SILA: An Incremental Learning Approach for Pedestrian Trajectory Prediction

Golnaz Habibi
MIT
golnaz@mit.edu

Nikita Jaipuria
Ford Motor Company
niksjaipuria@gmail.com

Jonathan P. How
MIT
jhow@mit.edu

Abstract

The prediction of pedestrian motion is challenging, especially in crowded roads and intersections. Most of the current approaches apply offline methods to learn motion behaviors, but as a result, they are not able to learn continuously and typically do not generalize well to new environments. This paper presents Similarity-based Incremental Learning Algorithm (SILA) for pedestrian motion prediction with the ability of improving the learned model over the time as data is obtained incrementally. To keep the model size efficient, the motion primitives learned from the new data are compared with the previously known ones, and similar motion primitives are fused while novel motion primitives are added to the model. Results show that the SILA model growth rate is about 1/3 that of an incremental approach that does not fuse motion primitives. SILA is evaluated on different datasets and scenarios including intersections and busy streets. The results show that, even though SILA learns incrementally, it performs comparably to (and sometimes outperforms) state-of-the-art algorithms in pedestrian prediction. Additionally, SILA learning time only depends on the size of the data added incrementally, which makes SILA more efficient in terms of time and space compared to batch learning.

1. Introduction

Safe navigation of autonomous vehicles in urban environments requires accurately predicting motions of other moving agents, including cars, cyclists and pedestrians. Pedestrian motion prediction is challenging as compared to other agents, as the rules are less clear and more frequently violated. Moreover, pedestrian behavior is also complicated by their interactions with other road users.

Machine learning techniques have been used to model complex pedestrian motion behavior by incorporating contextual features [1–3] and/or the interaction between other road users [4–8]. However, most of these techniques are limited to offline (batch) setting. The offline setting becomes impractical when training data is incrementally available or when the autonomous agent has to explore new environments. Batch learning does not have the option of learning *incrementally* and hence, limits the generalization of the pre-trained model to new pedestrian behaviours and environments. Moreover, incremental learning provides the flexibility of learning new behaviors using relatively fewer data points as opposed to learning from scratch. Such a setup is significantly more efficient and better suited to autonomous driving applications where there is a need for real-time learning. The need to maintain pedestrian privacy might further limits the possibility of storing past data and re-learning from the past and new data combined (batch learning). Instead, if an agent can learn incrementally, its prediction model can be improved continually from new streams of data or "few-shot" examples without the need to access past data. Additionally, incremental learning enables systems with limited resources to process big data gradually in scenarios where offline learning is impractical. Although there has been a lot of work in incremental learning in different domains [9–11], there are very few prior works on incremental learning of pedestrian behaviors.

This paper presents a Similarity-based Incremental Learning Algorithm (SILA) for incremental learning of motion primitives for pedestrian motion prediction. To the best of our knowledge, the only prior work on incremental learning for pedestrian motion prediction is presented by Vasquez *et al*. [12] and is based on Growing Hidden Markov Models (GHMMs). Similar to other Markovian approaches, GHMMs depend on goal estimation for inferring pedestrian intent which requires complete un-occluded trajectories for training. However collecting such datasets is impractical in busy and crowded domains such as intersections, campus areas and shopping centers, where often a part of the pedestrian trajectory is occluded by obstacles and other agents.

In another work, Chen *et al*. [13] combined the merits of Markovian-based and clustering-based methods for the task of pedestrian trajectory prediction. In their approach, trajectories are segmented based on motion primitives learned using a sparse coding algorithm. The transitions between trajectory segments are then modeled by Gaussian Processes (GPs). Their approach does not require estimating the goal as opposed to [12] and can learn from partial trajectories. Inspired by this work, the trajectory prediction framework of SILA follows [13] and comprises of *motion primitives* and pair-wise *transitions* between them. The motion primitives are learned using Augmented Semi Non-negative Sparse Coding (ASNSC) [13] and the pair-wise transitions between motion primitives are modeled using sparse GPs based on "pseudo inputs" [14]. We leverage the idea of a "curbside/common frame" as introduced in [15] to build a transferable model by mapping trajectory data into a normalized and environment independent common frame.

In SILA, whenever new data becomes available, motion primitives and their transitions are learned from the new batch of data only. To update the pre-trained model, we propose a similarity measure for comparing the newly learned motion primitives with those in the pre-trained model. Similar motion primitives and transitions are fused based on a *similarity graph* constructed using the proposed similarity measure.

To summarize, the **main contributions** of this work are: (i) A novel incremental learning algorithm - SILA - for pedestrian motion prediction, with the ability of updating the prediction model *incrementally* while simultaneously *transferring* knowledge across different environments; (ii) SILA proposes a new method for fusion of motion primitives based on the similarity graph; (iii) SILA, on an average, outperforms prior works (including batch learning methods) in terms of prediction error (Average Displacement Error: ADE [16]) with 5% improvement; (iv) SILA achieves up to 1/3 smaller model growth rate (in terms of number of motion primitives and transitions) as compared to a naive incremental learning baseline (in which new motion primitives are simply added without any fusion); (v) Possibility of real-time learning because of fast model adaptation to newly available dataset: SILA's training time only depends on the size of the mini batch it learns from incrementally, as a result, it can learn up to 20 times faster than batch learning algorithms.

2. Related Work

Related prior works mainly fall under the domains of pedestrian motion prediction and incremental learning of motion behaviors.

Pedestrian motion prediction Mohamed *et al.* [5] present the most recent work in pedestrian motion prediction. Their approach models pedestrian interaction with an attention graph, which is similar to [6] but typically results in models with fewer parameters and better accuracy. In Huang *et al.* [6]'s approach, graph attention networks are leveraged to model the spatio-temporal interaction between pedestrians. Gupta *et al.* [7] extend Social-LSTM [4], one of the first few methods incorporating social interaction in pedestrian trajectory prediction, by making use of recurrent Generative Adversarial Networks (GANs). Chen *et al.* [13] presented Augmented Semi Non-negative Sparse Coding (ASNSC). In their method, trajectories are represented in terms of motion primitives and pair-wise transitions between them. Jaipuria *et al.* [15] introduced Transferable Augmented Semi Non-negative Sparse Coding (TASNSC) as an extension of ASNSC [13]. In TASNSC, trajectories are mapped into a normalized, environment independent coordinate frame that helps learn a prediction model that can generalize to environments with different geometries. We make use this frame for learning environment agnostic features in an incremental way. Inspired by TASNSC and ASNSC, we build their"incremental" variant - SILA.

Incremental learning of motion behaviors There are very few prior works in this category. Vasquez *et al.* [12] introduced a Growing Hidden Markov Model (GHMM) based incremental learning algorithm for pedestrian motion prediction. However, their method requires complete pedestrian trajectories to train on, which are impractical to collect in crowded environments. In contrast, our algorithm can learn from incomplete or partially occluded trajectories. Kulić *et al.* [17] presented an incremental learning approach for full-body human motion prediction. In their method, a set of motion primitives is pre-defined and the relationship between them is incrementally learned using HMMs. Such an approach cannot be directly applied to the task of pedestrian trajectory prediction because of the wide variety in plausible motion primitives that must be learned. Ferguson *et al.* [18] introduced a method for online learning of motion patterns. In their work, newly learned behaviors are compared with those in the pre-trained model using GPs to detect novel behaviors and add them to the model. Our work is similar to [18] in terms of adding novel behaviors to the pre-trained model. However, we also fuse similar motion primitives to enrich the learned model.

3. Background and problem statement

This section introduces notations and building blocks for SILA, which is an incremental variation of ASNSC [13] and TASNSC [15]. Training trajectories are represented as a set of tuples of 2D position and velocity - $(x(t), y(t), v_x(t), v_y(t))$. Similar to Ref. [13], trajectories are mapped into a grid world with $N = r \times c$ cells, where r and c are the number of rows and columns respectively. Let the training dataset denoted by $\mathbf{X}_{N\times p}$ consist of p trajectories. The q-th trajectory can be represented as a column vector $\mathbf{tr}_q \in \mathbb{R}^N$ such that the k-th element of $\mathbf{tr}_q$ is the normalized velocity vector of the q-th trajectory in the k-th grid cell. Given this vectorized representation of training trajectories, a set of L motion primitives (dictionary atoms),

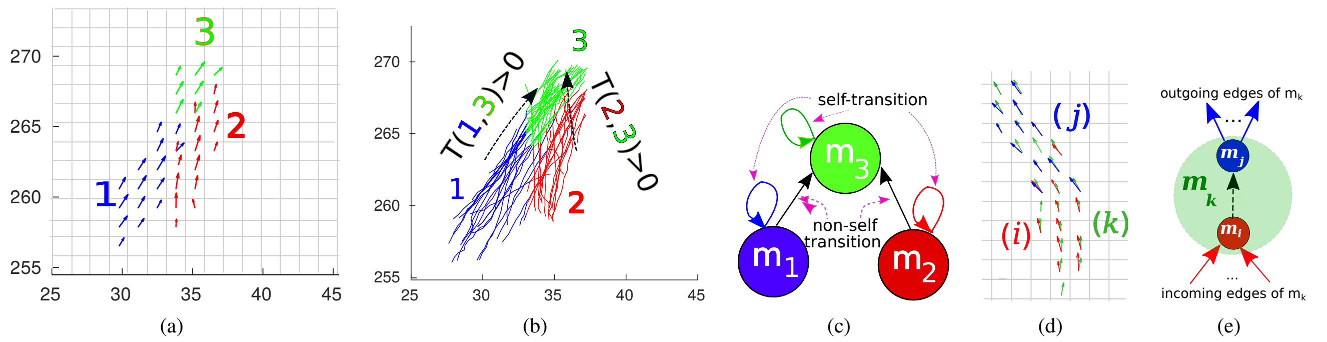


Figure 1. **(a)** Motion primitives are learned in a grid world using ASNSC [13]. Each color represents a motion primitive $\mathbf{m}_i$ (here $i \in 1,2,3$); **(b)** Segmentation of training trajectories into clusters using motion primitives shown in (a). Each cluster is best explained by the motion primitive of the same color in (a) and the black arrows show the transition between motion primitives for cases in which the corresponding element in the transition matrix $\mathbf{T}$ is non-zero; **(c)** Motion primitives and their transitions in (b) modeled as a directed graph, *i.e*. a motion primitive graph. For brevity, only 3 motion primitives are shown in (a); **(d)** Two motion primitives $\mathbf{m}_i$ and $\mathbf{m}_j$ (red and blue) from one model are similar to a motion primitive $\mathbf{m}_k$ (green) from another model. In this example, $\mathbf{m}_i$ and $\mathbf{m}_j$ together provide a richer and more primitive representation of motion behaviors as compared to $\mathbf{m}_k$ alone. Thus, in fusing the two models, $\mathbf{m}_k$ is replaced by $\mathbf{m}_i$.**(e)** Replacing $\mathbf{m}_k$ (green circle) with $\mathbf{m}_i$ and $\mathbf{m}_j$, as explained in (d). All incoming edges of $\mathbf{m}_k$ (red arrows) now enter $\mathbf{m}_i$ and all outgoing edges of $\mathbf{m}_k$ (blue arrows) now exit from $\mathbf{m}_j$.The transition from $\mathbf{m}_i$ to $\mathbf{m}_j$ (dashed line) is kept as-is (figure best seen in color).

$\mathbf{D} = \{\mathbf{m}_1, \ldots, \mathbf{m}_L\}$, are learned using sparse coding [13]. Each color in Figure 1(a) represents a single motion primitive $\mathbf{m}_i$, learned from the trajectories shown in Figure 1(b). Each motion primitive $\mathbf{m}_i$ is represented as a set of normalized cell-wise velocities $\{\mathbf{v}_i^k\}$. Here, $\mathbf{v}_i^k$ is the velocity of $\mathbf{m}_i$ in the k-th grid cell (in $N = 14 \times 15$ grid cells in Fig. 1(a)). $\mathbf{D}$ is used to segment the training trajectories into clusters (see Figure 1(b)). Each color in Figure 1(b) is one such cluster, best explained by the motion primitive in Figure 1(a) in the same color. These clusters are used to create the transition matrix $\mathbf{T}_{L\times L}$, where $T(i,j)$ denotes the number of training trajectories exhibiting a transition from $\mathbf{m}_i$ to $\mathbf{m}_j$. Following Ref. [13], each of these transitions is modeled as a two-dimensional GP flow field [19, 20]. Thus, $\mathbf{T}$ is used to create the set of transitions $\mathbf{R}$ such that $\mathbf{R} = \{r_{ij} | \mathbf{T}(i,j) \neq \emptyset\}$ The learned motion primitives and transitions are then represented as a *dual motion primitive graph* [21], wherein graph nodes represent motion primitives and graph edges represent the transition between motion primitives (see Figure 1(c)).Each transition r_{ij} in $\mathbf{R}$ are modeled by consist two independent GPs: $GP_{ij} = (GP_x, GP_y)$, that learn a mapping from the two-dimensional position features (x,y) to velocities v_x and v_y respectively similar to [13]. These GPs are also referred to as "motion patterns". Note that we define two types of transitions - "self transitions" which denote GP models representing a single motion primitive; and "non-self transitions" which denote GP models representing the pairwise transition between two motion primitives (see Figure 1(c)). We use these definitions later in Section 4. The sparse GP regression model presented by Snelson *et al*. [14] is used to learn motion patterns.

Problem statement: Our goal is to design an algorithm that incrementally learns and updates motion primitives and transitions. Assuming new data becomes available in n-th episode, a new model ($M'(n)$) is learned from this data. The following section describes how SILA updates the pre-trained model ($M(n-1)$) to $M(n)$ using $M'(n)$.

4. Proposed Algorithm

This section explains how SILA learns incrementally from new data in three main steps: (1) Learn $M'(n)$ from data received in the n-th training episode; (2) Use similarity between motion primitives in $M(n-1)$ and those in $M'(n)$ to create a similarity graph; (3) Using the similarity graph, fuse $M(n-1)$ with $M'(n)$ to give $M(n)$.

4.1. Pre-processing step: Normalize training trajectories in the common frame

Trajectories are normalized and mapped into a common frame prior to being used for training to aid knowledge transfer across environments with different geometries. For non-intersection datasets, trajectories are simply normalized based on their x-y range. To account for different intersection geometries in intersection datasets, we leverage the common frame of Ref. [15] (with origin at the intersection corner of interest and axes aligned with the intersecting curbsides). In this common frame, each data point is represented by its *contravariant distance* [22] from the curbsides. The transformed trajectories are also normalized by the sidewalk width to account for scale differences across intersections. All these operations combined help learn and fuse motion primitives with similar underlying behaviors even in scenarios in which the primitives are spatially different in the original coordinate frame (see Ref. [15] for more details).

4.2. Step 1: Learn motion primitives and transitions

In each training episode, first $M'(n)$ is learned from the new batch of data. This model is comprising of motion primitives (learned using ASNSC [13]) and transitions (modeled using sparse GPs [14]) .

4.3. Step 2: Compute similarity score between motion primitives of two models

Given $M'(n)$, the next step is to compute the similarity between motion primitives in the pre-trained model $M(n-1)$ and those in $M'(n)$. Since the learned motion primitives are essentially dictionary atoms [13], the measure of similarity between pairs of motion primitives used in this work is inspired by the concept of *coherence of dictionary atoms* [23, 24]. Mathematically, it is computed as the normalized inner product of two dictionary atoms $\mathbf{m}_i$ and $\mathbf{m'}_j$

$$S(\mathbf{m}_i, \mathbf{m}'_j) = \frac{\langle \mathbf{m}_i, \mathbf{m}'_j \rangle}{|\mathbf{m}_i||\mathbf{m}'_j|} \quad (1)$$

where, $\langle \cdot, \cdot \rangle$ is the inner product, $\mathbf{m}_i$ and $\mathbf{m'}_j$ denote the i-th motion primitive in $M(n-1)$ and the j-th motion primitive in $M'(n)$ respectively. Given Eq. (1), pairs of motion primitives $\mathbf{m}_i$ and $\mathbf{m'}_j$ with similarities greater than a pre-defined threshold, β, are considered as *matched motion primitives*. SILA finds all matched pairs of motion primitives between $M(n-1)$ and $M'(n)$ and fuses them, as discussed below.

For the fusion process, $M(n-1)$ and $M'(n)$ are represented by their dual motion primitive graphs [21]. Figure 1(c) shows the dual motion primitive graph of the model shown in Figure 1(a), wherein all motion primitives are represented by nodes and there exists a directed edge from nodes (*e.g.* $\mathbf{m_i} \rightarrow \mathbf{m}_k$) if, and only if, at least one *switching* behavior from $\mathbf{m}_i$ to $\mathbf{m}_k$ is observed. For brevity, the fusion mechanism is explained further using an example in Figure 2. Figure 2(a) illustrates the dual motion primitive graphs of models $M(n-1)$ and $M'(n)$. The similarity score between nodes in two graphs is computed using Eq. (1) to find the matched nodes (*i.e.* matched motion primitives). Figure 2(b) extends the graphs (original graphs are faded) in Figure 2(a) by adding extra edges in red that link the matched nodes from the two original graphs such that:

$$e_{i,j'} = \{(i,j') | S(\mathbf{m}_i, \mathbf{m}'_j) \geq \beta\}, \quad \omega_{i,j'} = S(\mathbf{m}_i, \mathbf{m}'_j). \quad (2)$$

Eq. (2) indicates that an *indirect weighted* edge $e_{i,j'}$ exists if, and only if, the similarity score between motion primitives $\mathbf{m}_i$ and $\mathbf{m}'_j$ is greater than the pre-defined threshold, β. The weight of such an edge is equal to the similarity score. This resulting new graph (called a similarity graph and denoted by G_s), has been considered for map merging and graph matching [25, 26]. We leverage this abstract graph to design an efficient fusion mechanism. Note that G_s by definition does not include unmatched nodes which implies that, as expected, the unmatched nodes are not going to be fused. We can consider the similarity graph as a set of connected components [27], as shown in dashed blue in Figure 2(b). The rest of this section presents fusion strategies based on the topology of the connected components. Recall each motion primitive is modeled by two GPs:(GP_x, GP_y), which is represented by self-transition in dual motion primitive graph. For the rest of this section, whenever two motion primitives or more are merged, their self- transitions, *i.e.* their corresponding GPs, are also merged. The fusion of motion primitives and GPs are explained later.

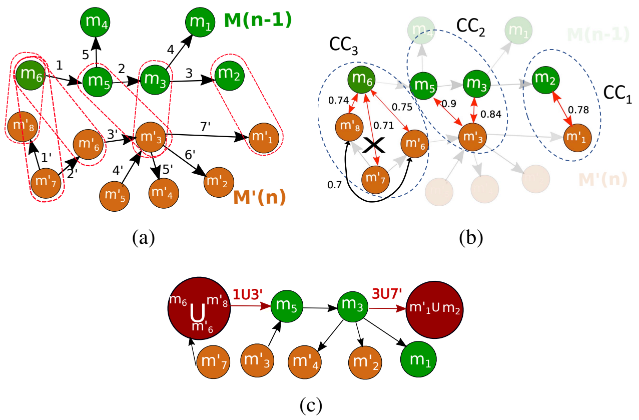


Figure 2. **(a)** Motion primitive graph of pre-trained $M(n-1)$ in green and new model $M'(n)$ in brown (self transitions are omitted for brevity), black arrows represent the directed transitions and they are indexed. Red dashed rectangles show pair-wise matched nodes between two models. **(b)** similarity graph G_s with threshold of $\beta = 0.6$ is shown on top of the original graphs in (a), unmatched nodes are faded for brevity. Edges of similarity graph with their assigned weight (similarity score) are shown in red arrows.(note: black arrow shows the intra-similarity between two nodes from same model which is considered in case 2) As shown, G_s has three connected components (dashed blue circles): CC_3, CC_2, CC_1. In CC_3 the edge with least similarity is relaxed (crossed).**(c)** final motion primitive graph corresponding to updated model $M(n)$ which is created after fusing the matched nodes based on the topology of connected components in (b) as described in the paper. Nodes with no fusion are colored by their original model color and fused nodes and edges are colored in dark red. Fusion is denoted by 'U'.

Connected components with one edge Also referred to as *one-to-one matching*, this is the case when the connected component comprises of two motion primitives ($\mathbf{m}_i$ and $\mathbf{m}'_j$), one from each model, that are similar. In this case, the two matched primitives and their corresponding self-transitions are merged (see Figure 2(b)-CC_1, $\mathbf{m}'_1$ and $\mathbf{m}_2$).

Connected components with two edges This case happens when two nodes from one model ($\mathbf{m}_i$ and $\mathbf{m}_j$) are matched with a single node ($\mathbf{m}'_k$) from another model[1] (see Figure 2(b)-CC_2, $\mathbf{m}_5$ and $\mathbf{m}_3$ are matched with $\mathbf{m}'_3$). The fusion strategy now depends on the relationship between $\mathbf{m}_i$ and $\mathbf{m}_j$ in their original dual motion primitive graph. These relationships are investigated in the following order:

Case 1: There exists a transition from $\mathbf{m}_i$ to $\mathbf{m}_j$. This implies that $\mathbf{m}_i$ and $\mathbf{m}_j$ together have a richer and more primitive representation of motion behaviours as compared to $\mathbf{m}'_k$ alone. Thus, $\mathbf{m}'_k$ is replaced with $\mathbf{m}_i$ and $\mathbf{m}_j$. All incoming edges of $\mathbf{m}'_k$ now enter $\mathbf{m}_i$ and all outgoing edges of $\mathbf{m}'_k$ now leave $\mathbf{m}_j$ (*e.g.* Figure 2(b)-CC_2, and procedure shown in Figure 1(d) and Figure 1(e)).

Case 2: No transition exists between $\mathbf{m}_i$ and $\mathbf{m}_j$ but they are similar (*i.e.* $S(\mathbf{m}_i,\mathbf{m}_j) \geq \beta$, when two motion primitives from one model are similar, we call it intra-similar). This implies that the learned motion primitives $\mathbf{m}_i$ and $\mathbf{m}_j$ are redundant, so the three motion primitives $\mathbf{m}_i$, $\mathbf{m}_j$ and $\mathbf{m}'_k$ and their self-transitions are fused to reduce this redundancy (see Figure 2(a)-CC_3, black arrow shows $\mathbf{m}'_6$ and $\mathbf{m}'_8$ are intra-similar).

Case 3: No transition exists between $\mathbf{m}_i$ and $\mathbf{m}_j$ and they are not similar (*i.e.* $S(\mathbf{m}_i,\mathbf{m}_j) < \beta$). This case happens when two motion primitives in one model are matched to a part of another primitive in another model, while themselves are not overlapping. In our experiments, we observed that such a case only arises when similarity threshold β is too small and the matching is not meaningful. No action is required in this case, however, selecting appropriate β minimizes these cases.

Connected components with three or more edges In this case, the edge with the least similarity value is successively relaxed (removed) until the connected component has only two edges left, which then allows for the use of Case 2 fusion strategy (see Figure. 2(b)-CC_3). We did not observe adverse effects on algorithm performance because of this relaxation as this case was rarely encountered. However, further investigation would benefit future work.

4.4. Step3: Model Fusion

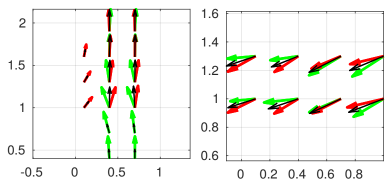
As discussed, transitions and motion primitives between two models $M'(n)$ and $M(n-1)$ are sometimes fused. This part explains fusion process for transitions and primitives.

4.4.1 Fusion motion primitives and transitions

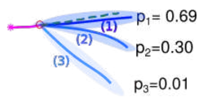
To effectively fuse a pair of motion primitives $\mathbf{m}_i$ and $\mathbf{m}'_j$, we simply compute the element-wise average between the two vectors, denoted by $avg(\mathbf{m}_i,\mathbf{m}'_j)$, is given by:

$$avg(\mathbf{m}_i,\mathbf{m}'_\mathbf{j}) = \frac{1}{N}\sum_{k=1}^{N} \mathbf{m}_i^k + \mathbf{m'}_j^k \qquad (3)$$

[1] All cases are applicable to the reverse order of notations *e.g.* when $\mathbf{m}'_i$ and $\mathbf{m}'_j$ are matched with node $\mathbf{m}_k$, it is also considered as the case of connected components with two edges.



(a)



(b)

Figure 3. (a) Similar motion primitives of $M(n-1)$ (green) and $M'(n)$ (red). Matched primitives are fused (black) by considering the cell-wise average of each vector. Here, $\beta = 0.6$. (b) Example prediction using SILA. Trajectory is observed for 3.2 s (magenta) and predicted for 4.8 s (blue). Prediction is a set of Gaussian distributions (blue shading). The number next to each distribution shows the prediction likelihood. Ground truth is the dashed line.

where, N is the dimension of motion primitives, and $\mathbf{m}_i^k$ is the k-th element of $\mathbf{m}_i$. For example, in Figure 3(a), the average of $\mathbf{m}_i$ (red arrows) and $\mathbf{m}'_j$ (green arrows) gives the fused motion primitive $\mathbf{m}''_i$ (black arrows). The updated set of motion primitives is denoted by $\mathbf{D}''$ and comprises of all fused nodes and remaining non-fused nodes from the two models. The rest of this section describes the process of fusing the GP-based transitions (refer Section 3) between motion primitives. Fusion of the transitions means the fusion of their corresponding GPs.

As described in Section 3, each transition, either self transitions or non-self transitions, is modeled as a pair of two dimensional GP flow fields (GP_x,GP_y), where each GP is represented by a sparse number of *pseudo inputs*. Let matched transition in $M(n-1)$ be denoted by r. For the new model $M'(n)$, the data associated with matched transition is available and is denoted by d'. To fuse GPs associated with matched transitions in $M(n-1)$ and $M'(n)$, the GP regression method in Ref. [28] is used to update r incrementally using d' (note that data associated with r is assumed not be available, because it is a transition in pre-trained model and we have only access to pseudo inputs corresponding to transitions in $M(n-1)$. Otherwise, we could simply learn a new GP from new data and old data). (see edges 1 and $3'$ are fused in Figure 2(c), also edges 3 and $7'$ are fused.).

4.5. Prediction phase

Given an observed trajectory, the motion primitive corresponding to the most likely self-transition GP is obtained and is referred to as the observed primitive. All transitions from the observed primitive, as can be found in the transition set $\mathbf{R}$, are considered as possible future directions. Recall each transition is modeled using GPs. Thus, predicted future directions are mathematically represented as a set of Gaussian distributions (refer Figure 3(b) - blue shades). The set of predicted trajectories is then obtained by sampling from these predicted Gaussian distributions.

5. Experimental Results

Two sets of experiments were conducted: (1) evaluate the SILA learning process on different intersections, and (2) compare SILA's performance with prior work using non-intersection datasets.

5.1. Evaluation metrics

Consistent with prior works [5–7], the following two metrics were used to evaluate SILA's performance:

1. Average Displacement Error (ADE) [16] defined as the average Euclidean distance between the predicted and ground truth trajectory over a fixed time horizon H:

$$ADE = \frac{1}{H} \sum_{i=t_{obs}+1}^{t_{obs}+H} \left\| p^{i}_{pred} - p^{i}_{gt} \right\|_2, \tag{4}$$

where p^t_{pred} and p^t_{gt} are points at time t in the predicted and ground truth trajectories respectively. t_{obs} is the time corresponding to the last observed point, before prediction starts.

2. Final Displacement Error (FDE) defined as the prediction error at the end of the fixed time horizon H [7]:

$$FDE = \left\| p^{t_{obs}+H}_{pred} - p^{t_{obs}+H}_{gt} \right\|_2. \tag{5}$$

5.2. Transfer knowledge across intersections

As motivated in Section 1, intersections are challenging domains for pedestrian trajectory prediction because of several reasons. One such reason is the influence of intersection geometry such as sidewalk boundaries on pedestrian behavior. The experiments in this section analyze the ability of SILA to transfer knowledge across intersections of different geometries. In this experiment, the model is trained on the dataset $I_{train} = \{I_A, I_B, I_C, I_D\}$, the components of which were collected in three busy intersections $\{A, C, D\}$ located in two crowded cities of Boston and Cambridge, including residential and business areas. I_B was collected in Mcity [29] to add more diversity. The training dataset includes 611 trajectories in total (see Figure 4). Post incremental learning from these 4 datasets, the model is tested on a fifth dataset $\{I_E\}$ with 114 trajectories collected at an intersection near MIT campus with a completely different geometry. Trajectories are annotated 2.5 fps. Each test trajectory is observed for 3.2 sec and predicted for 4.8 sec. We sample 20 trajectories from the prediction distribution and the best one is selected for evaluation. Figure 5(a) shows SILA incrementally improves the prediction model while simultaneously transferring knowledge across intersections with different geometries. The order in which SILA trains on the different datasets can influence the learning trend. To analyze this effect, the training datasets are fed in different permutations. Results from five different permutations are shown in Figure 5(a). Prediction error, on an average, at the end of the 4th training episode is given by $ADE = 0.65 \pm 0.03$, while batch learning (TASNSC) on the four datasets combined results in $ADE = 0.69 \pm 0.006$. Learning using SILA results in a model comprising of 63 motion primitives and 194 transitions. In contrast, batch learning learns 52 motion primitives and 198 transitions. In this experiment, batch learning took 1 sec in average, however, SILA 's learning on each mini batch data only took 0.05 sec which is 20 times faster than batch learning.

5.3. Ablation study

In order to select similarity threshold, We ran another set of experiments on datasets (I_A) collected in intersection A. Data was collected using two 2D Lidars and three RGB cameras and contains 1228 trajectories annotated at a frequency of 2.5 fps. 1044 trajectories were used for training by splitting into mini-batches of 29 trajectories each that were incrementally fed. This experiment is repeated 12 times in which the order of feeding mini-batches is shuffled. ADE on the held-out test dataset is used for quantitative analysis. We follow similar procedure in section 5.2 for evaluation (*i.e.* 8 frames observation, 12 frames for prediction and choose the best trajectory among 20 samples). To find the best similarity threshold, the experiments are repeated for thresholds ranging from $\beta = 0.5$ to $\beta = 1.0$. SILA-β denotes SILA with a similarity threshold of β. Figure 5(b) summarizes the results from these experiments. Note the consistent decrease in ADE as the model is exposed to more data. Increasing the similarity threshold discourages the fusion of motion primitives (for $\beta = 1$ only pairs of motion primitives that are exactly identical are matched and fused). As shown in Fig. 5(c), $\beta = 0.6$ gives the best results in terms of the trade-off between model size (as a linear function of number of motion primitives and transitions) and prediction error. We also studied SILA-Naive in which the motion primitives are not fused (same as $\beta > 1$). Figures 5(c) shows SILA-Naive model size is up to $3\times$ larger than that of SILA-0.6.

5.4. Comparison with Baselines

To compare SILA with prior works, we run another set of experiments on datasets and setups similar to the ones used in previous works [5–7]. The datasets used for this set of experiments are: ETH [16] and UCY [30]. ETH consists of two scenes - ETH and Hotel, and UCY contains three scenes - ZARA1, ZARA2 and UNIV, which makes a total of five datasets. These datasets are annotated at a frequency of 2.5 fps. Similar to prior works, we use the leave-one-out method in which 4 out of the 5 datasets are selected for training and the model is evaluated on the fifth dataset. Sim-

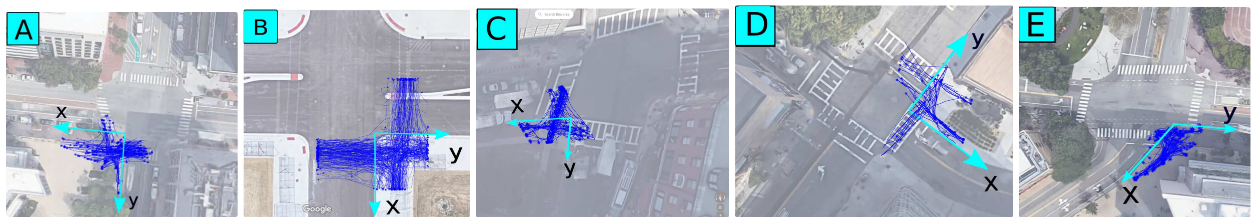


Figure 4. Pedestrian trajectory data (blue curves) collected in 5 intersections denoted by the letters $A-E$ with different geometries (right, closed and open angle corners) described in the paper. Cyan arrows show intersection axes with the origin at the corner of interest. The intersections and trajectories are shown in a bird's eye view taken from Google Maps, however the trajectory data is collected locally on the ground of each intersection, except for the data collected in B using GPS.

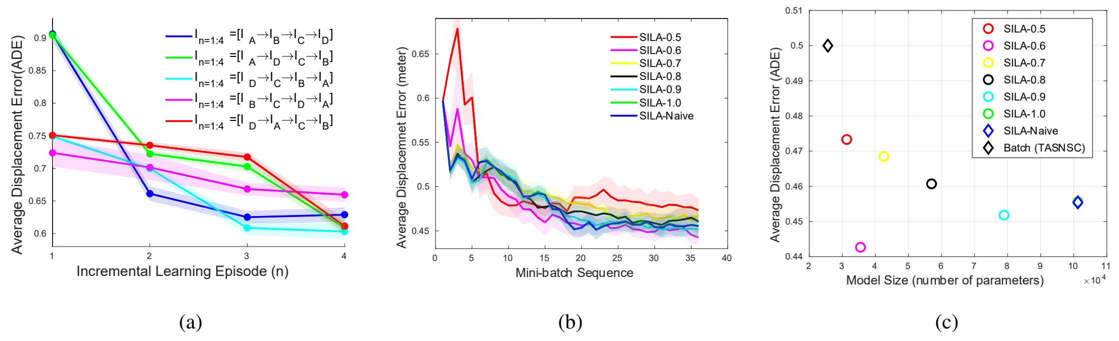


Figure 5. (a) SILA prediction error (ADE) on the test data (I_E) using models incrementally trained on data from intersections shown in Fig. 4. The datasets from intersections are used in the various ordering shown in different color, leading to different results. (b) Average of ADE over 12 trials when the data is incrementally available in mini-batches of 29 trajectories from I_A in Fig. 4, The similarity threshold varies from 0.5 to 1. (c) Ablation study for choosing similarity threshold. Threshold $\beta = 0.6$ is selected for the best trade-off between model size growth and prediction error (ADE).

ilar to [5–7], two additional data (ZARA3 and uni example from UCY) are used for training but not for evaluation.

In the evaluation phase, each trajectory is observed for 3.2 s (8 frames) and future trajectory is predicted for the next 4.8 s (12 frames). Except for the linear model (refer Section 5.4.1), all other baselines and SILA are probabilistic models that predict a Gaussian distribution. Following prior works [5–7], we sample 20 trajectories from the predicted distribution and select the sampled trajectory closest to ground truth for evaluation. Models trained using SILA and SILA-Naive (refer Section 5.4.1) learn incrementally and thus are sequentially fed datasets. There are many permutations of the order of feeding training datasets (*i.e.* $n!$ for n datasets) and we randomly choose one such permutation for these experiments, as summarized in Table 1.

Test dataset	Order of providing training datasets
ETH	uni examples, st3, st1, Zara3, Hotel, Zara2, Zara1
Univ(st1,st3)	Hotel, Zara3, uni examples, Zara2, Zara1, ETH
Hotel	uni examples,st3, st1, Zara3, ETH, Zara2, Zara1
Zara1	uni examples,st3, st1, Zara3, ETH, Zara2, Hotel
Zara2	uni examples,st3, st1, Zara3, ETH, Zara1, Hotel

Table 1. Order of feeding dataset to SILA and SILA-Naive models

5.4.1 Baselines

SILA-0.6 (*i.e.* with $\beta = 0.6$) is compared with the eight baselines listed in Table 2 that lie in four categories: (1) Linear model assuming constant velocity for prediction; (2) TASNSC [15] which is an offline (batch) version of SILA; (3) SILA-Naive which simply *adds* motion primitives and transitions learned from new data (refer Section 5.3); (4) Recent works on interaction-aware pedestrian motion prediction: GAT [31], SGAN [7], STGAT [6], STSGN [32], and Social-STGCNN [5].

5.4.2 Results

Figure 6(a) and Figure 6(b) show the trend of improving ADE and FDE respectively as training datasets are incrementally fed to models trained using SILA. Note not all datasets help improve prediction error. For example, when testing on Zara2 dataset, adding training data from ZARA1 and Hotel does not improve prediction. Figure 6(c) shows that the model size growth rate for SILA is slower than SILA-Naive which indicates the size efficiency of SILA. Table 2 shows that while SILA outperforms all baselines in terms of ADE on 2 out of the 5 test datasets, Social-STGCNN (and in one case STGAT) consistently outper-

Algorithm	ETH	Hotel	Univ	Zara1	Zara2	Average
Linear*	0.92/2.20	0.37/0.84	0.62/1.44	0.45/1.04	0.58/1.33	0.53/1.24
SGAN-20V [7]	0.71/1.27	0.48/1.03	0.56/1.18	0.34/0.68	0.31/0.64	0.48/0.96
SGAN-20VP [7]	0.77/1.40	0.43/0.87	0.75/1.50	0.35/0.70	0.36/0.72	0.51/1.02
STGAT [6]	0.72/1.45	**0.24/0.44**	0.50/1.09	0.34/0.72	0.28/0.63	0.41/0.86
CGNS [33]	0.62/1.40	0.70/0.93	0.48/1.22	0.32/0.59	0.35/0.71	0.49/0.97
STSGN [32]	0.75/1.63	0.63/1.01	0.48/1.08	0.30/0.65	**0.26**/0.57	0.48/0.99
GAT [31]	0.68/1.29	0.68/1.40	0.57/1.29	**0.29**/0.60	0.37/0.75	0.52/1.07
Social-STGCNN [5]	0.63/**1.08**	0.49/0.86	**0.44/0.79**	0.34/**0.53**	0.30/**0.48**	0.44/**0.74**
TASNSC [15]	0.78/1.40	0.42/0.90	0.61/1.34	0.50/1.05	0.52/1.14	0.56/1.16
SILA-Naive (ours)	0.77/1.83	0.28/0.63	0.60/1.31	0.38/0.84	0.43/0.90	0.49/1.10
SILA (ours)	**0.56**/1.23	0.27/0.63	0.55/1.25	**0.29**/0.63	0.32/0.72	**0.39**/0.89

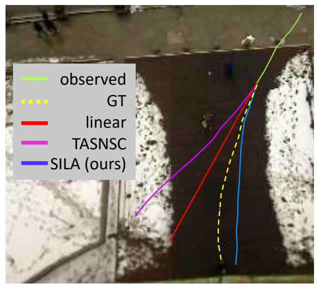


Table 2. Left: ADE/FDE of baselines compared to SILA for each test dataset using leave-one-out method. All models excepts linear model are probabilistic. To evaluate these models, the best out of 20 sampled trajectories is considered similar to in [5, 6], samaller is better. Results of SGAN, STGAT, social-STGCNN and TASNSC are obtained by running their code. Other baselines results are obtained from their papers. Right: qualitative comparison of SILA with TASNSC, linear model and ground truth (GT) in a scene from ETH dataset.

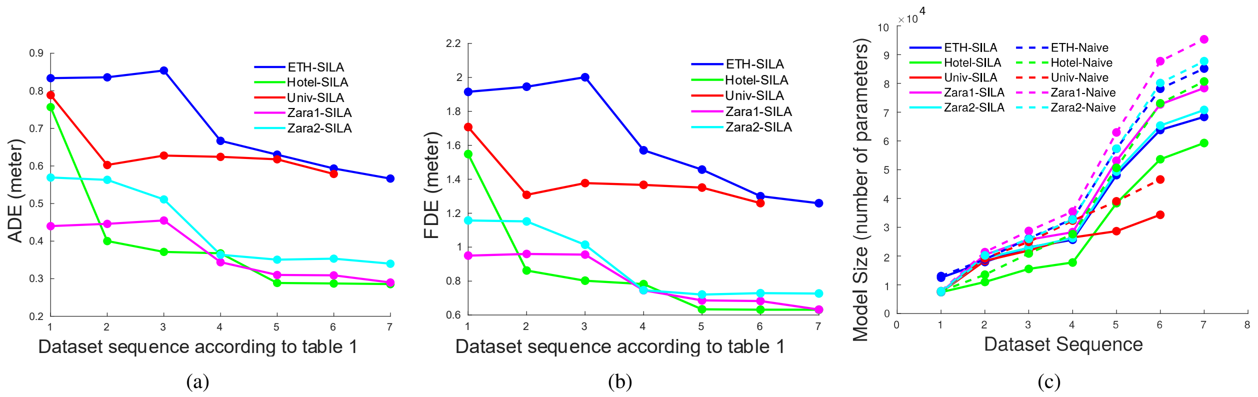


Figure 6. (a) SILA prediction error (ADE) on a held-out test set when training datasets are incrementally added as summarized in Table 1 (each experiment is shown in a different color). (b) SILA's FDE for the experiment in (a). (c) Model size (number of parameters) growth with SILA (solid lines) compared to that with SILA-Naive (dashed lines) as data is added incrementally according to (a).

form other methods, including SILA, in terms of FDE as it accounts for pedestrian interaction and hence does better in terms of long-term prediction. However, SILA had the additional advantages of (i) transferring knowledge across environments with different geometries; (ii) learning incrementally from new data which requires lesser compute and memory; and (iii) multi-modal prediction (*i.e*. SILA predicts a set of Gaussian distribution as opposed to a single distribution). As a result SILA-0.6 can robustly generalize to new data and achieve better ADE on average (refer last column in Table 2). Moreover, SILA always outperforms TASNSC, even though both have similar structure and learn from similar data (see qualitative comparison on the right side of Table 2). Fusion in SILA decreases redundancy in motion primitives and thus reduces overfitting, which leads to better generalization to new datasets than batch learning as well as SILA-Naive, where adding motion primitives without fusion may lead redundant motion primitives and increase overfitting. Fusion of motion primitives can also create new motion primitives that are better representative of pedestrian behaviors as compared to batch learning (with the objective of minimizing the number of motion primitives over the entire dataset) which can discourage learning motion primitives and transitions that represent underrepresented behaviors. Instead, these rare behaviors can be captured and memorized through incremental learning.

6. Conclusion

This paper presents SILA for pedestrian trajectory prediction in scenarios where the training data is incrementally available. SILA consistently outperforms its batch learning version (*i.e*. TASNSC) and on an average outperforms prior batch learning methods in terms of ADE with 5% improvement. Future work includes extension of SILA to multiagent learning problems.

Acknowledgement

Special thanks to Luke Qi and Justin Miller for their great help in data collection. This project has been supported by Ford Motor Company.

References

[1] F. Schneemann and P. Heinemann, “Context-based detection of pedestrian crossing intention for autonomous driving in urban environments,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2243–2248, IEEE, 2016.

[2] J. F. P. Kooij, N. Schneider, F. Flohr, and D. M. Gavrila, “Context-based pedestrian path prediction,” in *European Conference on Computer Vision*, pp. 618–633, Springer, 2014.

[3] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, “Context-aware trajectory prediction,” in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 1941–1946, IEEE, 2018.

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

[5] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, “Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction,” *arXiv preprint arXiv:2002.11927*, 2020.

[6] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, “Stgat: Modeling spatial-temporal interactions for human trajectory prediction,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6272–6281, 2019.

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

[8] J. Amirian, J.-B. Hayet, and J. Pettré, “Social ways: Learning multi-modal distributions of pedestrian trajectories with gans,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[9] K. Shmelkov, C. Schmid, and K. Alahari, “Incremental learning of object detectors without catastrophic forgetting,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3400–3409, 2017.

[10] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *International journal of computer vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[11] Z. Wang, C. Chen, H.-X. Li, D. Dong, and T.-J. Tarn, “Incremental reinforcement learning with prioritized sweeping for dynamic environments,” *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 2, pp. 621–632, 2019.

[12] D. Vasquez, T. Fraichard, and C. Laugier, “Incremental learning of statistical motion patterns with growing hidden markov models,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 403–416, 2009.

[13] Y. F. Chen, M. Liu, and J. P. How, “Augmented dictionary learning for motion prediction,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2527–2534, IEEE, 2016.

[14] E. Snelson and Z. Ghahramani, “Sparse gaussian processes using pseudo-inputs,” in *Advances in neural information processing systems*, pp. 1257–1264, 2006.

[15] N. Jaipuria, G. Habibi, and J. P. How, “Learning in the curbside coordinate frame for a transferable pedestrian trajectory prediction model,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3125–3131, IEEE, 2018.

[16] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, “You’ll never walk alone: Modeling social behavior for multi-target tracking,” in *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, IEEE, 2009.

[17] D. Kulić, W. Takano, and Y. Nakamura, “Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains,” *The International Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008.

[18] S. Ferguson, B. Luders, R. C. Grande, and J. P. How, “Real-time predictive modeling and robust avoidance of pedestrians with uncertain, changing intentions,” in *Algorithmic Foundations of Robotics XI*, pp. 161–177, Springer, 2015.

[19] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, “A bayesian nonparametric approach to modeling motion patterns,” *Autonomous Robots*, vol. 31, no. 4, p. 383, 2011.

[20] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns,” *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[21] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, “Incremental learning of full body motion primitives and their sequencing through human motion observation,” *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[22] R. M. Bowen and C.-C. Wang, *Introduction to vectors and tensors*, vol. 2. Courier Corporation, 2008.

[23] S. Ubaru, A.-K. Seghouane, and Y. Saad, “Improving the incoherence of a learned dictionary via rank shrinkage,” *Neural computation*, vol. 29, no. 1, pp. 263–285, 2017.

[24] H. Bai, S. Li, and Q. Jiang, “An efficient algorithm for learning dictionary under coherence constraint,” *Mathematical Problems in Engineering*, vol. 2016, 2016.

[25] T. M. Bonanni, B. Della Corte, and G. Grisetti, “3-d map merging on pose graphs,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1031–1038, 2017.

[26] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, “Graph matching networks for learning the similarity of graph structured objects,” *arXiv preprint arXiv:1904.12787*, 2019.

[27] J. Hopcroft and R. Tarjan, “Algorithm 447: Efficient algorithms for graph manipulation,” *Commun. ACM*, vol. 16, pp. 372–378, June 1973.

[28] H.-I. Suk, Y. Wang, and S.-W. Lee, “Incremental sparse pseudo-input gaussian process regression,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 08, p. 1250019, 2012.

[29] Y. Dong, Y. Zhong, W. Yu, M. Zhu, P. Lu, Y. Fang, J. Hong, and H. Peng, “Mcity data collection for automated vehicles study,” *arXiv preprint arXiv:1912.06258*, 2019.

[30] A. Lerner, Y. Chrysanthou, and D. Lischinski, “Crowds by example,” in *Computer graphics forum*, vol. 26, pp. 655–664, Wiley Online Library, 2007.

[31] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, “Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks,” in *Advances in Neural Information Processing Systems*, pp. 137–146, 2019.

[32] L. Zhang, Q. She, and P. Guo, “Stochastic trajectory prediction with social graph network,” *arXiv preprint arXiv:1907.10233*, 2019.

[33] J. Li, H. Ma, and M. Tomizuka, “Conditional generative neural system for probabilistic trajectory prediction,” *arXiv preprint arXiv:1905.01631*, 2019.

A. Implementation of SILA

Figure 7 summarizes all steps of SILA and the pseudo code for main algorithm SILA is summarized in Algorithm 1. In this section, some details on SILA implementation such as indexing motion primitives and transitions are explained.

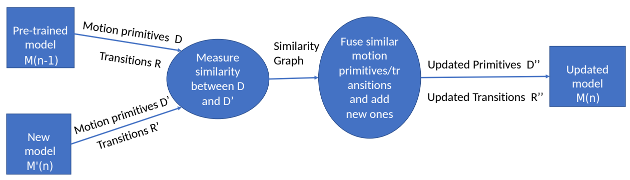


Figure 7. Block diagram of SILA

A.1. Initialize Indexes of Motion Primitives and Transitions

SILA's ultimate goal is to efficiently merge the pre-trained model $M(n-1)$ with the new model $M'(n)$ to get a single model $M(n)$ that is best representative of all motion behaviours seen so far. The first step towards this goal is to re-index the motion primitives and transitions from $M(n-1)$ and $M'(n)$ to avoid repetition of indices. Thus, initially the accumulated model $\hat{M}(n)$ simply combines $M(n-1)$ with $M'(n)$, such that $\hat{\mathbf{D}} = \{\mathbf{D}, \mathbf{D}'\}$ is of size $N_1 + N_1'$. Similarly, the set of accumulated transitions $\hat{\mathbf{R}} = \{\mathbf{R}, \mathbf{R}'\}$ is of size $N_2 + N_2'$ (Alg. 1,line 4) Each node is represented by two *virtual nodes*, denoted by *in* and *out*, and a *virtual transition* (edge): (in, out). Given this *virtual* representation of each node, we need to re-index edges such that all edges entering $\mathbf{m}_k$ now enter $\mathbf{m}_{k,in}$; and all edges exiting $\mathbf{m}_k$ now exit $\mathbf{m}_{k,out}$. This implies that for every transition edge (i, j) (*i.e*. edges exiting from $\mathbf{m}_i$ and entering $\mathbf{m}_j$) in $\hat{\mathbf{R}}_{edges}$ re-indexing is done as follows: $(i, j) \mapsto (\mathbf{m}_{i,out}, \mathbf{m}_{j,in})$. In the beginning of each learning episode, indices *in* and *out* for all nodes are initialized equivalently. For instance, $\mathbf{m}_{k,in} = \mathbf{m}_{k,out} = k$ for the k-th motion primitive in accumulated motion primitive set.

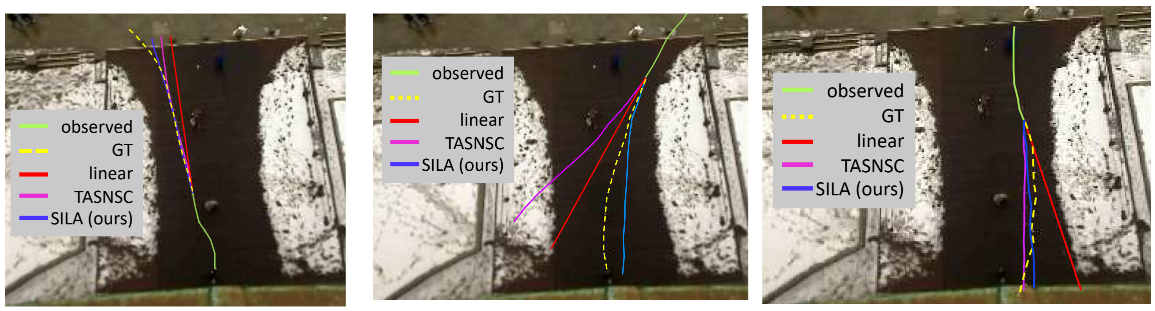
A special set, called the *fusion set*, is also defined and initialized as an empty set in this step. Eventually, it contains a list of tuples of nodes that need to be fused. *e.g*. the tuple $(\mathbf{m}_i, \mathbf{m}_j)$ implies $\mathbf{m}_i$ and $\mathbf{m}_j$ need to be fused in the model fusion step.

A.2. Qualitative results

Figure 8 shows SILA performance compared to two baselines TASNSC, as its batch version, and linear model (constant velocity) in three different scenarios in ETH. This results show SILA always outperforms TASNSC and linear model in predicting the pedestrian trajectory. (note: Figure 8(b) was also reported in main result section.)

Algorithm 1: $M(n) = SILA(M(n-1), M'(n))$

```
1  Input: previous model $M(n-1)\{\mathbf{D},\mathbf{R}\}$ and incremental model $M'(n)\{\mathbf{D}',\mathbf{R}'\}$
2  Output: updated model $M(n)\{\mathbf{D}'',\mathbf{R}''\}$
3  $\hat{M}(n) \leftarrow \{\hat{D}\{D:D'\}, \hat{R}\{R:R'\}\}, \emptyset \leftarrow FuseSet$
4  $[\hat{D},\hat{R}] = Reindex(\hat{D},\hat{R})$
5  $G_s = SimilarityGraph(D,D')$
6  $CC = ConnectedComponents(G_s)$
7  for $k = 1:|CC|$ do
8      $ee = |edges(CC_k)|, [h,c] = nodesIdx(CC_k)$
9      if $ee == 1$ then
10         $\mathbf{m}_{i,in} = \mathbf{m}_{i,out} = min(\mathbf{m}_{h,in}, \mathbf{m}_{c,in}), i \in \{h,c\}$
11         $FuseSet \leftarrow FuseSet \cup \mathbf{m}_h \cup \mathbf{m}_c$
12     else if $ee == 2$ then
13         for $\mathbf{m}_i \in nodes(CC_k)$ do
14             if $Degree(\mathbf{m}_i) == 2$ then
15                 $[w,q] = nbrsIdx(\mathbf{m}_i)$//neighbor index
16                 if $(\mathbf{m}_w, \mathbf{m}_q) \in \hat{\mathbf{R}}$ then
17                     $\mathbf{m}_{i,in} \leftarrow \mathbf{m}_{w,out}$
18                     $\mathbf{m}_{i,out} \leftarrow \mathbf{m}_{q,in}$
19                 else if $S(\mathbf{m}_w, \mathbf{m}_q) \geq \beta$ then
20                     $\forall j \in \{w,q,i\}, \mathbf{m}_{j,in} = \mathbf{m}_{j,out} = min(\mathbf{m}_{w,in}, \mathbf{m}_{q,in}, \mathbf{m}_{i,in})$
21                     $FuseSet \leftarrow FuseSet \cup \mathbf{m}_i \cup \mathbf{m}_w \cup \mathbf{m}_q$
22     else
23         $CC_k = Relaxedges(CC_k, 2)$
24         Go to line 13
25 $\mathbf{D}'' = FusePrimitives(FuseSet)$
26 $\mathbf{R}'' = Reindex\&FuseEdges(\hat{\mathbf{R}}, \mathbf{D}'')$
27 return $M(n)\{\mathbf{D}'',\mathbf{R}''\}$
```



(a) (b) (c)

Figure 8. Comparison of SILA prediction with two baselines (TASNSC and linear), when the trajectory is observed for 3.2 sec (green) and predicted for 4.8 sec, the predictions are compared with ground truth (GT). (a) In this scenario all algorithms predict closely to the ground truth. (b) In this scenario TASNSC and linear model fails to predict the pedestrian trajectory accurately. (c) In this scenario TASNSC and SILA perform well, while linear model fails to predict accurately.