Representing Spatial Trajectories as Distributions

Dídac Surís
Columbia University
didac.suris@columbia.edu

Carl Vondrick
Columbia University
vondrick@cs.columbia.edu

Abstract

We introduce a representation learning framework for spatial trajectories. We represent partial observations of trajectories as probability distributions in a learned latent space, which characterize the uncertainty about unobserved parts of the trajectory. Our framework allows us to obtain samples from a trajectory for any continuous point in time—both interpolating and extrapolating. Our flexible approach supports directly modifying specific attributes of a trajectory, such as its pace, as well as combining different partial observations into single representations. Experiments show our method's advantage over baselines in prediction tasks. See trajectories.cs.columbia.edu for video results and code.

1 Introduction

The visual world is full of objects moving around in predictable ways. Examples of these *spatial trajectories* include human motion, such as people dancing or exercising; objects moving, such as a ball rolling; trajectories of cars and bicycles; or animal migration patterns. Evidence suggests that the human perceptual system encodes motion into high-level neural codes that represent the motion holistically, going beyond the specific input observations [22]. Humans use this abstract representation for downstream tasks like inferring intention [6]. Computer vision systems likely need similar mechanisms to encode trajectories and motions into global representations.

Representation learning has been transformative in other domains such as images and text for its ability to obtain high-level representations that reorganize the information in the input, and are better at downstream tasks than the original signals. A global representation of trajectories would allow us to evaluate a trajectory at any point in time, even ones not yet observed. However, modeling trajectories presents a series of challenges for representation learning. First, in real-time scenarios, the future of the trajectory is never observed. Second, temporal and spatial occlusions may impede observing part of a trajectory. Third, trajectories are by nature continuous in time. And finally, a trajectory-level metric is usually not well defined and application-dependent.

We propose a representation learning framework for trajectories that deals with all these challenges in a unified way. Our key contribution is the representation of a partial observation of a trajectory as a probability distribution in a learned latent space, that represents all the possible trajectories the observation could have been sampled from. Our framework's simplicity and generality allows it to be flexible: it does not constrain the input-space metric, accepts observations of different lengths and at any (irregularly sampled) point in time, can be implemented using different families of latent space distributions, and is capable of performing inference-time tasks for which it has not been explicitly trained. Our experiments on human movement datasets show that our method can accurately predict the past and future of a trajectory segment, as well as the interpolation between two different segments, outperforming autoregressive baselines. Additionally, it can do so for any continuous point in time. We also show how we can modify given trajectories by manipulating their representations.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

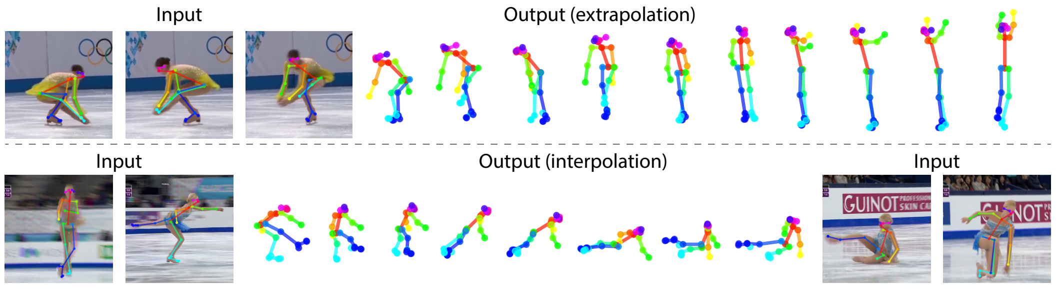


Figure 1: **Predictions on figure skating data (FisV)**. Our model is capable of predicting the future (top row), past, and interpolation (bottom row) of a trajectory given partial observations, at any continuous time. The inputs to the model are the keypoints in the images. See more examples in Fig. 4.

2 Method

2.1 Framework, Definitions and Notation

The input to our framework is a sequence of samples obtained from a (continuous in time and infinite) spatial trajectory u, which we define as the continuous temporal evolution of a set of spatial coordinates. We call each sample a *point* x, which lives in the *input space* $\mathbb{R}^K$. We call the sequence of points, together with the times t at which they were sampled, a *segment* s, which can be understood as a partial observation of u. We define a distance metric δ between points x in the input space.

Our goal is to transform these measurements of motion s into a representation z that will be useful for downstream tasks. We define a *latent space* $\mathbb{R}^N$ of *trajectories* z. Each z in this space represents the full extent of a trajectory, both in time and in space. We use Q to represent probability distributions over trajectories z, in the latent space. We define a distance function D between distributions of trajectories, which assumes an underlying distance function d between trajectories z.

We use an encoder Θ to encode every segment s to a probability distribution $Q(\cdot; s) = \Theta(s)$ over trajectories, where $Q(z; s)$ represents the probability that s was sampled from the trajectory represented by z. Additionally, we can decode a trajectory z at a specific time t by using a decoder Φ, obtaining a point $x = \Phi(z, t)$. Φ takes any continuous t as input. See Fig. 2 for a schematic.

2.2 Representation Learning

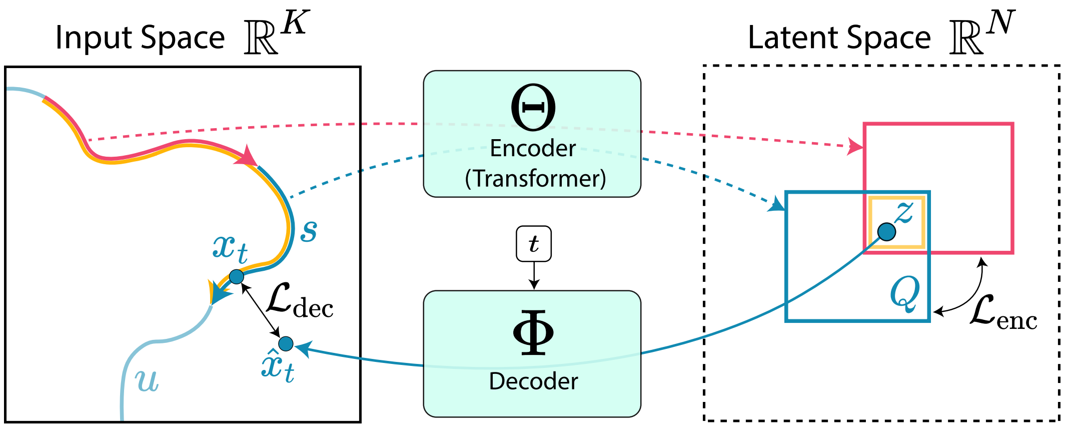
When observing a segment s, one may have some uncertainty about the specific trajectory it was sampled from. For instance, a segment showing a person jumping may correspond to a trajectory that continues with the person falling, or to a trajectory that proceeds with them doing a backflip and landing on their feet, but it will not belong to a trajectory of a person swimming. Therefore, we represent the segment as a distribution over trajectories, where $Q(z; s)$ represents the likelihood of a trajectory given the segment. During training, the goal is to learn this mapping from the input space (segments of trajectories) to the latent space (distributions over trajectories).

Concretely, given two segments s^a, s^b that have been obtained from the same underlying trajectory, we want some z to exist such that its likelihood under the distributions Q^a and Q^b representing each of the segments is high. To encourage this, we train the model to maximize the overlap between the distributions Q^a and Q^b. Similarly, we minimize the overlap between (the distribution representations of) segments sampled from different trajectories, under the assumption that no trajectory z exists that contains both segments. Specifically, we minimize a self-supervised triplet loss:

$$\mathcal{L}_{\text{enc}} = \sum_{(i,k^+,k^-)\in\mathcal{T}} \max\left[D\left(Q^i, Q^{k^+}\right) - D\left(Q^i, Q^{k^-}\right) + \alpha, 0\right], \tag{1}$$

where α is a margin hyperparameter, and $\mathcal{T}$ is a set of triplets: for every segment i in the dataset, we define several triplets by sampling pairs consisting of a positive segment k^+ (such that (i, k^+) is a positive pair) and a negative segment k^- (such that (i, k^-) is a negative pair).

In addition to learning representations of trajectories, we also wish to be able decode them back to input-space points. To achieve this, we train a decoder Φ that allows us to obtain the specific value



Input Space $\mathbb{R}^K$
Latent Space $\mathbb{R}^N$
Θ
Encoder (Transformer)
t
Φ
Decoder
s
x_t
$\mathcal{L}_{\text{dec}}$
$\hat{x}_t$
u
z
Q
$\mathcal{L}_{\text{enc}}$

Figure 2: **Schematic of our framework**. We show the input space $\mathbb{R}^K$, the latent space $\mathbb{R}^N$, and the mappings between the two (encoder Θ and decoder Φ). A segment s belonging to a trajectory u is encoded into a distribution Q, from which a trajectory z is sampled and decoded at a time t, to get $\hat{x}_t$.

of any trajectory at any continuous time t. In order to train the decoder Φ, we sample trajectories $z \sim Q(\cdot; s)$ from each segment representation, and decode them at specific time-steps t which were contained in s, obtaining a prediction $\hat{x}_t = \Phi(z, t)$ for which we have ground truth x_t. There is no uncertainty in this prediction, as x_t was part of the segment s in the first place; the decoder is only explicitly trained for reconstruction, not extrapolation. We train the decoder via regression, using the *point-wise* distance δ. Note that we never explicitly define a trajectory-level distance in the input space; it is implicitly learned by the model. The reconstruction loss is mathematically defined as:

$$\mathcal{L}_{\text{dec}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{z \sim \Theta(s^i)} \sum_{t}^{T^i} \delta\left(\Phi(z, t), x_t^i\right), \tag{2}$$

where N is the number of segments in the dataset, and T^i is the number of points in segment s^i. We minimize Eqs. (1) and (2) jointly and end-to-end. We implement the encoder Θ using a Transformer Encoder architecture [54], and the decoder Φ using a ResNet [17]. See Appendix C for more details.

2.3 Creating Positive and Negative Pairs

In order to define positives and negative pairs for Eq. (1), we use the following:

- **Input-space relationships**. The simplest way is to take segments from the same trajectory as positives and segments from other random trajectories as negatives. The initial segments can have different relationships, such as precedence, containment, or overlap [3]. In our experiments, we sample three segments for every trajectory: a *past* segment (**P** in Fig. 3), a *future* segment (**F**) whose starting time comes right after the end of the past segment, and a *combination* segment (**C**), which contains both the past and the future segments.
- **Intersection**. An intersection **I** of two distributions Q in the latent space will represent all the trajectories that have a high likelihood for both intersected segments. Note that an intersection in the latent space is a union in the input space: the intersection constrains the possible trajectories to those that are consistent simultaneously for the two segments. Similarly, an intersection in the input space (assuming an overlap between segments) is a union in the latent space. In the latent space, the intersection of the past and future segments should be equal to the representation of the combination segment, and therefore the pair (**C**, **I**) is a positive one.
- **Re-encoding**. Given a trajectory z, we can decode it into any set of times t, obtaining a new segment. This segment can be (re-)encoded using Θ, and a representation Q can be obtained for it, resulting in a new positive or negative for other segment representations. For example, when given the past we randomly sample a possible the future, the resulting segment (**FP** - *future given past*) will be *different* than the ground truth future, so the pair (**F**, **FP**) will be treated as a negative.

We exemplify a combination of these possibilities in Fig. 3. In order to determine which pairs of segments are positive, and which are negative, the rule is always the same: if they can belong to the

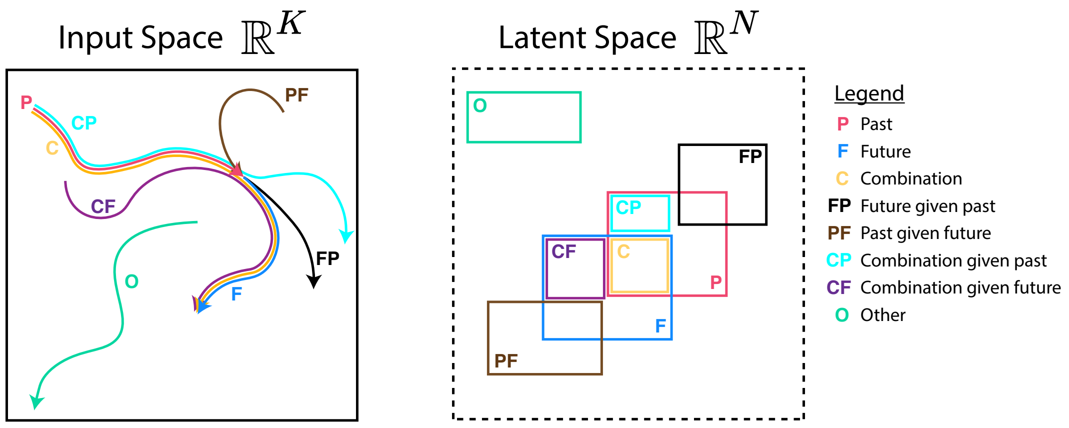

Input Space $\mathbb{R}^K$
Latent Space $\mathbb{R}^N$
P
CP
C
CF
O
F
FP
PF
Legend
P Past
F Future
C Combination
FP Future given past
PF Past given future
CP Combination given past
CF Combination given future
O Other

Figure 3: **Examples of segments**. We illustrate how spatial trajectories (left) are ideally encoded into the latent space (right). The intersection between two segment representations (boxes in the figure) represents the trajectories that contain the two segments. "Future given past" represents a segment decoded at a future time, from a trajectory sampled from the past representation. It is effectively a sample of a possible future given the past. Other segments are defined similarly. For clarity, we do not show other options like "past given past", which would be the same box as past **P**. Best viewed in color.

same trajectory they are positives, otherwise they are negatives. For example, looking at Fig. 3 it is clear that, as discussed above, there is no trajectory that can contain both **F** and **FP**. We list all negative and positive pairs in Appendix C.1.

2.4 Comparing Distributions

Eq. (1) uses the distance function D to compare distributions of trajectories. In this section, we introduce two different ways of designing D, resulting in different intuitions about the latent space.

Symmetric Distance If two segments can belong to the same trajectory, the distributions Q representing each segment should be similar and close to each other (positives), and the (symmetric) distance D between them should be small. For example, in Fig. 3, the representations of the past **P** and future **F** segments belonging to the same trajectory are treated as positives.

Conditional Instead of computing a distance or a similarity, we compute the probability that a segment s^a belongs to the same trajectory as another segment s^b. We model this as a conditional probability $P(Q^a|Q^b)$. There are four possibilities:

1. $P(Q^a|Q^b) = 1$, when s^b includes s^a, like the combination segment **C** including the past **P**.
2. $0 < P(Q^a|Q^b) < 1$, when Q^a is possible but not necessary given by Q^b, like **P** and **F** in Fig. 3.
3. $0 < P(Q^b|Q^a) < 1$, defined in a similar manner.
4. $P(Q^a|Q^b) = 0$, for unrelated segments, like **C** and **O**.

We treat the first three cases as positives, and the last case as a negative. Because pairs belonging to the first case have a stricter correspondence than those belonging to the second and third cases, we sample them more often during training. Note that under this interpretation, a past **P** and a future **F** from the same trajectory do not have a strong correspondence (first case), but a softer one (second and third cases): one does not fully define the other. This approach results in probability values that we either maximize (positives) or minimize (negatives), so we define $D(A, B) = 1 - P(A|B)$.

The previous approaches require a way of computing either a distance between the distributions Q, or a conditional probability between them. In the next section, we show two families of distributions for which these can be defined.

2.5 Trajectory Segments as Distributions

In order to obtain Q, the encoder Θ predicts the parameters of a distribution family. Conditions for the distribution families are: 1) we can sample from it in a differentiable way, 2) we can parameterize it, 3) we can compute, in closed form, an intersection that returns a distribution from the same family, and 4) we can compute either a similarity function or a conditional probability, or both (see Sec.2.4). Next, we introduce two distribution families that meet the previous criteria.

Normal distributions We use uncorrelated multivariate normal distributions, and parameterize them with a mean $\boldsymbol{\mu}$ and a standard deviation $\boldsymbol{\sigma}$. We compute the intersection as the product of two normal distributions, which remains normal when the dimensions are uncorrelated (see Appendix D). We use the symmetrized Kullback-Leibler (KL) divergence between distributions as a distance function. This distance is not a proper metric; alternatives are discussed in Appendix D. Normal distributions assume an underlying Euclidean distance metric d between trajectories z.

Box embeddings Box embeddings [55] represent objects with high-dimensional products-of-intervals (or boxes), parameterized by their two extreme vertices $z^{\wedge}$ and $z^{\vee}$. The intersection between box embeddings is well defined and results in another box embedding. This makes them a natural choice to represent conditional probabilities, which can be computed as $P(A|B) = \mathrm{Vol}(A \cap B)/\mathrm{Vol}(B)$, where $\mathrm{Vol}(A) = \prod_i^N \max(z_i^{\vee} - z_i^{\wedge}, 0)$ is the volume of the box, and $\cap$ represents the intersection operation. These operations are straightforward to compute. Boxes are not actual distributions, as they need not integrate to one. However, they are easily normalized by dividing by their volume, and therefore they can be treated as distributions for all the practical purposes required in our framework (*i.e.* sampling, where we approximate the boxes with a uniform distribution). Symmetric distance functions can also be defined on box embeddings; we define a few in Appendix D.2.

In both cases, we use the reparameterization trick [24] in order to sample from the distributions while keeping gradient information. We found the best-performing option was using box embeddings under the conditional scenario; the values reported in Section 3.2 use this setting.

2.6 Inference

Once trained, our decoder Φ is able to decode a trajectory at any continuous time t, including times that were not part of the input. For example, our framework can decode a future segment given an input past segment, by sampling from its representation, and evaluating that sample at some future times. This future segment will not necessarily be equal to the ground truth future segment (in case it exists), because a single past can have multiple futures.

Overall, our framework is capable of doing 1) future and past prediction, by decoding a segment at times outside of its range; 2) continuous reconstruction given a discrete input, by decoding at any continuous time t; 3) interpolation between two segments, by decoding trajectories in their latent-space intersection; and 4) modifying existing trajectories, by manipulating the latent space. All the previous tasks are possible without explicitly training to do any of them. We show examples in Sec. 3.

3 Experiments

3.1 Datasets

For our experiments, we selected data adhering to the following criteria. First, there has to be uncertainty in the trajectory when given just a segment (for instance, the future is not fully specified given the past). Second, the prediction should not require external contextual information. Context can be seamlessly added to our architecture, but it involves additional task-specific engineering decisions, and we want our evaluation to be orthogonal to them. Similarly, we avoid trajectories that require highly-engineered point-level distances δ. Finally, we prefer our trajectories to be obtained from real-world data. For all the previous reasons, we implement our framework on *human movement datasets*.

Specifically, we extract keypoints from human action datasets using OpenPose [10]. For every video, we keep the most salient human trajectories. This results in sequences of dimension $[L, 25, 2]$, where L is the number of frames in the trajectory, 25 is the number of joints in a human skeleton extracted by OpenPose, and 2 corresponds to the number of spatial coordinates for every joint. We refer to the

Table 1: **Prediction results**. We report the mean squared error (the lower the better) across keypoints, after normalizing each trajectory to be contained in a region of size 100×100. F, P and I stand for "future", "past" and "interpolation", respectively. Values are obtained over 10 runs with different test-time random seeds (changes include sampled segments and sampled z). An extended table with standard deviations is in Appendix E.

(a) Long sequences

	FineGym			**Diving48**			**FisV**		
	F	P	I	F	P	I	F	P	I
VRNN [13]	15.85	15.93	16.10	23.51	27.97	25.66	14.95	15.03	15.08
Trajectron++ uni. [44]	9.54	9.98	9.73	11.67	16.52	11.98	11.42	11.85	11.68
Trajectron++ [44]	9.72	10.01	9.89	11.59	16.23	12.68	11.41	11.71	11.63
TrajRep (ours, ablation)	8.82	9.07	7.57	10.00	**11.74**	10.06	10.62	11.27	9.70
+ re-encoding (ours)	**8.50**	**8.83**	**7.11**	**9.81**	12.00	**9.58**	**10.32**	**10.77**	**9.22**

(b) Short sequences

	FineGym			**Diving48**			**FisV**		
	F	P	I	F	P	I	F	P	I
VRNN [13]	12.77	13.20	13.40	18.36	20.14	19.86	13.26	13.44	13.45
Trajectron++ uni. [44]	7.80	8.28	7.48	9.05	10.36	8.29	9.23	9.68	8.86
Trajectron++ [44]	7.26	7.93	6.94	8.74	11.35	8.31	8.70	9.28	8.28
TrajRep (ours, ablation)	6.49	6.59	5.15	6.94	6.99	**5.00**	7.83	8.17	6.01
+ re-encoding (ours)	**6.20**	**6.36**	**4.88**	**6.76**	**6.85**	5.04	**7.54**	**7.78**	**5.88**

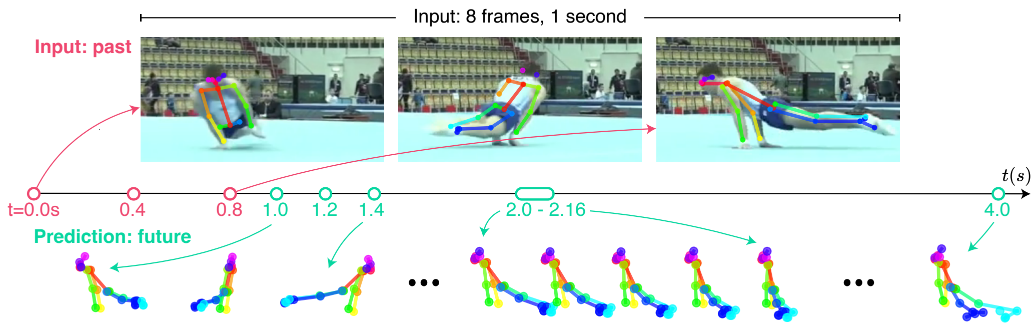
whole skeleton at every time-step —the combination of all joints, resulting in a $K = 50$-dimensional vector— as a *point*. As a distance function δ between points (*i.e.* skeletons) we use l^2-norm distance per-joint, and average across all visible joints. We extract human movement trajectories from the FineGym [45], Diving48 [29] and FisV [57] datasets, which correspond to gymnastics, diving and figure skating, respectively.

For each of the datasets, we experiment with short sequences (up to 10 time-steps, or slightly over one second) and long ones (up to 30 time-steps, representing slightly under four seconds of the trajectory), and report results for both. We provide more details on the dataset creation in Appendix B.

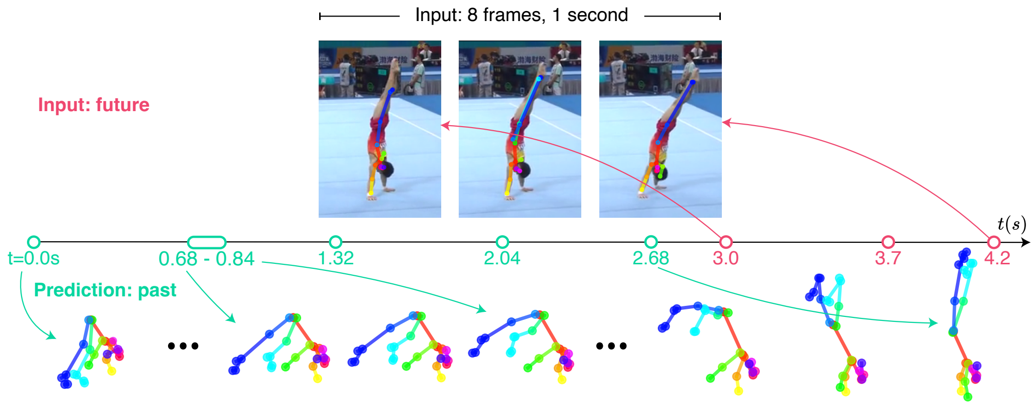
3.2 Quantitative Experiments

Baselines and ablations As baselines, we select trajectory-modeling methods that are capable of encoding uncertainty about the future. **Variational RNNs** [13] extend recurrent neural networks (RNNs) to the non-deterministic case, by modeling every step with a variational auto-encoder (VAE) [24]. **Trajectron++** [44] is a state-of-the-art trajectory-modeling framework which also builds on top of RNNs and (conditional) VAEs [23]. Uncertainty is modeled as a Gaussian mixture model (GMM). We adapt Trajectron++ to our data, making the encoding and decoding as similar to our setting as possible (for fairness), while keeping the core of the framework intact. We train two Trajectron++ versions, one with uniformly-sampled inputs and outputs ("Trajectron++ uni."), and a second one with non-uniform sampling, following the setup in our models ("Trajectron++"). We also ablate our model, and report results with and without training with re-encoded segments.

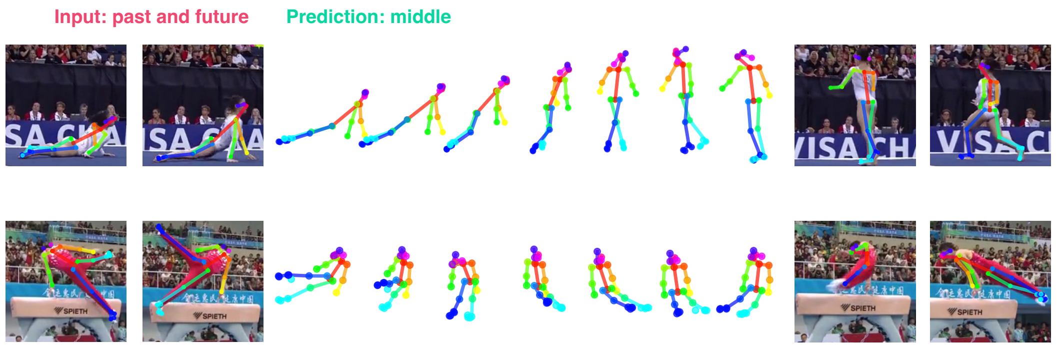
Tasks and metric We evaluate our framework on three different tasks: future prediction, past prediction, and interpolation between two segments. Future prediction consists in predicting points from a future segment given a past segment. Past prediction is defined symmetrically. In the interpolation task, we input two *separated* segments (past and future) from a trajectory, and predict the segment in between them. In our model, we do so by decoding from the latent-space intersection of the two input segments. Baselines (which are autoregressive) are not capable of doing this combination, so we only use the past segment as input. Baselines are also not capable of directly



(a) **Future prediction**. We show an example of a future prediction, where the input are eight irregularly sampled frames during one second (we only show three of them), and we predict up to three seconds into the future.

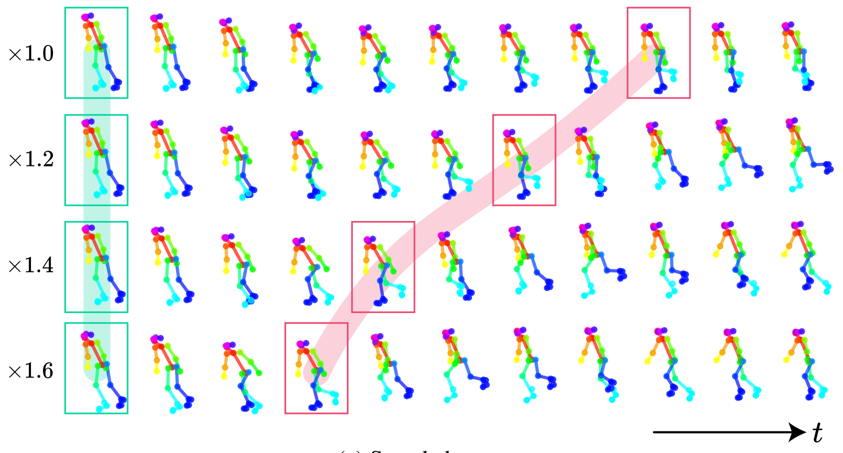


(b) **Past prediction**. We show an example of a past prediction, where the input are eight irregularly sampled frames during one second (we only show three of them), and we predict up to three seconds into the past.

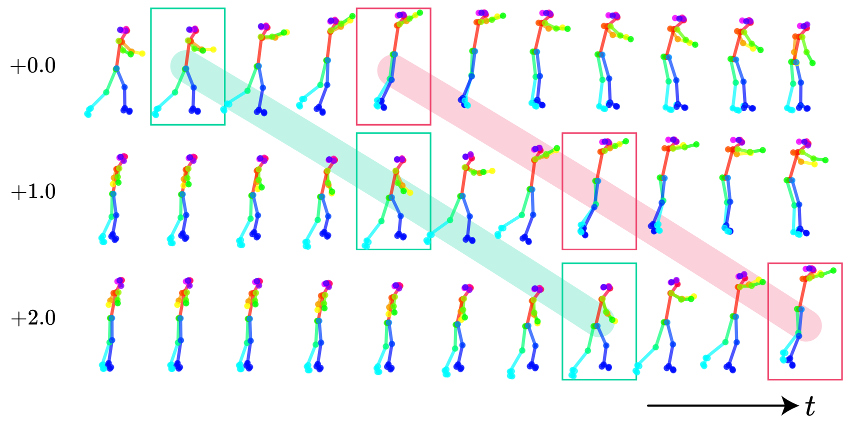


(c) **Interpolation**. We provide the model with skeleton keypoints coming from two separate segments, and sample points at times in between these two segments, from the intersection of the two segments in latent space. The model produces sensible interpolations that are not simply a linear interpolation at the joint level: in the first example, the gymnast first stands, then turns; in the second example, the gymnast swings right and left, just in time to end up meeting the future segment at the right position.

Figure 4: **Predictions on gymnastics data (FineGym)**. We show examples of past, future, and interpolation predictions. The input to our model are (irregularly in time) sampled keypoints obtained from human movement datasets, and the outputs are predictions of the trajectories at different continuous times (past, future, or in between the inputs). The only input to the model are the keypoints, not the images. Results show our model's capabilities for modeling trajectories well outside of the input's temporal range, for dealing with spatial and temporal occlusions, and for doing so at a large temporal resolution. See Section 3.3 for a deeper analysis.



(a) Speed change.



(b) Temporal offset.

Figure 5: **Temporal editing**. We decode, for the same times t, different trajectories z in the latent space. These trajectories have been obtained by encoding the segment in the top row, and moving in small increments in the latent space along directions that represent speed (a) or temporal offset (b). We highlight in green and pink some correspondences between different decoded trajectories, to emphasize that the spatial trajectories are the same but with variations in some time-related attribute, such as speed or temporal offset.

performing past prediction, so we predict the future of the reversed trajectory instead. As a metric, we use the average of the l^2-norm distance across joints, which is used by all methods during training, and report the best out of $M = 10$ samples, to account for multiple modes and uncertainty in the prediction. Note that our model has never been explicitly trained to perform any of the previous tasks.

We show results in Tables 1a and 1b, for long and short trajectories respectively. Our model outperforms baselines in all the tasks, which proves its value and flexibility. We also show how creating more interesting negatives with the re-encoding of decoded trajectories results in more accurate prediction results. However, our method performs well even without re-encoding.

3.3 Qualitative Experiments

We show examples of our model's inputs and outputs in Fig. 4. Specifically, we show future, past and interpolation predictions. The results reflect that the model learns to predict sequences up to four times longer than the input. They also show the large temporal resolution of our model: the model predictions evolve smoothly and sensibly for time-steps separated by a few hundredths of a second (Figs. 4a and 4b). When not all joints are present in the input (first frame in Fig. 4a), our model is still capable of reconstructing the full spatial extent of the position. Finally, note how the model can take

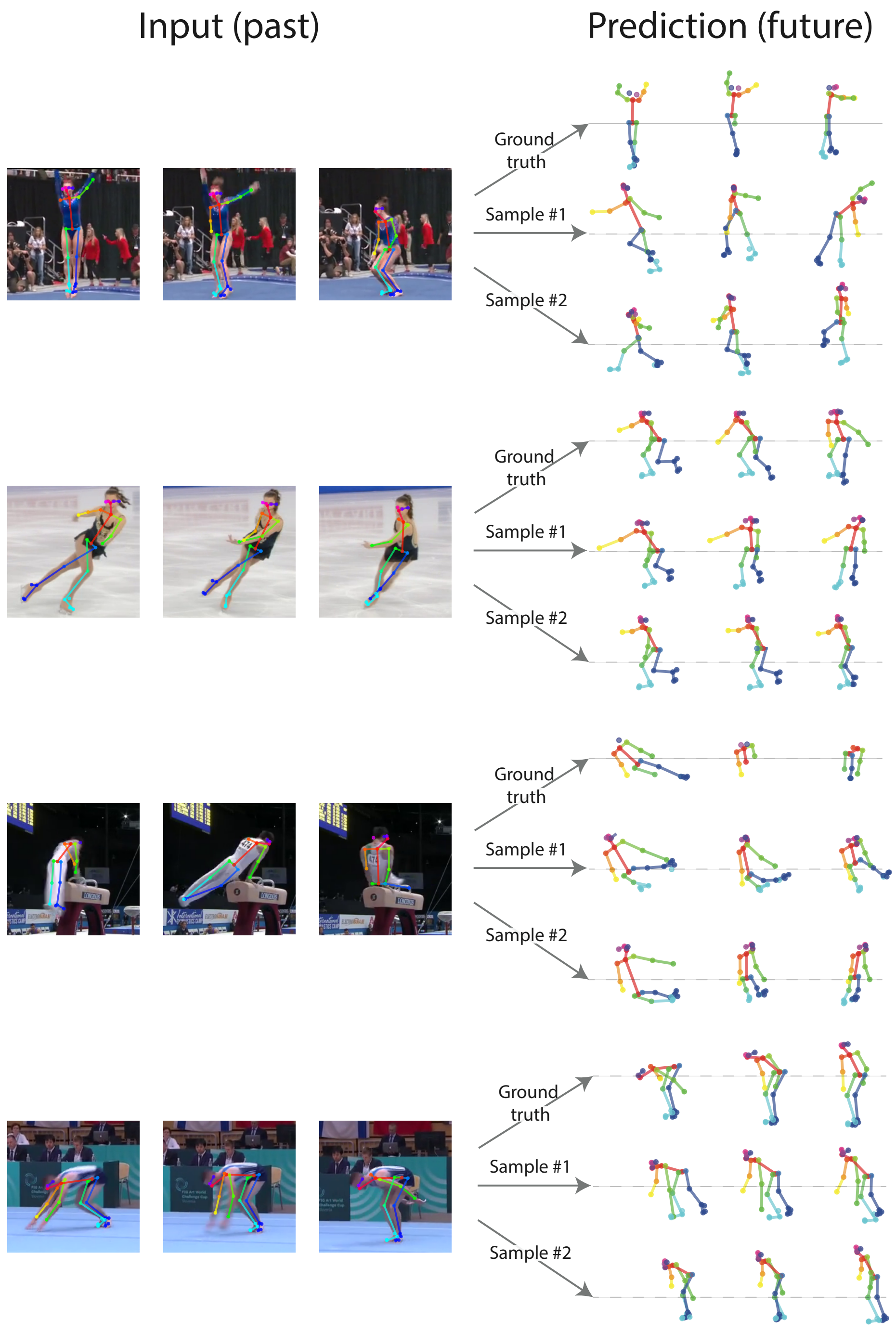


Figure 6: **Multiple futures**. Given a few input frames, our model is capable of predicting the future. It does so by modeling a distribution over the possible trajectories: by sampling from this distribution, we can obtain different plausible futures given the input (past) segment. In this figure we show, for specific inputs, the ground truth future, as well as two different futures sampled from the input segment distribution, which have been sampled randomly. This figure shows that our model is indeed capturing the multi-modal nature of the trajectories under uncertainty.

as inputs irregularly sampled time-steps (Fig. 4b), which makes it adaptable to temporal occlusions. Baselines are only capable of predicting the future, and are restricted to predicting uniform time-steps.

Temporal editing The segments are directly tied to the temporal span they represent. For example, two segments with the exact same coordinates and evolution across time, but starting at different times will result in different (albeit similar) representations. The speed of a movement and the time in the trajectory where the movement is done are important attributes of that movement, and the representation should not be invariant to them: they belong to different trajectories. However, because these trajectories are very similar, our model learns to represent them close in the latent space. In Fig. 5, we show that the model encodes different temporal variations. Moving along specific directions in the latent space results in progressively faster trajectories (Fig. 5a), or in trajectories with an increasing temporal offset with respect to the original one (Fig. 5b). See Appendix E for details.

Representing multiple futures A crucial aspect of our formulation is the assumption that the future is uncertain, and that our model has to be capable of modeling the different modes of the trajectory distribution. In Fig. 6 we show examples of multiple predicted futures given a single past segment, proving that our model captures the multi-modal nature of the trajectories under uncertainty.

4 Related work

Modeling trajectories Spatial trajectories are usually modeled in the literature in an autoregressive (AR) fashion [47, 33, 26, 2, 49, 20, 44, 30, 58, 52, 25, 19], where trajectories are defined conditioned on previous time-steps. Despite their success, AR models are incapable of dealing with some of the challenges stated in Section 1, most notably they do not represent time as a continuous variable, they cannot model the full extent of a trajectory (simultaneously both past and future), and no learned trajectory-level metric can be obtained from them. Some of them model the uncertainty in the prediction [52, 25, 19, 20, 44, 13, 49], and we use two representative ones [13, 44] as our baselines. A different line of work is focused on representing segments of trajectories (not just points) as points in a latent space [62, 65, 60, 61, 28, 64, 9, 31]. However, they are not capable of modeling the full extent of a trajectory outside the limits of the considered segment. Additionally, the segment-level metric is either unstructured [62], or is explicitly given [65, 63, 64].

Continuous time Modeling time as a continuous signal has gained traction recently in fields such as graphics [56, 53, 39, 46] or physics modeling [8, 11], because it accurately represents the underlying (continuous) world being modeled. In the graphics neural implicit functions literature, time is used to condition the prediction of the network. We adopt the same approach in our decoder. We encode the set of continuous times in a segment by using a Transformer network [54], which by construction is permutation invariant, but allows temporal embeddings to be concatenated with the input, both discrete [5, 4] and continuous [53, 51].

Self-supervised representation learning Finding self-supervised representations for temporal data has been the subject of a large amount of work in domains such as trajectories [31], video [35, 16, 40], or audio processing [21, 42, 15]. Most methods, however, represent segments as simple points in a Euclidean space. Structured representations for temporal data [50, 36] allow the latent space to follow certain inductive biases, like our framework's idea that segments compose trajectories. We model segments as either normal distributions or box embeddings [55]. The latter have been used to represent hierarchical relationships in domans such as text [38, 34], knowledge bases [1], or images [41]. We use them to represent temporal information. In recent work, Park et al. [36] also model segments using normal distributions, where trajectories are weighted sums of the segment representations.

Acknowledgments and Disclosure of Funding

We thank Arjun Mani and Mia Chiquier for helpful feedback. This research is based on work partially supported by the NSF NRI Award #2132519 and the DARPA MCS program. DS is supported by the Microsoft PhD Fellowship.

References

[1] Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:9649–9661, 2020.

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[3] James F. Allen. Maintaining knowledge about temporal intervals. In *Communications of the ACM, Volume 26 Issue 11*, 1983.

[4] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[6] Sarah-Jayne Blakemore and Jean Decety. From the perception of action to the understanding of intention. *Nature reviews neuroscience*, 2(8):561–567, 2001.

[7] Paul Bromiley. Products and convolutions of gaussian probability density functions. *Tina-Vision Memo*, 3(4):1, 2003.

[8] Michael Bronstein. Beyond message passing: a physics-inspired paradigm for graph neural networks. *The Gradient*, 2022.

[9] Judith Butepage, Michael J Black, Danica Kragic, and Hedvig Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6158–6166, 2017.

[10] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[11] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.

[12] Tejas Chheda, Purujit Goyal, Trang Tran, Dhruvesh Patel, Michael Boratko, Shib Sankar Dasgupta, and Andrew McCallum. Box embeddings: An open-source library for representation learning using geometric structures. In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2021.

[13] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

[14] Shib Sankar Dasgupta, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Lorraine Li, and Andrew McCallum. Improving local identifiability in probabilistic box embeddings. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[15] Eduardo Fonseca, Diego Ortego, Kevin McGuinness, Noel E O'Connor, and Xavier Serra. Unsupervised contrastive learning of sound event representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 371–375. IEEE, 2021.

[16] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR) Workshops*, pages 0–0, 2019.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[18] John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–317. IEEE, 2007.

[19] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. *Advances in neural information processing systems*, 31, 2018.

[20] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019.

[21] Aren Jansen, Manoj Plakal, Ratheet Pandya, Daniel PW Ellis, Shawn Hershey, Jiayang Liu, R Channing Moore, and Rif A Saurous. Unsupervised learning of semantic audio representations. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 126–130. IEEE, 2018.

[22] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14(2):201–211, 1973.

[23] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[25] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 31, 2018.

[26] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Prediction of human full-body movements with motion optimization and recurrent neural networks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1792–1798. IEEE, 2020.

[27] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. In *International Conference on Learning Representations (ICLR)*, 2019.

[28] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 617–628. IEEE, 2018.

[29] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *European Conference on Computer Vision (ECCV)*, pages 513–528, 2018.

[30] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[31] Xiang Liu, Xiaoying Tan, Yuchun Guo, Yishuai Chen, and Zhe Zhang. Cstrm: Contrastive self-supervised trajectory representation model for trajectory similarity computation. *Computer Communications*, 2022.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[33] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2891–2900, 2017.

[34] Yasumasa Onoe, Michael Boratko, Andrew McCallum, and Greg Durrett. Modeling fine-grained entity types with box embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, page 2051–2064, 2021.

[35] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11205–11214, 2021.

[36] Jungin Park, Jiyoung Lee, Ig-Jae Kim, and Kwanghoon Sohn. Probabilistic representations for video contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035. Curran Associates, Inc., 2019.

[38] Dhruvesh Patel, Shib Sankar Dasgupta, Michael Boratko, Xiang Li, Luke Vilnis, and Andrew McCallum. Representing joint hierarchies with box embeddings. In *Automated Knowledge Base Construction*, 2020.

[39] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, 2021.

[40] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6964–6974, 2021.

[41] Anita Rau, Guillermo Garcia-Hernando, Danail Stoyanov, Gabriel J Brostow, and Daniyar Turmukhambetov. Predicting visual overlap of images through interpretable non-metric box embeddings. In *European Conference on Computer Vision (ECCV)*, pages 629–646. Springer, 2020.

[42] Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3875–3879. IEEE, 2021.

[43] Antoine Salmona, Julie Delon, and Agnès Desolneux. Gromov-wasserstein distances between gaussian distributions. *arXiv preprint arXiv:2104.07970*, 2021.

[44] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision (ECCV)*, pages 683–700. Springer, 2020.

[45] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2616–2625, 2020.

[46] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020.

[47] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[48] Nishant Subramani, Samuel Bowman, and Kyunghyun Cho. Can unconditional language models recover arbitrary sentences? *Advances in Neural Information Processing Systems*, 32, 2019.

[49] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *International Conference on Learning Representations (ICLR)*, 2019.

[50] Dídac Surís, Ruoshi Liu, and Carl Vondrick. Learning the predictability of the future. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[51] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 7537–7547, 2020.

[52] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. *Advances in Neural Information Processing Systems*, 32, 2019.

[53] Basile Van Hoorick, Purva Tendulkar, Dídac Surís, Dennis Park, Simon Stent, and Carl Vondrick. Revealing occlusions with 4d neural fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[55] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 263–272, 2018.

[56] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021.

[57] Chengming Xu, Yanwei Fu, Bing Zhang, Zitian Chen, Yu-Gang Jiang, and Xiangyang Xue. Learning to score figure skating sport videos. *IEEE transactions on circuits and systems for video technology*, 30(12):4578–4590, 2019.

[58] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[59] Sijie Yan, Yuanjun Xiong, Jingbo Wang, and Dahua Lin. Mmskeleton. `https://github.com/open-mmlab/mmskeleton`, 2019.

[60] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *2019 IEEE 35th international conference on data engineering (ICDE)*, pages 1358–1369. IEEE, 2019.

[61] Di Yao, Gao Cong, Chao Zhang, Xuying Meng, Rongchang Duan, and Jingping Bi. A linear time approach to computing time series similarity based on deep metric learning. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[62] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. Trajectory clustering via deep representation learning. In *2017 international joint conference on neural networks (IJCNN)*, pages 3880–3887. IEEE, 2017.

[63] Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47(1):123–144, 2017.

[64] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. Dsdnet: Deep structured self-driving network. In *European Conference on Computer Vision (ECCV)*, pages 156–172. Springer, 2020.

[65] Hanyuan Zhang, Xingyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. Trajectory similarity learning with auxiliary supervision and optimal matching. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

Table 2: **Dataset details.**

	FineGym		**Diving48**		**FisV**	
	Long	Short	Long	Short	Long	Short
Training trajectories	404k	665k	787	27k	231k	276k
Validation trajectories	50k	83k	98	3k	29k	35k
Test trajectories	50k	83k	98	3k	29k	35k
Total trajectories	505k	832k	984	33647	289k	345k
Total videos	13k		18k		500	

A Limitations and Societal Impact

Limitations Our approach assumes all the information about a trajectory can be represented by a single latent-space embedding. While this holds true in our experiments, it is unclear how well it can scale to more complex data (like pixel-space videos) or longer trajectories (of the order of minutes). See Subramani et al. [48] for a related discussion. Another limitation we noticed from our model is that, while it is in principle capable of generating diverse trajectories from given segments, it tends to predict "average" trajectories when asked to predict far into the future (or past). Baselines have a similar behavior. A generative approach to the decoding, which would naturally fit into our framework, would probably help generating diverse trajectories far into the future; we leave it to future work. Finally, we mention in Section 3.1 that, by construction, contextual information can easily be added to the model, but we did not try it. We leave this exploration to future work.

Negative societal impact Our approach is data-driven, and thus it will replicate the biases seen in the data. Trajectory data is not as sensitive as image data on its own. However, trajectory-modeling methods can be used in sensible applications such as tracking of people for surveillance purposes.

B Dataset Details

For each one of the considered datasets, we process all the frames individually with OpenPose [10] to extract human skeleton keypoints. OpenPose returns the keypoints of all the detected humans. We then post-process the extracted keypoints, in order to group the ones belonging to the same person over time. We use heuristics that include spatial closeness between two consecutive frames and similar size of the skeleton. Once the correspondences have been obtained, and we get a series of trajectories, we filter out those that are either too small, too short, or too noisy. The minimum length (in seconds) of the trajectories is set to be the maximum length of the combination segments being sampled, so that, from the point of view of the segments, the trajectories are not limited in time.

In the short-segment experiments, the segment length is up to 30 frames, from which we randomly (read: non-uniformly) sample a third of them (this is, up to 10), for slightly over a second long segments. In the long-segment experiments, the segment length is up to 90 frames, from which we randomly sample a third of them (this is, up to 30), for slightly under four seconds long segments. Note that our model is not limited to predicting trajectories within this time-span, and it can extrapolate to larger time values.

We randomly divide the obtained trajectories into training, validation and test splits in a 80/10/10 proportion. We report the size of each dataset in Table 2. Because Diving48 is a small dataset, we fine-tune the FineGym-trained models for it, both in our models and in baselines. The reason there are very few long trajectories for Diving48 is that OpenPose extracts noisy keypoints due to out-of-distribution positions, which get filtered-out in our post-processing. The amount of videos reported for FineGym and Diving48 corresponds to the number of events; there is more than one event in each original video, but we pre-process them into separate event-level videos before extracting the keypoint trajectories.

C Implementation Details

Architecture We implement the encoder network Θ using a Transformer Encoder [54] with two layers and two heads, and hidden size 512. The latent space dimensionality N is also 512. The input to the Transformer are the spatial positions at every sampled time-step, as well as a [CLS] token to represent the whole trajectory. We use the output of the [CLS] token to represent the distributions Q. We append a temporal embedding to each input, corresponding to the time each point was sampled. We use Fourier encodings [51] to encode the continuous time value to its vector representation. We found using MLPs had similar results but slightly worse generalization to extrapolation to out-of-distribution time-steps.

The decoder network Φ is implemented using a ResNet with four blocks and hidden size 512. The temporal indices are also encoded using Fourier encodings, and they are concatenated to the z value to be decoded, following prior work [53]. We use the same architecture for the decoder network in the baselines. The capacity of the baselines is the same as our method's (~10M parameters).

Inputs In order to avoid learning shortcuts by Θ, the combination segment has the same temporal span as the combination of the past and future segments, but the specific sampled points are different (different times), so that the model cannot simply identify that the values are exactly the same and use that information to bring representations together. For every trajectory in the dataset, we randomly sample a start and end point for the segments to be used.

Losses We classify segment pairs as hard positives, which we deem very important, soft positives, soft negatives, and hard negatives, and sample hard positives and negatives more often. For example, the pair consisting of the past **P** and its reconstruction **PP** is deemed important enough to be a hard positive, and samples from other random elements in the batch are considered soft negatives. We found in initial experiments that the selection of which pairs are hard or soft is only marginally important, as long as the overall intuition described in Section 2.2 is followed. We report the specific choices in Fig. 7. Note that in the conditional setting, where we are minimizing and maximizing probability values, optimizing a binary cross-entropy loss instead of a triplet loss may seem more adequate. However, we found in initial experiments that the triplet loss performed better, so we kept it.

We found using l^1-norm for the distance function δ to perform similarly to the l^2-norm, so we kept using l^2-norm but l^1-norm is a viable alternative. This shows that our model is general and not dependant on the specific δ.

In the triplet loss in Eq. (1), we set the margin $\alpha = 1$.

When sampling trajectories to be decoded, we sample 3 trajectories for every distribution Q. These will be either positives or negatives among themselves, depending on the case (see Fig. 7).

Optimization The model parameters are optimized using AdamW [32] with weight decay of 0.05. For the learning rate, we use a cosine annealing strategy, with ranges $[1e-6, 1e-4]$, a period of 4k steps, and a warmup of 1k steps. Additionally, we clip the gradients by norm for values larger than 0.01. We did not run a hyperparameter search on the previous parameters, as we found that the initial ones worked well, except for the gradient clipping, which we found necessary to stabilize the training of the Transformer. Finally, we trained the model using mixed precision.

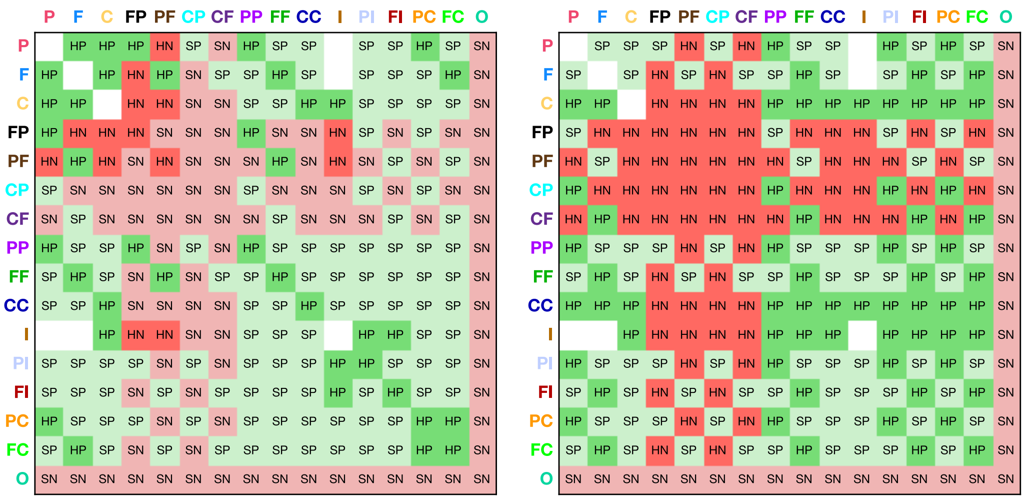
We use PyTorch [37], and the box embeddings library [12]. The models take two days to train on four RTX208-Ti GPUs.

Code, models, and data are provided.

C.1 All pair-to-pair negative/positive relationships

We follow the notation in Fig. 3, and show pairwise negatives and positives in Fig. 7. Note that some segments have a negative or positive value defined with respect to themselves: this corresponds to the relationship between different samples. For example, given a past **P**, multiple futures **FP** can be defined, with a specific relationship between them (in this case, they will be hard negatives). Additionally, we add other options that are not shown in Fig. 3. We list all of them here:

P Past.
F Future.
C Combination.
FP Future given past. Representation that has been obtained from encoding a segment that has been decoded from the past **P** representation at times t obtained from the future **F** segment.



	P	F	C	FP	PF	CP	CF	PP	FF	CC	I	PI	FI	PC	FC	O
P		HP	HP	HP	HN	SP	SN	HP	SP	SP		SP	SP	HP	SP	SN
F	HP		HP	HN	HP	SN	SP	SP	HP	SP		SP	SP	SP	HP	SN
C	HP	HP		HN	HN	SN	SN	SP	SP	HP	HP	SP	SP	SP	SP	SN
FP	HP	HN	HN	HN	SN	SN	SN	HP	SN	SN	HN	SP	SN	SP	SN	SN
PF	HN	HP	HN	SN	HN	SN	SN	SN	HP	SN	HN	SN	SP	SN	SP	SN
CP	SP	SN	SN	SN	SN	SN	SN	SP	SN	SN	SN	SP	SN	SP	SN	SN
CF	SN	SP	SN	SN	SN	SN	SN	SN	SP	SN	SN	SN	SP	SN	SP	SN
PP	HP	SP	SP	HP	SN	SP	SN	HP	SP	SP	SP	SP	SP	SP	SP	SN
FF	SP	HP	SP	SN	HP	SN	SP	SP	HP	SP	SP	SP	SP	SP	SP	SN
CC	SP	SP	HP	SN	SN	SN	SN	SP	SP	HP	SP	SP	SP	SP	SP	SN
I			HP	HN	HN	SN	SN	SP	SP	SP		HP	HP	SP	SP	SN
PI	SP	SP	SP	SP	SN	SP	SN	SP	SP	SP	HP	HP	SP	SP	SP	SN
FI	SP	SP	SP	SN	SP	SN	SP	SP	SP	SP	HP	SP	HP	SP	SP	SN
PC	HP	SP	SP	SP	SN	SP	SN	SP	SP	SP	SP	SP	SP	HP	HP	SN
FC	SP	HP	SP	SN	SP	SN	SP	SP	SP	SP	SP	SP	SP	HP	HP	SN
O	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN

(a) **Symmetric approach**

	P	F	C	FP	PF	CP	CF	PP	FF	CC	I	PI	FI	PC	FC	O
P		SP	SP	SP	HN	SP	HN	HP	SP	SP		HP	SP	HP	SP	SN
F	SP		SP	HN	SP	HN	SP	SP	HP	SP		SP	HP	SP	HP	SN
C	HP	HP		HN	HN	HN	HN	HP	HP	HP	HP	HP	HP	HP	HP	SN
FP	SP	HN	HN	HN	HN	HN	HN	SP	HN	HN	HN	SP	HN	SP	HN	SN
PF	HN	SP	HN	HN	HN	HN	HN	HN	SP	HN	HN	HN	SP	HN	SP	SN
CP	HP	HN	HN	HN	HN	HN	HN	HP	HN	HN	HN	HP	HN	HP	HN	SN
CF	HN	HP	HN	HN	HN	HN	HN	HN	HP	HN	HN	HN	HP	HN	HP	SN
PP	HP	SP	SP	SP	HN	SP	HN	HP	SP	SP	SP	HP	SP	HP	SP	SN
FF	SP	HP	SP	HN	SP	HN	SP	SP	HP	SP	SP	SP	HP	SP	HP	SN
CC	HP	HP	HP	HN	HN	HN	HN	HP	HP	HP	HP	HP	HP	HP	HP	SN
I			HP	HN	HN	HN	HN	HP	HP	HP		HP	HP	HP	HP	SN
PI	HP	SP	SP	SP	HN	SP	HN	HP	SP	SP	SP	HP	SP	HP	SP	SN
FI	SP	HP	SP	HN	SP	HN	SP	SP	HP	SP	SP	SP	HP	SP	HP	SN
PC	HP	SP	SP	SP	HN	SP	HN	HP	SP	SP	SP	HP	SP	HP	SP	SN
FC	SP	HP	SP	HN	SP	HN	SP	SP	HP	SP	SP	SP	HP	SP	HP	SN
O	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN	SN

(b) **Conditional approach**

Figure 7: **Positive and negative pairs**. HP, SP, SN, HN stand for "hard positive", "soft positive", "soft negative" and "hard negative", respectively. Note that the two matrices are the same if we consider hard and soft to be equal. The symmetric approach results in symmetric negatives and positives, while the conditional one results in an asymmetric one, because $P(A|B) \neq P(B|A)$ in general. However, as expected, it is symmetric if we consider hard and soft negatives to be equal. In that case, the two approaches result in the same matrices. Best seen in color.

PF	Past given future.
CP	Combination given past.
CF	Combination given future.
PP	Past given past. As in the previous cases, this corresponds to decoding the past **P** representation into the times in the past segment, and then re-encoding the obtained segment.
FF	Future given future.
CC	Combination given combination.
I	Intersection. Distribution that results from intersecting the past **P** and future **F** representations in the latent space.
PI	Past given intersection.
FI	Future given intersection.
PC	Past given combination.
FC	Future given combination.
O	Other (segment from a different trajectory in the batch).

In practice, not all positive and negative pairs are equally important. We differentiate between soft and hard positives and negatives, where hard ones are deemed more relevant and we give them a more important role in the optimization, by sampling them more often during training. The motivation behind distinguishing between hard and soft pairs is that we want to mostly rely on strong signals, and not add too much noise to the training. However, the distinction between hard and soft pairs is not as crucial and fundamental to our framework as the distinction between positives and negatives. Different criteria to select hard and soft pairs could be used, and the rules we used to differentiate between hard and soft positives and negatives are as follows. First, let us define the representations of segments that come directly from the data as "first encodings", to distinguish them from the re-encoded ones.

For the symmetric approach, where positives are defined as those segments that can belong to the same trajectory, *hard* positives are the positive pairs where *either* of these conditions is met:

- The pair consists of two first encodings. For example, (**P**, **F**).
- Pairs that encode *exactly* the same segment, and thus the distribution should be exactly the same. For example, (**P**, **PP**).
- Additionally, we add four extra hard positives that do not meet either of the previous conditions, but are necessary so that some hard negatives can act as such. In a triplet loss, a (hard) negative

requires a (hard) positive to be contrasted to. Some hard negatives like (**PF**, **PF**) would not have a hard positive associated to them if we only followed the two previous conditions (**PF** would not be hard positive with any other segment), and for this reason we explicitly create these four hard positives, which are (**P**, **FP**), (**F**, **PF**), (**PP**, **FP**) and (**FF**, **PF**).

Hard negatives in the symmetric case are the negative pairs where *both* of these conditions are met:

- One element of the pair represents either the past or the future.
- The other element is either a first encoding (*e.g.* the pair (**PF**, **P**)), or it represents the same segment as the first element of the pair, but a different sample (*e.g.* the pair (**PF**, **PF**)).

For the conditional approach, the distinction between hard and soft positives is more clear: hard positives are those where $P(A|B) = 1$ (this is, the first case in the list, where given B we can be certain of A), for example for $P(\textbf{P}|\textbf{C}) = 1$. Other positives (second and third cases) are treated as soft. All negatives are treated as hard negatives except for the ones coming from different elements in the batch.

An ablations where we make all positives hard is included in Tab. 3.

D Distribution Families

D.1 Normal Distributions

Multivariate normal (also called Gaussian) distributions have the following probability density function (PDF):

$$p(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})\right], \tag{3}$$

where $\mathbf{z}$ is a real N-dimensional column vector, μ is the mean, and $\boldsymbol{\Sigma}$ is the covariance matrix. In the main paper all variables z and x are vectors, so we do not use their bold versions $\mathbf{z}$ and $\mathbf{x}$, as they cannot possibly be confused with scalars.

We assume an uncorrelated multivariate normal distribution, so the previous equation is simplified to:

$$p(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{(z_i-\mu_i)^2}{2\sigma_i^2}\right], \tag{4}$$

where $\boldsymbol{\sigma}$ is a vector representing the individual-dimension standard deviations.

The product of two normal PDFs (*not* the product of two normal random variables) results in a scaled normal distribution when the N variables are uncorrelated. This is *not* general for other multivariate normal distributions. See Bomiley [7] for derivations in the two scenarios. The resulting PDF is the product of the individual dimension density functions, which follow the equation:

$$\begin{gathered} p(z)q(z) = \frac{S_{pq}}{\sqrt{2\pi\sigma_{pq}^2}} \exp\left[-\frac{(z-\mu_{pq})^2}{2\sigma_{pq}^2}\right], \\ \text{where} \quad \sigma_{pq} = \sqrt{\frac{\sigma_p^2\sigma_q^2}{\sigma_p^2+\sigma_q^2}} \quad \text{and} \quad \mu_{pq} = \frac{\mu_p\sigma_q^2+\mu_q\sigma_p^2}{\sigma_p^2+\sigma_q^2}. \end{gathered} \tag{5}$$

In the previous equation, $p(z)$ and $q(z)$ are univariate normal PDFs with mean and variance μ_p, σ_q and μ_p, σ_q, respectively, and the scaling factor S_{pq} is itself a normal PDF on both μ_p and μ_q with standard deviation $\sqrt{\sigma_p^2+\sigma_q^2}$:

$$S_{pq} = \frac{1}{\sqrt{2\pi(\sigma_p^2+\sigma_q^2)}} \exp\left[-\frac{(\mu_p-\mu_q)^2}{2(\sigma_p^2+\sigma_q^2)}\right]. \tag{6}$$

In order to compute the intersection of the two original normal PDFs, we simply ignore the scaling factor.

Kullback–Leibler divergence For two distributions P and Q of a continuous random variable, the Kullback–Leibler (KL) divergence is defined to be the integral:

$$D_{\mathrm{KL}}(P \| Q) = \int_{-\infty}^{\infty} p(z) \log \frac{p(z)}{q(z)}\, dz, \tag{7}$$

where p and q denote the probability densities of P and Q. The KL divergence measures how well a probability distribution Q represented another distribution P. In the specific case of normal distributions, the previous equation is [18]:

$$\begin{aligned} D_{\mathrm{KL}}(P \| Q) &= \frac{1}{2}\left[\log \frac{|\boldsymbol{\Sigma}_q|}{|\boldsymbol{\Sigma}_p|} - N + \operatorname{Tr}\left[\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\Sigma}_p\right] + \left(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\right)^T \boldsymbol{\Sigma}_q^{-1}\left(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\right)\right] \overset{\text{uncorrelated}}{=} \\ &= \sum_i^N \log \sigma_{q_i} - \log \sigma_{p_i} - \frac{1}{2} + \frac{{\sigma_{p_i}}^2 + \left(\mu_{q_i} - \mu_{p_i}\right)^2}{2{\sigma_{q_i}}^2}. \end{aligned} \tag{8}$$

Note that $D_{\mathrm{KL}}(P \| Q)$ is technically not a distance metric, just a divergence. Also, $D_{KL}(p \| q)$ is asymmetric, so in the symmetric approach we use the symmetrized version $D_{\mathrm{KL}}(P, Q) = (D_{\mathrm{KL}}(P \| Q) + D_{\mathrm{KL}}(Q \| P))/2$.

Proper distance metrics between normal distributions can be defined. For example, the p^{th} Wasserstein distance between two distributions P and Q is generally defined as:

$$W_p(P, Q) := \left(\inf_{\gamma \in \Gamma(P,Q)} \int_{Z \times Z} d(z_p, z_q)^p \,\mathrm{d}\gamma(z_p, z_q)\right)^{1/p} \tag{9}$$

where $\Gamma(P, Q)$ denotes the collection of all measures on $Z \times Z$ with marginals P and Q on the first and second factors, respectively. When the underlying distance metric d is the l^2-norm distance function, the Wasserstein distance between normal distributions has the closed-form expression [43]:

$$\begin{aligned} W_2^2(P, Q) &= \left\|\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\right\|^2 + \operatorname{Tr}\left[\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q - 2\left(\boldsymbol{\Sigma}_p^{\frac{1}{2}}\boldsymbol{\Sigma}_q\Sigma_p^{\frac{1}{2}}\right)^{\frac{1}{2}}\right] \overset{\text{uncorrelated}}{=} \\ &= \sum_i^N (\mu_{q_i} - \mu_{p_i})^2 + \sigma_{p_i}^2 + \sigma_{q_i}^2 - 2\sigma_{p_i}\sigma_{q_i}. \end{aligned} \tag{10}$$

Defining a distribution metric using an optimal transport approach has two main advantages. First, it results in an actual metric, as opposed to just a divergence or non-metric distance function. And second, it makes it very explicit what the underlying distance function d is in the latent space. However, it is less clear than in the KL case that minimizing the W_2 distance between normal distributions will result in a large overlap between them, as opposed to just spatial proximity. Therefore, we use KL divergence instead.

D.2 Box Embeddings

Box embeddings are N-dimensional hyperrectangles that can represent relationships such as intersection and containment. Boxes are parameterized by their two extreme points $z^\wedge$ and $z^\vee$. They are designed to represent unary and joint probabilities of events (segments, in our case), where large boxes represent highly probable and general concepts. The original paper [55] defines boxes as step functions (rectangles), but posterior papers smooth the edges of the boxes so that all pairs of boxes have positive intersections. Specifically, Li et al. [27] use Gaussian convolutions, and Dasgupta&Boratko et al. [14] improve on the previous paper using min and max Gumbel distributions. This results in better gradients during optimization, while keeping the intuition and parameters the same. We use the latter, which is conveniently implemented in an open source library [12].

We sample from a box by assuming the edges are hard instead of soft, and assuming a uniform distribution in the range $[z^\wedge, z^\vee]$. The volume of a box is computed as:

$$\operatorname{Vol}(A) = \prod_i^N \max(z_i^\vee - z_i^\wedge, 0) \tag{11}$$

In practice, the $\max$ operation is replaced by soft versions. The intersection between two boxes A and B can be computed as $z_\cap^\wedge = \max(z_a^\wedge, z_b^\wedge)$ and $z_\cap^\vee = \min(z_a^\vee, z_b^\vee)$. Note that if the intersection is zero (e.g. if $z_a^\vee < z_b^\wedge$), this will result in $z_\cap^\wedge > z_\cap^\vee$. Gumbel boxes [14] naturally handle these cases and the volume of such "negative" boxes is close to zero.

The conditional probability of one box given another one can be defined as:

$$P(A|B) = \frac{\text{Vol}(A \cap B)}{\text{Vol}(B)}. \tag{12}$$

When using the conditional approach (Section 2.4), the values we maximize in Eq. (1) are the conditional probability values obtained in Eq. (12). For the symmetric approach, box embeddings offer a variety of possibilities. We list a few of them next (Sim stands for "similarity function"):

- **Symmetric conditional**. $\text{Sim} = (P(A|B) + P(B|A))/2$
- **Intersection over Union (IoU)**. $\text{Sim} = \text{IoU} \rightarrow D = 1 - \text{IoU} = \text{Vol}(A \cup B)/\text{Vol}(A \cap B)$. This distance is known as the Jaccard distance, and it is a proper metric.
- **Sørensen–Dice coefficient**. $\text{Sim} = \text{Vol}(A \cup B)/(\text{Vol}(A) + \text{Vol}(B))$
- **Symmetric difference**. $\text{Sim} = \text{Vol}(A \triangle B) = \text{Vol}(A \cup B) - \text{Vol}(A \cap B)$.

For the values defined as similarities, we obtain the distance functions (not necessariliy metrics) as $D = 1 - \text{Sim}$.

E Additional Results and Experimental Details

Prediction into the future Our model is better than baselines for every time into the future that we tested. In Figure 10 we show a graph of error for future prediction, with respect to the time elapsed from the end of the past (input) segment.

Representing multiple futures In addition to Fig. 6 in the main paper, in Fig .9, we show how the evaluation results change depending on how many samples M we use at inference time (the reported value is the best out of the M predictions). Sampling $M > 1$ clearly improves the results, which demonstrates that our model represents (and predicts) more than one mode in the distribution. This applies not only to future prediction, but also to past and interpolation prediction.

Error bars We also report the quantitative results with standard deviations. At test time there are two factors of randomness. First, the input segments can be sampled at different times. Second, there is a sampling process to obtain trajectories z given segment representations Q. We run the test with 10 different random seeds, using the same seeds for all the experiments (this is, given a seed, both our method and the baselines use the same sampled segments). In Table 1 we report the average of the obtained values across the 10 random seeds. In this section we repeat the same results, but add information about the standard deviation across the seeds, shown in parentheses. See Table 4. Again, we report the l^2 error (the lower the better) across keypoints, after normalizing each trajectory to be contained in a region of size 100×100. FU, FR, P and I stand for "future uniform", "future random", "past" and "interpolation", respectively. The low standard deviation values imply that the significance of the average values reported in Table 1 is strong. The larger standard deviation values in the long version of the Diving48 dataset reflect the small number of test samples.

Ablations We report additional ablations of our framework in Tab. 3, for the FineGym short dataset. Specifically, we show the following ablations:

- **No re-encoding**. Same ablation as the main paper one, included here for completeness.
- **No trajectory loss**. We train without Eq. 1, which leads to a significant increase in prediction error.
- **Gaussian symmetric**. We use Gaussian distributions instead of box embeddings, and train using the symmetric scenario with KL divergence. The conditional case trained with box embeddings ("TrajRep (ours)" in the table) obtains better results, but that the symmetric case with Gaussian distributions is also competitive, and clearly outperforms the baselines.
- **Modify margin** α. The choice of α is rather arbitrary. Because the range of the distance function (for the distances used in our experiments) is in $[0, \infty)$, α influences the norm of

Table 3: Ablations on the FineGym short dataset. Values represent mean squared error. See Appendix E for a discussion.

	F	P	I
TrajRep (ours)	6.20	6.36	4.88
w/o re-encoding	6.49	6.59	5.15
w/o trajectory loss (Eq. 1)	7.37	7.55	7.02
w/ Gaussian symmetric	6.64	6.65	5.05
w/ margin $\alpha = 0.1$	**6.18**	**6.32**	4.89
w/ margin $\alpha = 10$	6.25	6.40	**4.83**
w/ all hard positives	6.32	6.51	5.03
w/ uniform sampling	6.58	6.70	5.36
w/ ST-GCN	6.77	6.99	5.26

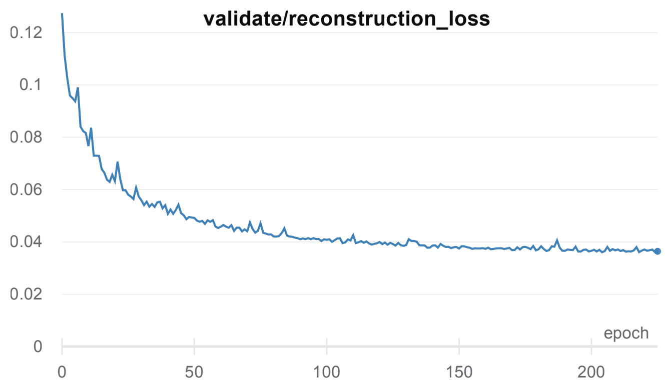
the distances, but not the relative distances between segments. We run two experiments with different values of α (0.1 and 10), on top of the default $\alpha = 1$ and show that the model performance is not very sensitive to this hyperparameter.

- **All hard positives**. We replace all soft positives by hard negatives, in order to assess how important is to distinguish them. We notice an increase in error with respect to the original model, showing that distinguishing between hard and soft negatives has some influence in the final model.
- **Uniform sampling**. We train using uniform time-steps, and test regularly, with irregular time-steps. This model does not generalize as well to irregular time-steps as the one trained directly on irregular time-steps.
- **ST-GCN**. We implement the encoder Θ using an ST-GCN network [58] instead of a Transformer. ST-GCN is the most established model to process temporal skeleton data. We use the implementation in Yan et. al. [59]. ST-GCN performs worse than the Transformer network (although the results are competitive), probably because temporal information cannot be added to an out-of-the-box ST-GCN architecture.

Stable optimization The optimization process is stable across the training. We show loss curves in Fig. 8 both for $\mathcal{L}_{\text{enc}}$ in Eq. 1 (trajectory loss) and for $\mathcal{L}_{\text{dec}}$ in Eq. 2 (reconstruction loss).

Fig. 5 details We find the directions in Fig. 5 by sampling segments from the test set, and for every segment, we modify their time indices in a progressive way, obtaining 10 modified segments for each original one. These modified time-steps are created either by multiplying their time indices by a constant (for Fig. 5a), or by adding a constant to their time indices (for Fig. 5b). We compute their representations, and for every set of segments we find the main direction of variation across the set by computing the first PCA component. The overall direction is found by averaging these directions across the different test-set samples. This overall direction (for example, the direction representing "speed change") is general for all trajectories, not trajectory-specific, and therefore we can apply it to any encoded segment.

Baseline details Regarding the baselines, the Trajecton++ models the GMM with a first step consisting of a Categorical random variable, followed by a normal distribution. In order to selec the $M = 10$ future predictions, we sample the top-10 options in the Categorical variable, and for each one we sample once from the Gaussian. Note that this is exactly how the Trajectron++ is trained, both in the original paper and in our implementation. For the VRNN baseline, at test time we sample the 10 samples in the first time-step, and then sample the mode of the latent distributions for the rest of the samples.



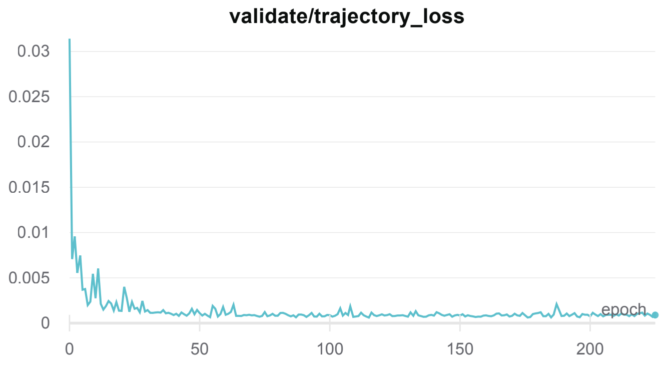


Figure 8: Loss curves during training, corresponding to the FineGym short experiment, trained with bounding boxes.

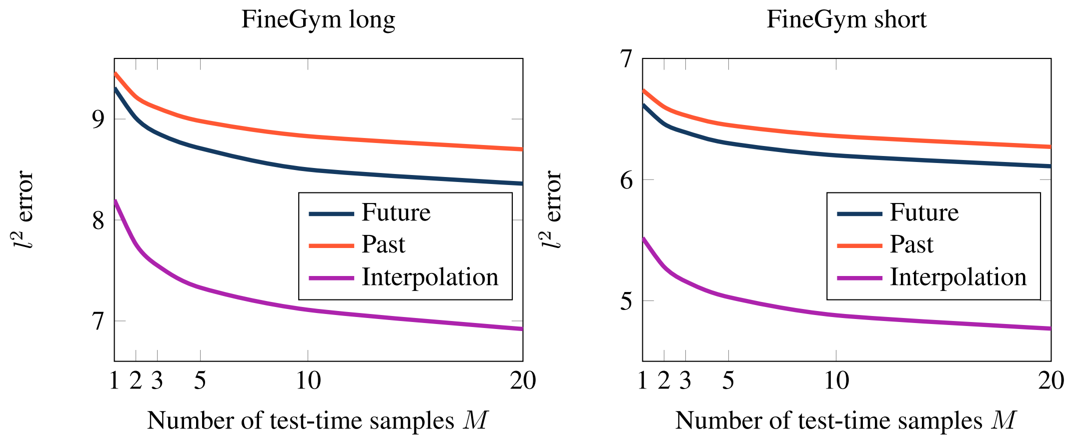


Figure 9: In our evaluation, we report the *Best-of-M* samples, where $M = 10$. In this figure, we show how the evaluation results change for different choices of M. These plots show how our model captures different modes in the distribution, as adding more samples results in better values, indicating that the samples result in different trajectories. This is qualitatively reinforced by Fig. 6. The figure on the left corresponds to the FineGym long dataset, and the figure on the right corresponds to the FineGym short dataset.

Table 4: Results from Table 1 extended with standard deviation information.

(a) **FineGym - Long sequences**

	F	P	I
VRNN [13]	15.85 (0.00)	15.93 (0.00)	16.10 (0.00)
Trajectron++ uni. [44]	9.54 (0.01)	9.98 (0.01)	9.73 (0.00)
Trajectron++ [44]	9.72 (0.00)	10.01 (0.00)	9.89 (0.00)
TrajRep (ours, ablation)	8.82 (0.01)	9.07 (0.00)	7.57 (0.00)
+ re-encoding (ours)	8.50 (0.01)	8.83 (0.00)	7.11 (0.00)

(b) **Diving48 - Long sequences**

	F	P	I
VRNN [13]	23.51 (0.13)	27.97 (0.17)	25.66 (0.06)
Trajectron++ uni. [44]	11.67 (0.22)	16.52 (0.55)	11.98 (0.10)
Trajectron++ [44]	11.59 (0.12)	16.23 (0.21)	12.68 (0.09)
TrajRep (ours, ablation)	10.00 (0.13)	11.74 (0.17)	10.06 (0.13)
+ re-encoding (ours)	9.81 (0.16)	12.00 (0.11)	9.58 (0.18)

(c) **FisV - Long sequences**

	F	P	I
VRNN [13]	14.95 (0.00)	15.03 (0.01)	15.08 (0.00)
Trajectron++ uni. [44]	11.42 (0.01)	11.85 (0.01)	11.68 (0.01)
Trajectron++ [44]	11.41 (0.00)	11.71 (0.00)	11.63 (0.00)
TrajRep (ours, ablation)	10.62 (0.01)	11.27 (0.01)	9.70 (0.00)
+ re-encoding (ours)	10.32 (0.01)	10.77 (0.00)	9.22 (0.00)

(d) **FineGym - Short sequences**

	F	P	I
VRNN [13]	12.77 (0.00)	13.20 (0.01)	13.40 (0.00)
Trajectron++ uni. [44]	7.80 (0.01)	8.28 (0.01)	7.48 (0.01)
Trajectron++ [44]	7.26 (0.01)	7.93 (0.01)	6.94 (0.00)
TrajRep (ours, ablation)	6.49 (0.00)	6.59 (0.01)	5.15 (0.00)
+ re-encoding (ours)	6.20 (0.01)	6.36 (0.01)	4.88 (0.00)

(e) **Diving48 - Short sequences**

	F	P	I
VRNN [13]	18.36 (0.05)	20.14 (0.07)	19.86 (0.02)
Trajectron++ uni. [44]	9.05 (0.02)	10.36 (0.05)	8.29 (0.03)
Trajectron++ [44]	8.74 (0.03)	11.35 (0.03)	8.31 (0.03)
TrajRep (ours, ablation)	6.94 (0.03)	6.99 (0.03)	5.00 (0.02)
+ re-encoding (ours)	6.76 (0.02)	6.85 (0.03)	5.04 (0.02)

(f) **FisV - Short sequences**

	F	P	I
VRNN [13]	13.26 (0.01)	13.44 (0.01)	13.45 (0.00)
Trajectron++ uni. [44]	9.23 (0.01)	9.68 (0.01)	8.86 (0.01)
Trajectron++ [44]	8.70 (0.01)	9.28 (0.01)	8.28 (0.00)
TrajRep (ours, ablation)	7.83 (0.01)	8.17 (0.01)	6.01 (0.01)
+ re-encoding (ours)	7.54 (0.01)	7.78 (0.01)	5.88 (0.01)

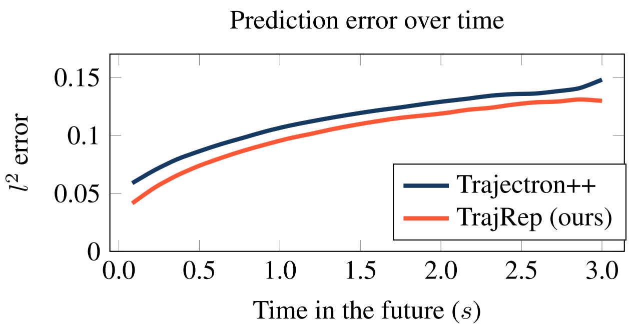


Figure 10: l^2 error with respect to the the time elapsed from the past (input) segment, where we evalute future prediction ($t = 0$ represents the end of the past segment). This result corresponds to the FineGym long dataset.