Using Deep Siamese networks for trajectory analysis to extract motion patterns in videos

Using Deep Siamese networks for trajectory analysis to extract motion patterns in videos

Jonathan Boyle,[1,✉] Tahir Nawaz,[2] and James Ferryman[1]

[1]*Department of Computer Science, School of Mathematical Physical and Computational Sciences, University of Reading, Reading, UK*

[2]*Department of Mechatronics Engineering, National University of Sciences and Technology, Islamabad, Pakistan*

✉Email: j.n.boyle@reading.ac.uk

This paper investigates the use of Siamese networks for trajectory similarity analysis in surveillance tasks. Specifically, the proposed approach uses an auto-encoder as a part of training a discriminative twin (Siamese) network to perform trajectory similarity analysis, thus presenting an end-to-end framework to perform an online motion pattern extraction in the scene with an ability to incorporate new incoming trajectory(ies) incrementally. The effectiveness of the proposed method is evaluated on four challenging public real-world datasets containing both vehicle and person targets, and compared with five existing methods. The proposed method consistently shows better or comparable performance than the existing methods on all datasets.

Introduction: Motion patterns represent spatiotemporal trends of moving targets in a scene and can be extracted based on the analysis of motion information [1–4]. Indeed, these extracted patterns of motion of targets offer useful information that aid in performing activity analysis [5], behaviour prediction [6], tracking [7], and abnormality detection [8].

Existing motion pattern extraction methods are generally classified as interframe-motion-based and multiframe-motion-based methods. Interframe-motion-based methods [9–11] rely on using motion information of targets between two consecutive frames to extract motion patterns. These methods generally perform more robustly in crowded scenarios but are often considered not suitable for extracting long-range motion patterns [12]. Multiframe-motion-based methods [1–3, 13] instead use motion information across multiple frames, e.g. short-duration tracks or complete trajectories, as estimated with video tracking and are regarded to be more helpful in extracting long-range patterns. These methods generally involve first estimating tracked trajectories of targets [1–3, 12], followed by encoding trajectory information into feature space(s) [2, 13–15], and performing trajectory clustering [1, 2, 4, 13, 14] or classification [3, 15] to determine dominant patterns. Some of these approaches [2, 4, 13, 14] work in an offline manner as they assume availability of all estimated trajectories a priori with an inability to incorporate new trajectory(ies) at clustering or classification stage incrementally. Online approaches [1, 3, 15] do exist that enable updating extracted clusters incrementally with incoming trajectory(ies). Recently there has been a growing focus on the use of deep learning techniques in performing various vision tasks including those involving trajectory analysis [16, 17] but are not specifically aimed at motion pattern extraction.

This paper presents a framework that is built upon using auto-encoder architecture as part of training of the Siamese networks to perform trajectory similarity analysis for extracting motion patterns. Unlike existing methods [2, 4, 13, 14], this paper proposes an online approach that allows incorporating new incoming trajectory(ies) in an incremental manner. Moreover, unlike existing online approaches [1, 3, 15], the proposed method presents an end-to-end deep learning based trained network without the need for separate explicit feature extraction and clustering/classification stages; hence no requirement either for an enhanced discriminative ability of feature(s) as in [1, 3, 15] or an appropriate choice of clustering techniques as in refs. [1, 3]. We show the effectiveness of the proposed method by evaluating and comparing the performance with several state-of-the-art methods on four challenging public real-world datasets with significant variability.

Problem definition: Let $\mathcal{X}$ be a set of trajectories estimated by a tracker in a video sequence, V: $\mathcal{X} = \{\mathfrak{X}_j\}_{j=1}^{J}$, where J is the number of estimated trajectories. $\mathfrak{X}_j$ is the estimated trajectory for target j: $\mathfrak{X}_j = (X_{k,j})_{k=k^j_{\text{start}}}^{k^j_{\text{end}}}$, where k^j_{start} and k^j_{end} are the first and final frame numbers of $\mathfrak{X}_j$, respectively. $X_{k,j}$ is the estimated state of target j at frame k : $k = 1, \ldots, K$ with K as the total number of frames in V. $X_{k,j} = (x_{k,j}, y_{k,j})$, where $(x_{k,j}, y_{k,j})$ denotes at frame k the position of target j on the image plane. The analysis of trajectories ($\mathcal{X}$) aids in identifying the motion patterns that refer to the representative spatiotemporal trends of moving targets (people, vehicles etc.) in a scene.

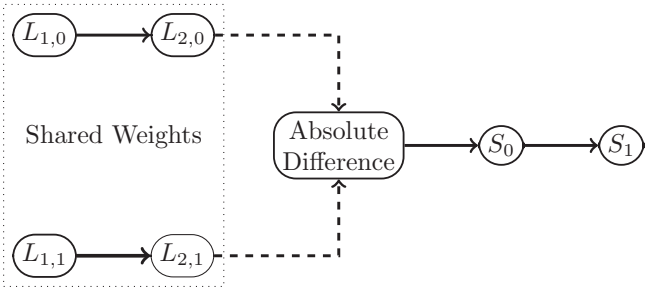


Fig. 1 *Architecture of the full Siamese network with all layers fully connected*

Neural network architecture: The overall architecture of the neural network is split into two parts: an encoding part trained as an auto-encoder and a second part containing its extension into a twin (or Siamese) neural network. The full architecture can be seen in Figure 1. An auto-encoder was chosen as the first stage as it has been shown to be very effective in creating a trajectory feature vector to discriminate between clusters [2] and is therefore expected to be a good discriminator between trajectories as part of a Siamese network.

In order to generate the feature vector $\mathbf{f}_j$ for a trajectory $\mathfrak{X}_j$, an auto-encoder, also known as auto-associator or Diabolo network [18], is first trained to reproduce the set of trajectories $\mathcal{X}$. Once a network has been trained, the output of its smallest layer is used as the feature vector $\mathbf{f}j$. A separate network has to be trained for each dataset due to the varying length of the input vectors described above. A network consisting only of fully-connected layers cannot handle input vectors of varying sizes without introducing further methods of normalisation. Due to this reason as well as the difference in scenes of different datasets, we opted to use separate networks.

An auto-encoder is an arrangement of a neural network where the output, once trained, is an estimation of the provided input vector. Outputs of any of the layers in a trained network can be used as a representation of the input, due to the ability of rest of the network to reproduce the input vector. In this case, the input is the vectorisation of the data of $\mathfrak{X}_j$ and is denoted as $\mathbf{V}_{\mathfrak{X}_j}$: $\mathbf{V}_{\mathfrak{X}_j} = [x_{k,j}, y_{k,j}]_{k=k^j_{\text{start}}}^{k^j_{\text{end}}}$. A neural network is a combination of small units called neurons that are built up into multiple layers. The type of neuron used in this paper is based on the McCulloch–Pitts neuron [19]; this multiplies a single-dimensional input vector with a weight vector summed with a bias node, and outputs a single value: $y_{l,n} = \sum_i^{I_l} (w_{i,n}\mathbf{x}_{l,i}) + b_n$, where $y_{l,n}$ denotes the output of pre-activation function y of the neuron n in layer l. I_l is the length of the input vector X_l for a particular layer, w_i and $\mathbf{x}_{l,i}$ is a value in the weight vector and input vector, respectively, and b is the bias value. The weight vector is initialised to random values. A sigmoidal function is used for the activation function:

$$Y_{l,n} = \frac{1}{1 + e^{-y_{l,n}}}, \tag{1}$$

where Y is the output of the neuron. The neurons are then placed alongside each other as a layer. The output of a layer l can be described as follows: $L_l = [Y_{l,1}, \ldots, Y_{l,I_l}]$, where the layer L_l is a vector containing all of the outputs of the neurons in the layer. The next layer is then provided with the output of the previous one as its input vector (known as a fully-connected layer), excluding the first layer ($l = 1$) for which the input vector is the vectorised trajectory $\mathbf{V}_{\mathfrak{X}_j}$ rather than a previous layer:

$$\mathbf{X_l} = \begin{cases} \mathbf{V}_{\mathfrak{X}_j}, & \text{if } l = 1; \\ L_{l-1}, & \text{if } l > 1. \end{cases} \tag{2}$$

Table 1. The hidden layer sizes of auto-encoder for different datasets

Dataset	Input	L_1 size	L_2 size
Traffic Junction	358	35	17
Parking Lot	1092	109	54
Train Station	1462	146	73
Students003	994	99	49

The architecture of this type of network contains two stages: an encoding and a decoding stage. For encoding, the number of neurons in each layer decreases so that the dimensionality of the original input vector is effectively reduced when passed through the network. The decoding stage is the opposite: a set of layers incrementing in size up to the original length of the input vector. Both of these stages consist of one or more layers. As mentioned above, this method uses pre-defined scales based on the length of the original feature vector to determine the number of neurons in each layer in the encoding stage. Two layers are used: the first is 10% of the length of the vector, the second 5% of the vector. Only one layer is used in the decoding stage of the same size as the original vector. These scales are static across all of the datasets used in this study. The values of layer sizes are listed in Table 1.

The concept of a Siamese network [20] is to create a network of two parts. The network takes two inputs ($\mathbf{V}_{\mathfrak{X}_{j0}}$ and $\mathbf{V}_{\mathfrak{X}_{j1}}$) through the previously trained feature network (excluding the final decoding layer) to produce two outputs, in this case: $L_{2,0}$ and $L_{2,1}$. For the purposes of training and testing, $L_{2,1}$ is an average of all the trajectories in $L_{2,0}$'s class. A depiction of the full network can be seen in Figure 1. The absolute difference of the two vectors is calculated for the input of the Siamese layer: $D = |L_{2,0} - L_{2,1}|$. The input is then forwarded into a layer equal to the size of L_2, S_0, into a single layer of a single neuron S_1. Both layers use a sigmoidal activation function as in Equation 1.

The datasets are split into various sets with the following ratios: training 0.4, validation 0.2, testing 0.4. There are two stages for training the overall network. Both parts of training use the normalised direction preserving ADAM optimiser (ND-ADAM) [21] using Pytorch [22].

At the first stage, an initial pre-training is performed at encoding part of the network before moving on to train the Siamese network as a whole. Specifically, it involves training the auto-encoder separately to the rest of the network. As described in the "Auto-Encoder" section above, the auto-encoder is trained to reproduce its input. This was trained for 20,000 epochs with a learning rate of $1e^{-3}$ with a weight decay of $5e^{-6}$—these values have been previously validated as optimal values for this phase of training via a grid search on parameters. The mean squared error (MSE) is used as the loss function, with the validation set being monitored at each epoch.

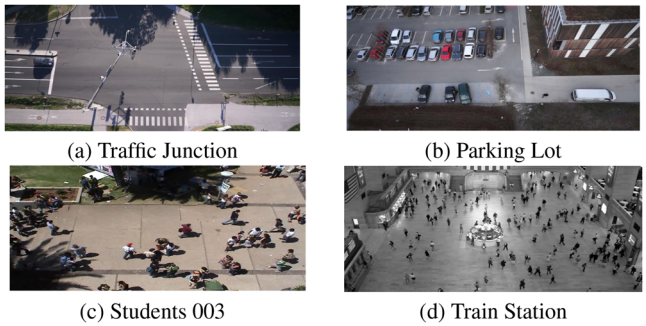
The second stage involves training the network as a twin neural network. As described previously, the final layer of the auto-encoder is removed and a new layer is added for the similarity output. The layers of the auto-encoder are frozen so no further optimisation is performed, and training occurs for 10,000 iterations on the singular output neuron after the differencing of feature vectors. The training of the overall Siamese network is approached with a one-shot approach [23] over multiple epochs, also known as a few-shot due to training over multiple iterations. In every iteration, for each class a random trajectory is chosen to be compared to both another of the same class, as well as random trajectory from another class. Rather than training on all the data in a traditional epoch, this few-shot methodology mitigates overfitting to any one class as each class gets equal training.

A Bayes optimisation search is performed on the learning rate and weight decay for the training of the second stage. Rather than a grid search as above, this was chosen due to the much larger training time of the "Siamese" part of the network. The best network is chosen based on the largest validation set F1-score across all models for each dataset. As mentioned above, both training phases extract the best state of the network based on the validation set. This is another method to attempt to reduce overfitting to the training set.

Experimental validation and analysis: In this section, we present the experimental validation of the proposed method by first describing the datasets and evaluation criteria followed by an analysis and comparison of the results with existing approaches.

Table 2. Summary of the datasets used in the study

Dataset	Frame size	Frame count	Trajectory count	FPS
Traffic Junction	540 × 960	16,154	162	30
Parking Lot	1080 × 1920	9517	37	30
Students003	576 × 720	5405	301	25
Train Station	480 × 720	46,009	379	23



(a) Traffic Junction (b) Parking Lot

(c) Students 003 (d) Train Station

Fig. 2 *Sample frame for each dataset*

Table 3. Evaluation of the output of Siamese network with model trained on each dataset

	Traffic junction			Parking lot			Students003			Train station		
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Traffic junction	**1.00**	**1.00**	**1.00**	0.73	0.77	0.72	0.66	0.61	0.60	0.53	0.54	0.50
Parking lot	0.95	0.88	0.90	**0.96**	**0.94**	**0.94**	0.45	0.47	0.43	0.63	0.54	0.50
Students003	0.92	0.85	0.85	0.87	0.77	0.75	**0.91**	**0.91**	**0.91**	0.61	0.51	0.49
Train station	0.92	0.77	0.81	0.74	0.82	0.77	0.53	0.52	0.50	**0.95**	**0.94**	**0.94**

In order to show the effectiveness of the proposed approach, we used four challenging publicly available real-world datasets (Table 2, Figure 2), namely Traffic Junction [13], Parking Lot [13], Train Station [6], and Students003 [24]. Traffic Junction and Parking Lot are recorded from a mobile aerial platform and contain vehicle and person targets. Real trajectories are used with the induced camera motion already compensated for both datasets, as provided by the original authors [13]. Train Station and Students003 are recorded from a top-down(ish) fixed camera, offering scenes that are highly crowded with people moving in varying directions inside a train station and an outside square. For both datasets we used the available trajectories by respective authors [6, 24]. In Train Station we use longer trajectories (length$>$600) as this dataset also contains short-duration tracklets generated by repeated tracker initialisations that are not within the scope of this work.

As for preparing a trajectory dataset, there are multiple ways to do so [25]. Broadly, these come under four categories: transformation, re-sampling, substitute or adding additional features. In this paper, we used a re-sampling methodology to achieve a static overall size per dataset based on the mean average trajectory length of its dataset. The x and y co-ordinates are also normalised between 0 and 1 based on the minimum and maximum of the dataset's original co-ordinate space.

For evaluating the performance of the extracted motion patterns, we use the precision (P), recall (R) and F-score ($F1$) measures. P provides the assessment by penalising the correct (true positive) patterns with respect to incorrect (false positive) patterns. R provides the assessment by penalising the correct (true positive) patterns with respect to missed (false negative) patterns. If an extracted pattern belongs to a ground-truth cluster, it is deemed correct. We used the ground truth as provided by authors [13]. Figure 3 provides a visualisation of the extracted patterns in terms of the predicted clusters on each dataset by the proposed method.

We tested the generalisation ability of the proposed method by training the Siamese network separately with the model corresponding to each dataset and then evaluating the performance across all other datasets (Table 3). Expectedly, the proposed method has shown the best

Table 4. Evaluation results of methods in terms of P, R and F1 scores; the higher the score, the better. Top two methods are shown in bold

Dataset		DFTfeat [1]	MULTfeat [14]	DWTfeat [13]	DEEPfeat [2]	Movelets [26]	Proposed
Traffic Junction	P	0.67	0.40	0.52	0.80	**1.00**	**1.00**
	R	0.27	0.33	0.50	0.50	**1.00**	**1.00**
	$F1$	0.38	0.36	0.51	0.61	**1.00**	**1.00**
Parking Lot	P	0.48	0.63	**0.65**	0.38	0.64	**0.96**
	R	0.53	0.33	**1.00**	0.83	0.71	**0.94**
	$F1$	0.51	0.43	**0.79**	0.52	0.68	**0.94**
Students003	P	0.90	0.60	0.58	0.64	**0.96**	**0.91**
	R	0.40	0.28	0.51	0.60	**0.96**	**0.91**
	$F1$	0.55	0.38	0.54	0.62	**0.96**	**0.91**
Train Station	P	0.60	0.35	0.45	0.47	**0.93**	**0.95**
	R	0.18	0.18	0.50	0.56	**0.89**	**0.94**
	$F1$	0.28	0.24	0.47	0.51	**0.90**	**0.94**

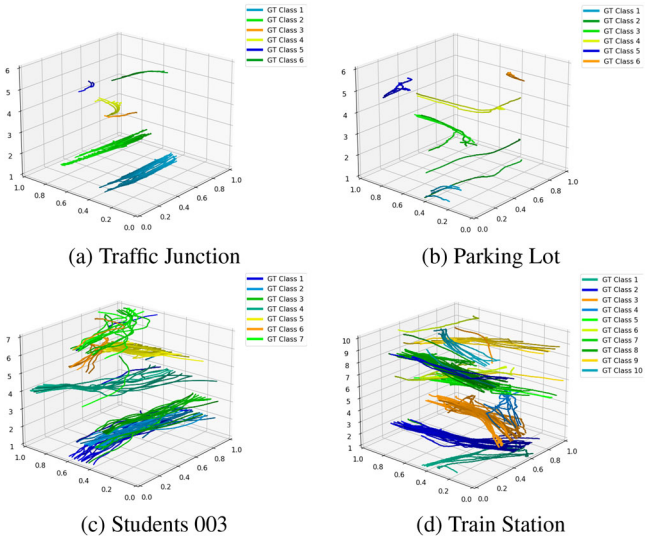


Fig. 3 *Visualisation of the extracted patterns in terms of the predicted clusters. The predicted clusters are shown on planes along z-axis in each plot. The colour of a trajectory corresponds to the relevant ground truth class as indicated in the legend. The darker side of a trajectory indicates the start of the trajectory*

performance when trained and tested on the same dataset. When trained on Traffic Junction and Parking Lot each, the performance deteriorated significantly more both for Train Station and Students003 that is understandable as the latter datasets offer substantially different scenarios and challenges.

It is however interesting to note that when trained on Train Station and Students003, the performance degradation is comparatively lesser ($F1 = 0.94$ on Train Station reduces to $F1 = 0.81$ and $F1 = 0.77$ on Traffic Junction and Parking Lot, and $F1 = 0.91$ on Students003 reduces to $F1 = 0.85$ and $F1 = 0.75$ on Traffic Junction and Parking Lot), thus showing a greater generalisation with these model training. This could be due to a larger number of similar classes in the Students003 and Train Station dataset a—thus the network trains to be better at separating the class clusters. Another point to mention is that when trained on Train Station (Students003), the performance substantially degrades on Students003 (train station); $F1 = 0.94$ ($F1 = 0.91$) on train station (students003) decreases to $F1 = 0.50$ ($F1 = 0.49$) on Student003 (Train station). This is likely due to the dissimilarity in two scene types: Train Station offers an indoor crowded scenario, whereas Students003 is an outdoor crowded scene.

To further demonstrate the usefulness of the proposed method, we also compared the performance with five state-of-the-art approaches, namely DFTfeat [1], MULTfeat [14], DWTfeat [13], DEEPfeat [2] and Movelets [26]. Table 4 summarises the evaluation results of all methods in terms of P, R and $F1$ scores on all datasets. On Traffic Junction, the proposed method and Movelets have outperformed existing approaches based on P, R and $F1$ scores. On Train Station, the proposed method again shows the best performance based on P, R and $F1$, followed by Movelets. On Parking Lot, the proposed method is the best based on P and $F1$, and shows slightly lower $R = 0.94$ than DWTfeat ($R = 1.00$). It is important to note that while R is a bit higher for DWTfeat than that of the proposed method, the former has a significantly lower $P = 0.65$ (due to a much higher number of false positives) that the latter with $P = 0.96$. On Students003, Movelets showed the best performance ($P = R = F_1 = 0.96$), with the proposed method actually also showing a comparable performance ($P = R = F_1 = 0.91$). This slight inferior performance is apparently attributed to the fact that the proposed method has mostly categorised trajectories belonging to the three clusters (clusters 1–3 in Figure 3c) as a single cluster. Indeed, evidently the motion patterns in these three clusters are largely similar in terms of their length and shape (all spanned horizontally across the image). Seemingly, the proposed method relying on the Siamese network is comparatively much better suited for detecting longer patterns than distinguishing among alike patterns differing slightly only in vertical placement.

Conclusion: We presented an end-to-end deep learning framework based on trained Siamese networks, which enabled trajectory analysis-based extraction of dominant motion patterns in an online manner allowing an incremental incorporation of new incoming trajectory(ies). We performed an experimental validation and comparison of the proposed method on four challenging publicly available real-world datasets. The results show that the proposed method has outperformed existing methods on three datasets (Traffic Junction, Parking Lot, Train Station), while achieving a comparable performance on the fourth one (Students003). Future work aims to not only look into testing on more datasets with different scenarios, but also extending the few-shot methodology to train across multiple datasets as this is expected to generalise the network even better.

Conflict of interest: The authors declare no conflict of interest.

Data availability statement: All datasets used are publicly available from their respective authors.

Received: *7 January 2022* Accepted: *13 February 2022*
doi: 10.1049/ell2.12460

References

1 Hu, W., Li, X., Tian, G., Maybank, S., Zhang, Z.: An incremental DPMM-based method for trajectory clustering, modeling, and retrieval. *IEEE Trans. PAMI* **5**(35), 1051–1065 (2013)
2 Boyle, J., Nawaz, T., Ferryman, J.: Deep trajectory representation-based clustering for motion pattern extraction in videos. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6. IEEE, Piscataway, NJ (2017)

3 Portelo, A., Aguiar, A., Lemos, J.: Multi-agent detection and labelling of activity patterns. *SIViP* **14**, 1207–1215 (2020)
4 Sekh, A., Dogra, D., Kar, S., Roy, P.: Video trajectory analysis using unsupervised clustering and multi-criteria ranking. *Soft Comput.* **24**, 16643–16654 (2020)
5 Wang, X., Tieu, K., Grimson, W.E.L.: Correspondence-free activity analysis and scene modeling in multiple camera views. *IEEE Trans. PAMI* **1**(32), 56–71 (2010)
6 Zhou, B., Wang, X., Tang, X.: Understanding collective crowd behaviors: learning a mixture model of dynamic pedestrian-agents. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2871–2878. IEEE, Piscataway, NJ (2012)
7 Peng, J., Qiu, F., See, J., Guo, Q., Huang, S., Duan, L.-Y., Lin, W.: Tracklet Siamese network with constrained clustering for multiple object tracking. In: *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4. IEEE, Piscataway, NJ (2018)
8 Patino, L., Ferryman, J.: Multiresolution semantic activity characterisation and abnormality discovery in videos. *Appl. Soft Comput.* **25**, 485–495 (2014)
9 Ali, S., Shah, M.: A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–6. IEEE, Piscataway, NJ (2007)
10 Saleemi, I., Hartung, L., Shah, M.: Scene understanding by statistical modeling of motion patterns. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2069–2076. IEEE, Piscataway, NJ (2010)
11 Jodoin, P.-M., Benezeth, Y., Wang, Y.: Meta-tracking for video scene understanding. In: *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 1–6. IEEE, Piscataway, NJ (2013)
12 Zhou, B., Wang, X., Tang, X.: Random field topic model for semantic region analysis in crowded scenes from tracklets. In: *2011 Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3441–3448. IEEE, Piscataway, NJ (2011)
13 Nawaz, T., Cavallaro, A., Rinner, B.: Trajectory clustering for motion pattern extraction in aerial videos. In: *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 1016–1020. IEEE, Piscataway, NJ (2014)
14 Anjum, N., Cavallaro, A.: Multi-feature object trajectory clustering for video analysis. *IEEE Trans. CSVT* **18**(11), 1555–1564 (2008)
15 Saini, R., Kumar, P., Roy, P., Pal, U.: Trajectory classification using feature selection by genetic algorithm. In: *Proceedings of 3rd International Conference on Computer Vision and Image Processing*, pp. 377–388. Springer, Cham (2020)
16 Deng, Z., Vahdat, A., Hu, H., Mori, G.: Structure inference machines: recurrent neural networks for analyzing relations in group activity recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4772–4781. IEEE, Piscataway, NJ (2016)
17 Shu, T., Todorovic, S., Zhu, S.-C.: CERN: confidence-energy recurrent network for group activity recognition. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4255–4263. IEEE, Piscataway, NJ (2017)
18 Bengio, Y.: Learning deep architectures for AI. *Found. Trends Mach. Learn.* **1**(2), 1–127 (2009)
19 McCulloch, S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **4**(5), 115–133 (1943)
20 Bromley, J., Bentz, J., Bottou, L., Guyon, I., Lecun, Y., Moore, C., Sackinger, E., Shah, R.: Signature verification using a "Siamese" time delay neural network. *Int. J. Pattern Recognit Artif Intell.* **7**, 25 (1993)
21 Zhang, Z., Ma, L., Li, Z., Wu, C.: Normalized direction-preserving ADAM. arXiv:1709.04546 (2018)
22 Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. arXiv:1912.01703 (2019)
23 Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1–8. Microtome Publishing, Brookline, MA (2015)
24 Šochman, J., Hogg, D.C.: Who knows who - inverting the social force model for finding groups. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 830–837. IEEE, Piscataway, NJ (2011)
25 Bian, J., Tian, D., Tang, Y., Tao, D.: Trajectory data classification: a review. *ACM Trans. Intell. Syst. Technol.* **4**(10), 1–34 (2019)
26 Ferrero, C., Petry, L., Alvares, L., de Silva, C., Zalewski, W., Bogorny, V.: MASTERMOVELETS: discovering heterogeneous movelets for multiple aspect trajectory classification. *Data Min. Knowl. Discovery* **34**(11), 652–680 (2020)