Deep Siamese Metric Learning: A Highly Scalable Approach to Searching Unordered Sets of Trajectories

CHRISTOFFER LÖFFLER* and LUCA REEB*, Fraunhofer IIS, Germany and Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Germany
DANIEL DZIBELA, Fraunhofer IIS, Germany
ROBERT MARZILGER, Fraunhofer IIS, Germany
NICOLAS WITT, Fraunhofer IIS, Germany
BJÖRN M. ESKOFIER, Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Germany
CHRISTOPHER MUTSCHLER, Fraunhofer IIS, Germany

This work proposes metric learning for fast similarity-based scene retrieval of unstructured ensembles of trajectory data from large databases. We present a novel representation learning approach using Siamese Metric Learning that approximates a distance preserving low-dimensional representation and that learns to estimate reasonable solutions to the assignment problem. To this end, we employ a Temporal Convolutional Network architecture that we extend with a gating mechanism to enable learning from sparse data, leading to solutions to the assignment problem exhibiting varying degrees of sparsity.

Our experimental results on professional soccer tracking data provides insights on learned features and embeddings, as well as on generalization, sensitivity and network architectural considerations. Our low approximation errors for learned representations and the interactive performance with retrieval times several magnitudes smaller shows that we outperform previous state of the art.

CCS Concepts: • **Computing methodologies** → **Machine learning**; • **Information systems** → **Information retrieval**.

ACM Reference Format:
Christoffer Löffler, Luca Reeb, Daniel Dzibela, Robert Marzilger, Nicolas Witt, Björn M. Eskofier, and Christopher Mutschler. 2022. Deep Siamese Metric Learning: A Highly Scalable Approach to Searching Unordered Sets of Trajectories. *ACM Trans. Intell. Syst. Technol.* 13, 1, Article 6 (February 2022), 22 pages. https://doi.org/10.1145/3465057

1 INTRODUCTION

The wide spread of devices and systems that generate positioning data allows for trajectory mining in a variety of applications. This has the potential to improve and revolutionize how we are currently working with unstructured multi-agent trajectory data, e.g., in sports [31], in mobility monitoring [33] or when we optimize transportation [11, 21]. However, as such trajectory data is often unstructured and unlabeled, their useful exploitation is still limited.

*Both authors contributed equally to this research.

Authors' addresses: Christoffer Löffler, christoffer.loeffler@iis.fraunhofer.de; Luca Reeb, reebla@iis.fraunhofer.de, Fraunhofer IIS, Nordostpark 84, Nuremberg, Germany, 90411 and Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Erlangen, Germany; Daniel Dzibela, Fraunhofer IIS, Nuremberg, Germany, daniel.dzibela@iis.fraunhofer.de; Robert Marzilger, Fraunhofer IIS, Nuremberg, Germany, robert.marzilger@iis.fraunhofer.de; Nicolas Witt, Fraunhofer IIS, Nuremberg, Germany, nicolas.witt@iis.fraunhofer.de; Björn M. Eskofier, Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Erlangen, Germany, bjoern.eskofier@fau.de; Christopher Mutschler, Fraunhofer IIS, Nuremberg, Germany, christopher.mutschler@iis.fraunhofer.de.

For instance, in sports applications, a central problem is the search and retrieval of similar occurrences in trajectory databases [31]. Such occurrences, i.e., *scenes*, can be formally defined as ensembles of trajectories, i.e., trajectories of the movements of multiple agents. Before we can compare (i.e., derive the similarity between) two scenes, we need to find an optimal assignment (*assignment problem*) for the particular trajectories, i.e., we need to find the correct match between the corresponding actors/players. Next, we can calculate a suitable distance metric between scenes. An incorrect assignment in the first step induces a large error in the estimated distance, i.e., the similarity between the two scenes. However, alignment and pair-wise comparison quickly becomes infeasible for larger databases (such as those that are available in the sports industry).

Previous work that deals with the infeasible computational complexity that comes with large databases either filters the data [10, 31, 41], learns approximations of the optimal assignment algorithm [40], or constrains the possible assignments themselves [31, 32] and thereby introduces approximation errors. While these approaches do in fact accelerate search and retrieval, their primary goal is to reduce the search space. However, this not only results in less optimal assignments and quality but also yields sub-optimal retrievals due to the limited amount of searchable data.

Fig. 1 illustrates two scenes from a trajectory database of soccer players of the German *Bundesliga*. Each scene shows variations of a reoccurring pattern that can be found throughout the season. Each single trajectory follows an inherent strategy and role, that may differ between teams and even for each player over the course of the game. Our primary goal is to find such similar scenes without constraining neither trajectory assignments nor the total search space over all samples in the database. A fast search scheme for high-dimensional trajectory sets is beneficial to sports-related applications and especially team sports (such as football, soccer, basketball, etc.), but it is not limited to that: it is also beneficial for work that is as diverse as searches similar trajectories in cellular positioning data for human mobility analysis [33], mining of frequent patterns in urban transportation [21], clustering air traffic trajectories for optimization [11], and even wildlife management in ecological behaviour analysis [8].

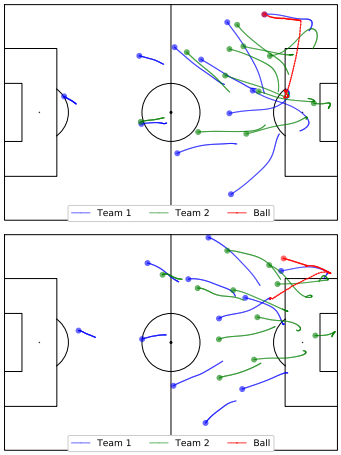


Fig. 1. Two scenes from a soccer database.

Current scene retrieval systems (especially in sports) are mainly based on manual annotations by analysts such as [20], or other automatic classification attempts that are equally expensive, e.g., highlight extraction using video data [12]. Still, tracking data becomes ubiquitously available and should become more usable despite it being unstructured. However, among the main challenges are the indexing and retrieval within these large trajectory datasets [44], given (i) multiple moving agents and (ii) the quantity of data. First, multiple trajectories of different roles are present, e.g., player formations in adversarial team sports. To calculate the distance between two scenes correctly, a correct assignment between trajectories is key. Simplifying this task by solving assignments with available meta-information only works for few applications such as player roles [31] in sports. However, this neither generalizes to other applications nor does it work in all sports. E.g., in soccer alone at least ten different formations exist [2] and players randomly break formations [2]. On the other hand, methods to optimally solve the problem such as combinatorial optimization [31] do not scale well. Second, given a manually annotated *query scene*, finding similar scenes in an entire sports season worth of data is computationally expensive. Suitable trajectory similarity functions

use metrics such as Dynamic Time Warping [29], Longest Common Sub-Sequence [37], or the Euclidean distance [31] that all need pair-wise comparisons.

This article proposes an approach which presents several solutions to the assignment problem, and which approximates the optimal distance between ensembles of trajectories, speeding up the computation needed for scene retrieval. We train a *Siamese Neural Network* (SNN) which learns a lower dimensional embedding for a given dataset, and which preserves distances between trajectories independent from use case specific distance metrics. This accelerates pair-wise comparisons while keeping the approximation error low (as we show for the *Euclidean distance* metric for a whole season of trajectories from games played in the German *Bundesliga*). This in turn enables novel interactive applications that have previously been intractable. Our contributions are as follows:

- We propose sparse and dense estimations to the assignment problem suitable for learning with deep neural networks. To this end, we extend the learning model with *Gated Temporal Convolutions* as a masking mechanism.
- We show that the learned embedding of a Siamese Neural Network, using these Gated Temporal Convolutions, approximates trajectory distances with a low approximation error.
- We show and discuss results for the *scene similarity* on a large soccer trajectory dataset. Our study includes an analysis of the embedding neighborhood and considers real wall-clock time compared to prior state of the art. We furthermore evaluate the generalizability of our method and perform detailed ablation studies.

The rest of the article is structured as follows. Sec. 2 discusses related work on trajectory similarity, sports scene retrieval systems, and approaches which solve the assignment problem. Sec. 3 formalizes the problem and Sec. 4 presents the details of our method. We show our experimental setup in Sec. 5 and discuss the results in Sec. 6. Sec. 7 concludes.

2 RELATED WORK

We discuss related work on the assignment problem (Sec 2.1), learning distance metrics (Sec. 2.2), and the search and retrieval of trajectories (Sec. 2.3), as well as the search and retrieval's special cases with event data (Sec. 2.3.1) or with tracking data (Sec. 2.3.2).

As we primarily target sports scene analysis, we give a brief overview of relevant work from that area. Usually, methods applied here do not retrieve similar scenes but analyze sports event- and position-data to predict different properties, e.g., classification and clustering. Recently, Wenninger et al. [39] evaluated modern machine learning models, e.g., convolutional or recurrent networks, to predict tactical behavior in beach volleyball such as play direction and their success. An older branch of research, among others [13, 26, 30], investigated the use of self-organizing maps that use the learned map for downstream tasks such as classification. The specific problem of classifying ensembles of trajectories was discussed in [34], however, the assignment problem was solved by design (the data was provided through skeletal tracking). The analysis of spatio-temporal trajectories of multiple players was proposed in [6, 7]. Trajectories are approximated with splines that are filtered and normalized. Several similarity measures are compared in terms of classification accuracy using the reduced representations. However, as assignments must be provided manually, the applicability is limited.

2.1 The Assignment Problem

The Hungarian algorithm [18] is an optimal solver for the assignment problem. It is used in object detection [5] to match proposed and actual object bounding boxes, in object tracking [43] to associate the identity of objects over time, and in identification [40] to match words based on their semantic similarity. Approximations [19] use neural networks based on similarity or cost matrices. As neural networks are also capable of processing unaligned data, the pre-processing

step can be skipped, either as part of the network architecture [19, 40] or as part of the learning objective [5, 43]. In our work we choose to learn it as part of the learning objective and jointly optimize a distance-preserving lower dimensional representation of the data that accelerates pair-wise comparisons. To this end, we extend the model architecture with masks to support sparse data, i.e., with Gated Temporal Convolutions, to learn assignment methods from the data.

2.2 Distance Metric Learning

In metric learning, relevant approaches either learn a better *approximation* of the distance function or instead a *lower dimensional* transformation of the data that preserves a distance. Approximations as proposed in [17] are motivated by the fact that exact metrics are either not smooth enough (over small perturbations in the inputs) or too unreliable for the application. They formulate metric learning as a regression problem that is solved with Siamese Neural Networks [4] that learn the similarity. Similar to our work, a distance acts as a regression target to the network. However, the proposed model only emits a distance given two inputs, i.e., the Siamese network must be applied to a pair of data. In contrast, our approach also allows to process single inputs, so that we can construct their embedding offline, before using them in search. This allows us to quickly search for similar examples. `Sentence-BERT` [27] approximates similarity for natural language processing, specifically for sentence-pair regression, using ideas similar to dimensionality reduction. The authors propose an attention-based transformer network together with Siamese or Triplet Networks to learn a semantically meaningful embedding based on a pair-wise similarity metric as regression target (i.e., they estimate semantic textual similarity). This approach aims to learn a lower dimensional embedding for pairs of sentences. The authors do not focus on ensembles of sentences and have no assignment problem to deal with. However, we build upon the basic methodology and extend it appropriately for learning an ensemble similarity.

2.3 Trajectory retrieval

Most trajectory retrieval systems make special assumptions such as temporal bounds of interesting search spaces or focus on other data-inherent properties that cannot be generalized. For instance, Yadamjav et al. [41] propose a framework for query retrieval of similarly co-moving trajectories, e.g., convoys of objects. An indexing structure filters the dataset based on temporal constraints (i.e., discards old samples to improve computational performance) and is fast due to an in-memory table lookup. Similarity metrics such as a point-wise max-min distance, i.e., Hausdorff distance, are extended to work on subsets of cluster points. In contrast, our approach builds a distance-preserving approximated representation of the complete dataset and thus allows scaling to larger problems, while at the same time accelerating lookup speed.

2.3.1 Event-Data Based Sports Play Retrieval. Instead of predicting similarities directly from trajectory data, the methods that search in event-data develop *query formulations* on annotated event-data. The work closest to ours is presented by Richly et al. [28]. They aim to retrieve scenes using a graphical query language composed of action sequences in user-specified areas. Another approach by Decroos et al. [9] uses an SQL-like query language instead, which also allows the OR-operator to relax search constraints over characteristic, user-defined functions.

However, annotated event data is both expensive to acquire and relatively sparse. Moreover, as such search approaches operate over event-data, they are inherently limited in their performance. Queries defined on tracking-data on the other hand allow to search for very specific situations. More importantly though, the need for event-data makes these approaches only applicable when (expensive) event-data annotations of sufficiently high quality are available.

2.3.2 Tracking-Data Based Sports Play Retrieval. There is also work that uses trajectory ensembles directly as queries for similarity-based retrieval. Such methods must handle the assignment problem and the expensive distance computation.

`Chalkboarding` [31] approximates the assignment problem using player roles and a match- and team-specific role/formation template, which are learned from data as proposed in [3]. While this introduces an unquantified error, it speeds up retrievals considerably. However, the approach does not generalize well to other domains such as soccer, where the role assignments are not static within matches and teams. Sha et al. [32] address the assignment problem with hierarchical templates which are iteratively aligned over all matches. This relaxes the fixed role assignment, making it applicable to more applications. However, the hierarchy depth is not determined by semantic properties of the data, but with respect to computation speed, e.g., cluster size. For each cluster, a suboptimal assignment template is then used for the expensive distance computations on data of (unchanged) high dimensionality. To compare distant elements, even less optimal assignment templates are used. In contrast, we propose a data-driven approach to the assignment problem that uses optimal solutions as learning targets instead of finding arbitrary approximations.

An acceleration of distance computations can be achieved by a pre-clustering of the search space [10, 32]. Sha et al. [32] propose a k-means clustering that strongly depends on the Euclidean distance as its similarity measure to cluster the scenes. An interesting way to refine search results was proposed by Di et al. [10], who extended `Chalkboarding` by using a ranking function that is sensitive to user preferences, which are determined from simulated click-through data. In contrast to clustering approaches we learn a low-dimensional representation and solve scene retrieval without a pre-clustering of the scenes, making our approach agnostic to the underlying similarity metric.

An alternative to `Chalkboarding` is proposed by Wang et al. [38] who matches segments into coherent parts, i.e., ball possession phases, which they process analogously to sentences in natural language processing. Their network learns relative similarities and relations between these segments. While a qualitative user-study proves it to be superior to `Chalkboarding`, it lacks flexibility. As user-drawn query-sketches cannot be used as queries, their framework cannot handle sparsity in queries. Furthermore, single trajectories from an ensemble are indiscernible to the model, e.g., users cannot give higher weight to the ball's trajectory if that is not learned during model training. The embedding only represents a relative ordering that is specific to a similarity metric that includes relevance features for the whole ensemble.

3 PROBLEM FORMULATION

In this work, we investigate the more general case of trajectories ensemble retrieval (i.e., scene retrieval) where the ordering between trajectories is unknown. We consider trajectory ensembles as unordered, fix-sized sets of trajectories. Trajectories are temporally ordered sets of data such as spatial positions or other arbitrary types of information, and we assume that the trajectories are spaced equidistant. We represent trajectories as $S \times T$ matrices, where S is the spatial dimension of the trajectory, as follows:

$$\vec{x} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1T} \\ x_{21} & x_{22} & \dots & x_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ x_{S1} & x_{S2} & \dots & x_{ST} \end{bmatrix}$$

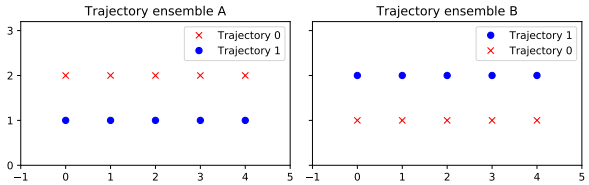


Fig. 2. Two trajectory ensembles with time increasing from left to right (-1 to 5). In the left ensemble A, the trajectories are ordered top-down, in the right ensemble B, they are ordered bottom-up. Aside from the different indexing, the data are identical.

For 3D trajectories we set $S = 3$ and for 2D trajectories $S = 2$, respectively. The distance d between two trajectories $\vec{x}$ and $\vec{x'}$ is given by the average Euclidean distance at each point in time:

$$d(\vec{x}, \vec{x'}) = \frac{1}{T} \sum_{t=1}^{T} ||\vec{x}_{:,t} - \vec{x'}_{:,t}||_2 \tag{1}$$

We chose the Euclidean distance based on the evaluation of various distance functions for the sports use case [31]. However, our method does not depend on it, as we use Siamese Networks as general function approximators.

Trajectory ensembles are given by $X = \{\vec{x}^{(1)}, \vec{x}^{(2)}, ..., \vec{x}^{(N)}\}$ where $\vec{x}^{(1)}, .., \vec{x}^{(N)}$ are trajectories. Analogous to trajectories, we represent trajectory ensembles as tensors of shape $N \times S \times T$. Note that we indexed the trajectories containing X with ascending integers. This ordering between trajectories is arbitrary and differs from the ordering of data-points within trajectories: within a trajectory the order has physical meaning, i.e., the time stamp. This has no equivalent for trajectories in an ensemble. Swapping two differently-valued and indexed data-points in a trajectory destroys its identity. For a trajectory ensemble, this is not the case: $\{\vec{x}^{(1)}, \vec{x}^{(2)}\} = \{\vec{x}^{(2)}, \vec{x}^{(1)}\}$. This arbitrary indexing also represents one of the core problems, formally known as the *assignment problem*.

Fig. 2 illustrates the assignment problem. The optimal assignment of ensemble A and B results in a distance of 0, because they match perfectly, i.e., trajectory 0 from A with trajectory 1 from B. However, there exist more possible permutations for which the distance is much greater than 0. With every new ensemble, this optimal assignment has to be calculated again, before further algorithms like distance calculations or clustering can be applied. In this work, we propose a learning approach to this problem, which approximates the assignment optimization and preserves distances, so that distance calculations or clustering remain possible. This example shows that defining the distance between trajectory ensembles is more complex than for pairs of trajectories. To compute their distances, a permutational optimization of their component's order is necessary for all pairs of ensembles X and X', that minimizes the sum over all distances between component trajectories. The Hungarian algorithm [18] solves this in polynomial time of $O(n^3)$ by iterating over the cost adjacency matrix, but is still too slow for large databases.

Let $P(X, X')$ be the permutation function found by the Hungarian algorithm which *aligns* X to X'. When $P(X, X')$ is applied to X, the indices that give the ordering are re-assigned in such a way that the sum of distances between trajectories with equal indices in X and X' is globally minimal:

$$d(X, X') = \sum_{i=1}^{N} f(X_{P(X,X')(i)}, X'_i), \tag{2}$$

with $P(X, X') = \min_{\sum_{i=1}^{N} f(X_{g(i)}, X'_i)} \{g : g \in \{1, \ldots, N\} \rightarrow \{1, \ldots, N\}\}$ and g being invertible.

Until now, we considered trajectory ensembles as unordered sets and treated trajectories equally. However, in special cases, such as team sports, we can apply some obvious simplifications without loss of generality. For soccer, we treat both teams and the ball differently, i.e., we always align attacking to attacking and defending to defending players, the ball trajectory to the ball trajectory, goalkeepers to goalkeepers. To compute the overall distance between two scenes, we sum up all individual trajectory ensemble distances, e.g., for soccer the distance between the attacking or defending team and the ball. These simplifications reduce the problem size of the trajectories that have to be assigned, in the case of soccer from 23 to 10.

We draw an example based on the two scenes from Fig. 1. Each trajectory ensemble has a color, i.e., the attacking team in blue, the defending in green and the ball in red. We compute the distance between the two scenes as follows: (1) we compute the optimal assignment of the first team's trajectories, (2) we compute d_0 by adding up the trajectory distances between all matched trajectories, (3) we repeat step 1 and 2 for the second team (yielding d_1), (4) we add up all trajectory distances between the ball trajectories and both goal-keeper trajectories (yielding d_2), and (5) we finally sum up all distances: $d_0 + d_1 + d_2$ which gives the total distance d.

4 DEEP SIAMESE METRIC LEARNING

With *Deep Siamese Metric Learning*, we propose a principled approach to approximate the exact similarity between trajectory ensembles. Instead of a hand-crafted approximation, we directly learn the exact similarity while also improving retrieval speed. The similarity metrics' accuracy directly depends on how the assignment problem is approximated, hence we propose four approaches that aim to alleviate previous limitations.

As traditional methods do not scale well, their search scope is limited to small scene datasets. This is due to the high time complexity, as described previously. For (optimal) trajectory assignments, the time complexity is $O(s \cdot n^3)$ using the optimized Hungarian algorithm for s scenes. In our approach, we simplify the search space using metric learning and different assignment methods, reducing the cost to $O(s \cdot M)$, where M is the embedding size of the metric learner.

First, for metric learning, we follow a similar idea as `Sentence-BERT` [27], that uses a Siamese Neural Network to map natural language sentences into an embedding in which the cosine distance corresponds to the semantics' similarity in a supervised regression setting. In our work we transfer and extend that idea to trajectory ensemble similarity. We map scenes into an embedding space in which the distance can be computed more efficiently (while being close to the original distance). To select the best distance for the sports use case we rely on previous work [31], where the authors evaluated a set of commonly used trajectory similarity metrics with regard to their suitability to classify sports plays into 38 distinct categories chosen by experts. Their results showed the suitability of the L_2 similarity. However, our approach is not limited to this choice as we can resort to any other similarity measure as well.

Second, we deal with the assignment problem in our framework, as well. As calculating the optimal assignment is too expensive to compute for datasets of practically relevant size, approximations were used previously, e.g., based on roles and formations in sports [31]. However, these fixed assignments of roles do not generalize well [32], not even to variations in the same sport, and neither to other use cases with more trajectories and formations [2]. Moreover, this fundamental assignment problem is intrinsic to sports, as creative changes to fixed strategies are especially beneficial for success [25]. These circumstances motivate alternative solutions to the assignment problem, i.e., forms of sparse encoding, location-based grid assignments or even random assignments, that we propose in Sec. 4.3.

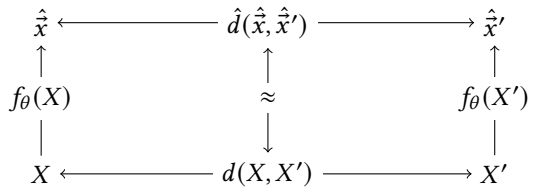


Fig. 3. Schematic of the approach that projects the high dimensional data to a low dimensional representation. The trajectory ensembles X and X' are passed through a *Siamese* network f_θ producing the embedded scenes $\hat{\vec{x}}$ and $\hat{\vec{x}}'$ respectively. The network is then trained such that the distance in the embedding $\hat{d}$ matches the ground-truth d.

4.1 Objective and Metric Learning Scheme

We consider a non-linear function approximator f_θ with parameters θ that maps inputs $X \in \mathcal{S} = \mathbb{R}^{N \times 2 \times T}$ (e.g., a trajectory ensemble) into an embedding $\hat{\mathcal{S}} = \mathbb{R}^M$ such that the distance $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ over the input space is preserved in $\hat{\mathcal{S}}$. More specifically, the distance function $\hat{d} : \hat{\mathcal{S}} \times \hat{\mathcal{S}} \to \mathbb{R}$ over the embedding preserves the property $\hat{d}(\hat{\vec{x}}, \hat{\vec{x}}') \approx d(X, X')$ with $\hat{\vec{x}} = f_\theta(X)$ for all $X \in \mathcal{S}$. Fig. 3 illustrates the approach. If d and $\hat{d}$ are equal for every scene pair X and X' we can evaluate $\hat{d}$ instead of calculating the computationally expensive ground-truth distance d. This speeds up the retrieval time considerably.

We use the same distance function, i.e., the Euclidean distance $\hat{d}(\hat{\vec{x}}, \hat{\vec{x}}') = ||\hat{\vec{x}} - \hat{\vec{x}}'||_2$, as the distance function $\hat{d}$ between embedded inputs. This choice enables a wide range of machine learning algorithms to the embedding for further processing, e.g., to restrict the search space via clustering [32]. For f_θ, we use a neural network composed of several residual temporal convolution layers for the feature extraction together with fully-connected layers to map these features to the embedding space. The observations are hence pairs of trajectory ensembles, i.e., (X, X').

The dataset size is quadratic in the number of trajectory ensembles available and training on all permutations hence quickly becomes infeasible. Instead, we sample pairs uniformly at random and minimize the *empirical risk*, i.e., $\min_\theta \mathbb{E}_{x,y \sim \mathcal{D}}[L(f_\theta(x), y)]$ where $\mathcal{D}$ is the data generating distribution. We use the following loss function:

$$L(X, X') = (||f_\theta(X) - f_\theta(X')||_2 - d(X, X'))^2, \tag{3}$$

i.e., the Mean Squared Error (MSE) and include two regularization terms:

$$||f_\theta(X)||_2 + ||f_\theta(X')||_2 + ||\theta||_2. \tag{4}$$

The first term penalizes embedded points far from the origin and the second is a standard L_2 weight regularization term that penalizes large weights in the parameters of the neural network. Note that the first term is required for convergence as the norm of the difference between two points is shift-invariant, i.e., as there are infinite optimal solutions if no order is imposed.

We use two identical Siamese Neural Networks (SNN) [4] that process distinct inputs that are joined by a metric function, i.e., the distance function. Each twin projects the input into the embedding space, on which we compute the loss. In contrast to the conventional use, we never join the outputs of the twins in a joint layer. Instead, the Euclidean distance of their outputs is directly compared to the ground-truth distance. This is due to the different problem formulation, as conventional SNNs learn a similarity metric based on class labels, whereas we use the networks to directly learn an existing metric [27]. Note that the structure of this solution is well tailored to the problem setting: f_θ is deterministic, therefore $d(\vec{X}, \vec{X}) = \hat{d}(\hat{\vec{x}}, \hat{\vec{x}}) = 0$ (identity is indiscernible)

and $\hat{d}(\hat{\vec{x}}, \hat{\vec{x}}') = \hat{d}(\hat{\vec{x}}', \hat{\vec{x}})$ (symmetry) is given by construction. In essence, the SNN learns a distance-preserving dimensionality reduction.

4.2 Siamese Network Architecture

Our SNN processes sparse temporal data. While historically recurrent networks such as Long Short-Term Memory (LSTM) [15] have been the dominant approach to process time-series data, more recent variants of convolutional networks such as the Temporal Convolutional Network (TCN) are more successful [35, 36]. Bai et al. [1] apply TCN to a wide variety of datasets with en par or better results than LSTMs while being faster to train.

The core difference of TCNs compared to classical convolutional networks is the left-padded same-convolution. Intuitively, this means that the history is padded while the present is not. This mechanism implicitly assigns weight to data based on its recency, as values in the past are replaced by zero-padding. A large benefit is the performance: TCNs are inherently parallelizable as they compute responses locally and stationary by using the last k time-steps. TCNs are hence well suited for data with a maximum dependency length, such as the trajectory ensembles in this work, for which we require a common length by construction. In addition, the assignment methods may produce sparse input data (see Sec. 4.3) as, for instance, there are at least 47 roles in soccer with only 23 trajectories including the ball. Hence, we require an architecture that can easily handle missing inputs. Prior work on TCN [35, 36, 42] provides methods that explicitly handle sparsity successfully in the domains of image and audio processing.

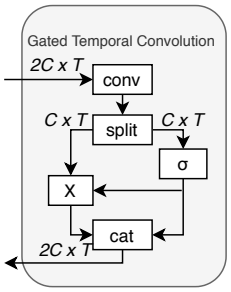


Fig. 4. Schematic of the gated temporal convolution operation.

We adapt the gating mechanism from [42] where the masks are passed to convolution layers alongside the data. The neural network continuously updates the masks and uses their information to extract features. Following a convolution operation, the masks are applied to the data in a *differentiable* way, i.e., by multiplying the data with the sigmoid of the masks. This is visualized in Fig. 4. The input $\mathrm{x} \in 2C \times T$ is composed of $C \times T$ values and an equally shaped mask. The temporal convolution layer extracts features in x and updates the mask. The mask channels are then passed through the sigmoid function and multiplied element-wise to the value channels. In the end, value and mask channels are concatenated and returned. We adapt this gating mechanism to 1-dimensional time-series data for TCNs and refer to it as *Gated Temporal Convolution*.

The network architecture itself is similar to the one in [1] and is based on a Residual Network (ResNet) [14]. While we selected this architecture due to its previous use with gating mechanisms (although on data from a different domain), its general popularity, and typically good performance, our proposed method is not limited to the use of this specific architecture. We adapt the temporal convolution block from [1] by replacing the regular convolutions with Gated Temporal Convolutions. The network is composed of a series of residual Gated Temporal Convolution blocks, see Fig. 5 with a fully-connected layer as an embedding space at the end. The ResNet's typical skip-connections add the input of the residual path to the output to enable deeper networks, see Fig. 5. The Rectified Linear Unit (ReLU) activations [23] after each convolution do not change values in the mask channels, as their range is $[0, 1]$. The channel size C is constant throughout the network but the dilation rate is increased with depth.

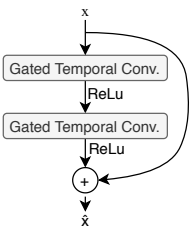


Fig. 5. Residual Gated Temporal Convolution.

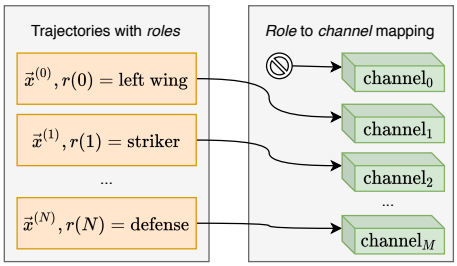


Fig. 6. Channel assignment by role: the trajectories on the left are assigned to channels based on their role r. Each channel corresponds to only one role, e.g., channel_1 always contains left wingers or remains empty (such as channel_0.)

4.3 The Assignment Problem

To estimate the similarity between two trajectory ensembles we need to have a reliable assignment of trajectories between the two ensembles with each trajectory itself being of dimension $C \times T$. This is different from convolutional network learning on image data where we assume a stable ordering of color channels. Our idea is to still consider trajectory ensembles like images when we learn convolutional filters. We map each trajectory to a channel, so that the kernel learns on the C-dimensional space with C as the temporal information for the multiple *intensity values* of T.

We require a stable ordering of channels (as it is the default case for color channels in image processing). A channel encodes information that convolution kernels learn, and neural networks presumably even extract an abstract *meaning* from these. Hence, we also construct assignment schemes that aim for relatively stable orderings. However, we also investigate if a random assignment (that breaks with the assumption) is also a suitable *channel assignment method*.

(1) Random Assignment. For each game, the trajectory assignment to a channel is stable, but then random in between games. This introduces randomness as the assignments preserve no roles. We hypothesize that this unbiased assignment looses valuable additional meta-information, e.g., player roles or positional grids, that the more sophisticated schemes can capture.

(2) By Role. For each trajectory, we may know meta information on its specific *role*, e.g., in soccer, there exist a multitude of formations. We can construct an assignment that assigns each *role* a separate channel. Given trajectory $\vec{x^{(i)}}$ with a *role* r, we assign $\vec{x^{(i)}}$ to the channel $c(r)$, where c is a bijective map from the set of roles to channel indices. The *roles* are unique in each scene. The twins of the Siamese Network can learn two trajectory ensembles where both have c_1 populated, but one twin has channel c_0 masked out in favor of another channel, see Fig. 6 for an illustration.

This assignment method leads to sparse input data when the roles in two compared scenes do not match up. For the soccer use case, a total of 23 roles are defined but only a subset is populated with data. With two teams, this results in a total of 47 channels, i.e., 23 per team and the ball, and a high degree of sparsity. We fill empty channels with zeros and use masks during the training procedure to deal with these missing values.

(3) Inferred from data. Alternatively to using meta information, we infer assignments from trajectory data. We construct these assignments either as *role positions*, using a fixed set of artificial *template* trajectories similar to [32], or as *grid positions*, that are uniformly spatially distributed. Both approaches are illustrated in Fig. 7. We calculate *role positions* using the Hungarian algorithm [18] and align the trajectory ensembles to the template. Each position p in the template is mapped to channel $c(p)$ bijectively. If trajectory $\vec{x^{(i)}}$ is assigned to p_j, $\vec{x^{(i)}}$ will be inserted into $c(j)$. Each cross (position) in Fig. 7a corresponds to a channel-trajectory pair and symbolizes a *role*. Hence,

Attribute	Range	Meaning
Identity	$\mathbb{N}$	The data-generating entity, e.g., player-id.
Role	$\{0, \ldots, 21\}$	Encoding of player role
Team	$\{0, 1\}$	Encoding of the team
Period	$\{0, 1\}$	Game period
X	$[-52.5, 52.5]$	Position in meters
Y	$[-34, 34]$	
T	$\mathbb{N}$	Time of recording
Ball possession	$\{0, 1\}$	Encoding of team in ball possession
Game activity	$\{0, 1\}$	Indicates if game was paused or active
T	$\mathbb{N}$	Time of recording

Table 1. Trajectory data and meta information in the dataset.

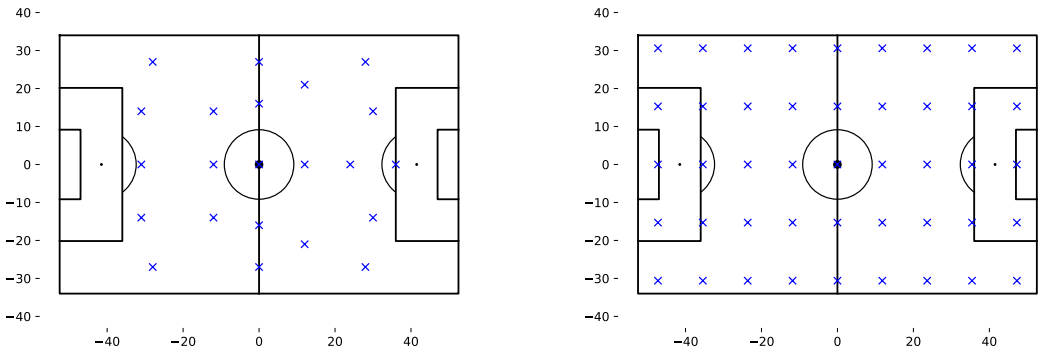
there are 22 positions in the template for the soccer use case, transforming the data into a sparse representation.

To calculate *grid positions* we use an unbiased template with increased spatial resolution. The network thus learns a set of roles from data instead of relying on pre-defined roles. This increases sparsity but also computational cost. Similarly to role positions, we again use the Hungarian algorithm to align the trajectory ensembles (during training). The Siamese networks rely on the gating mechanism to handle sparse inputs that result from some assignment methods, an architectural property that generalizes beyond the soccer domain. Section 5 shows a performance analysis and ablation study which discuss the benefits of our proposed solution.

5 EXPERIMENTAL SETUP

5.1 Dataset

The dataset we used for our experiments consists of 306 soccer matches from the *1. Bundesliga* from season 2014/15. We discarded two of these games due to incorrect or missing annotation. For each match, positions for each player and the ball were extracted from multi-perspective video feeds at a sampling rate of 25 Hz. The trajectory data structure is given in Tab. 1, and includes meta-information such as roles, ball possession or whether the game was active or not.



(a) Role position template.

(b) Grid position template.

Fig. 7. Channel assignment templates: each cross indicates the constant position of a trajectory in the template over time. Fig. 7a shows an assignment template based on the expected role position. Fig. 7b shows an assignment template based on a grid of trajectories. It is composed of nine different positions in the horizontal axis and five in the vertical axis. The difference between trajectory counts for each axis was chosen such that the positions are approximately spaced out equidistantly.

Pre-processing. We describe the pre-processing steps to convert the dataset into an appropriate scene representation for Siamese Neural Networks with Gated Temporal Convolutions. We extract game scenes of fixed length to generate trajectory ensembles. In our experiments, we set the length to either 5 s (as suggested in [10, 31, 32]) or 20 s (which was suggested by a number of sports scouts we interviewed). For our experiments, we simplify the implementation by applying several constraints to the extracted data[1]:

- One team has significantly more ball possession than the other (this circumvents an assignment over teams).
- The attacking team plays from left to right in order to use an absolute coordinate system. This ensures that differences if the playing direction in two otherwise identical scenes do not complicate the distance calculations.
- The game is active (*not paused*) for most of the time during the scene. This filters irrelevant parts of the game.
- No players are missing during the scene (even though our method could handle sparse data).

Normalization. Common normalization schemes such as mean subtraction and division by standard deviation (assuming a Gaussian distribution) is impractical for distance functions as maximum-likelihood estimation quickly becomes intractable on large datasets. The number of distances is quadratic in the number of scenes, so we approximate the normalization, while the statistics for the inputs $\vec{X}$ can be estimated directly. For the approximation, we use a running-average, i.e., similar to batch normalization, using the following (iterative) update rules:

$$\mu_{i+1}^{(x)} = (1-\beta)\mu_i^{(x)} + \beta \mathrm{E}[x_i] \tag{5}$$

$$\sigma_{i+1}^{(x)} = (1-\beta)\sigma_i^{(x)} + \beta \sqrt{\mathrm{Var}[x_i]} \tag{6}$$

with the momentum parameter $\beta \in [0, 1]$ determining the *mass* of the moving average and a series of values x_n, stopping after 100,000 steps. The inputs $\vec{X}$ are then normalized as:

$$\vec{X} \leftarrow \frac{(\vec{X} - \mu^{(\vec{X})})}{\sigma^{(\vec{X})}}. \tag{7}$$

For the distance d however, the mean-subtraction must be omitted, as otherwise it may take negative values, which the Siamese network cannot produce. Hence, we downscale the distances to preserve ordering between samples via

$$d(\vec{X}, \vec{X}') \leftarrow \frac{d(\vec{X}, \vec{X}')}{\sigma^{(d)}}. \tag{8}$$

5.2 Configuration

For our experiments, we extract in total (after pre-processing) about 1,200,000 scenes of 5 s and 400,000 scenes of 20 s from 304 games. We split the scenes extracted from the matches into three sets for training, validation and test at 80/10/10% ratio, e.g., for 20 s long scenes this yields 309,665/44,424/44,536 samples per dataset. However, due to its combinatorical nature, we cannot process all possible combinations and their distances. Hence, we construct large subsets for 5 s and 20 s by randomly sampling scene pairs and computing their ground-truth distances, resulting in (per scene length) 10 million pairs for the train dataset, and 1 million each for validation and test dataset.

For all experiments we use PyTorch [24] and the Adam optimizer [16] with an initial learning rate of $\eta = 1e^{-3}$ and $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for the momentum terms, the decay factor $s_\eta = 1e^{-1}$

[1]These are only meant to simplify our implementation but do not pose any limitations on our method in general.

every $N_\eta = 5$ epochs, for 15 epochs in total. We set the L_2 weight penalty λ_1 to 0.001. Preliminary experiments showed no convergence issues due to drifting, hence we set the embedding L_2 penalty $\lambda_2 = 0$. All inferences are computed using single-threaded code on an AMD Ryzen 7 2700 processor with 16GB RAM and a GeForce RTX 2070 GPU while we use NVidia Tesla V100 GPUs for training.

5.3 Evaluation Metrics

The *evaluation metrics* measure the errors introduced by the learned representations. For retrieval, we are interested in (1) the structural correspondence, (2) the ordering and the neighborhood structure, and (3) the retrieval set comparison.

(1) Structural correspondence. First, we measure the *structural correspondence* between the distance of pairs in the original and the learned representation using the MSE and the Mean Absolute Percentage Error (MAPE). MAPE is the expected absolute error relative to the ground-truth in percent, e.g., a MAPE of 10% means that on average, the prediction is off by 10 percent:

$$\text{MAPE}_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left| \frac{d(\vec{X}_i, \vec{X}_i') - \hat{d}(\hat{\vec{x}}_i, \hat{\vec{x}}_i')}{d(\vec{X}_i, \vec{X}_i')} \right|. \tag{9}$$

(2) Ordering and neighborhood. The ordering and neighborhood information of scenes is crucial. However, evaluating the performance of approximations of the nearest neighbor search on the full dataset is prohibitively expensive. Hence, we only work on the smaller subset of the validation dataset $\mathcal{M}_{sub}$, containing 5025 scenes with all $5025 \cdot 5024/2$ ground truth distance pairs. We use the top-N Mean Spearman Rank Correlation Coefficient (MSRCC) as a metric on $\mathcal{M}_{sub}$ to analyze the structure of the embedding, i.e., the scene ordering between the ground-truth and the Euclidean distance in the embedding. We use rank correlation to quantify the correspondence between the scene ordering in the ground-truth and in the embedding as approximation errors that change the ordering between samples are more relevant.

We calculate the MSRCC as follows. For a query scene $\vec{X}_{query,i}$ we find the N nearest neighbors $\{\vec{X}_0, ..., \vec{X}_N\}$ and the distances to them $A_i = \{d(\vec{X}_{query,i}, \vec{X}_0), ..., d(\vec{X}_{query,i}, \vec{X}_N)\}$ using the ground-truth distances. We compute their distances in the embedding $\hat{A}_i = \{\hat{d}(f_\theta(\vec{X}_{query,i}), f_\theta(\vec{X}_0)), ..., \hat{d}(f_\theta(\vec{X}_{query,i}), f_\theta(\vec{X}_N))\}$ and the *Spearman rank correlation coefficient* $r_i = \text{r}(A_i, \hat{A}_i)$ between distances in embedding and ground-truth. The MSRCC for $\mathcal{D}$ is then the average over every query scene:

$$\text{MSRCC}_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \sum_{i=0}^{|\mathcal{D}|} r_i. \tag{10}$$

We evaluate the top-N MSRCC for $N = 100$, as the nearest neighbors are the most relevant samples for the data retrieval use case, and for the whole dataset in order to assess the overall structure.

(3) Retrieval set comparison. To quantify how many of the closest N scenes are retrieved via the embedding, we calculate the top-N Mean Intersection over Union (MIoU). This is the intersection between the N closest scenes in the embedding and in the ground-truth divided by their union, averaged over every query scene. We can therefore relate it to an adaption of the accuracy metric that measures the ratio between the number of scenes that are correctly retrieved and the number of scenes that are incorrectly retrieved in the top-N. In Fig. 8, the MIoU is given as a function of the accuracy, e.g., the

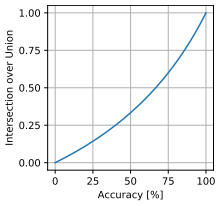


Fig. 8. Relation between the IoU of same-sized sets and the accuracy.

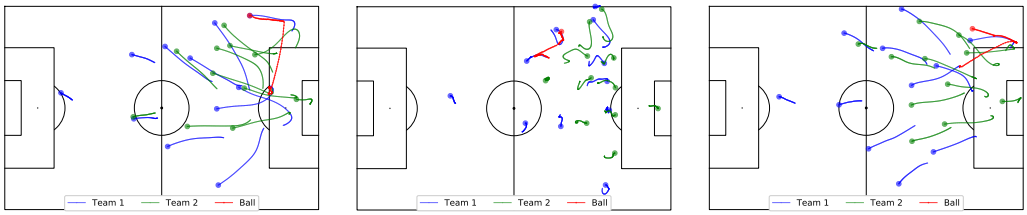
top-100 MIoU of 0.5 means that about 67 of the 100 closest scenes were also in the closest 100 scenes in the embedding.

6 EVALUATION

We first evaluate the quality of the retrievals by analyzing the scenes that our method returns (Sec. 6.1) and by analytically comparing the embedding neighborhood structure with ground-truth retrievals (Sec. 6.2). Next, we compare the retrieval speed of our approach against a naïve baseline and `Chalkboarding` (Sec. 6.3), and assess the system's generalization on a hold-out test dataset (Sec. 6.4). We conduct an expansive ablation study (Sec. 6.5) that performs a sensitivity analysis on important hyper-parameters (Sec. 6.5.1), ablates our proposed gating mechanism (Sec. 6.5.2) and learns distances for longer scene lengths (Sec. 6.5.3).

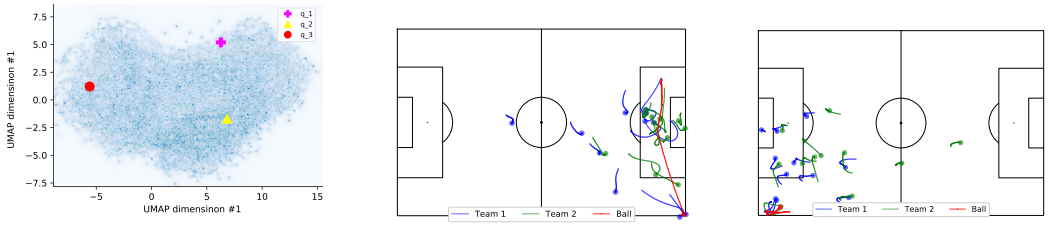
6.1 Scene Retrieval

We begin by showing exemplary retrieval queries and results for two different configurations of our method, i.e., $M = 2$ and $M = 64$. The query scene q_1 in Fig. 9a resembles a cross from the left flank. Its nearest neighbors (retrieved by our method) in Fig. 9b (for $M = 2$) and in Fig. 9c (for $M = 64$) make it very obvious that our method is capable of learning at very different levels of detail. In the query q_1 all players rush towards the defending team's goal, i.e., the scene is highly dynamic. In contrast, the query result for $M = 2$ shows a much more static game situation. The positional layout is also different: the query result for $M = 2$ is clinched relative to q_1 in the horizontal axis. The



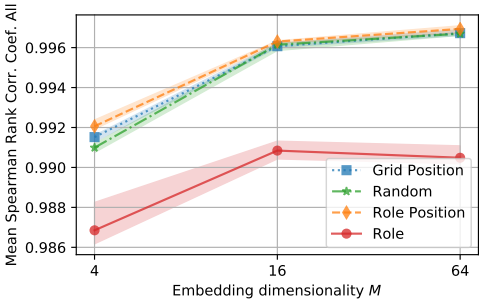
(a) Query scene q_1. (b) Neighbor of q_1 in $M = 2$. (c) Neighbor of q_1 in $M = 64$.

Fig. 9. Different scenes. The start of each trajectory is indicated by a circle. The attacking team is shown in blue, the defending in green, and the ball in red. Fig. 9a shows the query scene with a corner kick used for retrieval of the nearest neighbors in different embedding sizes M. Fig. 9b and 9c show the query results for q_1 retrieved via a 2- and 64-dimensional embedding, respectively.

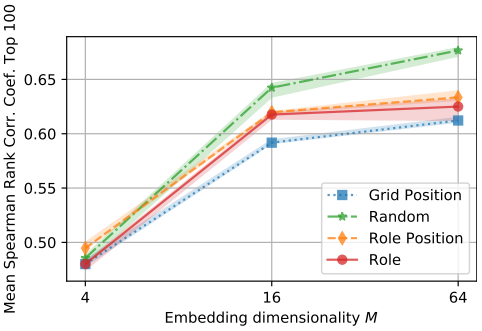


(a) 2D UMAP of 64D embeddings. (b) Query scene q_2. (c) Query scene q_3.

Fig. 10. The 64D embedding neighborhood that was fitted to a 2D manifold seemingly shaped like a curved plane in Fig. 10a (using UMAP [22]), and we highlight the queries q_1, q_2 and q_3. The scenes q_2 in 10b and q_3 in 10c show vastly different plays compared to q_1, especially in location. This is also visible on the manifold.



(a) Top-$|\mathcal{M}_{sub}|$ MSRCC.



(b) Top-100 MSRCC.

Fig. 11. Left: MSRCC for the subset $\mathcal{M}_{sub}$ over all scenes; Right: MSRCC for the top-100 with increasing embedding dimensionality M for each channel assignment method.

ball trajectory is also misplaced. The learned embedding resembles an approximation of the center of mass of a scene as $M = 2$ does not allow for a rich representation of the scene. However, with an increased size of $M = 64$ the model learns a very exact and powerful representation of scenes. The resulting scene in Fig. 9c exhibits similar game dynamics, i.e., both teams rush towards the defending team's goal. This is a representative qualitative analysis which shows that our method can retrieve subjectively similar scenes if the embedding dimension is appropriately set to capture the actual information hidden in a scene.

Next, we analyze the learned embedding for $M = 64$ in order to get a clearer understanding of what the network has learned. To this end, we visualize the validation dataset as a 2D UMAP [22] heat map in Fig. 10a, and highlight the representation for q_1 and for two additional query scenes scenes, q_2 in Fig. 10b and q_3 in Fig. 10c. These scenes q_2 and q_3 show very different games and take place at different corners. We see their positional arrangement on the learned manifold (here fitted to 2D), that has the shape of a curved plane. We flatten out the manifold to resemble the game pitch and see that the queries q_1 and q_2 are relatively close to where their games take place. Similarly, the embedding of q_3 mirrors this, as it takes place in the opposite corner. Note that while the 2D UMAP simplifies the learned representations to their main, positional features, we saw in Fig. 9 that additional properties such as game dynamics are also modeled.

We saw that the 64D embedding contains additional learned features, like game dynamics, and perform better than the 2D embedding. However, Fig. 10a can only give intuitive insights. Hence, we provide further performance metrics and more in-depth analysis in the remainder of the evaluation.

6.2 Analysis of Embedding Neighborhood

We analyze the structure of the embedding space and measure how well our method preserves inter-sample distances and hence the ordering of the retrieval results. For this, we compare the original ordering of scenes with their embedded form on two different scales: on a fine scale, i.e., relative to the 100 nearest neighbors, and on a coarse scale, i.e., the position of a scene in the embedding relative to *all others* sampled from the $\mathcal{M}_{sub}$ subset. The fine scale is important for evaluating how well the ordering of the most relevant retrievals is maintained between the learned representations, i.e., it measures the precision of our approach which is an important metric for the task of scene retrieval. The coarse scale evaluates the general feasibility of the approach, e.g., whether it can be used as a method for clustering as pre-processor.

Fig. 11a shows the coarse structural score for the whole $\mathcal{M}_{sub}$, i.e., the top-$|\mathcal{M}_{sub}|$ score. In this experiment, `Role Position` shows the best performance, closely followed by `Grid Position` and `Random` assignments. `Role` is left far behind as it has a strong inductive bias due to role specific assignments that are suboptimal on more diverse scenes. Nevertheless, all assignment methods perform better than the coefficient of 0.986, and hence are suitable for clustering at least. Furthermore, all assignment methods show better performance with increasing embedding dimensionality M until $M = 16$, from where they settle into a plateau. Overall, all assignment methods for any embedding dimensionality, even for $M = 2$, show high MSRCC, indicating that the coarse structure is captured very well.

Fig. 11b shows the fine scale Spearman rank correlation coefficient averaged over the nearest 100 neighboring scenes. This directly influences the retrieval quality. Here, the ordering of `Random` assignments is best according to MSRCC, followed by `Role`, `Role Position` and `Grid Position`. Even though `Role Position` was marginally ahead for the overall ordering, its ordering of the nearest neighbors is worse than `Random` and `Role`. Again, all assignment methods improve with increasing M until $M = 16$, after which they plateau. In contrast to the overall ordering of scenes, the ordering of the nearest neighbors changes strongly with differing assignment methods and sizes of embedding dimensionality. Even with relatively high MAPE, the overall ordering is captured well (Fig. 11a), but when observing the ordering at a smaller scale the errors become visible (Fig. 11b). The better fine scale performance of `Role` compared to its coarse scale results can be explained by a greater benefit of the inductive bias of explicit role assignments for very similar scenes, in contrast to detrimental effects when scenes, and thus the formation of players, are further apart.

As the high MSRCC indicates, the overall ordering is preserved well (which enables clustering of the search space [32]), while the fine scale ranking is not perfectly preserved. To alleviate the implications of this sub-optimal top-100 MSRCC, we may instead evaluate the ground truth distance to rank these 100 nearest results better, or even train a specific ranking algorithm for top-100 similarly to [10].

We furthermore show the differences in MAPE between the larger validation set and the smaller $\mathcal{M}_{sub}$ in Fig. 12. Differences are only marginal for `Role Position` and `Grid Position`, which perform nearly identical on both datasets. However, both `Random` and `Role` exhibit a lower MAPE in the validation set. These two assignment methods do not generalize as well as methods that adapt the channel assignment to the actual positional layouts of trajectories in a scene. In larger datasets, players may often switch actual roles depending on the game's situation [3]. Models without positional channel information have no additional indication of the dynamic role each player fills in a scene.

The high divergence of the $\mathcal{M}_{sub}$ subset from the validation set indicates that `Random` and `Role` only rely on static player roles, which do not necessarily match well with dynamic roles in scenes, whereas the methods `Grid Position` and `Role Position` generalize better. We consider the small differences for $M \in [2, 4]$ to be of less importance, because models learn little to no dynamics as the previous experiments show. This overly simplistic representation results in a lower generalization error, but ultimately is not really usable either. Overall, `Role Position` performs strong in the fine and coarse settings, and also generalizes very well to larger problems.

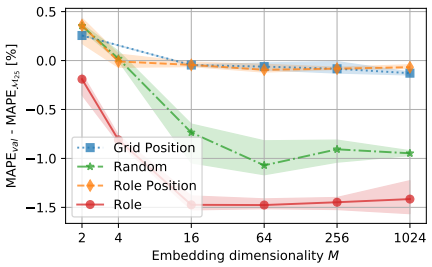


Fig. 12. Difference between the MAPE on the validation set and the subset $\mathcal{M}_{sub}$ for increasing embedding size M and channel assignment methods. Negative values indicate that the MAPE for $\mathcal{M}_{sub}$ was greater than for the validation set.

6.3 Retrieval Speed

Our main goal is to enable real-time scene search
on very large datasets of trajectory ensembles. Hence, we evaluate the required retrieval time for searches on 10K and 1M scenes and compare it with the previous state-of-the-art. For our experiments, we use scenes of 5 s duration, with 11 trajectories per team and both teams accounted for in the distance computation. The experiment is designed to be maximally fair, and we average over 100 repetitions. All database scenes are in memory, no clustering is performed to limit the search space, and the query is aligned to a template, i.e., both teams are aligned to their own template that contains 11 trajectories.

To measure the retrieval time, we compute the distance between query and each sample in the database. For our method, the retrieval speed is the time of one forward pass through the neural network, followed by the computation of the Euclidean distance from the query's embedding vector to other embedded scenes. For all methods, we search only for the 10 scenes with smallest distance, and do not sort the results.

We show the retrieval speed for varying sizes of embedding dimension in Tab. 2. As a `baseline` we implemented a naïve (brute force) distance computation and also include `Chalkboarding` [31] retrieval. With 10,000 scenes in the database, our method takes only about 10 ms for $M \leq 64$, while `Chalkboarding` takes 346ms and `Baseline` even takes 65 s, which is prohibitively long. At embedding sizes of $M \in [2, 4, 16, 64]$, searches are not limited by the distance function computations. With $M \in [256, 1024]$ it begins to affect retrieval times with 17 ms to 45 ms. With smaller M, the cost of computing forward passes through the network is nearly constant as only the last fully-connected layer varies with M. The largest part of the $9 - 11ms$ run time is data transfer overhead, e.g., GPU to CPU transfer and vice versa, while the distance computation time itself is minimal.

There are mainly two reasons why our method outperforms `Chalkboarding`. First, `Chalkboarding` expensively aligns queries to the learned template, and second, it computes the ranking based on 23 trajectories, hence one distance computation operates over a $23 \times 2 \times 125$ dimensional tensor in contrast to only M values in the embedding.

The larger experiment with 1M scenes corresponds to almost an entire season (305 games). As the data does not fit into main memory we could not compute results for `Baseline`, `Chalkboarding`, and our method with $M = 1024$. However, from the available experiments it is apparent that their

Method	Retrieval time 10k scenes	Retrieval time 1M scenes
`Baseline`	$68.887s$	> 1 h
`Chalkboarding`	$0.346s$	> 30 s
$M = 2$	$0.009s$	$0.052s$
$M = 4$	$0.009s$	$0.054s$
$M = 16$	$0.010s$	$0.100s$
$M = 64$	$0.011s$	$0.287s$
$M = 256$	$0.017s$	$0.998s$
$M = 1024$	$0.045s$	-

Table 2. Evaluation of the retrieval time using differently sized embeddings, the `Chalkboarding` [31] retrieval system and the baseline. All retrieval speed tests except for the baseline are averaged over 100 runs.

retrieval times are not competitive, i.e., >1 h and >30 s. Instead, our method yields an interactive retrieval time. Between $M = 16$ or $M = 64$ it offers an interesting trade-off. $M = 16$ requires only about 35% of the time of $M = 64$ but increases MAPE by about 18%. The lower error rate for $M = 64$ should be preferred if the resulting set of nearest neighbors is not further refined or ranked, i.e., similar to `Chalkboarding`'s two-step cluster-then-refine approach.

6.4 Generalization

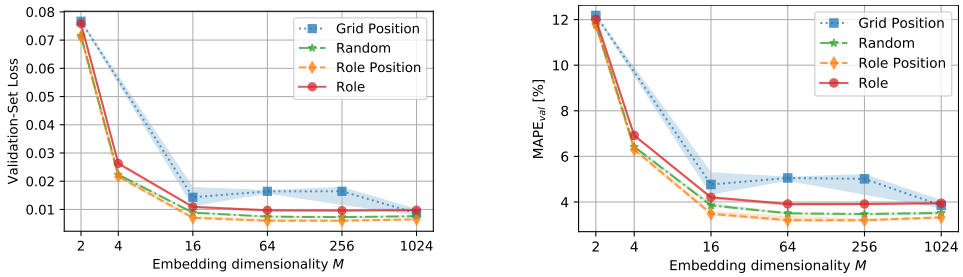
We also assess generalization on a hold-out dataset that contains 30 matches. We design our experiment as follows. We sample 1M pairs from this unseen test set and compute the MAPE between the pair-distances and their representation in the embedding. The difference between the MAPE computed on the test and validation set allows us to evaluate how our approach generalizes to unseen data.

We use the best performing channel assignment method `Role Position` and set the embedding size $M = 64$. This offers a speedy retrieval with relatively low error on the validation set with a MAPE of 2.66%. On the test set, this optimal model achieves a MAPE of 2.68%. This minimal generalization error of about 0.02% shows that our method learns a general representation which is not only a fit to the training data, but also works on previously unseen scenes.

6.5 Ablation Study

6.5.1 Sensitivity Analysis. We study the sensitivity of our approach to different sizes of the *embedding dimension* $M \in [2, 4, 16, 64, 256, 1024]$ and the channel assignment methods `Grid Position`, `Random`, `Role Position` and `Role` on the validation set, in order to measure their impact on the approximation error.

Fig. 13 shows the MSE to the left and the MAPE to the right. The experiments were performed the validation-set over three runs. Here, we report the evaluation for the network when the validation loss is minimal. The MSE on the left shows errors irrespective of the *location* of the ground-truth trajectory. However, a low MAPE does not necessarily follow from a low MSE. The MAPE is a measure of how relevant the scenes are ranked on average, by comparing the nearest neighbors in the embedding. `Role Position` shows best performance followed closely by `Random` and `Role`. We see low variability over repeated runs for each of the assignment methods. For all of them, performance increases rapidly until $M = 16$, and then plateaus from $M = 64$ to $M = 1024$. In contrast, the channel assignment by `Grid Position` shows slightly worse performance with a



(a) MSE over the validation set.

(b) MAPE over the validation set.

Fig. 13. Validation-set MSE loss (left) and MAPE (right) computed on predetermined set of pairs with increasing embedding dimensionality M for each channel assignment method. The points indicate the median over three runs. The colored areas correspond to the standard deviation centered around the mean. Note that the x-axis has a logarithmic scaling.

higher variability over repeated runs. However, with the larger capacity of $M = 1024$ it catches up to the other assignment methods (at the cost of compute time).

The decrease in error with increasing embedding dimension M is expected. With a larger M, more scene features can be embedded into the representation. Interestingly, the performance of most models improves only up to $M = 64$. We hypothesize that this is due to a lack of model capacity in the feature extractor. The maximum width of channels during the feature extraction is 128, the networks learn solely based on these features. Accordingly, embedding sizes greater than $M = 128$ use an under-determined projection matrix, i.e., some of the embedding axes linearly depend on each other and are redundant. Hence, scaling the width of the models could allow models with large M to improve further.

The channel assignment methods exhibit varying degrees of sparsity and of positional encoding. The evaluation results for `Grid Position`, `Role Position` and `Role` suggest that sparsity impacts performance negatively. We next examine the positive impact of our gating mechanism onto sparse data.

6.5.2 Ablating Gating Mechanism. In this work we introduced a novel gating mechanism for TCNs for sparse data to address the assignment problem. Here, we investigate the performance benefits in an ablation study. We show differences for sparse (channel assignment) inputs through the masking operation.

In Fig. 14 we show the differences of MAPE values between the full and the ablated TCN architecture without gating mechanism. The sparsest method `Role` shows clear performance degradation compared to the more dense methods `Grid Position`, `Random` and `Role Position`. The changes range within 0.3% MAPE, i.e., the expected worsening of performance due to missing information is more subtle. The absolute MAPE for `Random`, `Role Position` and `Grid Position` is slightly below 3% and `Role` around 3.75%. In essence, omitting the gating mechanism clearly leads to a decreased predictive performance, with the realistic real world configuration of $M = 64$ showing the largest difference.

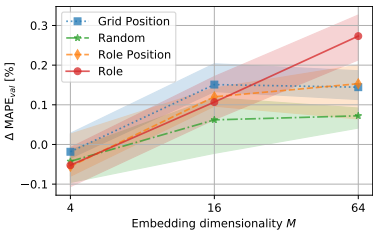


Fig. 14. ΔMAPE wo/ gating over different embedding dimensionalities $M \in [4, 16, 64]$.

Overall, the effect on all sparse channel assignment methods is larger than for the dense `Random` assignment, indicating that sparse assignment methods benefit from information in the masks. We explain the minimal change for `Random` due to the complete lack of additional information that masks encode for it, i.e., every mask value is one. Hence, the performance improvements can be partially accounted to the increased model capacity from additional mask channels. Interestingly, the degree of sparsity does not affect the effect size, as `Role` benefits to a larger extend than does `Grid Position`. To summarize, our gating mechanism improves performance in most (useful) scenarios, likely due to the information encoded through the masks and the increased model capacity, while the sparse channel assignment methods improve most.

6.5.3 Scene Length. Besides sport scenes of 5 s length for smaller scale analysis, sports scientists and scouts are also interested in larger strategic movements over longer periods of time. Generally, for other applications, the complexity of the data varies greatly, e.g., the amount of data per scene between bird foraging and urban transportation patterns. For these reasons we evaluate our learning method for longer scenes of 20 s length, which exhibit a larger feature complexity. In our experiment, we make use of all four assignment methods and use the previously best performing network architecture but extend the input width and receptive field size accordingly. We used a training set of 10 million scenes.

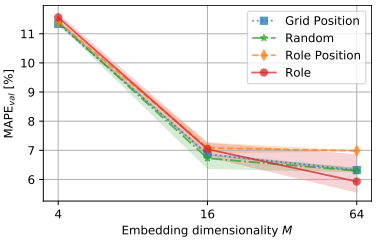


Fig. 15. MAPE for 20 s scene length.

The experimental results in Fig. 15 for $\mathcal{M} = 64$ show an increased absolute approximation error of a MAPE of 6.3% with `Random` assignments, and 5.8% for `Role Position`. This error increased compared to the previous 3.47% at 5 s scene length. With respect to the local neighborhood ranking for 20 s scene length, we find that the error did not increase similarly to MAPE, with very good performance of a top-$|\mathcal{M}_{sub}|$ MSRCC of still 0.97, and the fine granular neighborhood top-100 ranking MSRCC of 0.69. Also the top-100 ranking score MIoU is high (0.57). This means that the overall retrieval accuracy remains almost unchanged. For the longer scenes of 20 s duration, their pair-wise distances grow larger, and following from that, their neighborhood structure thins out, with neighbors farther apart from each other. Hence, the absolute approximation error MAPE has less impact on (local) ranking because absolute pairwise distances are also larger. This shows that our approach is especially well-suited for more complex data due to its typically larger distances and also due to the massive savings of computation time.

Moreover, using an estimate of the embedding density we could derive an optimal cluster size to restrict the search space in a principled way. Instead of directly using the embedding for scene ranking, computing the exact ground truth distance in a cluster is feasible. Here, MAPE serves as the expected radius in which most relevant scenes lie.

7 CONCLUSION

We proposed a novel approach to similarity-based scene retrieval of trajectory ensembles that uses approximations to the assignment problem at much lower computational costs than state of the art. Using Siamese Networks at its core we learn a low-dimensional representation that preserves the distance between sample pairs and thus accelerates the search by several orders of magnitude. The low approximation error allows fast search and subsequent ranking of the closest neighbors, while leaving the global neighborhood structure almost unchanged. We propose and evaluate four channel assignment methods, both application agnostic or biased for role-based (sports) trajectories, and found the hybrid `Role Position` to work best for the evaluated sports tracking application.

Our experimental results prove that our method learns non-trivial trajectory features like game play dynamics, and users can select an optimal trade-off between estimation accuracy and search speed, depending on the application. Furthermore, the proposed gating mechanism increases performance for sparse channel encoding.

In conclusion, our approach enables a highly accurate and interactive retrieval of similar scenes. A trivial extension with a two-step ranking system could additionally incorporate a refined second ranking step of the top-100 retrieved samples, using the exact distance computation on original sample representations. Furthermore, adapting our framework to applications in similar team sports like basketball or ice hockey is obvious, especially in datasets of unordered trajectory ensembles. The method may help the analysis of data from in-store tracking of customer movement by finding similar movement patterns. In the medical domain, learning the similarity of parameters of walking gait only from movement trajectories (specifying a human motion simulator) could benefit. In radio-frequency positioning, interference minimization in a changing environment is a hard problem, that could profit from accelerated search of historically similar channel/frequency and sender/receiver assignments.

ACKNOWLEDGMENTS

This work was supported by the Bavarian Ministry of Economic Affairs, Infrastructure, Energy and Technology as part of the Bavarian project Leistungszentrum Elektroniksysteme (LZE) and through the Center for Analytics-Data-Applications (ADA-Center) within the framework of "BAYERN DIGITAL II".

REFERENCES

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271 (2018).
[2] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. 2014. Identifying team style in soccer using formations learned from spatiotemporal tracking data. In International Conference on Data Mining Workshop. IEEE, 9–14.
[3] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. 2014. Large-scale analysis of soccer matches using spatiotemporal tracking data. In International Conference on Data Mining. IEEE, IEEE Computer Society, Los Alamitos, CA, USA, 725–730.
[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In Advances in neural information processing systems. 737–744.
[5] Nicolas Carion, F. Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. In European Conference on Computer Vision.
[6] Christopher Mutschler. 2010. Online Data–Mining of Interactive Trajectories in Realtime Location Systems. Master's thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU).
[7] Christopher Mutschler, Gabriella Kókai, and Thorsten Edelhäußer. 2011. Online Data Stream Mining on Interactive Trajectories in Soccer Games. In Proceedings of the 2nd International Conference on Positioning and Context-Awareness (Brussels). 15–22.
[8] Ian R Cleasby, Ewan D Wakefield, Barbara J Morrissey, Thomas W Bodey, Steven C Votier, Stuart Bearhop, and Keith C Hamer. 2019. Using time-series similarity measures to compare animal movement trajectories in ecology. Behavioral Ecology and Sociobiology 73, 11 (2019), 151.
[9] Tom Decroos, Jan Van Haaren, and Jesse Davis. 2018. Automatic discovery of tactics in spatio-temporal soccer match data. In Proceedings of the International Conference on Knowledge Discovery & Data Mining. 223–232.
[10] Mingyang Di, Diego Klabjan, Long Sha, and Patrick Lucey. 2018. Large-Scale Adversarial Sports Play Retrieval with Learning to Rank. ACM Transactions on Knowledge Discovery from Data (TKDD) 12, 6 (2018), 1–18.
[11] Esther Calvo Fernández, José Manuel Cordero, George Vouros, Nikos Pelekis, Theocharis Kravaris, Harris Georgiou, Georg Fuchs, Natalya Andrienko, Gennady Andrienko, Enrique Casado, et al. 2017. DART: a machine-learning approach to trajectory prediction and demand-capacity balancing. SESAR Innovation Days, Belgrade (2017), 28–30.
[12] X. Gao, X. Liu, T. Yang, G. Deng, H. Peng, Q. Zhang, H. Li, and J. Liu. 2020. Automatic Key Moment Extraction and Highlights Generation Based on Comprehensive Soccer Video Understanding. In International Conference on Multimedia Expo Workshops (ICMEW). IEEE, 1–6.
[13] Andreas Grunz, Daniel Memmert, and Jürgen Perl. 2012. Tactical pattern recognition in soccer games by means of special self-organizing maps. Human movement science 31, 2 (2012), 334–343.
[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition. 770–778.
[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.
[16] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, San Diego, CA, USA.
[17] Georg Kohl, Kiwon Um, and Nils Thuerey. 2020. Learning Similarity Metrics for Numerical Simulations. In International Conference on Machine Learning. PMLR, 5349–5360.
[18] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. Naval research logistics quarterly 2, 1-2 (1955), 83–97.
[19] Mengyuan Lee, Yuanhao Xiong, Guanding Yu, and Geoffrey Ye Li. 2018. Deep neural networks for linear sum assignment problems. IEEE Wireless Communications Letters 7, 6 (2018), 962–965.
[20] D. Link. 2014. A toolset for beach volleyball game analysis based on object tracking. International Journal of Computer Science in Sport 13 (01 2014), 24–35.
[21] Yingchi Mao, Haishi Zhong, Xianjian Xiao, and Xiaofang Li. 2017. A Segment-Based Trajectory Similarity Measure in the Urban Transportation Systems. Sensors 17, 3 (Mar 2017), 524.

[22] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. UMAP: Uniform Manifold Approximation and Projection. Journal of Open Source Software 3, 29 (2018), 861.
[23] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning (Haifa, Israel) (ICML'10). 807–814.
[24] 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. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32. Montréal, Canada, 8024–8035.
[25] Jürgen Perl. 2018. Formation-based modelling and simulation of success in soccer. International Journal of Computer Science in Sport 17, 2 (2018), 204–215.
[26] Jürgen Perl and Daniel Memmert. 2011. Net-Based Game Analysis by Means of the Software Tool SOCCER. International Journal of Computer Science in Sport (01 2011).
[27] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 3973–3983.
[28] Keven Richly. 2018. Leveraging spatio-temporal soccer data to define a graphical query language for game recordings. In 2018 IEEE International Conference on Big Data (Big Data). IEEE, 3456–3463.
[29] Pavel Senin. 2008. Dynamic time warping algorithm review. Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA 855, 1-23 (2008), 40.
[30] B. Serrien, M. Goossens, and J-P. Baeyens. 01 Jul. 2017. Issues in Using Self-Organizing Maps in Human Movement and Sport Science. International Journal of Computer Science in Sport 16, 1 (01 Jul. 2017), 1 – 17.
[31] Long Sha, Patrick Lucey, Yisong Yue, Peter Carr, Charlie Rohlf, and Iain Matthews. 2016. Chalkboarding: A new spatiotemporal query paradigm for sports play retrieval. In Proceedings of the 21st International Conference on Intelligent User Interfaces. 336–347.
[32] Long Sha, Patrick Lucey, Stephan Zheng, Taehwan Kim, Yisong Yue, and Sridha Sridharan. 2017. Fine-grained retrieval of sports plays using tree-based alignment of trajectories. arXiv preprint arXiv:1710.02255 (2017).
[33] Zhi-Hao Shen, W. Du, X. Zhao, and Jianhua Zou. 2019. Retrieving Similar Trajectories from Cellular Data at City Scale. ArXiv abs/1907.12371 (2019).
[34] Huong Yong Ting, Kok-Swee Sim, and Fazly Salleh Abas. 2015. Kinect-based badminton movement recognition and analysis system. International Journal of Computer Science in Sport 14, 2 (2015), 25–41.
[35] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In 9th ISCA Speech Synthesis Workshop. 125–125.
[36] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48. 1747–1756.
[37] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. 2002. Discovering similar multidimensional trajectories. In Proceedings 18th international conference on data engineering. IEEE, 673–684.
[38] Zheng Wang, Cheng Long, Gao Cong, and Ce Ju. 2019. Effective and efficient sports play retrieval with deep representation learning. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 499–509.
[39] Sebastian Wenninger, Daniel Link, and Martin Lames. 01 Jul. 2020. Performance of machine learning models in application to beach volleyball data. International Journal of Computer Science in Sport 19, 1 (01 Jul. 2020), 24 – 36.
[40] Han Xiao. 2020. Hungarian layer: A novel interpretable neural layer for paraphrase identification. Neural Networks 131 (2020), 172–184.
[41] Munkh-Erdene Yadamjav, Zhifeng Bao, Baihua Zheng, Farhana M. Choudhury, and Hanan Samet. 2020. Querying Recurrent Convoys over Trajectory Data. ACM Trans. Intell. Syst. Technol. 11, 5, Article 59 (Aug. 2020), 24 pages.
[42] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In Proceedings of the IEEE International Conference on Computer Vision. 4471–4480.
[43] Yu Zhao, Quan Chen, Wengang Cao, Jie Yang, Jian Xiong, and Guan Gui. 2019. Deep learning for risk detection and trajectory tracking at construction sites. IEEE Access 7 (2019), 30905–30912.
[44] Yu Zheng. 2015. Trajectory data mining: an overview. ACM Transactions on Intelligent Systems and Technology (TIST) 6, 3 (2015), 1–41.