Deep Variational Learning for Multiple Trajectory Prediction of 360° Head Movements

Quentin Guimard[⋆], Lucile Sassatelli[⋆§]
[⋆]Université Côte d'Azur, CNRS, I3S
[§]Institut Universitaire de France
Sophia Antipolis, France

Francesco Marchetti, Federico Becattini,
Lorenzo Seidenari, Alberto Del Bimbo
Università degli Studi di Firenze, MICC
Florence, Italy

ABSTRACT

Prediction of head movements in immersive media is key to design efficient streaming systems able to focus the bandwidth budget on visible areas of the content. Numerous proposals have therefore been made in the recent years to predict 360° images and videos. However, the performance of these models is limited by a main characteristic of the head motion data: its intrinsic uncertainty. In this article, we present an approach to generate multiple plausible futures of head motion in 360° videos, given a common past trajectory. Our method provides likelihood estimates of every predicted trajectory, enabling direct integration in streaming optimization. To the best of our knowledge, this is the first work that considers the problem of multiple head motion prediction for 360° video streaming. We first quantify this uncertainty from the data. We then introduce our discrete variational multiple sequence (DVMS) learning framework, which builds on deep latent variable models. We design a training procedure to obtain a flexible and lightweight stochastic prediction model compatible with sequence-to-sequence recurrent neural architectures. Experimental results on 3 different datasets show that our method DVMS outperforms competitors adapted from the self-driving domain by up to 37% on prediction horizons up to 5 sec., at lower computational and memory costs. Finally, we design a method to estimate the respective likelihoods of the multiple predicted trajectories, by exploiting the stationarity of the distribution of the prediction error over the latent space. Experimental results on 3 datasets show the quality of these estimates, and how they depend on the video category.

CCS CONCEPTS

• **Human-centered computing → Virtual reality**; • **Information systems → Multimedia streaming**; • **Computing methodologies → Neural networks**.

KEYWORDS

360° videos, head motion, trajectory prediction, deep learning

ACM Reference Format:
Quentin Guimard[⋆], Lucile Sassatelli[⋆§] and Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, Alberto Del Bimbo. 2022. Deep Variational Learning for Multiple Trajectory Prediction of 360° Head Movements. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3524273.3528176

1 INTRODUCTION

Immersive media are getting more and more popular, particularly with the announced advent of the so-called *Metaverse*[1]. The online experience of such immersive media, such as virtual reality (VR), is however hindered by their bandwidth and latency requirements. For example, 360° videos, meant to be watched in a head-mounted display (HMD), require data rates about two order of magnitude that of regular videos to obtain the same quality perception [47]. To reduce the required data rates, a simple principle is to allocate higher quality levels in the field of view (FoV) of the user. Doing so when streaming requires to predict where the user is going to look at in advance, which may be up to a few seconds ahead of time if one wants to enable playback buffers to absorb network bandwidth variations. The problem may get even more acute if we consider the increase in HMD resolution, where proper prediction should enable higher resolution in the restricted foveal area.

Numerous works have therefore looked into the problem of head motion prediction in 360° images and videos in the last couple of years [10, 11, 52, 73]. However, the performance of existing prediction models is limited by a main characteristic of the head motion data: its intrinsic uncertainty. Very few models have considered this characteristic so far [17, 29, 70], but only heuristically for 360° videos. We illustrate this uncertainty in Fig. 1-left, showing that close past trajectories often lead to diverse/distant future trajectories. This is exemplified for two different users in Fig. 9-left. This has long been identified in other application domains such as autonomous driving [7, 41] or human pose estimation [54]. Such an ambiguity in the data (a same input may be mapped to several outputs) leads to degraded performance and over-fitting. Considering uncertainty in optimization of resource allocation is therefore key to improve systems' performance, as shown in robotic planning [26] and regular video streaming considering bandwidth uncertainty [33, 69].

In this article, we present an approach to generate multiple plausible futures of head motion in 360° videos, given a common past trajectory. Our method provides likelihood estimates of every predicted trajectory, enabling direct integration in streaming optimization. To the best of our knowledge, this is the first work that considers the problem of multiple head motion prediction in 360° videos. Our contributions are:

[1]https://www.cnbc.com/2021/12/27/metas-oculus-virtual-reality-headsets-were-a-popular-holiday-gift.html

• We first analyze head motion data and show the substantial diversity of futures corresponding to close past trajectories, and the shortcoming of a recent predictor in such cases.
• We introduce our discrete variational multiple sequence (DVMS) learning framework, which builds on deep latent variable models. The latent variable is designed to modulate the function connecting the past to the future. Each sample of the latent variable leads to a different plausible future. We design a training procedure to obtain a flexible and lightweight stochastic prediction model compatible with sequence-to-sequence recurrent neural architectures. Experimental results on 3 different datasets show that our method DVMS outperforms competitors adapted from the self-driving domain by up to 37% on prediction horizons up to 5 sec., at lower computational and memory costs.
• We design a method to estimate the respective likelihoods of the multiple predicted trajectories, by showing that the distribution of the prediction error over the latent space has some stationarity, which we exploit. Experimental results on 3 datasets show the quality of these estimates, and how they depend on the video category.

Sec. 2 presents recent work on point-wise and uncertainty-aware prediction of head motion, as well as relevant work on trajectory prediction from the domains of robotics and autonomous driving. Sec. 3 formulates the prediction problem we tackle, positions formally the contribution in the framework of 360° streaming optimization, and motivates the approach with analysis of head motion data. Sec. 4 (i) provides necessary background on deep generative models, (ii) presents our DVMS stochastic prediction model, emphasizing its generality and exemplifying it with a simple recurrent architecture, and (iii) presents experimental results of DVMS on 3 datasets. Sec. 5 shows how to exploit the latent space to estimate trajectory likelihoods, and provides experimental assessment. Sec. 6 discusses the limitations of this work and the perspectives it opens for streaming optimization, and Sec. 7 concludes the article.

The code associated with this article is publicly available at https://gitlab.com/DVMS_/DVMS.

2 RELATED WORK

We first review head motion prediction in 360° videos, with methods producing point-wise trajectory estimates, and methods considering motion uncertainty. We then discuss recent relevant work on multiple trajectory prediction in the domain of robotics with human pose estimation and autonomous driving systems.

2.1 Head motion prediction in 360° videos

Point-wise prediction:

Several approaches have relied on simple regressors or hand-crafted features to produce single trajectory prediction. For example, Chen et al. [11] observed an equatorial posture attraction, and that video genre affects user behavior similarity. They then proposed a FoV prediction algorithm that explicitly balance between the current user's history and the history of other traces, for horizon of up to 4 seconds. Their method requires to have traces of previous users available for every video. Mao et al. [40] presented a coding scheme for interactive applications based on 360° video content, such as VR gaming or conferencing. They also adopt a simple linear FoV prediction method for horizons of 100ms. Recently, Chopra et al. [12] extracted trajectories of moving objects in the video and combine them with autoregressive-filtered past user trajectories to predict future trajectories.

Regression with deep neural networks (DNNs) have also been investigated in several works. Park et al. [48] designed a point-wise prediction method fed with the video content and the past trajectory of the current user. They then fed the predicted FoV coordinates to a model-predictive control (MPC)-based streaming logic. Hou et al. [28] also considered streaming optimization with FoV prediction based on an LSTM architecture predicting the tiles in FoV over the next 2 seconds. Both approaches are similar to a baseline considered by Romero et al. [52]. They first re-examined existing deep-learning approaches and showed that they achieve worse or similar performance as simple baselines (predicting the future position equal to the last past or predicting only from the past head coordinates and not considering the video content). Then they proposed a new deep architecture establishing state-of-the-art performance for head prediction on horizons of up to 5 seconds. Their method enables prediction for new videos where no previous user trace is available. Feng et al. [18] also considered FoV prediction for live content. They underlined that the challenge is to find features of the video content and user behavior that have high correlation with the user's future FoV. They designed a FoV prediction method by collecting the user's real-time trajectory and the semantic description of the attended video regions. FoV is then predicted by finding the tiles with semantic description similar enough with the user's past trajectory encoded as a phrase. Zhang et al. [73] designed a federated learning approach to predict the viewing probability of every tile, considering the video catalog known per user and focus on personalized model training from other users traces. Yu et al. [71] proposed LSTM-based architectures with attention to predict future FoV up to 3 sec. ahead, given past FoV coordinates. Chao et al. [10] proposed a similar approach but with a transformer-based architecture, and showed that it can outperform previous approaches. Finally, several approaches are based on a deep reinforcement learning (DRL) framework where FoV prediction, bandwidth prediction and tile quality decisions are not achieved separately but jointly. Wu et al. [65] proposed one of the most recent such DRL-based approaches for end-to-end control of 360° video streaming. The download horizon is up to 5 seconds.

Considering prediction uncertainty:

How users explore in VR and what commonalities do their viewing patterns exhibit have fostered a lot of interest in the last few years [4, 14, 58]. Almquist et al. [4] showed that the viewing congruence heavily depends on the type of scene, while other works [14, 58] have shown that, upon entering a new scene, the user first goes through an exploration phase where movements are not strongly correlated with the visual content.

To study and cope with user movement uncertainty, several approaches have relied on hand-crafted adaptations. Fan et al. [17] studied spurious head movements that are not related to the scene content. They ran user experiments and attempted to automatically classify such movements. Hu et al. [29] dealt with the uncertainty of FoV prediction by designing a FoV prediction method with a probabilistic model to prefetch video segments into the playback buffer, and enabled chunk replacements to maximize quality in the FoV. Prediction is achieved with a linear regressor trained on data from which are also extracted the parameters of the Gaussian distribution of prediction errors. Feng et al. [19] developed a FoV prediction scheme for live 360° videos that consider various levels of synchronization of the user with the moving objects in the scene. From object detection and optical flow calculation, they linearly predicted future FoV, and dynamically adapted the size of the predicted region to cope with arbitrary moves. Zhang et al. [74] considered FoV prediction over 50-300ms. They proposed a Markov model that learns stationary and transition distributions between discrete angle positions from past users' traces on this video, from the saliency map, and considering human head physical constraints. In contrast to these works relying on single trajectory prediction trying to consider the error distribution around a single mode, our method provides diverse trajectories by design, additionally to their estimated likelihood.

Recent works have presented deep learning approaches to consider prediction uncertainty [33, 69]. In contrast with the vast majority of approaches considering point-wise estimates of future bandwidth for adaptive streaming, both consider the uncertainty of bandwidth prediction in the decision problem of what encoding rate to choose for the next video chunks to send. Both derive probability distribution of the future throughputs, that they feed into an MPC algorithm. Yan et al. [69] designed a neural network to output a discretized probability distribution of predicted transmission times. Kan et al. [33] considered Bayesian neural networks (BNN) to output the probability distribution of future throughput, given the network's historical throughput. In a very recent work, Yang et al. [70] considered predicting multiple head trajectories but only for 360° images, not videos as we do. They consider head trajectory as a succession of fixations and saccades, and intend to learn to capture the uncertainty of head trajectories across different subjects. They resort to a Bayesian neural networks (BNN) approach, to predict, given an input 360° image, multiple head trajectories by sampling the weights of the neural network predictor, the inter-subject variance being modeled with a latent variable conditioning the weight distribution. This approach is the closest to our work, but it differs from ours in several aspects. It considers 360° images, not videos as we do. It generates trajectories for the entire viewing duration, and is meant to model the intrinsic variability between the users, generating the trajectory uncertainty. In our work, we generate future trajectories online over a prediction horizon of 5 seconds and considering past motion of the current user only. We therefore cope not only with inter-user variability, but also with intrinsic uncertainty of the data in how past is correlated with future motion, data uncertainty often referred to as aleatoric uncertainty. Also, BNN are computationally-heavy (the approach from Yang et al. [70] is not real-time) and fit accurately but to just one mode in the data [20]. In this article, we consider a lightweight approach to multiple trajectory prediction, able to predict multiple modes for the future trajectory.

2.2 Multiple trajectory prediction in robotics

Prediction of 360° head motion is closely related to human pose prediction, and more generally to human motion prediction. In the field of robotics, a major challenge is the study of human pose motion, with the aim of generating the future movement of a collection of joints that represent human body. Fragkiadaki et al. [21] developed an encoder-decoder model based on a recurrent neural network (RNN) to process the temporal dynamics of human pose. Afterwards, Jain et al. [32] combined the ability of temporal modeling of RNNs with a spatio-temporal graph to model the interactions between humans and the environment. Many of the state-of-the-art models are based on graph neural networks (GNN) and its evolutions such as the graph convolutional network (GCN) and graph attentional network (GAT) [1, 37, 59]. In these models, each joint is represented as a node and each relation between joints as an edge. In the literature, this problem is still handled in single-modal setting, despite a recent attempt to better consider randomness [49].

The problem of predicting a set of multiple and diverse trajectories has been extensively studied in the field of autonomous driving. There the task is to forecast future positions of moving agents such as cars and pedestrians. Compared to head motion prediction, where predictions are guided by content and user attitude, trajectory forecasting is a more constrained task due to social behavioral rules [3, 25, 31, 36, 55, 72], inertia of moving agents and environmental constraints [7, 9, 36, 42, 61]. Nonetheless, the ability to forecast a multimodal prediction is of fundamental importance for planning secure trajectories for autonomous vehicles.

The first method to generate multiple predictions has been Social-GAN [25], which uses a generative adversarial model to sample multiple outcomes by injecting random noise in an encoder-decoder architecture. Diversity is enforced with the introduction of a variety loss, which optimizes only the best prediction thus leaving the model free to explore the output space with multiple outcomes. The usage of a variety loss is now a common approach for generating multimodal predictions, not only for trajectory forecasting [5, 15, 24, 30, 35, 42, 64]. In the present article, we leverage this domain knowledge by considering the variety loss to enable the training of our DVMS model aiming to produce diverse plausible trajectories.

An extension of such loss, the multimodality Loss, has been introduced by Berlincioni et al. [7], where the authors rely on synthetic data to generate multiple ground truth futures and directly optimize the model to output multiple adequate predictions. This approach requires the ability to generate synthetic samples but replaces the exploration step with an explicit supervision signal. A recent trend in multimodal trajectory forecasting for autonomous driving is to divide the problem into two steps: first, possible goals or endpoints are estimated and then actual trajectories are regressed to reach such intents [16, 38, 75]. Similarly, other approaches use a set of anchors to guide motion prediction following some previously observed samples [27, 42]. We believe that such approaches are less suited for 360° head motion prediction, since motion is mostly guided by content and user attitude rather than constrained maneuvers.

3 MOTIVATION BEHIND MULTIPLE PREDICTION OF HEAD TRAJECTORIES

We first define the prediction problem in Sec. 3.1, then position formally the quantities we aim to approximate in the formulation of 360° streaming optimization in Sec. 3.2. In Sec. 3.3, we analyze head motion data to quantify the diversity of futures corresponding to similar past trajectories, and show the need for multiple future predictions.

3.1 Problem definition

The problem we consider is formally described as follows. We consider that a given 360° video v of duration T seconds is being watched by a user u. The head trajectory of the user is denoted $\mathbf{x}_{0:T}^{u,v}$, with $\mathbf{x}$ storing the head coordinates on the unit sphere (as, e.g., Euler angles, Euclidean coordinates or quaternions).

Online single prediction problem: At any time t in $[0, T]$, predict $\mathbf{x}_{t:t+H}^{u,v}$ with an estimate $\mathbf{y}_{t:t+H}^{u,v}$, that is predict the future trajectory over a prediction horizon H, assuming only $\mathbf{x}_{0:t}^{u,v}$ is known. That is, we do not assume any knowledge of traces other than u on this video v. Hence, for lighter notations, we drop indices u and v from $\mathbf{x}_{0:t}^{u,v}$ and only write $\mathbf{x}_{0:t}$ and $\mathbf{y}_{0:t}$.

Online multiple future prediction problem: At any time t in $[0, T]$, predict K possible future trajectories $\mathbf{y}_{t:t+H}^{k}$, for $k = 1, \ldots, K$, to estimate $\mathbf{x}_{t:t+H}$. This is the general problem definition considered in related work [6, 8]. However, for optimization of heterogeneous quality decisions in a video streaming system, it is also important to estimate the likelihood of every such possible future trajectory. We therefore augment the multiple future prediction problem with estimation of likelihood $Pr[\mathbf{y}_{t:t+H}^{k}|\mathbf{x}_{0:t}]$. This is addressed in Sec. 5 thanks to our variational model proposed in Sec. 4.2.

3.2 Positioning in 360° streaming optimization

The core motivation for our contribution is to provide a stochastic tool, the DVMS learning framework presented in Sec. 4.2, to take into account randomness of the environment in the optimization of resource allocation. Specifically, considering the optimization of spatial heterogeneous quality in streaming 360° videos, one has to consider the variations of network bandwidth and human head position, which both cannot be predicted perfectly. Such stochastic optimization can generally be approached in two ways. First, RL-based approaches [39, 65] do not split the problem into environment prediction and resource allocation, but rather tackle it end-to-end. Other recent works show the benefit, for regular video streaming [33, 69], of splitting the problem and designing a DNN to produce stochastic predictions of bandwidth, which are then considered as parameters in model predictive control (MPC). For example, Yan et al. [69] used dynamic programming to maximize the expected cumulative quality of experience (QoE) as shown in Eq. 1, where H is the look-ahead horizon for download, B_j is the playback buffer's level at chunk j, $QoE(\cdot)$ is the QoE function, K_i^s is chunk i in quality s, and $T(K_j^s)$ is the stochastic download time of this chunk.

$$\max_{K_i^s,\ldots,K_{i+H-1}^s} \sum_{j=i}^{i+H-1} \sum_{t_j} Pr[T(K_j^s) = t_j] QoE\left(K_j^s, K_{j-1}^s, B_j, t_j\right) \quad (1)$$

$$\max_{\{K_{i,l}^s\}_l,\ldots,\{K_{i+H-1,l}^s\}_l} \sum_{j=i}^{i+H-1} \sum_{l=1}^{L} \sum_{t_j} Pr[l \in FoV(j)] \\ Pr[T(K_j^s) = t_j] QoE\left(\{K_{j,l}^s\}_l, \{K_{j-1,l}^s\}_l, B_j, t_j\right) \quad (2)$$

Such formulation enables buffering of H chunks to absorb bandwidth variations. Kan et al. [33] formulated this optimization by projecting the estimated bandwidth distribution onto a confidence interval. In the case of 360° streaming, the equivalent problem can be formulated, incorporating the distribution of the FoV position over the look-head horizon [56] as shown in Eq. 2, with $l \in \{1, L\}$ denoting the tile index, if we consider a tile-based formulation.

In this article, we provide a way to estimate the distribution $Pr\,[l \in FoV(j)]$. To do so, we make a proposal to predict several K trajectories (series of centers of FoV) $\mathbf{y}_{t:t+H}^{k}$, for $k \in \{1, K\}$, with their estimated likelihood $Pr[\mathbf{y}_{t:t+H}^{k}|\mathbf{x}_{0:t}]$. If the problem is tile-based as above, then we can obtain $Pr\,[l \in FoV(j)]$ in Eq. 3.

$$Pr\,[l \in FoV(j)] = \sum_{k=1}^{K} Pr[l \in FoV(j)|\mathbf{y}_{i:i+H-1}^{k}] Pr[\mathbf{y}_{i:i+H-1}^{k}|\mathbf{x}_{0:i}] \\ = \sum_{k: l \in \text{ FoV of center } y_j^k} Pr[\mathbf{y}_{i:i+H-1}^{k}|\mathbf{x}_{0:i}] \quad (3)$$

Once we have at our disposal multiple trajectory estimates and their respective likelihoods, we can express the distribution of the FoV position as a function of these estimates as seen in Eq. 3. This distribution can in turn be used in conjunction with the appropriate QoE function to solve the optimization problem as formulated in Eq. 2. In this article, we focus on the design and evaluation of the prediction methods. System gains will be evaluated in future works.

3.3 Analysis of the need for multiple prediction in head motion data

We now analyze the need for multiple prediction from two perspectives: from the data only, and from the performance of a given predictor on this data. We consider the test data of the MMSys18 dataset, described in Sec. 4.4. In what follows, past (resp. future) trajectories are considered over a horizon of 1 sec. (resp. 5 sec.) as done in recent work [10, 52]. We consider the orthodromic distance defined in Sec. 4.4 between pairs of trajectories. The distance is normalized with the mean length of both trajectories. Every normalized orthodromic distance is augmented with the cosine distance between the last past segments of the pair, to consider orientations of the trajectories as well. We first investigate how the distance between past trajectories relates to the distance between their corresponding true futures. To do so, for each timestamp of each video in the dataset, we consider all pairs of users, and select 100 pairs per video with the closest past trajectories. Every pair of users yields the distance between both past trajectories, and the distance between both respective true future trajectories. Fig. 1-left represents the scatter plot of both distances for every pair. We observe that for 100 pairs of closest

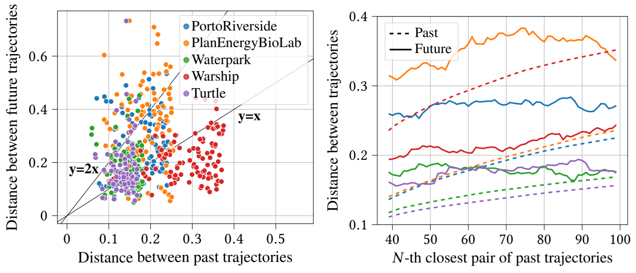

Figure 1: Distances between close past trajectories and their corresponding futures on the test videos of the MMSys18 dataset.

past trajectories, there is a non-negligible number of pairs whose distance between future elements is at least twice the distance between their past elements. Also, we observe that for close past elements, more distant futures are produced, on this dataset, for exploration-type videos *PortoRiverside* and *PlanEnergyBioLab*. Specifically, for *PortoRiverside*, 71% of points are above the $y = x$ line and 23% of points are above $y = 2x$. For *PlanEnergyBioLab*, 83% of points are above $y = x$ and 42% of points above $y = 2x$. Fig. 1-right represents the distance between past elements and the distance between future elements, for every N-th closest pair, with $N \leq 100$ (distances are smoothed with moving average). It confirms that the average future distance is generally higher that the past distance, with a greater difference obtained for exploration videos.

Finding: This is an indication that relatively close past trajectories may lead to distinct/farther apart future trajectories, which may create difficulties when attempting to train a prediction model on such data. Indeed, a (neural) regressor trained with the regular mean square error (MSE) cannot map similar inputs to different outputs. Second, we investigate predictions made by a recent deep predictor on this data. Thanks to the reproducible framework provided by Romero et al. [52], we consider their prediction model named TRACK. Fig. 2 represent how the distance between predicted future trajectories match the distance between their corresponding true futures, with the same form as Fig. 1, with the pairs still formed by closest past trajectories. Fig. 2-left shows that, given close past trajectories, the predicted trajectories are much closer together than the true future trajectories. Specifically, for all of the videos except *Warship*, the proportion of points above y=x ranges from 89% (*PlanEnergyBioLab*) to 100% (*PortoRiverside*), and the proportion of points above y=2x ranges from 67% (*Waterpark*) to 80% (*PortoRiverside*). This is confirmed in Fig. 2-right showing the difference between the average in-between true futures distances and in-between predicted futures distances.

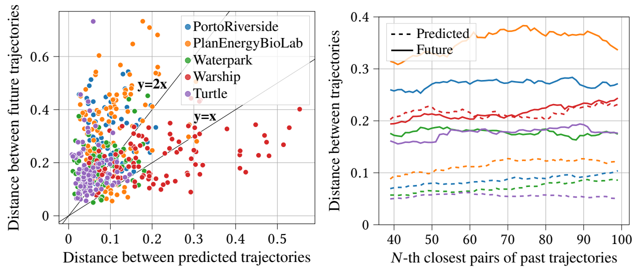

Figure 2: Distances between predicted trajectories with closest past and their corresponding futures, on the test videos of the MMSys18 dataset.

Finding: This is an indication that predicted trajectories have less diversity than true trajectories.

Finally, we investigate the connection between diversity (distance) of the true futures, and the prediction error. Fig. 3 shows the evolution of the maximum prediction error for each pair of the same 100 closest past trajectories per video against the distance between the respective future trajectories. Given close past trajectories, the prediction error tends to increase when the distance between the true futures is higher.

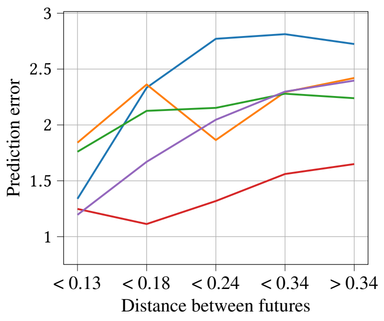

Figure 3: Prediction error of TRACK [52] against distance between pairs of true futures. The colors are associated with the video ids and are the same as in Fig. 2.

Finding: This is an indication that the predictor is less accurate when there is diversity in the true futures.

While this finding might have been expected, we considered important to experimentally verify that (i) there is diversity in head motion traces as seen in Fig 1, (ii) the diversity of ground truth data is not properly reproduced by a refined recent predictor considering both past motion and visual content as shown in Fig. 2, and (iii) this diversity of futures indeed contributes to the error of single trajectory predictor as illustrated in Fig. 3. Requirements for streaming optimization, the significant diversity of future trajectories relatively to their close respective past trajectories, and the increased error rate of existing predictors in such cases, are therefore solid rationale for designing multiple trajectory prediction methods.

4 DEEP STOCHASTIC PREDICTION OF MULTIPLE HEAD TRAJECTORIES

We first provide necessary background on deep generative models in Sec. 4.1. We present our proposal for a multiple prediction framework in Sec. 4.2, exemplified with an architecture in Sec. 4.3. Sec. 4.4 presents performance results.

4.1 Background on deep generative models for sequences

Trajectory prediction can be cast into conditional sequence generation, for which we provide some background next. Deep generative approaches are meant to generate data, modelling either explicitly or implicitly training data distribution. Variational auto-encoders (VAE) [34, 50] and generative adversarial networks (GAN) [23] are two prominent such families of approaches.

In this article, we focus on the VAE family owing to their capability not to narrowly focus on a few modes of the data distribution, hence being a better fit to the characteristics of head motion data as described in Sec. 3.3. VAE frameworks aim to enable the generation of high-dimensional data samples by sampling a normally distributed low dimensional latent variable $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. A sample x is then generated by passing z through a high-capacity model, particularly a deep neural network. The latent variable is meant to capture the minimum number of independent random dimensions from the data, while the decoding by a neural network of z into x is meant to capture the complex dependencies in a sample [34]. The typical representation of a VAE is illustrated in Fig. 4. Denoting the decoder's parameters with θ, the generative model is typically defined with $p(z)$ and $p_\theta(x|z)$. For the decoder network to be trained, the posterior distribution $p(z|x)$ is required, but can only be approximated with the output distribution, named the approximate posterior $q_\phi(z|x)$, of another neural network of parameters ϕ and usually referred to as the *encoder* or *inference network*. VAE can also be declined into conditional VAE (CVAE) when the goal is to generate output variables y from input variables x, drawing z from a prior distribution $p_\theta(z|x)$ to generate y from the decoder with $p_\theta(y|x, z)$ [60].

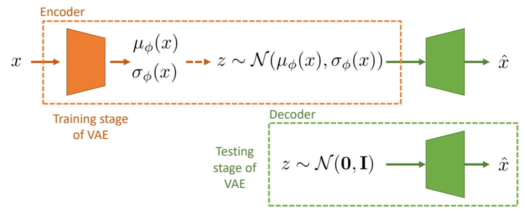

Figure 4: Schematic representation of a VAE.

When considering sequence prediction formalized in Sec. 3.1, prediction of a time series over a certain horizon is often made conditionally to the past of the time series. This is often translated into sequence-to-sequence architectures, where a so-called *encoder* processes the past (even at test time, different to VAE), produces an intermediate embedding, which is then decoded into a future trajectory (see Fig. 5). The architecture of the encoder and decoder networks are often based on recurrent neural networks (RNNs), such as LSTM or GRU [52, 62]. Note that the concept of encoder of past trajectory in a sequence-to-sequence architecture is different from the term *encoder* used in a variational context (aka *inference network*, as mentioned above).

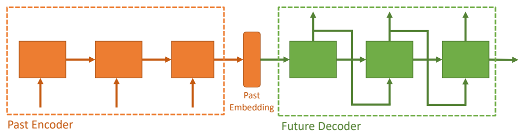

Figure 5: A sequence-to-sequence architecture.

Deep variational learning has initially been designed for image data. More recently, variational approaches have been proposed for sequence data and so-called *structured output prediction*. These approaches are diverse depending on where the random latent variables are considered in the recurrent architectures [6, 13, 49]. For example, Babaeizadeh et al. [6] performed conditional video prediction to predict future frames until final video time T, conditioned on c initial frames, by sampling from $p(x_{c:T}|x_{0:c-1})$. A random latent vector z is picked at random from the prior distribution $p(z) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at test time (while the training is made as usual with z sampled from the approximate posterior $q_\phi(z|x_{0:T})$). They show the performance in multiple future frame sequence prediction, specifically in PSNR and SSIM of the 10% best sequences obtained from 100 samples of z.

Alahi et al. [49] considered the randomness in human pose forecasting, which they decompose into trajectory forecasting and local pose forecasting. They advocate that the latter has higher randomness, which they tackle by considering an LSTM-based sequence-to-sequence architecture as done by Martinez et al. [43], but they set the initial hidden state of the decoder to a latent vector z drawn from $\mathcal{N}(\mu(h_t), \sigma(h_t))$ where h_t is the latest hidden state of the encoder of the past coordinates, and $\mu(\cdot)$ and $\sigma(\cdot)$ are functions implemented with fully connected layers. The training of such RNN-based VAEs can be difficult to converge and unstable [6, 8]. This is particularly due to the fact that during training, z is sampled from the approximate posterior $q_\phi(z|x_{0:T})$ while it can only be sampled from $p_\theta(z|x_{0:c-1})$ in test, and despite a KL divergence component in the training loss meant to nudge $q_\phi(z|x_{0:T})$ towards $p_\theta(z|x_{0:c-1})$.

With this background on variational approaches for sequence generation, we now present our learning framework for multiple prediction of head trajectories.

4.2 Discrete Variational Multiple Sequence (DVMS) prediction

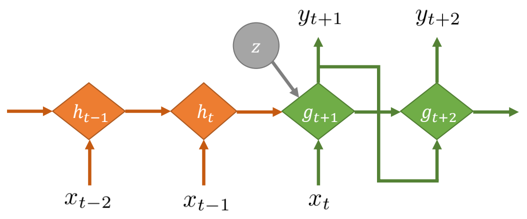

Figure 6: Probabilistic graphical model of the proposed stochastic discrete variational multiple sequence (DVMS) prediction framework. A random variable is represented with a circle, a deterministic state with a diamond.

We now present a new learning framework for multiple head motion trajectory prediction, named discrete variational multiple sequence (DVMS). It builds on deep latent variable models like VAEs. DVMS is designed to be compatible with any sequence-to-sequence architecture. The rationale for such design is as follows. Our goal is to design a framework for multiple prediction of head motion for deep architectures, which provides key properties:

P1 sufficiently diverse predictions $\mathbf{y}^k_{t:t+H}$, for $k = 1, \ldots, K$,
P2 state-of-the-art performance when $K = 1$,
P3 estimates of likelihoods of the predicted trajectories,
P4 flexibility and low computational cost.

Generative model: The probabilistic graphical model of DVMS is depicted in Fig. 6. For any encoder fed with past sequence $\mathbf{x}_{0:t}$, an embedding h_t is produced. This embedding is then concatenated with a unique latent variable z. The latent variable is key in our DVMS proposal. This latent variable is meant to capture the variations in the function relating the future sequence to the past sequence, hence acting as a parameter in the past-to-future mapping. The resulting concatenation produces the first hidden state g_t of the decoder. Considering that the encoder is made of recurrent connections with hidden state h_t, the generative model writes as Eq. 4, where $\mathbb{U}_{\mathcal{Z}_K}$ denotes the uniform distribution over discrete set $\mathcal{Z}_K$, and MLP stands for multi-layer perceptron to denote one or several fully connected (FC) layers. To generate multiple prediction, every $z_k \in \mathcal{Z}_K$

generates a future trajectory $\mathbf{y}^k_{t:t+H}$. To enable diverse predictions (P1), we do not constrain the distribution $p(z)$ we sample from to be conditioned on $x_{0:t}$ in test, in contrast to what was done by Alahi et al. [49], but instead draw z uniformly in $\mathcal{Z} \in [-1, 1]^d$ (where d is the dimension of vector z). To meet (P3), z is drawn from a discrete set $\mathcal{Z}$ with K elements. Indeed, z codes for latent features parameterizing the expression of the future trajectory from the past trajectory. In other words, different values of z allow for the representation of different *modes* of future trajectories, given the same past. If there is some stationarity in how likely is every trajectory produced from every z_k, then we can exploit this stationarity for likelihood estimation (P3). We therefore consider a discrete fixed set of possible z values to ease this exploitation, which we describe in Sec. 5.

$$\begin{aligned}
h_t &= \mathrm{RNN}_{enc}\left(h_{t-1}, \mathbf{x}_{t-1}\right), \quad h_0 = 0 \\
z &\sim \mathbb{U}_{\mathcal{Z}_K} \\
g_{t+1} &= \mathrm{MLP}(h_t, z) \\
\mathbf{y}_t &= \mathbf{x}_t \\
\mathbf{y}_{t+s} &= \mathrm{FC}(g_{t+s}) + \mathbf{y}_{t+s-1}\,, \quad \text{for } s \geq 1 \\
g_{t+s} &= \mathrm{RNN}_{dec}\left(g_{t+s-1}, \mathbf{y}_{t+s-1}\right)\,, \quad \text{for } s \geq 2
\end{aligned} \tag{4}$$

Training procedure: To ensure (P2), we enforce the prior distribution $p(z)$ z is sampled from at training time to be the same as in test (contrary to work from Babaeizadeh et al. [6]), i.e., we do not consider an inference network. This allows to avoid the mismatch between $p(z)$ and $q(z|x_{0:T})$, which impedes the training convergence, as described in Sec. 4.1. However, doing so also adds noise to the sequence decoder which, if trained with gradient descent performed over every sampled trajectory obtained from z_k, for all $k \in \{1, K\}$, learns to discard the z input and only produces a single trajectory corresponding to the baseline, as described by Babaeizadeh et al. [6]. To avoid this phenomenon, we instead train our architecture with the *best of many samples* (BMS) loss [8], also named the *variety loss* [25, 63], defined in Eq. 5.

$$\mathcal{L}(\mathbf{x}_{0:t}, \theta) = \min_{k \in \{1,K\}} D\left(\mathbf{y}^k_{t:t+H}, \mathbf{x}_{t:t+H}\right) \tag{5}$$

where $D(\cdot)$ can be any distance between two trajectories on the sphere. This loss thus consists, for every past trajectory sample, in selecting sample z_{k^*} generating the best match to the single ground truth future. The gradient descent is hence performed only on a single k^* sample out of the trajectories generated by the model. This prevents the architecture from learning to discard z as being an uninformative input for prediction.

DVMS is flexible (P4) because it can be used with any sequence-to-sequence architecture, being it an architecture processing video content [52] in case of streaming of stored content, or an architecture processing only the past user's trajectory [10] in case of live streaming. Indeed, Bayesian methods like BNN [45] and Monte-Carlo dropout [22] require to change how every network weight is considered in train (generating multiple weight samples). In contrast, DVMS only consists in adding a latent variable to modulate the initial state of the sequence-to-sequence decoder with a random component, independently of the actual structure of the sequence-to-sequence encoder and decoder.

DVMS is also lightweight (P4) because the additional training cost, w.r.t. the original sequence-to-sequence architecture, only comes from the latent variable z to be concatenated with the encoder's last hidden state (MLP to learn in Eq. 4). This additional cost is also limited because we do not learn an approximate posterior $q(z|\mathbf{x}_{0:t})$, that is an additional neural network (named *inference network* in Fig. 4 and used only in train), but rather directly sample z from $\mathcal{U}_{\mathcal{Z}_K}$ both in test and train.

All 4 properties (P1)-(P4) are experimentally evaluated in Sec. 4.4 and 5.3.

4.3 Proposal of a DVMS-based architecture

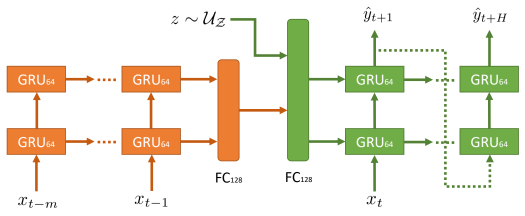

Figure 7: Proposed example of a DVMS-based architecture.

To demonstrate the interest of the proposed DVMS learning framework of multiple head trajectory prediction, in this section we propose a simple architecture akin to those presented in [49, Fig. 2] or [52, Fig. 4]. This architecture is of type sequence-to-sequence and is represented in Fig. 7. It is however simplified compared to the previous literature, as we consider double-stacked gated recurrent units (GRU) instead of single or double-stacked LSTM. Here we purposefully do not consider the visual content in order to simplify the presentation and analysis of our contribution which is on the variational framework DVMS for multiple future prediction, and not on a specific neural architecture. So other architectures can be incorporated in our framework, such as based on more advanced recurrent techniques like transformers [10] or fusion of multimodal input considering the visual content [52]. We discuss more this compatibility in Sec. 6.

Architecture: We set $d = 1$ as the dimension of z. The encoder is made of a doubly-stacked GRU with 64 neurons (and default GRU activations). The final GRU's hidden state is then fed to a 128-neuron fully connected layer. The output of this layer is concatenated with z and fed to another 128-neuron fully connected layer. The decoder is also a doubly-stacked GRU with same hyper-parameters as the encoder. The past sequence is restricted to $\mathbf{x}_{t-m:t}$ with $m =$ 1sec., matching recent work [10, 52], and we set $H =$ 5sec. as the prediction horizon. The sampling rate of the scanpaths is 5Hz, thus the past sequences are 5 sample-long, and the future sequences are 25 sample-long.

Training procedure: The model is trained using the loss described in Eq. 5. Distance $D(\cdot)$ is taken as the cumulated Euclidean distance, that is $D\left(\mathbf{y}^k_{t:t+H}, \mathbf{x}_{t:t+H}\right) = \sum_{s=0.2}^{H} \left\|\mathbf{y}^k_{t+s} - \mathbf{x}^k_{t+s}\right\|^2$. The optimizer is Adam with weight decay (AdamW), with a learning rate of $5{\times}10^{-4}$ and a batch size of 64.

4.4 Results on multiple trajectory prediction

In this section we assess (P1) the diversity of predictions, (P2) the performance for $K = 1$, and (P4) the computational cost. Likelihood estimation (P3) is addressed in Sec. 5.

4.4.1 Experimental settings.

Datasets: We consider 3 datasets of 360° videos with head motion traces:

- MMSys18 [14]: This dataset consists of the head positions of 57 subjects watching 19 360° videos lasting 20 seconds.
- PAMI18 [66]: It is made of both the eyes and head positions of 58 participants watching 76 360° videos, lasting 10 to 80 seconds (25 seconds on average).
- CVPR18 [67]: This dataset contains the gaze positions of 45 users over 208 360° videos, lasting 15 to more than 80 seconds (36 seconds on average).

For all of these datasets, we use the same split as described in the supplemental material from by Romero et al. [52], such that there is no overlap between the videos in the train and test sets for PAMI18 and CVPR18 as well as no overlap between the users for MMSys18. Additionally, we do not make predictions for the first 6 seconds of the video with any of the considered competitors, as done by Romero et al. [52] to skip the user's initial exploration phase.

Metrics: When it comes to evaluating the quality of the multiple predictions, the major challenge is that several plausible futures may correspond to a single input, but the datasets provide only a single ground-truth future. The best way to assess (P1) is therefore to check if the known ground truth is covered by one of the few predictions, while the others can efficiently explore the search space to cover the futures of close inputs. This can be done by using the *winner-take-all* or *best of many samples* (BMS) metric [8]. Therefore, as is usually done as standard practice in multiple sequence prediction [6, 41, 61], we report the BMS metric. Specifically, BMS at prediction step s is defined in Eq. 6, where the orthodromic distance between points P_1 and P_2 on the unit sphere is $\text{orthdist}(P1, P2) = \arccos(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\lambda$ with ϕ the latitude and λ the absolute difference in longitude, and k^* is defined in Eq. 7.

$$\frac{1}{U}\frac{1}{V}\frac{1}{T}\sum_u\sum_v\sum_t \text{orthdist}(y^{k^*}_{u,v,t+s}, x_{u,v,t+s}) \tag{6}$$

$$k^*(u,v,t) = \arg\min_k \sum_{s=0.2}^{H} \text{orthdist}(y^{k}_{u,v,t+s}, x_{u,v,t+s}) \tag{7}$$

We report the BMS metric in figures, and we report in a more compact form in tables the average prediction error, which is the average over $s \leq H$ of the BMS metric, similarly to what Marchetti et al. reported [41]. For $K = 1$, the BMS metric amounts to the orthodromic distance, hence enabling the assessment of (P2) with the same metric as used for single sequence prediction [10, 52].

Competitors: We compare our models with 3 competitors. As no competitor exists so far for multiple prediction of head motion, we adapt a recent method from the autonomous driving domain.

• *Deep-position-only*: *Deep-position-only* is a baseline introduced by Romero et al. [52]. It is a simple sequence-to-sequence LSTM taking past head positions as input. Additional details can be found in section 3.2 of [52]. Thanks to the reproducible framework they published [51], we were able to re-implement *Deep-position-only* using PyTorch and achieve the same performance as reported.

• MANTRA-*adapted*: MANTRA is an approach described by Marchetti et al. [41] to predict the trajectory of other vehicles. It uses an autoencoder in conjunction with a memory network. The autoencoder is first trained to reconstruct future trajectories from past and future trajectories. A memory-writing controller is then trained to fill the memory with embeddings from the encoder. The memory takes the form of a (key, value) dictionary, where the embeddings of past trajectories are the keys that are used to retrieve the values, embeddings of future trajectories. At prediction time, embeddings of yet unseen past trajectories are computed and matched with keys from the memory. The K most similar keys are used to retrieve the K corresponding values, which are then fused with the embedding of the actual past and decoded into K predicted future trajectories. Memory is built at training time with the following procedure. During training, if none of the predicted trajectories is close enough (defined by a manual threshold) to the ground truth future trajectory, the embeddings (past and future) of this trajectory are added as new key and value to the memory. The loss for the writing controller is designed so that it only writes relevant trajectories into the memory. Embeddings that are too similar and do not help to decrease the prediction error are not added to the memory. At test time, the memory is read-only and filled with embeddings from the training set. For this model to work properly, the trajectories have to be normalized so that they are translation and rotation-invariant.

Building from this approach, we build a MANTRA-*adapted* model as a multiple trajectory prediction baseline to be compared to our proposed model. The changes from the original MANTRA model are described as follows. The trajectories are 3-dimensional instead of 2-dimensional. We adapt the manual distance thresholds used for the writing controller with values that fit our data and give an acceptable memory size. We do not normalize the trajectories in the same way. As there is no rotation invariance in head motion, we carried out several tests with translation invariance (separating yaw and pitch). The results were best when only re-centering on yaw (longitude). The results were worse with re-centering both axes, only pitch or with no re-centering. Since video cue is not considered in DVMS, thus not providing any contextual information or map, MANTRA-*adapted* does not employ any contextual cue either, such as the "Iterative Refinement Module" [41], which integrates information from the map.

• VPT360: VPT360 is the recurrent transformer-based viewport prediction architecture presented by Chao et al. [10]. We do not reproduce their results because the code is not available at the time of writing, but we report the values presented in their work [10] and compare DVMS with VPT360 on the exact same settings.

4.4.2 Experimental results.

Prediction error: Fig. 8-left represents, for the MMSys18 dataset, the orthodromic distance and the BMS metric of DVMS for $K \in \{1, 5\}$. The shaded area represents the 95% confidence interval. We observe that DVMS achieves performance similar for $K = 1$ as *Deep-position-only*, which is the state of the art it builds on, and close to VPT360, hence meeting (P2). For (P1), we observe for $K = 2$ a 28% reduction in prediction error at prediction step $s = 3$sec. with DVMS, compared to the single prediction competitors *Deep-position-only* and VPT360. For higher K, the error reduction increases, and tends to saturate for $K = 4$ then $K = 5$. DVMS hence meets both (P1) and (P2) on this dataset. Still for the MMSys18 dataset, Fig. 8-right, compares the performance of DVMS with the MANTRA-*adapted* competitor. We first notice that for every $K = 1, \dots, 5$, DVMS yields a lower prediction error than MANTRA-*adapted*. Also, we observe that MANTRA-*adapted* does not match the state of the art performance of *Deep-position-only* for $K = 1$. For every value of K, DVMS yields lower prediction errors than MANTRA-*adapted*. At $s = 3$sec., for $K = 1, \dots, 5$, DVMS achieves a prediction error lower than MANTRA-*adapted* by 26.82%, 26.77%, 32.01%, 34.05% and 36.20%, respectively. The results on the CVPR18 and PAMI18 datasets are presented in Tables 1 and 2. VPT360 does not appear because its performance is not available for these datasets over $H = 5$sec.. Tables 1 and 2 show that DVMS achieves similar performance as *Deep-position-only* for $K = 1$, and outperforms MANTRA-*adapted* for every value of K, on both datasets. Constructing futures by combining past with future pieces from the training set does not seem sufficient for MANTRA-*adapted* to produce diverse enough futures, compared with DVMS which instead modulates the initial state of the sequence decoder with a random component.

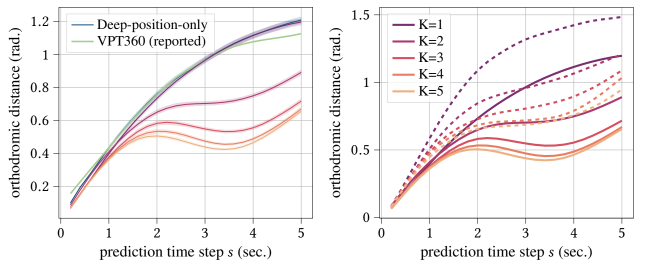

Figure 8: Prediction error (in orthodromic distance or BMS metric for $K > 1$) of predictors on the MMSys18 dataset. Left: DVMS (ours, same colors as right figure) vs *Deep-position-only* and VPT360. Right: DVMS (ours, solid lines) vs MANTRA-adapted (dashed lines).

Method		Average prediction error				
		$s \le 1$s	$s \le 2$s	$s \le 3$s	$s \le 4$s	$s \le 5$s
Deep-position-only		0.204	0.357	0.471	0.555	0.618
MANTRA-adapted	K=1	0.351	0.594	0.767	0.887	0.981
	K=2	0.323	0.503	0.608	0.678	0.755
	K=3	0.298	0.456	0.544	0.598	0.656
	K=4	0.292	0.433	0.500	0.544	0.596
	K=5	0.303	0.426	0.487	0.533	0.581
DVMS (ours)	K=1	0.200	0.355	0.470	0.555	0.618
	K=2	0.200	0.326	0.394	0.435	0.477
	K=3	0.190	0.305	0.357	0.384	0.419
	K=4	0.187	0.295	0.337	0.355	0.387
	K=5	**0.186**	**0.287**	**0.321**	**0.335**	**0.366**

Table 1: Prediction error over all $s \le H$ on the CVPR18 dataset.

The results on all 3 datasets therefore show that DVMS is able to produce diverse predictions (P1), outperforming the multiple prediction competitor MANTRA-*adapted*, while providing similar performance to single prediction when $K = 1$ (P2).

Fig. 9 shows qualitative examples of multiple trajectory prediction. It shows similar past trajectories of two different users, yielding distant future trajectories. When $K = 2$, we observe that DVMS produces different plausible trajectories, where one matches best the first user and the other the second user. **Computational cost**: Hardware used to train and test the methods is a Nvidia RTX 3080 with 10GB of video RAM on a station with 128GB of RAM. Table 3 shows that DVMS and MANTRA-*adapted* have significantly less weights than both single prediction methods *Deep-position-only* and VPT360. While DVMS has more neural network parameters than MANTRA-*adapted*, the execution time to generate a trajectory at test time is 14% less than MANTRA-*adapted*. This is due to MANTRA-*adapted* having to do memory lookup. Indeed MANTRA-*adapted* has an extra memory, which DVMS does not, and the size of this memory, shown in Table 4 in percentage of the training set size, varies with the target accuracy and the dataset (and hence cannot be generalized to other datasets before actual training). Also, training

Method		Average prediction error				
		$s \le 1$s	$s \le 2$s	$s \le 3$s	$s \le 4$s	$s \le 5$s
Deep-position-only		0.134	0.231	0.303	0.352	0.386
MANTRA-adapted	K=1	0.236	0.429	0.571	0.666	0.736
	K=2	0.211	0.343	0.426	0.479	0.530
	K=3	0.202	0.313	0.375	0.417	0.457
	K=4	0.186	0.291	0.351	0.389	0.428
	K=5	0.194	0.290	0.342	0.378	0.413
DVMS (ours)	K=1	0.135	0.233	0.304	0.353	0.387
	K=2	0.127	0.207	0.253	0.284	0.313
	K=3	0.128	0.202	0.239	0.262	0.289
	K=4	**0.124**	0.192	0.224	0.244	0.271
	K=5	0.125	**0.189**	**0.218**	**0.235**	**0.259**

Table 2: Prediction error over all $s \le H$ on the PAMI18 dataset.

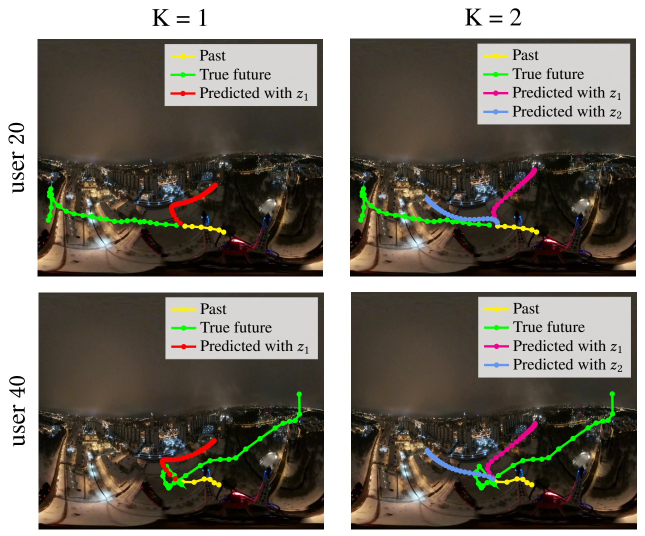

Figure 9: Examples of generated trajectories. Two different users (rows) have close past trajectories for the same timestamp $t = 8$sec. of the same video *DroneFlight* (MMSys18 dataset), but their future trajectories are significantly different. Predicting only one future (left column) does not enable good prediction of both futures, while predicting multiple (right column) does.

MANTRA-*adapted* requires two phases, the first to train the auto-encoder, the second for the memory writing controller.

Method	# parameters	Prediction time (ms)
Deep-position-only	4.21M	15.71
VPT360	6.3M	N/A
MANTRA-adapted	**76k**	6.81
DVMS (ours)	110k	**5.87**

Table 3: Computational cost of the different models.

Dataset	MMSys18	CVPR18	PAMI18
Training set size	12600	560342	271440
K=1	6413 (50.90%)	258758 (46.18%)	79503 (29.29%)
K=2	3601 (28.58%)	142113 (25.36%)	34564 (12.73%)
K=3	2041 (16.20%)	83702 (14.94%)	20059 (7.39%)
K=4	1227 (9.74%)	50265 (8.97%)	10470 (3.86%)
K=5	**811 (6.44%)**	**36325 (6.48%)**	**7632 (2.81%)**

Table 4: Memory size (in number and percentage of training samples) of the MANTRA-*adapted* method for different number of predicted trajectories K across all the datasets.

Hence, on 3 datasets of head motion data, DVMS achieves better prediction performance for lower computational resources than single prediction methods VPT360 and *Deep-position-only*, and for lower or equivalent resources than MANTRA-*adapted*.

5 EXPLOITING DVMS LATENT SPACE TO ESTIMATE TRAJECTORY LIKELIHOOD

In this section, we present how we leverage our variational predictor DVMS to estimate the likelihood of the multiple predicted trajectories. First, we present the general approach to likelihood estimation in Sec. 5.1. Second, we study in Sec 5.2 a stationarity hypothesis we make to produce our estimator. Third, we demonstrate in Sec. 5.3 the performance of our likelihood estimator, including disaggregated results and analysis over video categories.

5.1 Definition of the likelihood estimator

For a regression problem, the likelihood $Pr[\mathbf{y}^k_{t:t+H}|\mathbf{x}_{0:t}]$ of a future trajectory can be expressed with $\exp^{-D(\mathbf{y}_{t:t+H},\mathbf{x}_{t:t+H})}$, hence estimating the likelihood is equivalent to estimating the distance of a trajectory to the ground truth, that is the negative log-likelihood. We denote by $err^k_{u,v,t}$ the error of the k-th generated trajectory $\mathbf{y}^k_{t:t+H}$, the motion of user u on video v at timestamp t, defined in Eq. 8.

$$err^k_{u,v,t} = D\left(\mathbf{y}^k_{t:t+H}, \mathbf{x}_{t:t+H}\right) \tag{8}$$

With a variational framework, a standard approach to estimate the likelihood would be to rely on the model (whose parameters are set from the training data) and on the known past $\mathbf{x}_{0:t}$. In this work, we argue that this is not sufficient, and that other available information must be considered, namely the past generated trajectories $\mathbf{y}^k_{s:\min(s+H,t)}$, for all $k \in \{1, K\}$ and $s \in [0, t]$, and the errors obtained by every such trajectory when compared to the available ground truth at t $\mathbf{x}_{s:\min(s+H,t)}$. Indeed, these errors are informative of which z_k, for $k \in \{1, K\}$, have best coded the latent features connecting the future trajectory with the past trajectory. If the errors over the various z_k, for $k \in \{1, K\}$, have sufficient stationarity in time, then we can exploit such stationarity to estimate the likelihood of the predicted trajectories. We therefore define an estimate the estimate $\widehat{err}^k_{u,v,t}(r)$ of $err^k_{u,v,t}$ in Eq. 9.

$$\widehat{err}^k_{u,v,t}(r) = D\left(\mathbf{y}^k_{t-r:\min(t-r+H,t)}, \mathbf{x}_{t-r:\min(t-r+H,t)}\right) \tag{9}$$

where r is a past window of size controlling the age of the trajectory ground truth to produce the error estimate. Let us recall that the z-space is discrete, with $\mathcal{Z}_K = \{z_k\}^K_{k=1}$. This means that $err^k_{u,v,t}$ is predicted by the error produced by the trajectory $\mathbf{y}^k_{t-r:t-r+H}$ generated with the same z_k and predicted at time $t - r$ over a horizon H, but with the error only counted on the timestamps for which the ground truth $\mathbf{x}^{u,v}$ is available, i.e., on $[t - r, \min(t - r + H, t)]$.

The accuracy of this estimator therefore depends on the stationarity in time of the distribution of the error over the latent values z_k, for $k \in \{1, K\}$. We study this stationarity next.

5.2 Study of the stationarity of error distribution in the latent space of DVMS

We first analyze how $err^k_{u,v,t}$ evolves over $t = t_{start} : T$, for given test videos v and users u from the MMSys18 dataset. The MMSys18 test set is made of 5 videos of 4 categories [4, 52]: exploration (*PortoRiverside* and *PlanEnergyBioLab*), moving focus (*Turtle*), static focus with camera motion (*WaterPark*), static focus without camera motion (*Warship*). Fig. 10-left shows an example for video *PortoRiverside* and user $u = 56$. We observe that, for every $t \in \{t_{start} : T\}$, the future trajectory $\mathbf{y}^3_{t:t+H}$ produced by latent value z_3 consistently yields the lowest prediction error. Fig. 10-right shows that, for video *Turtle* and user $u = 28$, the lowest error is consistently produced by latent value z_2.

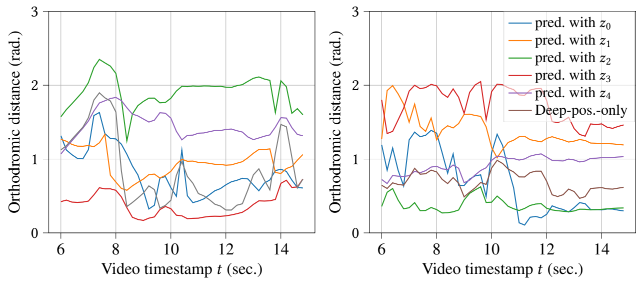

Figure 10: Prediction error for different latent values over time, test set of the MMSys18 dataset. Left: video *PortoRiverside*, user 56. Right: video *Turtle*, user 28.

$$A^v_{ij} = \frac{1}{UK} \sum_{u=1}^{U} \sum_{k=0}^{K} \left(err^k_{u,v,i} - err^k_{u,v,j}\right)^2, (i, j) \in \{t_{start}, \ldots, T\}^2 \tag{10}$$

Second, as Fig. 10 only shows examples obtained for specific choices of (v, u), we investigate how representative these cases are. To do so, we define a *latent stationarity matrix* (LSM) per video v defined in Eq. 10, where U is the total number of user traces per video. For

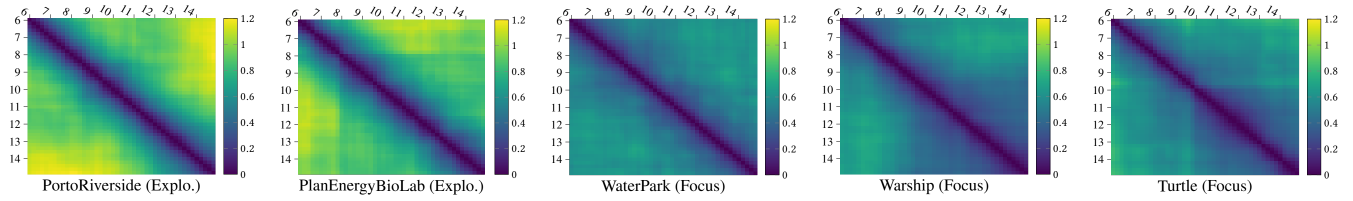

Figure 11: Latent Stationarity Matrix (LSM). The color scale codes for the error difference A^v_{ij}. Axes are in seconds ($t = 6$ sec. to $t = 15$ sec. so $t + H \leq T = 20$ sec.). Test videos of the MMSys18 dataset.

every user u, the main term in the summation represents the difference, between timestamps i and j, in how the error is distributed in the discrete latent space $\mathcal{Z}_K$. Fig. 11 depicts such error differences as heatmaps for all the five videos of the test set of the MMSys18 dataset. First, we observe that, when the timestamps i and j are equal, error difference is null, which is expected. Also, it is interesting to observe that the closer i and j, the lower the error distance. This shows that the results illustrated above for specific (u, v) pairs are general: the prediction error yielded by z_k varies more or less slowly over time (depending on the videos). Such stationarity may hence be exploited to produce error (i.e., likelihood) estimates. Similar qualitative results hold on the other datasets, but we do not show these maps owing to space limitation. However, estimation results on all 3 datasets are shown in Sec. 5.3. Second, it is interesting to observe that the level of stationarity/speed of variation of the error produced by every z_k, for $k \in \{1, K\}$, depends on the video category. For both exploration-type videos *PortoRiverside* and *PlanEnergyBioLab*, the error difference for a given timestamp difference is significantly higher than for the focus-type videos. The stationarity of latent errors is hence lower in exploration videos, and error estimates from the past timestamps should yields better estimates in Focus videos, which we investigate in the next section.

5.3 Results on trajectory likelihood estimation

Datasets: The results below are obtained on the same 3 datasets as those considered in Sec. 4.4.

Metrics: We measure the quality of the trajectory likelihood/error estimate with its Pearson correlation coefficient with the ground truth. Indeed, as the motivation for the present contribution is to benefit from such estimates in a stochastic formulation of resource optimization (see Sec. 3.2), we want to evaluate how much are the produced estimates linearly correlated with the ground truth error. If the correlation coefficient was 1, we would obtain the true likelihood. For any past window size r, for every tuple (v, u, t), we are interested in how the corresponding K samples $\left\{(err^k_{u,v,t}, \widehat{err}^k_{u,v,t}(r))\right\}^K_{k=1}$ are correlated. To allow considering a larger number of samples to obtain more confident estimates of the correlation coefficient, we consider $\left\{(err^k_{u,v,t}, \widehat{err}^k_{u,v,t}(r))\right\}_{k,u,t}$. However, to do so without artificially increasing the correlation coefficient owing to different average values of the samples over the (u, t) tuples, we first normalize $\left\{(err^k_{u,v,t}, \widehat{err}^k_{u,v,t}(r))\right\}^K_{k=1}$ independently for every (u, t). The correlation coefficient for every value of r is then computed on the normalized pairs of ground truth error and error estimate.

Results: Figures 12-14 show the evolution of the Pearson correlation coefficient (PCC) with the past window size $r \in [0, H]$, with the shaded area representing the 95% confidence interval. To assess the correlation strength, we follow the recommendation from recent literature [2, 68]: low: $0.1 \leq |\text{corr}| < 0.3$; moderate; $0.3 \leq |\text{corr}| < 0.6$; high: $0.6 \leq |\text{corr}| \leq 1.0$.

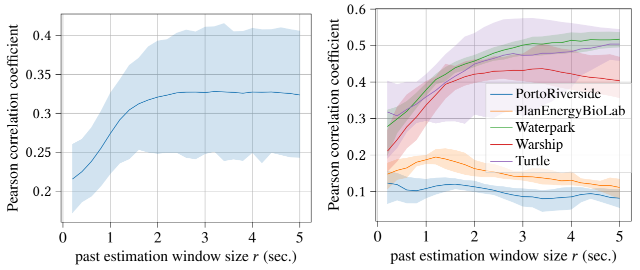

Figure 12: PCC between estimated and ground truth error of predicted trajectories, on the MMSys18 dataset. Left: average over all test videos. Right: average per test video.

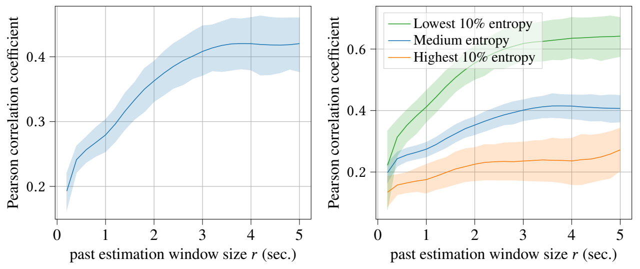

Figure 13: PCC between estimated and ground truth error, on the PAMI18 dataset. Left: average over all test videos. Right: average per group.

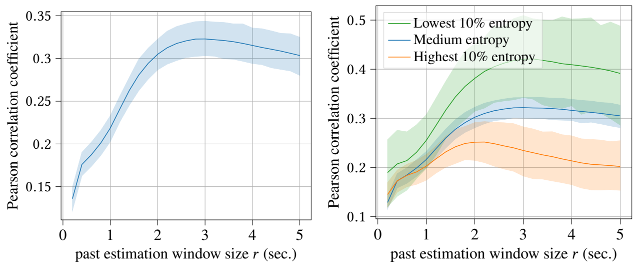

Figure 14: PCC between estimated and ground truth error, on the CVPR18 dataset. Left: average over all test videos. Right: average per group.

It is important to note that the LSM is a novel type of characterization for 360° video. While previous characterizations were directly based on the video content or the user traces [4, 44, 53], the LSM represents characteristics of the latent space in connection with prediction performance. Investigating how the LSM can be a novel characterization for different videos, and possibly different users, is left for future work. In this work, we develop a useful application of such characterization for deep learning-based predictors.

For the test sets of the PAMI18 and CVPR18 datasets, Fig. 13-left and 14-left show that there is a moderate significant correlation of the estimates with the ground truth errors. For the test set of the MMSys18 dataset, there is a moderate to low correlation, the significance being low possibly owing to the very low number of videos in the test set (only 5). We also observe that the correlation generally increases with r for the MMSys18 and PAMI18 datasets in Fig. 12 and 13 (reaching maximum level for $r \geq 2$ sec. and $r \geq 3.5$ sec., respectively), while for the CVPR18 dataset in Fig. 14, the average correlation reaches a maximum for $r = 3$ seconds then decreases. It is interesting to disaggregate the results and analyze the correlation per video type. Fig. 12-right shows the correlation results for each of the 5 test videos in the MMSys18 dataset. For both videos of type exploration, *PortoRiverside* and *PlanEnergyBiolab*, the correlation is significantly low. However for the focus-type videos (*Warship*, *WaterPark* and *Turtle*), the correlation is moderate, being significant for the first two for $2 \leq r \leq 3$ sec., while for *Turtle*, the performance in unstable (it is worth noting that *Turtle* is of type moving focus video, which is not present in the training set). We also notice that the maximum correlation is obtained for $r \leq 1.5$ sec. for exploration videos, while the maximum is obtained for $r \geq 3$ sec. for focus-type videos. The other two datasets have 16 (PAMI18) and 42 (CVPR18) test videos. Therefore, to categorize the video automatically between exploration and focus-type video, we resort to the method presented by Romero et al. [52], consisting in associating exploration (resp. focus) videos with low (resp. high) entropy values of the saliency maps obtained from the user traces. Fig. 13-right and 14-right show the correlation results broken down into the 10% of videos with highest entropy, 10% with lowest entropy, and the rest. In Fig. 13-right, we observe that for PAMI18 focus videos, the correlation is strong and significant for $r \geq 3$ sec., while it is low significant for exploration videos and moderate significant for the rest. Similar trends can be observed with CVPR18 videos, where the correlation is low and significant for exploration videos, and moderate and significant for the rest. For exploration videos, the correlation is maximum for $r \sim 2$ sec., while it is maximum for $r \sim 3$ sec. for the rest.

Therefore, we have shown that our method is able to predict multiple diverse trajectories, providing estimates of their respective likelihood to leverage in stochastic optimization of resource allocation. The estimates are shown to correlate with ground truth, the level of correlation and past window size r providing highest correlation depending on the video category.

6 DISCUSSION

To the best of our knowledge, this article presents the first proposal to generate multiple plausible 360° head trajectories. Our work therefore establishes a first baseline for comparison, and paves the way for more principled approaches to stochastic optimization of 360° streaming, and immersive streaming in general. Indeed, the DVMS approach proposed here is general and can be adapted to 6DoF motion, and more innovative streaming systems where, e.g., ultra-high resolution headset would require to narrow the high-quality region below the FoV to the foveated area, or interactive and user-adaptive strategies could be adaptively triggered based on the expected predictability of the user's movements [57].

Considering uncertainty in prediction is often approached with Bayesian neural networks [33, 45, 70]. However, these approaches are computationally intensive and do not allow to capture mode diversity in the data [20]. Alternatives exist to better learn data diversity, e.g., Monte-Carlo dropout [22], or approaches based on memory networks like [41]. In contrast, our DVMS method builds on deep latent variable models, and proves lightweight, flexible and suited to the head motion data diversity. It confirms the interest in investigating more dynamic VAE, to possibly design proper inference networks and conditional prior $p(z|\mathbf{x}_{0:t})$ in test.

We have exemplified the DVMS prediction framework with a video-agnostic neural architecture. A direct perspective is to investigate DVMS performance when used in conjunction with a content-aware architecture. DVMS is compatible by design with sequence-to-sequence architectures, such as those used in the head motion prediction literature [46, 52, 67]. Finally, it will be most interesting to analyze DVMS latent space in connection with trajectory features (such as angular acceleration between past and future, etc.), with user and video profiles to characterize attentional phases and motion predictability.

7 CONCLUSION

In this article, we presented the first method for multiple head motion prediction in 360° videos. We showed that the need for multiple trajectories is motivated by the diversity of future trajectories in real-world user data. Our main contribution is a new learning framework, called DVMS, which builds on deep latent variable models and allows to predict multiple future trajectories from a given past. We design a training procedure to obtain a flexible and lightweight stochastic prediction model compatible with sequence-to-sequence architectures. Experimental results on 3 datasets show that DVMS matches relevant baselines for single trajectory prediction and outperforms competitors adapted from the self-driving domain by up to 37%, on prediction horizons up to 5 seconds. By exploiting the stationarity of the prediction error over the latent space, our method provides likelihood estimates of every predicted trajectory, enabling direct integration in streaming optimization. DVMS paves the way for multiple head motion prediction in 360° videos, and future works will evaluate the gains when predicted trajectories and their likelihoods are used by a 360° adaptive streaming system.

ACKNOWLEDGMENTS

This work has been partly supported by the French government, through the UCA JEDI and EUR DS4H Investments in the Future projects ANR-15-IDEX-0001 and ANR-17-EURE-0004. This work was partly supported by EU Horizon 2020 project AI4Media, under contract no. 951911 (https://ai4media.eu/). This work was performed using HPC/AI resources from GENCI-IDRIS (Grant 2020-AD011012209).

REFERENCES

[1] Vida Adeli, Mahsa Ehsanpour, Ian Reid, Juan Carlos Niebles, Silvio Savarese, Ehsan Adeli, and Hamid Rezatofighi. 2021. TRiPOD: Human Trajectory and Pose Dynamics Forecasting in the Wild. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 13390–13400.

[2] Haldun Akoglu. 2018. User's guide to correlation coefficients. *Turkish Journal of Emergency Medicine* 18, 3 (2018), 91–93.

[3] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 961–971.

[4] Mathias Almquist, Viktor Almquist, Vengatanathan Krishnamoorthi, Niklas Carlsson, and Derek Eager. 2018. The prefetch aggressiveness tradeoff in 360° video streaming. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 258–269.

[5] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. 2019. Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories With GANs. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. https://doi.org/10.1109/CVPRW.2019.00359

[6] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. 2018. Stochastic Variational Video Prediction. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. ICLR.

[7] Lorenzo Berlincioni, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2021. Multiple Future Prediction Leveraging Synthetic Trajectories. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 6081–6088. https://doi.org/10.1109/ICPR48806.2021.9412158

[8] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. 2018. Accurate and Diverse Sampling of Sequences based on a "Best of Many" Sample Objective. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. https://doi.org/10.1109/CVPR.2018.00885

[9] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. 2019. Argoverse: 3D tracking and forecasting with rich maps. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 8748–8757. https://doi.org/10.1109/CVPR.2019.00895

[10] Fang-Yi Chao, Cagri Ozcinar, and Aljosa Smolic. 2021. Transformer-based Long-Term Viewport Prediction in 360° Video: Scanpath is All You Need. In *IEEE 23nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE. https://doi.org/10.1109/MMSP53017.2021.9733647

[11] Jinyu Chen, Xianzhuo Luo, Miao Hu, Di Wu, and Yipeng Zhou. 2021. Sparkle: User-Aware Viewport Prediction in 360-Degree Video Streaming. *IEEE Transactions on Multimedia* 23 (2021), 3853–3866.

[12] Lovish Chopra, Sarthak Chakraborty, Abhijit Mondal, and Sandip Chakraborty. 2021. PARIMA: Viewport Adaptive 360-Degree Video Streaming. In *Proceedings of the Web Conference 2021*. ACM, 2379–2391.

[13] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. 2015. A Recurrent Latent Variable Model for Sequential Data. In *Advances in Neural Information Processing Systems 27 (NIPS)*. MIT Press, Cambridge, MA, USA, 2980–2988. https://doi.org/10.5555/2969442.2969572

[14] Erwan J. David, Jesús Gutiérrez, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. 2018. A dataset of head and eye movements for 360° videos. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 432–437. https://doi.org/10.1145/3204949.3208139

[15] Lavinia De Divitiis, Federico Becattini, Claudio Baecchi, and Alberto Del Bimbo. 2021. Garment recommendation with memory augmented neural networks. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*. Springer, 282–295. https://doi.org/10.1007/978-3-030-68790-8_23

[16] Patrick Dendorfer, Aljosa Osep, and Laura Leal-Taixe. 2020. Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation. In *Proceedings of the 15th Asian Conference on Computer Vision (ACCV)*. Springer. https://doi.org/10.1007/978-3-030-69532-3_25

[17] Xiaoxiong Fan, Yun Cai, Yufei Yang, Tianxing Xu, Yike Li, Songhai Zhang, and Fanglue Zhang. 2021. Detection of scene-irrelevant head movements via eye-head coordination information. *Virtual Reality & Intelligent Hardware* 3 (2021), 14. https://doi.org/10.1016/j.vrih.2021.08.007

[18] Xianglong Feng, Weitian Li, and Sheng Wei. 2021. LiveROI: region of interest analysis for viewport prediction in live mobile virtual reality streaming. In *Proceedings of the 12th ACM Multimedia Systems Conference (MMSys '21)*. ACM, Istanbul Turkey, 132–145. https://doi.org/10.1145/3458305.3463378

[19] Xianglong Feng, Viswanathan Swaminathan, and Sheng Wei. 2019. Viewport Prediction for Live 360-Degree Mobile Video Streaming Using User-Content Hybrid Motion Tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 2 (2019), 43:1–43:22.

[20] Stanislav Fort, Clara Huiyi Hu, and Balaji Lakshminarayanan. 2020. Deep Ensembles: A Loss Landscape Perspective. https://openreview.net/forum?id=r1xZAkrFPr

[21] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent Network Models for Human Dynamics. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Los Alamitos, CA, USA, 4346–4354. https://doi.org/10.1109/ICCV.2015.494

[22] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. JMLR.org, New York, NY, USA, 1050–1059. https://doi.org/10.5555/3045390.3045502

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[24] Jiaqi Guan, Ye Yuan, Kris M Kitani, and Nicholas Rhinehart. 2020. Generative hybrid representations for activity forecasting with no-regret learning. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 173–182. https://doi.org/10.1109/CVPR42600.2020.00025

[25] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2255–2264.

[26] David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*. Curran Associates, Inc. https://doi.org/10.5555/3327144.3327171

[27] Zhao He and Richard P. Wildes. 2021. Where are you heading? Dynamic Trajectory Prediction with Expert Goal Examples. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE.

[28] Xueshi Hou, Sujit Dey, Jianzhong Zhang, and Madhukar Budagavi. 2021. Predictive Adaptive Streaming to Enable Mobile 360-Degree and VR Experiences. *IEEE Transactions on Multimedia* 23 (2021), 716–731.

[29] Han Hu, Zhimin Xu, Xinggong Zhang, and Zongming Guo. 2019. Optimal Viewport-Adaptive 360-Degree Video Streaming Against Random Head Movement. In *2019 IEEE International Conference on Communications (ICC)*. IEEE, Shanghai, China, 1–6. https://doi.org/10.1109/ICC.2019.8761189

[30] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. 2019. STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 6271–6280. https://doi.org/10.1109/ICCV.2019.00637

[31] Boris Ivanovic and Marco Pavone. 2019. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2375–2384. https://doi.org/10.1109/ICCV.2019.00246

[32] Ashesh Jain, Amir Roshan Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-RNN: Deep Learning on Spatio-Temporal Graphs. *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5308–5317. https://doi.org/10.1109/CVPR.2016.573

[33] Nuowen Kan, Chenglin Li, Caiyi Yang, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2021. Uncertainty-aware robust adaptive video streaming with bayesian neural network and model predictive control. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '21)*. ACM, New York, NY, USA, 17–24.

[34] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. ICLR.

[35] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, Hamid Rezatofighi, and Silvio Savarese. 2019. Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc. https://doi.org/10.5555/3454287.3454300

[36] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. 2017. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 336–345.

[37] Zhenguang Liu, Pengxiang Su, Shuang Wu, Xuanjing Shen, Haipeng Chen, Yanbin Hao, and Meng Wang. 2021. Motion Prediction Using Trajectory Cues. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 13299–13308. https://doi.org/10.1109/ICCV48922.2021.01305

[38] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. 2020. It Is Not the Journey But the Destination: Endpoint Conditioned Trajectory Prediction. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 759–776. https://doi.org/10.1007/978-3-030-58536-5_45

[39] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[40] Yixiang Mao, Liyang Sun, Yong Liu, and Yao Wang. 2020. Low-latency FoV-adaptive Coding and Streaming for Interactive 360° Video Streaming. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20)*. ACM, New York, NY, USA, 3696–3704. https://doi.org/10.1145/3394171.3413751

[41] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2020. MANTRA: Memory Augmented Networks for Multiple Trajectory Prediction. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 7141–7150. https://doi.org/10.1109/CVPR42600.2020.00717

[42] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. 2020. Multiple trajectory prediction of moving agents with memory augmented networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020). https://doi.org/10.1109/TPAMI.2020.3008558

[43] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On Human Motion Prediction Using Recurrent Neural Networks. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, 4674–4683. https://doi.org/10.1109/CVPR.2017.497

[44] Afshin Taghavi Nasrabadi, Aliehsan Samiei, Anahita Mahzari, Ryan P. McMahan, Ravi Prakash, Mylène C. Q. Farias, and Marcelo M. Carvalho. 2019. A Taxonomy and Dataset for 360° Videos. In *Proceedings of the 10th ACM Multimedia Systems Conference (MMSys '19)*. ACM, New York, NY, USA, 273–278.

[45] Radford M. Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer. https://doi.org/10.1007/978-1-4612-0745-0

[46] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. 2018. Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction. In *Proceedings of the 26th ACM International Conference on Multimedia (MM '18)*. ACM, New York, NY, USA, 1190–1198. https://doi.org/10.1145/3240508.3240669

[47] Jounsup Park, Philip A. Chou, and Jenq-Neng Hwang. 2019. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 149–162.

[48] Sohee Park, Arani Bhattacharya, Zhibo Yang, Samir R. Das, and Dimitris Samaras. 2021. Mosaic: Advancing User Quality of Experience in 360-Degree Video Streaming With Machine Learning. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 1000–1015.

[49] Behnam Parsaeifard, Saeed Saadatnejad, Yuejiang Liu, Taylor Mordan, and Alexandre Alahi. 2021. Learning Decoupled Representations for Human Pose Forecasting. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. IEEE, Montreal, BC, Canada, 2294–2303. https://doi.org/10.1109/ICCVW54120.2021.00259

[50] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*. JMLR.org, II–1278–II–1286. https://doi.org/10.5555/3044805.3045035

[51] Miguel Fabián Romero-Rondón, Lucile Sassatelli, Ramón Aparicio-Pardo, and Frédéric Precioso. 2020. A unified evaluation framework for head motion prediction methods in 360° videos. In *Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20)*. ACM, 279–284.

[52] Miguel Fabián Romero-Rondón, Lucile Sassatelli, Ramón Aparicio-Pardo, and Frédéric Precioso. 2021. TRACK: A New Method from a Re-examination of Deep Architectures for Head Motion Prediction in 360-degree Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[53] Silvia Rossi and Laura Toni. 2020. Understanding User Navigation in Immersive Experience: An Information-Theoretic Analysis. In *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE '20)*. ACM, New York, NY, USA, 19–24. https://doi.org/10.1145/3386293.3397115

[54] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. 2017. Learning in an Uncertain World: Representing Ambiguity Through Multiple Hypotheses. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 3591–3600. https://doi.org/10.1109/ICCV.2017.388

[55] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. 2019. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1349–1358. https://doi.org/10.1109/CVPR.2019.00144

[56] Lucile Sassatelli, Marco Winckler, Thomas Fisichella, Ramon Aparicio, and Anne-Marie Pinna-Déry. 2019. A New Adaptation Lever in 360° Video Streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '19)*. ACM, New York, NY, USA, 37–42. https://doi.org/10.1145/3304112.3325610

[57] Lucile Sassatelli, Marco Winckler, Thomas Fisichella, Antoine Dezarnaud, Julien Lemaire, Ramon Aparicio-Pardo, and Daniela Trevisan. 2020. New interactive strategies for virtual reality streaming in degraded context of use. *Computers & Graphics* 86 (2020), 27–41. https://doi.org/10.1016/j.cag.2019.10.005

[58] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. 2018. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1633–1642. https://doi.org/10.1109/TVCG.2018.2793599

[59] Theodoros Sofianos, Alessio Sampieri, Luca Franco, and Fabio Galasso. 2021. Space-Time-Separable Graph Convolutional Network for Pose Forecasting. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 11209–11218. https://doi.org/10.1109/ICCV48922.2021.01102

[60] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems 28 (NIPS)*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc.

[61] Shashank Srikanth, Junaid Ahmed Ansari, Sarthak Sharma, et al. 2019. INFER: INtermediate representations for FuturE pRediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)*. IEEE.

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*. MIT Press, Cambridge, MA, USA, 3104–3112.

[63] Luca Thiede and Pratik Brahma. 2019. Analyzing the Variety Loss in the Context of Probabilistic Trajectory Prediction. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, South Korea, 9953–9962. https://doi.org/10.1109/ICCV.2019.01005

[64] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. 2016. An uncertain future: Forecasting from static images using variational autoencoders. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*. Springer, 835–851. https://doi.org/10.1007/978-3-319-46478-7_51

[65] Chenglei Wu, Zhi Wang, and Lifeng Sun. 2021. PAAS: a preference-aware deep reinforcement learning approach for 360° video streaming. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'21)*. ACM, Istanbul Turkey, 34–41.

[66] Mai Xu, Yuhang Song, Jianyi Wang, Minglang Qiao, Liangyu Huo, and Zulin Wang. 2019. Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 11 (2019), 2693–2708.

[67] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. 2018. Gaze Prediction in Dynamic 360° Immersive Videos. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 5333–5342.

[68] Tong Xue, Abdallah El Ali, Gangyi Ding, and Pablo Cesar. 2021. Investigating the Relationship between Momentary Emotion Self-reports and Head and Eye Movements in HMD-based 360° VR Video Watching. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–8. https://doi.org/10.1145/3411763.3451627

[69] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 495–511. https://doi.org/10.5555/3388242.3388279

[70] Li Yang, Mai Xu, Yichen Guo, Xin Deng, Fangyuan Gao, and Zhenyu Guan. 2021. Hierarchical Bayesian LSTM for Head Trajectory Prediction on Omnidirectional Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). https://doi.org/10.1109/TPAMI.2021.3117019

[71] Jiang Yu and Yong Liu. 2019. Field-of-view prediction in 360-degree videos with attention-based neural encoder-decoder networks. In *Proceedings of the 11th ACM Workshop on Immersive Mixed and Virtual Environment Systems (MMVE '19)*. ACM, New York, NY, USA, 37–42. https://doi.org/10.1145/3304113.3326118

[72] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M. Kitani. 2021. AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 9813–9823. https://doi.org/10.1109/ICCV48922.2021.00967

[73] Ran Zhang, Jiang Liu, Fangqi Liu, Tao Huang, Qinqin Tang, Shangguang Wang, and F. Richard Yu. 2021. Buffer-Aware Virtual Reality Video Streaming with Personalized and Private Viewport Prediction. *IEEE Journal on Selected Areas in Communications* (2021). https://doi.org/10.1109/JSAC.2021.3119144

[74] Xue Zhang, Gene Cheung, Yao Zhao, Patrick Le Callet, Chunyu Lin, and Jack Z. G. Tan. 2021. Graph Learning Based Head Movement Prediction for Interactive 360 Video Streaming. *IEEE Transactions on Image Processing* 30 (2021), 4622–4636. https://doi.org/10.1109/TIP.2021.3073283

[75] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, Congcong Li, and Dragomir Anguelov. 2020. TNT: Target-driveN Trajectory Prediction. In *Proceedings of the 2020 Conference on Robot Learning*. PMLR.

A ARTIFACT APPENDIX

A.1 Abstract

This artifact appendix contains information about the source code for the DVMS-based architecture that we propose. Information about the model, datasets, and required libraries is provided.

We explain how to run the experiments and reproduce the results that we obtain in the article by including code as well as a Jupyter notebook. Model trained weights are also provided to allow for easy reproducibility.

A.2 Artifact check-list

- **Model:** DVMS-based sequence-to-sequence neural network implementation using PyTorch with 110k parameters.
- **Datasets:** 3 datasets described in Sec 4.4 included in the repository.
- **Run-time environment:** Python3 with required libraries.
- **Hardware:** GPU with at least 2GB of VRAM is recommended. Running the experiments without GPU is possible, but takes considerably more time.
- **Experiments:** the model can be trained and evaluated using different parameters.
- **Output:** Training output: LOG text file and PTH weights file. Evaluation output: Pickle error file and Pickle trajectory file.
- **Approximate disk space required:** 3GB including evaluation output.
- **Approximate time needed to complete experiments (GPU):** Training: 72 (14.4) hours for all K on all datasets sequentially (5 in parallel). Evaluation: 10 (2) minutes for all K on all datasets sequentially (5 in parallel). Notebook: 5 minutes. Training and evaluation can be done in parallel if more than 2GB of VRAM available.
- **Approximate time needed to complete experiments (CPU):** Training: 125 hours for all K on all datasets sequentially. Evaluation: 10 minutes for all K on all datasets sequentially. Notebook: 5 minutes. As PyTorch already uses multiple CPU cores during a single run, training and evaluation cannot be done in parallel without GPU.
- **Publicly available:** Yes, public GitLab repository.
- **Code license:** GNU GPL v3.0.

A.3 Description

A.3.1 How to access. The code and datasets are publicly available on a public GitLab repository. The repository may be cloned by running the following command:

```
git clone https://gitlab.com/DVMS_/DVMS.git
```

A.3.2 Software dependencies. Python3 is needed to run the experiments and visualize the results. All the required libraries are specified in a pip requirements file at the root of the repository. More details are available in the repository README file.

A.3.3 Datasets. The datasets used for the experiment and mentioned in the article are already included in the repository and ready to use. There is no need for additional download.

A.3.4 Model. An implementation of the DVMS-based architecture that we propose in Sec. 4.3 is available in a single dedicated file `models/dvms.py` and can be re-used separately. We also include the code to train and test the model in the same conditions as we did for reproducibility.

A.4 Installation

There is no need for specific installation apart from the required libraries mentioned in Sec. A.3.2.

We recommend using a Python virtual environment by using the Python `venv` module. More details are available in the repository README or in the official Python documentation[1].

A.5 Experiment workflow

A.5.1 Training the model. Training DVMS is done using the `--train` flag of `training_procedure.py`. Examples of commands to train the model are provided in `train.sh`. We also provide trained weights of the model corresponding to every command in `train.sh` to allow for direct evaluation.

A.5.2 Evaluating the model. Evaluating DVMS is done using the `--evaluate` flag of `training_procedure.py`. Examples of commands to evaluate the model are provided in `evaluate.sh`. Model weights resulting from a training with the same parameters are necessary for model evaluation.

A.5.3 Visualizing the results. Visualizing the results is done using the notebook under `notebooks/visualize_error.ipynb`. First, start a Jupyter server by following the instructions in the README. Running the notebook once the model has been evaluated will generate one figure and one table for each dataset the model has been tested on.

A.6 Evaluation and expected results

Notebook outputs are already provided and correspond to the outputs obtained when running the evaluation on the provided model weights. These outputs match the results presented in Fig. 8, Table 1, and Table 2 of the article.

A.7 Experiment customization

The model can be re-trained using different command-line parameters as detailed when typing `python training_procedure.py -h`. Internal parameters of the model may also be changed in the constructor of the `DVMS` class in `models/dvms.py`.

A.8 Running the experiments without a GPU

If you do not have a GPU, the requirements are slightly different and the version of PyTorch in the pip requirements file should be changed, as detailed in the repository README file. Running the experiments without a GPU also requires to change the command line options of `training_procedure.py`. More details are available in the repository README file.

[1]https://docs.python.org/3/library/venv.html