are very coarse and tied to a single context. In [56], anchor trajectories are created running k-means over training data and then performing uniform random sampling to reduce redundancy. To some extent, memory entries in our model can be interpreted as anchors encoding physically plausible futures instead of intentions. Differently from the described approaches though, we perform estimates based on any kind of past dynamics and road layout, without having to choose a reference agent to condition predictions or restrict the applicability to constrained scenarios. Moreover, the set of samples that we write in memory is chosen in order to explicitly take into account reconstruction error through a learned controller, thus minimizing redundancy in a principled manner.

Memory Networks Neural Networks with memory capabilities have been introduced to solve several machine learning problems which require to model a temporal dimension. The most common models are Recurrent Neural Networks (RNN) and their variants such as Long-Short Term Memories (LSTM) [62] and Gated Recurrent Units (GRU) [63]. However, in these models, memory is a single hidden state vector that encodes all the temporal information. So memory is addressable as a whole and they lack the ability to address individual elements of knowledge, necessary to apply algorithmic manipulation and rapid in-

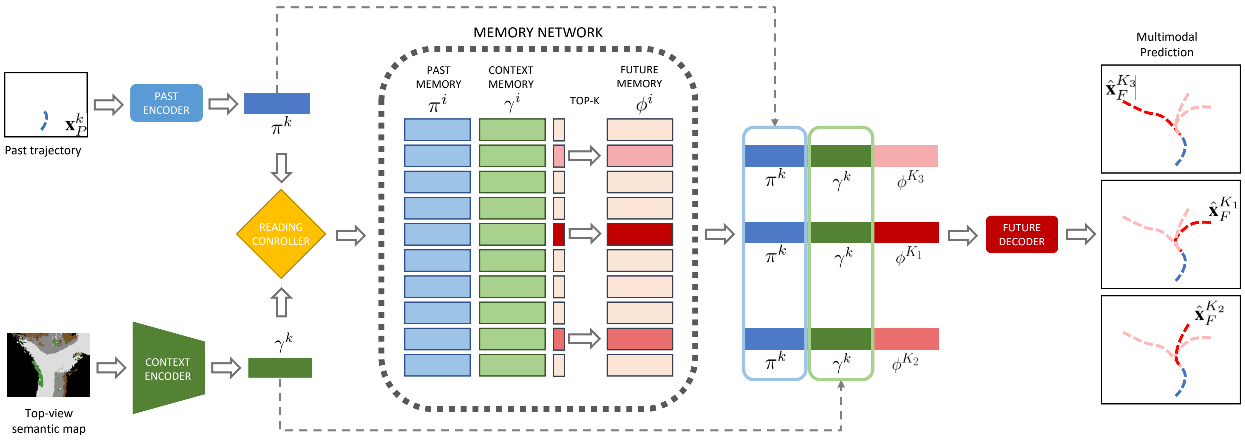


Fig. 2. Architecture of MANTRA. The encodings of an observed past trajectory and its contexts are used as key to read likely future encodings from memory. A multimodal prediction is obtained by decoding each future encoding, conditioned by the observed past and the context.

ference. Moreover, state to state transition is unstructured and global. Being the state updated at each time-step, eventually it fails to model very long term dependencies. Finally, the number of parameters is tied to the size of the hidden state. So, adding knowledge from the external environment, necessarily implies increasing the size of the state.

Recent works have proposed Memory Augmented Neural Networks, or simply Memory Networks, to overcome the limitations of RNNs [9], [64], [65], [66], [67], [68], [69], [70], [71]. The principal characteristic of these models is the usage of a controller network with an external element-wise addressable memory. This is used to store explicit information and access selectively relevant items. The memory controller is trained to dynamically manage memory content, optimizing predictions. Differently from RNNs, state to state transitions are obtained through read/write operations and a set of independent states is maintained. An important consideration is that in Memory Networks the number of parameters is not tied to the size of the memory, i.e., increasing the memory slots will not increase the number of parameters.

While introduced recently, a number of applications of these models have already appeared in literature. The first embodiment of a Memory Network was proposed in Neural Turing Machines (NTM) [9] to perform algorithmic tasks, such as sorting or copying, which require sequential manipulation steps. Thanks to a fully differentiable controller, the model interacts with the memory through read/write operations. The architecture was later extended to perform one-shot learning in [65]. Differently from NTM, they trained the MANN to implement a Least Recently Used memory access strategy to write into rarely used locations.

In [67], MANNs have been proved to be able to effectively address Question Answering tasks, where the model has to answer questions related to a series of sentences. In [66], the same problem is solved with an End-to-End Memory Network with attention weights to shift importance from one sentence to another. Recent approaches have proposed a MANN to address the more complex problem of Visual Question Answering [69], [70], training the MANN to learn uncommon question-answer pairs. Online learning has also been tackled using Memory Networks. Rebuffi *et al.* [68] learn a classifier adding classes incrementally. MANNs for object tracking have been proposed, where the model is trained to memorize templates, which are updated as the object is tracked [71].

All these MANNs rely on episodic memories. The system learns to write and read from memory, but the stored data is limited only to the current set of observations (such as a list of numbers to be sorted in [9] or a collection of sentences for question answering in [67]). Differently from prior work, we build a MANN with a memory that is not episodic. Instead, it acts like a persistent memory which stores an experience of relevant data to perform accurate predictions for any observation and not just for a restricted episode or set of samples. The rationale behind this approach is that instead of solving simple algorithmic tasks as a Neural Turing Machine, we learn how to create a pool of samples to be used for future trajectory predictions. The proposed model learns to store in memory only what is strictly needed to perform accurate predictions. Our usage of MANN is close to [72], but differs substantially. While they exploit the decoupling of embeddings to better fit data, we leverage the disjoint representation to create multiple outputs from a single input, leading to a fully multimodal predictive capability of the overall system.

3 Model

We formulate the task of predicting trajectories of moving agents as the problem of estimating $P(\hat{\mathbf{x}}_F|\mathbf{x}_P, \mathbf{c})$, where $\hat{\mathbf{x}}_F$ is the predicted future trajectory, $\mathbf{x}_P$ is the observed trajectory (or *past*) and $\mathbf{c}$ is a representation of the context (e.g., roads, sidewalks). The focus of our work lies on vehicles, but also includes other moving agents such as cyclists and pedestrians. We consider agent trajectories as a sequence of 2-dimensional spatial coordinates. The *past* $\mathbf{x}_P$ is given by its positions observed up to some reference point identified as *present*. Similarly, the *future* $\mathbf{x}_F$ is the sequence of positions in which it will find itself at the next time steps.

3.1 Memory Based Trajectory Prediction

Given a sample trajectory $\mathbf{x}^i = [\mathbf{x}_P^i, \mathbf{x}_F^i]$, let $\pi^i = \Pi(\mathbf{x}_P^i)$ and $\phi^i = \Phi(\mathbf{x}_F^i)$ be two encoding functions that map the 2D

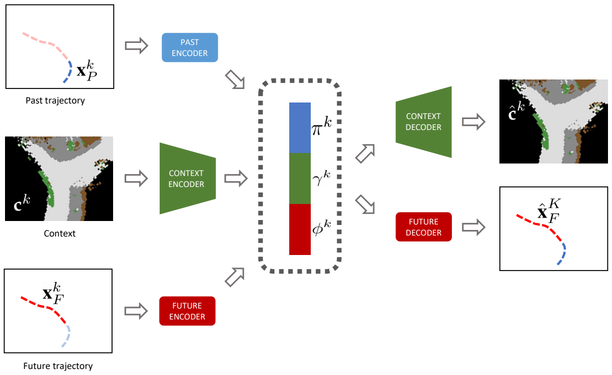


Fig. 3. Representation learning: past, context and future are encoded separately; a decoder reconstructs future trajectory and context.

coordinates of past and future trajectories into two separate latent representations. Similarly, let $\gamma^i = \Gamma(\mathbf{c}^i)$ be an encoding function that generates a latent vector representing the top-view map $\mathbf{c}^i$ of the surrounding context. Finally, let $\Psi_F(\pi^i, \gamma^i, \phi^i)$ and $\Psi_C(\pi^i, \gamma^i, \phi^i)$ be two decoding functions that take as input a triplet of past, context and future encodings. Ψ_F and Ψ_C respectively decode the input into the future sub-trajectory $\mathbf{x}_F^i$ and the context $\mathbf{c}^i$.

We define $M = \{\pi^i, \gamma^i, \phi^i\}$ as an associative key-value memory containing $|M|$ triplets of past, context and future encodings. Past and context tuples (π^i, γ^i) act as keys to access memory and future embeddings ϕ^i are the values. When a new trajectory $\mathbf{x}_P^k$ is observed, its encoding and context encoding (π^k, γ^k) are used as key to retrieve meaningful samples from memory. Note that observed trajectories are all considered to be *past* trajectories, since the future counterpart is yet to be observed and is what we want to predict. Memory addressing is performed by a reading controller that transforms a memory key into a read probability $P(r)^i$ for each stored sample. The controller is trained in order to maximize the read probability for samples that exhibit similar past and context to the observed one.

According to these similarity scores, the future encodings of the top-K elements ϕ^j are separately combined with the encodings of the observed past π^k and context γ^k. The novel triplets of encodings are transformed into 2D coordinates using the decoding function Ψ_F: $\hat{\mathbf{x}}_F^j = \Psi_F(\pi^k, \gamma^k, \phi^j)$, with $j = 1, ..., K$.

Note that π^k and γ^k are fixed while ϕ^j varies depending on the sample read from memory. Future encodings ϕ^j are used to suggest possible outcomes based on the past observation. This strategy allows the model to look ahead into likely futures in order to predict the correct one. Since multiple ϕ^j can be used independently, we can decode multiple futures and obtain a multimodal prediction in case of uncertainty (e.g., a bifurcation in the road). An overview of the model is shown in Fig. 2.

3.2 Feature Representation Learning

The encoding-decoding functions $\Pi, \Gamma, \Phi, \Psi_F, \Psi_C$ are trained jointly as an autoencoder, as shown in Fig. 3. The encoders Π and Φ learn to map past and future points into a meaningful representation and the decoder Ψ_F learns to reconstruct the future. To aid this process, we also include knowledge about the context, represented as a top-view semantic map. Instead of using just the future as input, we condition the reconstruction process also with an encoding of the past and the context. This is useful for two aspects. First, we are able to train different encoders and therefore learn meaningful representations for each component. We need this in order to obtain separate representations for both keys (past and context) and values (future) in memory.

Second, despite not explicitly constraining the decoding process, we observe that the usage of both past and context influences future reconstruction, depending on the size of the autoencoder latent state, as discussed in more detail in Section 5.3.

In general, this conditioned reconstruction of the future, also allows to generate trajectories that differ from the ones in memory and are not just a simple copy of already observed samples. We observed that the model was not able to represent contexts effectively, focusing just on the past and future coordinates. To avoid this and be able to condition reconstructed futures also with top-view maps, we added the auxiliary decoder Ψ_C that reconstructs the input context $\mathbf{c}^k$. We use this decoder only for training effective representations and we ignore it in the rest of the model.

3.3 Memory Writing Controller

Traditional Memory Augmented Neural Networks [9], [66], [67] are designed to observe collections of data, usually referred to as episodes. The models are equipped with a working memory to store relevant information about the episode in order to generate a meaningful output. Yet memory is cleared for each episode and what is trained is the controller that decides what to read/write. The supervision for training stems from the cost function at the end of the episode, tracing gradients back to the controller.

As in standard memories, we train a controller to emit a write probability $P(w)$ every time that a sample is observed, but, differently from these approaches, we build a compact and permanent memory.

Training such a controller might be challenging since $P(w)$ does not depend only on the intrinsic importance of the observed sample but also on the current state of the memory. To solve this issue, we do not rely on the prediction loss for supervision. We instead feed the reconstruction error e to the controller, which decides if the network reconstruction was sufficiently close to the ground truth. To enforce this behavior we define the writing controller loss $\mathcal{L}_w$ as:

$$\mathcal{L}_w = e \cdot (1 - P(w)) + (1 - e) \cdot P(w) \tag{1}$$

where e is assumed to have values in $[0, 1]$. When the error is low, i.e., $e \to 0$, then

$$\mathcal{L}_w \approx P(w) \tag{2}$$

therefore the write probability is minimized.

Conversely, when $e \to 1$, then

$$\mathcal{L}_w \approx 1 - P(w) \tag{3}$$

and the controller maximizes the write probability.

In this way the controller adaptively learns a threshold on the reconstruction error, that allows to store in memory

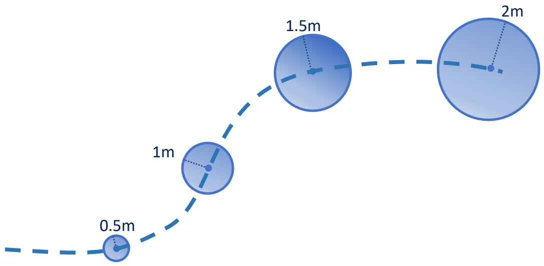


Fig. 4. The writing controller estimates a bounded trajectory error as an adaptive miss-rate. For further timesteps an increasingly higher error is tolerated.

only what is useful for predicting accurately, limiting redundancy. If the model exhibits a large prediction error, the controller writes the current sample with its ground truth future encoding in memory. When this happens, it indicates that the memory lacks samples to accurately reconstruct the future, hence, by writing the sample in memory, the model will improve its prediction capacity. This behavior is ensured by the reconstruction capabilities of the model. In fact we use a pretrained decoder Ψ_F that is trained on the same samples used to populate the memory (and will therefore have a small reconstruction error). Without a well trained decoder, the controller might risk to store redundant samples without improving the overall prediction capabilities of the system. More details on training and memory population are provided in Sec. 3.5.

To satisfy the assumption of a bounded error function with values in $[0,1]$ for the writing controller loss of Eq. 1, we introduce an adaptive miss rate error function with a threshold depending on the timestep:

$$e = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_i(\hat{\mathbf{x}}_F, \mathbf{x}_F) \qquad (4)$$

where $\mathbb{1}_i(\hat{\mathbf{x}}_F, \mathbf{x}_F)$ is an indicator function equal to 1 if the i-th point of the prediction $\hat{\mathbf{x}}_F$ lies within a threshold th from the ground truth and 0 otherwise. We use a different threshold for each timestep, allowing a given uncertainty for the farthest point (4 seconds) and linearly decreasing towards 0 for previous ones. Interestingly, we have noticed that changing these thresholds affects memory size, introducing redundancy when a small radius is used. In our experiments, we use th_{4s} equal to 2 meters, which offers a good balance between memory size and prediction error (Fig. 4).

Since the memory controller is learned exploiting reconstruction errors, it stores embeddings corresponding to previously unseen futures. These are representations of both frequent and rare trajectories. Keeping rare trajectories in an element-wise addressable memory is desirable, as they are necessary to predict similar instances that may happen again in the future. However, in a real world scenario, encodings of spurious trajectories due to the early perception modules may be written in memory (e.g., bad trajectories caused by detector or tracker failures). This could lead to noisy future reconstructions. However, this negative impact is mitigated by the fact that observed trajectories pass through an encoding-decoding process before being stored in the memory, which attenuates the effect of noise.

3.4 Memory Reading Controller

To access memory we use a reading controller. The reading controller uses both the observed past trajectory and the context (π^k, γ^k) as key and generates a read probability $P(r)^i$ over each memory location i. An ablation study of controller variants is reported in Sec. 5.1.

Our reading controller is based on cosine similarity between the observed sample and memory keys. We first compute a past read similarity s_π and a context read similarity s_γ as follows:

$$s_\pi^i = \frac{\pi^k \cdot \pi^i}{\|\pi^k\|\|\pi^i\|} \quad i = 0, ..., |M| \qquad (5)$$

$$s_\gamma^i = \frac{\gamma^k \cdot \gamma^i}{\|\gamma^k\|\|\gamma^i\|} \quad i = 0, ..., |M| \qquad (6)$$

We then feed s_π^i and s_γ^i to a multilayer feed-forward neural network F that blends the read similarities, weighing past and context importance and is trained to output high scores for relevant samples and low scores for the others. The final read probability is therefore obtained by:

$$P(r)^i = F(s_\pi^i, s_\gamma^i) \qquad (7)$$

Since each memory sample can be read and decoded independently, to obtain multimodality, we simply read the top-K samples with the highest $P(r)^i$ at inference time.

3.5 Training

We train our model to observe 2 seconds trajectories and predict up to 4 seconds in the future. To achieve translation and rotation invariance, each trajectory is normalized by shifting the present in the origin and rotating the trajectory in order to make it tangent with the Y-axis in the origin. In this way all futures start from (0, 0) in an upward direction.

First, a pretraining of both the encoders and the decoders is done jointly as an autoencoder. To do so, we feed triplets comprising past trajectories $\mathbf{x}_P$, future trajectories $\mathbf{x}_F$ and their context $\mathbf{c}$, all belonging to the same observation. The context is a semantic top-view map of $120px \times 120px$ which covers an area of 60×60 meters in front of the moving agent. The decoders reconstruct only future coordinates and the semantic map.

We then train the memory controllers, exploiting the learned past encoder and future decoder. The trained writing controller allows the memory to be filled with useful and non-redundant training samples by iterating over the training set and measuring reconstruction errors. While in principle the order in which samples are presented to the memory for writing may result in different final content, in our experiments we found that this does not affect the final prediction. During training, we reset the memory after each epoch until convergence.

The reading controller is trained to output a read probability for memory samples. For each training sample to be predicted, we select a memory subset taking the top-K elements with the highest similarity according to the past. In fact, past is a stronger cue than context for reconstructing plausible trajectories, as will be shown in Sec. 5.1. In our experiments, we set $K = 20$ for training the reading controller, in order to obtain a set of diverse samples that have similar dynamics with the observed one. Some of these trajectories will likely go off road since context is not observed yet. We identify the best and worst candidate in the set by decoding and comparing their reconstructions to the ground truth. We then use these memory elements as training samples for the controller: we train the reading controller F with a binary cross-entropy, assigning a positive label to the best candidate and a negative label to the worst one. The architecture of the reading controller is a simple Multi-Layer Perceptron (MLP) with two layers, separated by a ReLU activation. The controller takes two inputs (past and context similarities), projects them in a 4-dimensional space with the first layer of the MLP and then blends them into a single output. This allows us to obtain a simple non-linear fusion of past and context similarities which is learned from the data. The effect of this learning process, which adapts to different datasets, is shown in Sec. 5.1. To populate the final memory, we use the controllers to store a non-redundant set of samples by iterating for an epoch on the training set.

The trajectory encoders and decoder are implemented as Gated Recurrent Units with a 48-dimensional hidden state for each encoder and 96-dimensional for the decoder. The trajectories are first processed with a 1D convolution with 16 filters before being fed to the recurrent layer of the encoder. In the decoder instead, after the GRU, a fully connected layer generates spatial displacements to obtain the future reconstruction.

The context encoder is a Convolutional Neural Network composed as follows: a convolutional layer with 4 filters 3×3 with stride 2 and padding 1; a max pooling layer which halves the feature map size; a second convolutional layer with 8 filters 3×3 with stride 2 and padding 1 which generates a $15 \times 15 \times 8$ feature map; a fully connected layer which condenses the representation into a 48-dimensional vector. Each convolutional layer has batch normalization and ReLU activation. The latent feature of the autoencoder is obtained by concatenating the three representations π, γ, ϕ of past, context and future, and is therefore $48 \times 3 = 144$ dimensional. The first two components of the latent vector correspond to what is going to be stored in memory as key (past and context) and the last one (future) as value. To encourage the autoencoder to exploit information from all three inputs, we apply dropout with a 0.5 rate on the latent vector during training.

The context decoder, used for training the autoencoder is made by an initial fully connected layer which generates a 225 vector then reshaped into a 15×15 map. We upsample it with 3 transposed convolutional layers with stride 2, padding 1 and ReLU activation, that yields a 120×120 output, which has the same size of the original context.

We optimize $\mathcal{L}_w$ defined in Eq. 1 to train the writing controller, a Mean Squared Error loss for the decoder and a cross-entropy loss for the context decoder and the reading controller. All components are trained with the Adam optimizer using a learning rate of 0.0001.

4 Experiments

MANTRA produces multiple trajectory predictions of nearby moving agents. While these are observed from an egocentric perspective, contextual elements such as road layout in the observer neighborhood are also considered to predict more accurately. Given 2 seconds observations, MANTRA is capable of predicting 4 seconds ahead in the future, providing the capability of reacting appropriately. This capability can be used both for pedestrians and people driving bicycles, motorcars or cars, as well as for autonomous vehicles. For the large set of cases considered, and the variety of traffic conditions covered, in the experiments we used datasets designed for trajectory prediction of vehicles in an autonomous driving context: KITTI [73], Argoverse [11], Oxford RobotCar [74] and Cityscapes [75].

4.1 Datasets

KITTI [73] The dataset includes a large variety of annotations such as Velodyne LiDAR 3D scans, object bounding boxes and tracks, calibration, depth and IMU data. Not all data is always present for every video so we used the ones categorized as *KITTI Raw Data*, following the split of DESIRE [51]. Although the split is known, how to divide trajectories in data chunks is not. To obtain samples we collect 6 seconds chunks (2 seconds for past and 4 for future) in a sliding window fashion from all trajectories in the dataset, including the ego-vehicle. We obtain 8613 top-view trajectories for training and 2907 for testing. Note that these numbers are different from the original DESIRE split since they claim to gather 2509 trajectories in total. To favor reproducibility and future comparison we will publicly release our version of the dataset upon publication. Since top-view maps are not provided by KITTI, we project semantic labels of static categories obtained with DeepLab-v3+ [76] from all frames in a common top-view map using the Velodyne 3D point cloud and IMU. The resulting maps have a spatial resolution of 0.5 meters, and will be released along with the trajectories.

Another smaller version of the KITTI dataset for trajectory prediction has been recently proposed by [52] and is publicly available. The authors propose 5 different train/test splits and average results over all runs, so we follow this evaluation protocol. We report experiments on both variants of KITTI. In the following, we refer to KITTI as our split obtained following DESIRE, unless expressly stated otherwise.

Argoverse [11] This dataset provides data for two different tasks, 3D tracking and motion forecasting. Vehicle trajectories are collected in top-view in the cities of Pittsburgh and Miami, covering an area of more than $1000 km^2$. In total, there are approximately 325k annotated trajectories, gathered from over 1000 hours of video. Maps are also available with lane centerlines, traffic direction, ground height and drivable areas. Trajectories are all divided into 5 seconds long chunks (2 seconds for past and 3 for future). The dataset is split into train, validation and test. We report results on the validation set v1.1, for which ground truth data is publicly available. Argoverse has a much larger scale compared to KITTI and exhibits more diversity in trajectory patterns, proving to be a suitable and challenging benchmark on which to evaluate trajectory prediction methods.

Oxford RobotCar [74] and Cityscapes [75] The two datasets RobotCar and Cityscapes have been adapted for trajectory prediction in [52] to show zero-shot transfer capabilities on different domains. Of particular interest is

TABLE 1
Results on the KITTI dataset. Results obtained by DESIRE are given as reference even if not comparable, due to the data collection process.

	ADE				FDE			
Method	1s	2s	3s	4s	1s	2s	3s	4s
Kalman	0.51	1.14	1.99	3.03	0.97	2.54	4.71	7.41
Linear	0.20	0.49	0.96	1.64	0.40	1.18	2.56	4.73
MLP	0.20	0.49	0.93	1.53	0.40	1.17	2.39	4.12
MANTRA (top 1)	0.25	0.59	1.04	1.79	0.49	1.40	2.81	4.70
MANTRA (top 5)	0.17	0.38	0.67	1.06	0.32	0.83	1.63	2.79
MANTRA (top 10)	**0.16**	0.30	0.48	0.74	0.27	0.60	1.09	1.91
MANTRA (top 20)	**0.16**	**0.28**	**0.43**	**0.63**	**0.26**	**0.53**	**0.90**	**1.60**
DESIRE (top 1) [51]	-	-	-	-	0.51	1.44	2.76	4.45
DESIRE (top 5) [51])	-	-	-	-	0.28	0.67	1.22	2.06
DESIRE (top 20) [51])	-	-	-	-	-	-	-	2.04

TABLE 2
Results on the KITTI dataset (INFER split).

	ADE				FDE			
Method	1s	2s	3s	4s	1s	2s	3s	4s
Kalman	0.33	0.54	0.93	1.4	0.46	1.18	2.18	3.32
Linear	0.31	0.56	0.89	1.28	0.47	1.13	1.94	2.87
MLP	0.30	0.54	0.88	1.28	0.46	1.12	1.94	2.88
RNN Enc-Dec [78]	0.68	1.94	3.20	4.46	-	-	-	-
Markov [52]	0.70	1.41	2.12	2.99	-	-	-	-
Conv-LSTM (top 5) [52]	0.76	1.23	1.60	1.96	-	-	-	-
INFER (top 1) [52]	0.75	0.95	1.13	1.42	1.01	1.26	1.76	2.67
INFER (top 5) [52]	0.56	0.75	0.93	1.22	0.81	1.08	1.55	2.46
MANTRA (top 1)	0.37	0.67	1.07	1.55	0.60	1.33	2.32	3.50
MANTRA (top 5)	0.33	0.48	0.66	0.90	0.45	0.78	1.22	2.03
MANTRA (top 10)	0.31	0.43	0.57	0.78	0.43	0.67	1.04	1.78
MANTRA (top 20)	**0.29**	**0.41**	**0.55**	**0.74**	**0.41**	**0.64**	**1.00**	**1.68**

the ability to transfer to RobotCar since the sequences are acquired in the UK where cars drive on the left-side of the road. RobotCar has 6 seconds trajectories divided into 2 seconds for past and 4 for future. Cityscapes instead has shorter videos and predictions are made only up to one second in the future, as done in [52].

4.2 Evaluation Metrics and Baselines

We report results in two common metrics for vehicle trajectory prediction: *Average Displacement Error* (ADE) and *Final Displacement Error* (FDE), where ADE is the average L2 error between all future timesteps and FDE (sometimes referred to as Horizon error) is the error at a given timestep. As in [51], [52] we take the best out of K predictions to account for the intrinsic multimodality of the task. We compare our approach with several baselines: a linear coordinate regressor (*Linear*); a Multi-Layer Perceptron with two layers trained as a coordinate regressor (*MLP*); a Kalman filter [77], with a constant speed model used to propagate the estimate without incorporating measures (*Kalman*). We implemented and tested the baselines on KITTI and Argoverse to show comparable results. When available we also report existing baselines from the literature.

4.3 Results

Table 1 shows the results on the KITTI dataset. Simply propagating the trajectory with a Kalman filter proves to be insufficient to accurately predict future positions, especially over long time spans, with an FDE@4s higher than 7m. Learning based baselines all perform better than the Kalman filter, with the MLP performing slightly better than the linear regressor.

TABLE 3
Results on the Argoverse dataset.

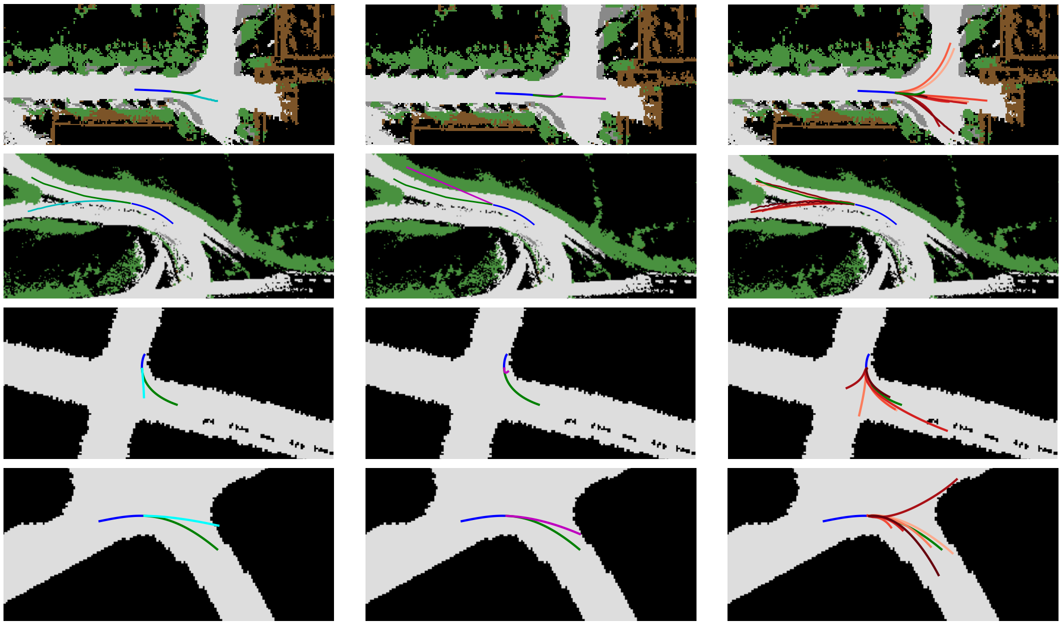
	ADE		FDE	
Method	1s	3s	1s	3s
Kalman (top 1)	0.72	2.70	1.29	6.56
Linear (top 1)	0.58	1.95	0.98	4.58
MLP (top 1)	0.53	1.68	0.87	3.90
NN [11] (top 1)	0.75	2.46	1.28	5.60
NN + map [11] (top 6)	0.72	2.28	1.33	4.80
LSTM ED [11] (top 1)	0.68	2.27	1.78	5.19
LSTM ED + map [11] (top 6)	0.80	2.25	1.35	4.67
MFP [57] (top 6)	-	1.39	-	-
MANTRA (top 1)	0.72	2.36	1.25	5.31
MANTRA (top 6)	0.56	1.22	0.84	2.30
MANTRA (top 10)	0.53	1.00	0.77	1.69
MANTRA (top 20)	**0.52**	**0.84**	**0.73**	**1.16**

Models that generate a single prediction fail to address the multimodality of the task, since they are trained to lower the error with a single output, even when there might be multiple equally likely desired outcomes. What may happen is that in front of a bifurcation, the model predicts an average of the two possible trajectories, trying to satisfy both scenarios. Examples of this behavior are shown in Fig. 5. Each prediction of MANTRA instead follows a specific path, ignoring the others. This leads to high errors on some examples when generating only one future, since the model may decide to follow a different likely path. On the other hand as soon as we generate K multiple predictions, the top-K error drastically decreases since we are able to cover diverse future paths. We also report results from DESIRE [51] varying K. Even though these results are not directly comparable as explained in Section 4.1, it is interesting to observe how DESIRE does not report significant improvements when increasing K from 5 to 20, while our method reduces the error significantly. This suggests that MANTRA samples a higher diversity of futures both at a coarse level (i.e., taking one road or another) and at a fine level (i.e., taking different behaviors on the same road).

Additionally, we evaluate MANTRA on the KITTI split proposed in [52], as shown in Table 2. Here we also report some available baselines from the state of the art, both for single and multimodal predictions. With $K = 1$ our method performs better or on par with INFER [52] at low timesteps, yet losing some precision at 4s. Increasing K instead we are able to largely outperform INFER over all timesteps.

A similar analysis is obtained when we move to a more recent and larger scale dataset: Argoverse. Differently from KITTI, the standard evaluation observes 2 seconds in the past and predicts up to 3 seconds in the future, using K=6 predictions to demonstrate multimodality. Results are shown in Table 3. Along with the Kalman, Linear and MLP baselines, we report results from the state of the art [11], [57], obtaining better results especially for 3 seconds predictions. Some qualitative results on KITTI and Argoverse are shown in Fig. 5, comparing them with the baselines.

Interestingly, our model does not include social interaction modeling as [11], [51], [52]. Although the presence of other agents in the surrounding context is an influencing variable, in our setting, the observation time span is sufficiently large for a vehicle to adapt its motion to what is



(a) Linear (b) Kalman (c) MANTRA

Fig. 5. MANTRA compared to Linear regression (a) and Kalman filter (b). Methods (a),(b) lack multi-modal capability. Past trajectories are depicted in blue, ground truth in green and future predictions are cyan (a), purple (b) and red (c). In (c) highly ranked are darker. The first two rows show samples from the KITTI dataset, while the other two from Argoverse.

TABLE 4
Zero-shot transfer evaluation on the Oxford RobotCar dataset. MANTRA was trained on KITTI and evaluated on Oxford RobotCar.

	ADE				FDE			
Method	1s	2s	3s	4s	1s	2s	3s	4s
INFER (top 1) [52]	1.06	1.35	1.48	1.68	1.31	1.71	1.70	2.56
INFER (top 5) [52]	0.85	1.14	1.29	1.50	1.18	1.58	1.58	2.41
MANTRA (top 1)	0.55	0.77	1.01	1.30	0.60	1.15	1.82	2.63
MANTRA (top 5)	0.55	0.68	0.82	1.03	0.58	0.88	1.37	2.07
MANTRA (top 10)	0.44	0.56	0.72	0.94	0.48	0.73	1.33	1.98
MANTRA (top 20)	**0.31**	**0.43**	**0.59**	**0.83**	**0.35**	**0.61**	**1.24**	**1.96**

perceived and prepare a reaction. This is true for social interactions (avoiding dynamic obstacles) as well as compliance with the environment (avoiding static obstacles). We believe that an observed past trajectory, incorporates to some extent these reactions reflecting also into the future prediction.

Following [52], we also showcase the ability of our model to zero-shot transfer from KITTI to other datasets, namely Oxford RobotCar and Cityscapes. We first train our model on KITTI and then we keep our memory frozen, without any additional training or finetuning. Results are shown in Table 4 and Table 5. On Oxford RobotCar, MANTRA is still able to provide satisfactory results, consistently outperforming INFER across timesteps for multimodal predictions. Analogously, on Cityscapes the model obtains a lower error than the other methods. Here we report only errors at 1s in the future, which is the maximum length of the trajectories in the dataset. Note that top-view semantic maps are not available for these two datasets, therefore we use a memory controller that relies solely on past embeddings via cosine distance.

4.4 Incremental Setting

Differently from prior work on trajectory prediction, MANTRA is able to improve its capabilities online, i.e., observing other agents' behaviors while driving. We simulate an online scenario on KITTI, iteratively removing N samples from the test set and feeding them to the memory writing controller without retraining the encoder. In this way, the controller can write in memory novel useful samples according to $P(w)$. After every new batch of samples, we evaluate the model on the remaining trajectories in the test set, until the test set has been completely observed. We start from a pre-trained memory of approximately 250 samples, belonging to the training set of KITTI.

TABLE 5
Zero-shot transfer evaluation on the Cityscapes dataset at 1s in the future. MANTRA was trained on KITTI and evaluated on Cityscapes.

Method	ADE	FDE
Conv-LSTM (top 1) [52]	1.50	-
Conv-LSTM (top 3) [52]	1.36	-
Conv-LSTM (top 5) [52]	1.28	-
INFER (top 1) [52]	1.11	1.59
INFER (top 3) [52]	0.99	1.45
INFER (top 5) [52]	0.91	1.38
MANTRA (top 1)	0.81	1.42
MANTRA (top 3)	0.66	1.15
MANTRA (top 5)	0.60	1.00
MANTRA (top 10)	0.54	0.86
MANTRA (top 20)	**0.49**	**0.79**

In Fig. 6, memory growth and test error are shown for K=5 multiple futures. Similar behaviors can be observed varying K. We plot memory size and prediction error after having observed every new batch of N samples. We use N=50 and we indicate in the plots the percentage of observed samples in the test set, which is composed of 2907 trajectories. Memory size is measured as the number of samples in memory. Interestingly, memory size grows slowly while the error keeps decreasing. The memory con-

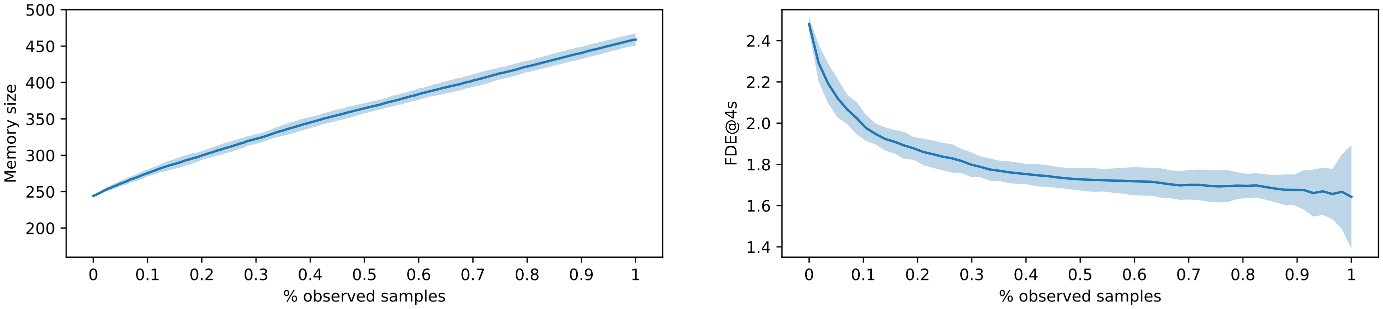


Fig. 6. Incremental setting. The model observes batches of test samples online, that are used as training data, and is evaluated on the remaining portion of the test set. Mean and variance of memory size (left) and prediction error (right), averaged over 100 runs, are shown. On the x-axis we report the percentage of test samples incrementally observed.

troller stores only 16% of the newly seen examples. Since the error variance increases when the test set size decreases, we average results over 100 runs, randomizing the test set.

5 Model Analysis

In the following, we analyze the model under several aspects. First, we perform ablation studies to highlight the importance of distinct components. Then, we investigate what samples the model writes in memory and how it decodes them. Finally, we report the execution time for different memory sizes.

5.1 Ablation Studies

We investigate several modifications of MANTRA, reporting results in Table 6 on Argoverse. We test the following variations: (i) with a reading controller based on cosine similarity with the past; (ii) with a reading controller based on cosine similarity with the context; (iii) without decoder, i.e., reading from memory using encodings but just copying the correspondent future coordinates; (iv) without rotation invariance, i.e., using trajectories with random rotations.

First, we discuss the importance of the memory reading controller introduced in Sec. 3.4. which weighs past and context importance to decode relevant samples from memory. In principle the reading controller can be any function that manipulates either of the two inputs. Note that the reading controller reflects also on memory size, since samples are stored based on reconstruction error.

At a first look, it appears that the past-based controller (Controller Past) performs on par with the learned controller (Full), with the context-based approach achieving far worse results, both in terms of reconstruction error and memory size. While basing decisions only on context will obviously not lead to meaningful reconstructions (Controller Context), we observed that the best-of-K metrics ADE and FDE do not point out the substantial differences between the other two controllers. It appears that often past information alone is sufficient to generate a good prediction, since the dynamics of the agent pose a strong constraint over future outcomes. However, it might happen that several trajectories are predicted off-road, thus not being compatible with the road layout. Using best-of-K metrics does not allow to detect such cases. To overcome this limitation, for each observation we take into account all the K predictions and measure the off-road percentage. We consider a prediction as off-road when at least one predicted coordinate lays outside the road. Similar metrics have been adopted in prior work [59], [79] to assess the violation of environmental constraints posed by obstacles or lanes.

TABLE 6
Ablation study of MANTRA on Argoverse. Errors are at K=6. Memory size is shown as number of samples and % of the training set.

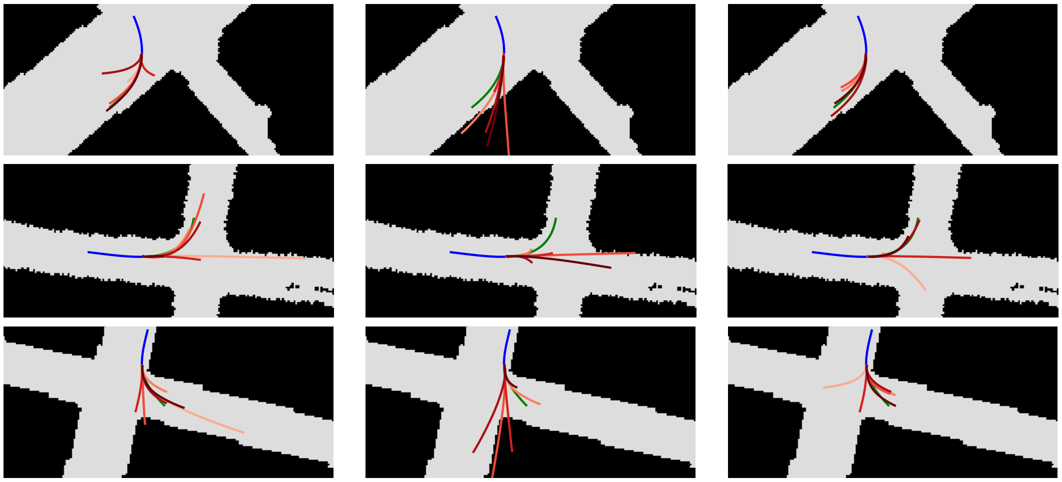
	ADE		FDE			
Method	1s	3s	1s	4s	Off-Road (%)	Memory Size
MANTRA (Full)	0.56	**1.22**	0.84	**2.30**	3.15	6397 (3.1 %)
MANTRA (Controller Past)	**0.52**	**1.22**	**0.79**	2.38	8.14	**6242 (2.9 %)**
MANTRA (Controller Context)	0.73	1.87	1.19	3.70	**3.08**	21992 (10.5 %)
MANTRA w/o dec.	0.80	1.47	1.12	2.44	6.01	6397 (3.1 %)
MANTRA w/o rot. inv.	1.11	2.54	1.80	4.63	40.26	75674 (36.3 %)

It appears that taking context into account along with the past, lowers this percentage considerably (Full), resulting in a set of trajectories that satisfy both the dynamics of the past and the surrounding environment (Fig. 7). Interestingly, we found that even when predicting a large number of futures, the off-road percentage stays low with the learned reading controller (e.g., 3.8% with K=20). The capacity to learn even a simple nonlinear function to blend past and context information allows the controller to adapt to the data it is trained on. In fact, we observe two different behaviors on KITTI and Argoverse, as shown in Fig. 8. On KITTI, the controller focuses more on the past, discarding context information when past similarity is low. On Argoverse, the higher complexity of contexts reflects on the weights learned by the controller, and context is taken more into account even when the past is very dissimilar.

Removing the decoder from the model, instead, leads to worse results and also doubles the percentage of off-road predictions. What appears to be of great importance is data normalization, since when we remove rotation invariance the model does not manage to achieve a good reconstruction error, which leads to a 10× bigger memory and a 40% rate of off-road predictions.

5.2 Memory Inspection

To better understand what the model is learning, we inspect what is stored in memory by the controller. We take each sample and plot its decoded future to depict a snapshot of the memory. In Fig. 9 all samples from a memory filled on the KITTI dataset for K=5 predictions are shown.



(a) Controller Context (b) Controller Past (c) Full

Fig. 7. Difference between reading controllers. Past is important to correctly model trajectory dynamics; context is relevant for making predictions that are feasible with the road layout. Past trajectories are blue, ground truth green and future predictions red (highly ranked are darker).

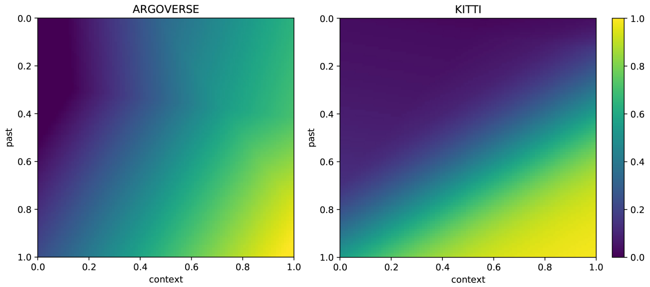


Fig. 8. Reading controller scores varying past and context similarities. Different blending functions are learned for different datasets, privileging the past on KITTI and increasing the relevance of context on Argoverse.

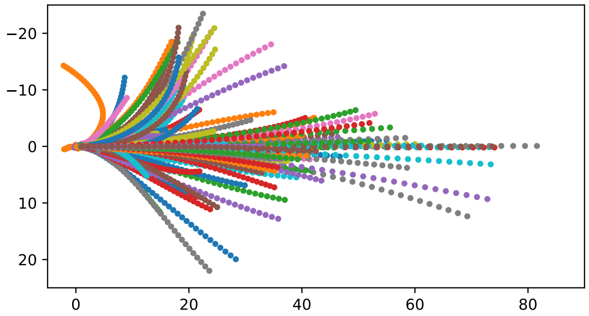


Fig. 9. Decoded trajectories from memory.

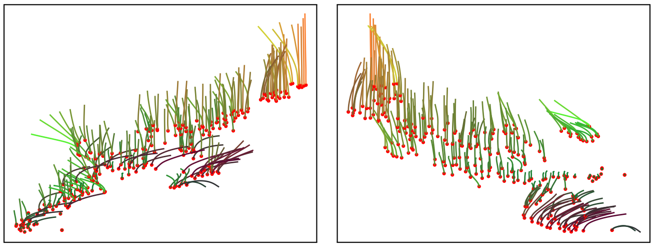


Fig. 10. t-SNE representations of past (left) and future (right) encodings stored in memory. Each point in the embedding space is shown along with the decoded trajectory. Trajectories are color coded by orientation (green tones) and speed (red tones).

In Fig. 10 we plot t-SNE projections [80] of past and future encodings stored in memory, as points. On the left we plot past embeddings, while on the right we report future embeddings. For each projected sample we also show the future trajectories generated by the decoder, displayed starting from the t-SNE points. All the trajectories in the image have an upward trend due to the rotation invariance. It can be seen that similar trajectories are clustered together, indicating that the encoders are learning a manifold where samples with similar patterns are close. Interestingly, observing the t-SNE of past encodings, the multimodal nature of the problem emerges. In fact, the space appears to be organized mostly by trajectory speed and for each point several possible future directions are present. When trajectories have lower speed, futures are free to span over many possible directions, while when trajectories have higher speed, the futures vary more in length rather than curvature.

5.3 Decoder Analysis

Here we inspect the behavior of the decoder and the influence that different pasts and contexts have on future reconstructions. Encoder and decoder are jointly trained, but differently from standard autoencoders, only part of the input is reconstructed, i.e., the future. Past and context, though, have the important role of conditioning the reconstruction so that we can generalize to unseen examples. First, we examine the influence of the past.

In Fig. 11 we show several reconstructions of the same future, changing only the past encoding and keeping fixed the future one. The reconstructions of the original past yields a precise reconstruction. By changing the past by shortening it or stretching it, i.e., changing the velocity, the reconstruction gets accelerated or decelerated, affecting its curvature. As a control experiment we also use a vector of zeros or a random embedding. In both cases the generated

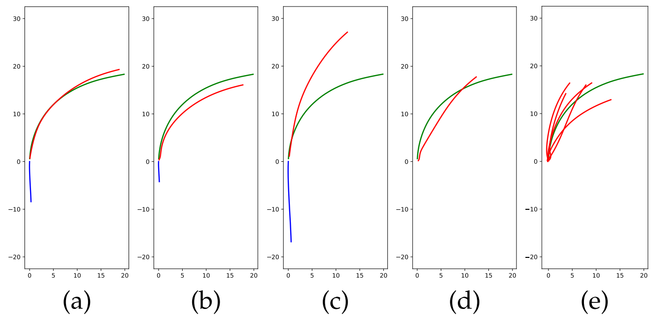


Fig. 11. Influence of past in the decoder. (a) observed past; (b) slower past; (c) faster past; (d) past embedding zeroed; (e) multiple randomized past embeddings. Blue: past trajectory. Red: future reconstruction. Green: original future.

trajectories are very imprecise but still follow approximately the original trend.

Similarly, we test the influence of different contexts on future reconstruction. In Fig. 12 we show multiple reconstructions varying only the context representation. It can be seen that using a map with a different layout, such as a straight road instead of a curve, the prediction is affected and shifted to follow the road. When the model is blinded, i.e., when the context embedding is replaced with a vector of zeros, the reconstruction is still accurate, even if slightly less precise. Feeding to the decoder a random vector instead, leads to noisy outcomes. These tests justify using the decoder feeding a combination of encodings belonging to different samples, as we do at test time. In fact the generated trajectories are new compared to the samples in memory and they adapt to the current observation.

The degree of past and context influence on future reconstruction depends on the size of the autoencoder latent state. In fact, models with a large sized encoding learn to pass the input directly to the output. Conversely, a reduced sized encoding forces the decoder to rely on information encoded in the past and context to reconstruct the output signal. To verify this, we perform a control experiment by training different autoencoders with different future embedding sizes $|\phi|$. Similarly to the previous experiments, we swap past and context with a different sample to observe how these influence the reconstruction. In Fig. 13 we show an example of the reconstructed future using $|\phi| = 24, 48, 72$. When the embedding size is small, the new context and past have a considerable influence on the reconstruction, shifting the future towards the lane. With a large embedding, however, the decoder reconstructs the original future, disregarding past and context entirely. We have observed this behavior consistently over different samples.

5.4 Execution Time Analysis

Since automotive applications are time-critical, in this paragraph we discuss the execution time of MANTRA. Model inference can be broken down into different steps: *encoding*, *memory access* and *decoding*. Encoding has a fixed cost, while decoding depends on K. However, the effect of K is negligible thanks to GPU parallel execution. Memory access, instead, has a linear dependency on the number of samples in memory. We measure the inference time of MANTRA on Argoverse, using a memory with 6397 entries as in the experiments in Table 3 and K=6 as in the evaluation protocol. To simulate a scenario with multiple agents, we predict futures for a batch of 5 different vehicles simultaneously. On an Nvidia Titan RTX GPU, the total inference time, averaged over 100 runs, is of 15.37ms. Figure 14 (left) shows the timing breakdown, highlighting that the most expensive stage is decoding. This is due to the fact that we decode autoregressively for 30 timesteps (3 seconds), which is less efficient than directly generating the whole trajectory. However, the model is still capable of running in real-time at approximately 65Hz. In Fig. 14 (right), we show inference times as a function of memory size. It can be noticed that memory access time increases linearly, up to 15ms with one million stored samples. Morever, in case of very large memories, fast access techniques could be employed [81]. We conclude that in a real-time system, the main computational bottleneck would likely be given by the detection and segmentation pipelines used to extract trajectories and semantic labels.

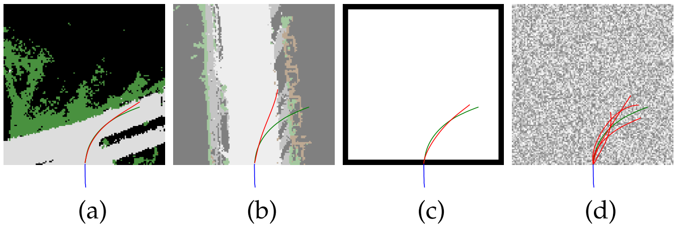


Fig. 12. Influence of context in the decoder. (a) original context; (b) different context; (c) context embedding zeroed; (e) multiple randomized context embeddings. Blue: past trajectory used for decoding. Red: future reconstruction. Green: original future.

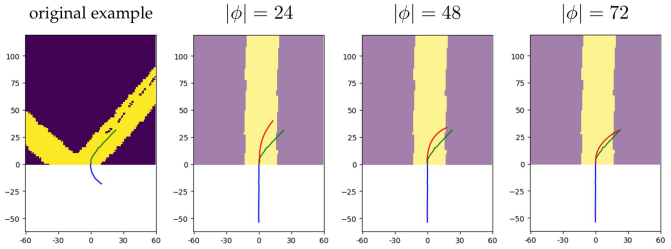


Fig. 13. Decoding is affected by the embedding size $|\phi|$ when changing past and context. With a small embedding, the reconstruction adapts more to past and context, while they are ignored with large embeddings. Blue: past trajectory. Red: future reconstruction. Green: original future.

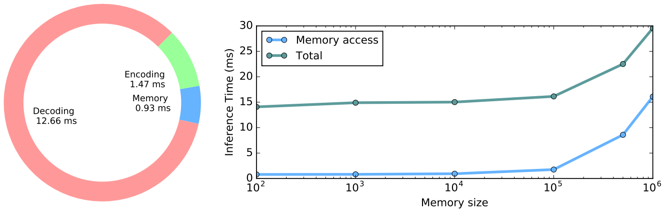


Fig. 14. Inference time. Left: breakdown of timings on Argoverse (memory size 6397) with K=6 futures. Right: inference time dependency from number of samples in memory is linear (log x axis).

6 Conclusions and Future Work

In this paper we have presented MANTRA, a novel framework exploiting a Memory Augmented Network for multiple trajectory prediction of moving agents, observed from an ego-centric point of view. The memory module is used to store encodings of past, context and future observations and is central to the model prediction capabilities. Two memory controllers are trained respectively to write a compact set of samples relevant for the task and to read the most appropriate ones, considering both the past trajectory and the surrounding road layout. The experiments presented on four public datasets, show that MANTRA achieves state of the art performance. The presented method also performs well in zero-shot transfer to unseen datasets and is able to improve incrementally online.

Currently, this work has focused on predicting trajectories of multiple moving agents. Future work will address agent interactions, extending MANTRA by adding an episodic memory along with the persistent one, to reason on social behaviors of the surrounding agents. Another aspect which we plan on improving is the online learning capability. Now, we are incrementally expanding MANTRA's memory after having trained the model. We believe that an end-to-end solution capable of updating the weights of the model at runtime would be an interesting development.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[3] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2174–2182.

[4] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1672–1678.

[5] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 399–404.

[6] 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*, 2016, pp. 961–971.

[7] 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*, 2018, pp. 2255–2264.

[8] A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1685–1692.

[9] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[10] F. Marchetti, F. Becattini, L. Seidenari, and A. Del Bimbo, "Mantra: Memory augmented networks for multiple trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[11] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[12] M. Leo, G. Medioni, M. Trivedi, T. Kanade, and G. M. Farinella, "Computer vision for assistive technologies," *Computer Vision and Image Understanding*, vol. 154, pp. 1–15, 2017.

[13] M. Leo, A. Furnari, G. G. Medioni, M. Trivedi, and G. M. Farinella, "Deep learning for assistive computer vision," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[14] M. Bolanos, R. Mestre, E. Talavera, X. Giró-i Nieto, and P. Radeva, "Visual summary of egocentric photostreams by representative keyframes," in *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2015, pp. 1–6.

[15] M. Bolanos, M. Dimiccoli, and P. Radeva, "Toward storytelling from visual lifelogging: An overview," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 1, pp. 77–90, 2016.

[16] A. Furnari, S. Battiato, and G. M. Farinella, "Personal-location-based temporal segmentation of egocentric videos for lifelogging applications," *Journal of Visual Communication and Image Representation*, vol. 52, pp. 1–12, 2018.

[17] H.-C. Wang, R. K. Katzschmann, S. Teng, B. Araki, L. Giarré, and D. Rus, "Enabling independent navigation for visually impaired people through a wearable vision-based feedback system," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 6533–6540.

[18] A. Fiannaca, I. Apostolopoulous, and E. Folmer, "Headlock: a wearable navigation aid that helps blind cane users traverse large open spaces," in *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility*, 2014, pp. 19–26.

[19] D. Dakopoulos and N. G. Bourbakis, "Wearable obstacle avoidance electronic travel aids for blind: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 25–35, 2009.

[20] T. Miyaki and J. Rekimoto, "Lidarman: reprogramming reality with egocentric laser depth scanning," in *ACM SIGGRAPH 2016 Emerging Technologies*, 2016, pp. 1–2.

[21] D. Lindlbauer and A. D. Wilson, "Remixed reality: manipulating space and time in augmented reality," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–13.

[22] K. K. Singh, K. Fatahalian, and A. A. Efros, "Krishnacam: Using a longitudinal, single-person, egocentric dataset for scene understanding tasks," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.

[23] H. Soo Park, J.-J. Hwang, Y. Niu, and J. Shi, "Egocentric future localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4697–4705.

[24] J. Bai, S. Lian, Z. Liu, K. Wang, and D. Liu, "Smart guiding glasses for visually impaired people in indoor environment," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 3, pp. 258–266, 2017.

[25] C. Feng, S. Valaee, A. W. S. Au, S. Reyes, S. Sorour, S. N. Markowitz, D. Gold, K. Gordon, and M. Eizenman, "Anonymous indoor navigation system on handheld mobile devices for visually impaired," *International Journal of Wireless Information Networks*, vol. 19, no. 4, pp. 352–367, 2012.

[26] Y. J. Lee and K. Grauman, "Predicting important objects for egocentric video summarization," *International Journal of Computer Vision*, vol. 114, no. 1, pp. 38–55, 2015.

[27] A. Furnari, S. Battiato, K. Grauman, and G. M. Farinella, "Next-active-object prediction from egocentric videos," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 401–411, 2017.

[28] Y. H. Lee, T.-S. Leung, and G. Medioni, "Real-time staircase detection from a wearable stereo system," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012, pp. 3770–3773.

[29] D. Damen, T. Leelasawassuk, O. Haines, A. Calway, and W. W. Mayol-Cuevas, "You-do, i-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video." in *BMVC*, vol. 2, 2014, p. 3.

[30] M. Liu, S. Tang, Y. Li, and J. Rehg, "Forecasting human object interaction: Joint prediction of motor attention and egocentric activity," *arXiv preprint arXiv:1911.10967*, 2019.

[31] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori, "Object level visual reasoning in videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 105–121.

[32] D. Damen, H. Doughty, G. Maria Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price *et al.*, "Scaling egocentric vision: The epic-kitchens dataset," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 720–736.

[33] R. Yonetani, K. M. Kitani, and Y. Sato, "Recognizing micro-actions and reactions from paired egocentric videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2629–2638.

[34] G. Abebe, A. Catala, and A. Cavallaro, "A first-person vision dataset of office activities," in *IAPR Workshop on Multimodal Pattern Recognition of Social Signals in Human-Computer Interaction*. Springer, 2018, pp. 27–37.

[35] S. Su, J. Pyo Hong, J. Shi, and H. Soo Park, "Predicting behaviors of basketball players from first person videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1501–1510.

[36] A. S. Razavian, O. Aghazadeh, J. Sullivan, and S. Carlsson, "Estimating attention in exhibitions using wearable cameras," in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014, pp. 2691–2696.

[37] F. Ragusa, A. Furnari, S. Battiato, G. Signorello, and G. M. Farinella, "Ego-ch: Dataset and fundamental tasks for visitors behavioral understanding using egocentric vision," *Pattern Recognition Letters*, vol. 131, pp. 150–157, 2020.

[38] L. Seidenari, C. Baecchi, T. Uricchio, A. Ferracani, M. Bertini, and A. D. Bimbo, "Deep artwork detection and retrieval for automatic context-aware audio guides," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 3s, pp. 1–21, 2017.

[39] Y.-S. Hsieh, Y.-C. Su, and L.-G. Chen, "Robust moving object tracking and trajectory prediction for visual navigation in dynamic environments," in *2012 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2012, pp. 696–697.

[40] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6120–6127.

[41] L. Berlincioni, F. Becattini, L. Galteri, L. Seidenari, and A. Del Bimbo, "Road layout understanding by generative adversarial inpainting," in *Inpainting and Denoising Challenges*. Springer, 2019, pp. 111–128.

[42] B. Bescos, J. Neira, R. Siegwart, and C. Cadena, "Empty cities: Image inpainting for a dynamic-object-invariant space," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5460–5466.

[43] Y. Yao, M. Xu, Y. Wang, D. J. Crandall, and E. M. Atkins, "Unsupervised traffic accident detection in first-person videos," *arXiv preprint arXiv:1903.00618*, 2019.

[44] S. Malla and C. Choi, "Nemo: Future object localization using noisy ego priors," *arXiv preprint arXiv:1909.08150*, 2019.

[45] Y. Yao, M. Xu, C. Choi, D. J. Crandall, E. M. Atkins, and B. Dariush, "Egocentric vision-based future vehicle localization for intelligent driving assistance systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9711–9717.

[46] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[47] 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*. IEEE, 2009, pp. 261–268.

[48] 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 Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1349–1358.

[49] M. Lisotto, P. Coscia, and L. Ballan, "Social and scene-aware trajectory prediction in crowded spaces," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[50] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2375–2384.

[51] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[52] S. Srikanth, J. A. Ansari, S. Sharma *et al.*, "Infer: Intermediate representations for future prediction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)*, 2019.

[53] A. Zyner, S. Worrall, and E. Nebot, "Naturalistic driver intention and path prediction using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[54] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2821–2830.

[55] C. Choi, A. Patil, and S. Malla, "Drogon: A causal reasoning framework for future trajectory forecast," *arXiv preprint arXiv:1908.00024*, 2019.

[56] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv preprint arXiv:1910.05449*, 2019.

[57] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 398–15 408.

[58] S. Malla, B. Dariush, and C. Choi, "Titan: Future forecast using action priors," *arXiv preprint arXiv:2003.13886*, 2020.

[59] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," *arXiv preprint arXiv:2001.03093*, 2020.

[60] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 204–211.

[61] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1179–1184.

[62] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[63] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[64] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," *arXiv preprint arXiv:1703.03129*, 2017.

[65] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, 2016, pp. 1842–1850.

[66] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[67] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[68] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[69] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *International conference on machine learning*, 2016, pp. 1378–1387.

[70] C. Ma, C. Shen, A. Dick, Q. Wu, P. Wang, A. van den Hengel, and I. Reid, "Visual question answering with memory-augmented networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6975–6984.

[71] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 152–167.

[72] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," *arXiv preprint arXiv:1606.03126*, 2016.

[73] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[74] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[75] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset

for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[76] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

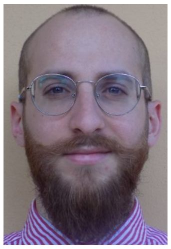
[77] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[78] J. Virdi, "Using deep learning to predict obstacle trajectories for collision avoidance in autonomous vehicles," Ph.D. dissertation, UC San Diego, 2017.

[79] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[80] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[81] G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou, "Large memory layers with product keys," in *Advances in Neural Information Processing Systems*, 2019, pp. 8546–8557.



Francesco Marchetti He received a master degree cum laude in 2019 in computer engineering from the University of Florence with the thesis "Trajectories Prediction for Autonomous Driving with Memory Networks" in collaboration with the research institute IMRA Europe.

Currently he is a research fellow at Media Integration and Communication Center (MICC) and the research work focuses on trajectories forecasting in the automotive field.

Federico Becattini obtained his PhD in 2018 from the University of Florence under the supervision of Prof. Alberto Del Bimbo and Prof. Lorenzo Seidenari. Currently he is a PostDoc at MICC, where he is involved in numerous collaborations, mostly focusing on Autonomous Driving and Scene Understanding. He attended several international conferences both as speaker and volunteer, as well as summer schools. He served to the scientific community as a reviewer for scientific journals and conferences.

Lorenzo Seidenari is currently an Assistant Professor at the Media Integration and Communication Center of the University of Florence. He received his Ph.D. degree in computer engineering in 2012 from the University of Florence. His research focuses on deep learning for object and action recognition in video and images. On this topics he addressed RGB-D activity recognition, embedding learning for multimodal-fusion, anomaly detection in video and people behavior profiling. He was a visiting scholar at the University of Michigan in 2013. He organized and gave a tutorial at ICPR 2012 on image categorization. He is author of 14 journal papers and more than 40 peer-reviewed conference papers. He has an h-index of 19 with more than 1300 citations.

Alberto Del Bimbo is a Full Professor of Computer Engineering, and the Director of the Media Integration and Communication Center with the University of Florence. His scientific interests are multimedia information retrieval, pattern recognition, image and video analysis, and human–computer interaction. From 1996 to 2000, he was the President of the IAPR Italian Chapter and the Member-at-Large of the IEEE Publication Board from 1998 to 2000. He was the General Co-Chair of ACM MM 2010 and ECCV in 2012. He was nominated as ACM Distinguished Scientist in 2016. He received the SIGMM Technical Achievement Award for Outstanding Technical Contributions to Multimedia Computing, Communications and Applications. He is an IAPR Fellow, and an Associate Editor of Multimedia Tools and Applications, Pattern Analysis and Applications, the Journal of Visual Languages and Computing, and the International Journal of Image and Video Processing, and was an Associate Editor of Pattern Recognition, the IEEE Transactions on Multimedia, and the IEEE Transactions on Pattern Analysis and Machine Intelligence. He serves as the Editor-in-Chief of the ACM Transactions on Multimedia Computing, Communications, and Applications.