A Convolutional Neural Network Approach for Modeling Semantic Trajectories and Predicting Future Locations

Antonios Karatzoglou[1,2](✉), Nikolai Schnell[1], and Michael Beigl[1]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
{antonios.karatzoglou,michael.beigl}@kit.edu,
nikolai.schnell@student.kit.edu

[2] Robert Bosch, Corporate Sector Research and Advance Engineering, Stuttgart, Germany
antonios.karatzoglou@de.bosch.com

Abstract. In recent years, Location Based Service (LBS) providers rely increasingly on predictive models in order to offer their users timely and tailored solutions. Current location prediction algorithms go beyond using plain location data and show that additional context information can lead to a higher performance. Moreover, it has been shown that using semantics and projecting GPS trajectories on so called semantic trajectories can further improve the model. At the same time, Artificial Neural Networks (ANNs) have been proven to be very reliable when it comes to modeling and predicting time series. Recurrent network architectures show a particularly good performance. However, very little research has been done on the use of Convolutional Neural Networks (CNNs) in connection with modeling human movement patterns. In this work, we introduce a CNN-based approach for representing semantic trajectories and predicting future locations. Furthermore, we included an additional embedding layer to raise the efficiency. In order to evaluate our approach, we use the MIT Reality Mining dataset and use a Feed-Forward (FFNN) -, a Recurrent (RNN) - and a LSTM network to compare it with on two different semantic trajectory levels. We show that CNNs are more than capable of handling semantic trajectories, while providing high prediction accuracies at the same time.

Keywords: Convolutional Neural Networks · Semantic trajectories
Location prediction · Embedding layer

1 Introduction

With the rise in the use of smartphones, wearables and other IoT devices over the past decade, applications that use location data have become increasingly popular. In addition, in recent years, providers attempt progressively to predict

V. Kůrková et al. (Eds.): ICANN 2018, LNCS 11139, pp. 61–72, 2018.
https://doi.org/10.1007/978-3-030-01418-6_7

the locations to be visited next by the users, in order to be able to offer them timely and personalised services. This makes the location prediction research particularly important. Patterns mined from location data can provide a deep insight into the behaviour of mobile users. The usage of semantic knowledge helps diving even deeper into their behaviour. So called *semantic trajectories* encapsulate additional knowledge that can be crucial for the predictive model.

The purpose of our paper is to present and evaluate a Convolutional Neural Network (CNN) architecture in a semantic location prediction scenario. First, we describe some related work that has been done in the realms of semantic location prediction, semantic location mining and CNNs. Next, we elaborate on the way CNNs work, by providing some relevant term definitions at the same time. In Sect. 4 we outline our own architecture together with some basic implementation details. Finally, in Sects. 5 and 6, we discuss our evaluation outcome and draw our final conclusions with regard to our findings.

2 Related Work

Spaccapietra et al. depict as one of the first in their work [12] the importance of viewing trajectories of moving objects in a conceptual manner. They show that, by defining and adding semantic information, such as the notion of application-specific *stops* and *moves*, to the raw trajectories, they can significantly enhance the analysis of movement patterns, and provide further insights into object behaviour. Elragal et al. depict in [5] the benefits of integrating semantics into trajectories as well. It is shown that semantic trajectories help improve both pattern extraction and decision-making processes in contrast to raw trajectories. For this reason, several papers have emerged in recent years presenting approaches to transforming raw location data into so called *semantic locations* (Sect. 3.1). Alvares et al. for instance introduce a semantic enrichment model aiming at simplifying the query and analysis of moving objects [1]. Bogorny et al. [2] extend the previous approach by introducing a more general and sophisticated model, capable of handling more complex queries, while providing different semantic granularities at the same time.

The notion of semantic trajectories has also grown in importance in the field of location prediction during the last years. Ying et al. [13] for example present a location prediction framework based on previously mined semantic trajectories from the users' raw geo-tracking data. Their prefix tree decision based algorithm shows good performance, especially in terms of recall, f-score and efficiency.

In their recent work, Karatzoglou et al. [7], explore the modeling and prediction performance of various artificial neural network (ANN) architectures, e.g., Feed-Forward (FFNN), Recurrent (RNN) and Long-Short-Term-Memory (LSTM) network on semantic trajectories. Similar to Ying et al. they evaluate their models using the MIT Reality Mining dataset [4], with the LSTM achieving the best results with up to 76% in terms of accuracy and outscoring the other methods on f-score and recall as well. In addition, they investigate the role of the semantic granularity of the considered trajectories in the overall performance of the networks. They show that the higher the semantic level, the better the modeling quality of the networks.

Lv et al. explore in [10] the possibility of using Convolutional Neural Networks (CNNs) (Sect. 3.2) to predict taxi trajectories. Their approach projects past trajectories upon a map and models them in turn as 2D images, on which the CNN is finally applied to estimate about future trajectories. By modeling trajectories as 2D images, they are able to make use of the inherent advantage of CNNs, namely their good performance in image analysis. This is also confirmed by their results. However, their approach is applied on raw, non-semantic GPS trajectories.

To our knowledge, there is no work exploring the performance of CNNs on semantic trajectories. Moreover, it seems that there is no work using trajectories (semantic or non-semantic ones) in combination with CNNs directly, e.g., without transforming them in an intermediate step into 2D images, but handling them in their raw form instead, as 1D vectors. In the presented work, we examine exactly these two points in terms of prediction performance in a semantic location prediction scenario. For this purpose, we focused on the Natural Language Processing (NLP) use case where, similar to our case, the data are also 1D and some work has already been done in combination with CNNs. Particularly interesting is the work of Collobert et al. [3], who propose a CNN architecture for solving several NLP problems including named entity recognition and semantic role labelling. Their framework features an unsupervised training algorithm for learning internal representations, e.g., by using an embedding layer and learning low-dimensional feature vectors of given words through backpropagation, yielding a good performance both in terms of accuracy and speed. The benefit of using embeddings has been recently shown also in connection with modeling human trajectories by Gao et al. in [6].

3 Theoretical Background

In this section, we give a brief insight into the fundamental components of our work.

3.1 Semantic Trajectories

Movement patterns, so called *trajectories*, describe sequences of consecutive location points visited by some object or person. In ubiquitous and mobile computing, trajectories refer usually to GPS sequences like the one displayed in Eq. 1, whereby $long_i$, lat_i and t_I refers to longitude, latitude and point of time respectively.

$$(long_1, lat_1, t_1), (long_2, lat_2, t_2), \ldots, (long_i, lat_i, t_i) \tag{1}$$

In the attempt to add more meaning when modeling movement, researchers like Spaccapietra et al. [12] and Alvares et al. [1] went beyond such numerical sequences and lay focus on conceptual, semantically enriched trajectories, so called *semantic trajectories*. A semantic trajectory is defined as a sequence of semantically significant locations (*semantic locations*, e.g., “home”, “burger joint”, etc.) as follows:

$$(SemLoc_1, t_1), (SemLoc_2, t_2), \ldots, (SemLoc_i, t_i) \tag{2}$$

A significant location usually refers to a location at which a user stays more than a certain amount of time, e.g. 20 min. Some researchers add further thresholds, like popularity, in order to extract the most significant common or public locations (see [13]). Locations can be described hierarchically over a number of various semantic levels, e.g., “restaurant” → “fast food restaurant” → “burger joint”. In this work, we evaluate the modeling performance of CNNs on two different semantic levels.

3.2 Convolutional Neural Networks (CNNs)

The most popular application area of Convolutional Neural Networks (CNNs) is the image classification and recognition [9]. However, CNNs can be applied to other areas as well, such as speech recognition and time series [8]. A CNN example architecture concerning the image classification use case can be seen in Fig. 1.

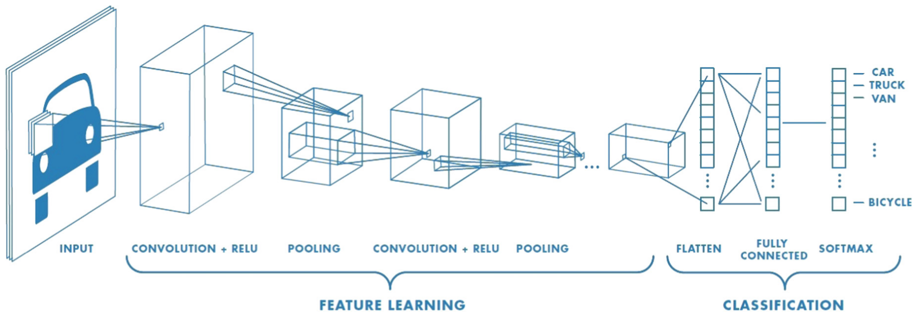

Fig. 1. Typical CNN architecture for Image Classification (source: [11]).

Here, the CNN first receives an image, which is supposed to classify, as its input. Next, a set of convolution operations takes place in order to for the features to be extracted. These operations are realised by filter kernels of fixed size, containing learnable weights, which are sled over the input image to “search” for certain features. Each convolution filter output results in a new layer that contains the findings of that filter in the input image. These layers are then further processed by a pooling operation set. Pooling operations combine multiple outputs from filter kernels in a feature layer into a single value (e.g. by taking the average or maximum value of the outputs in question). The resulting pooled layers can then be further processed, as shown here, by more Convolution + Pooling operations and as such features of a higher level can be extracted. The last pooled layer is flattened i.e. transformed into a single long vector containing all of its weights. These are then connected to a fully connected layer, which is further connected to the output of the network, which in this case is a Softmax layer, containing a field for every classifiable object, and as such representing the classification estimation of the network for the given input.

4 CNNs for Semantic Trajectories - Our Approach

As already mentioned, our network (CNN) takes semantic trajectories as input, like the ones defined in Sect. 3.1. For this purpose, each semantic location is given a unique index. After being fed into the CNN, each index value in the trajectory gets passed to a hash table (*embedding layer*) which assigns each index, and as such each semantic location, a k-dimensional feature vector (*embedding*), whereby k represents a hyperparameter set by us (Sect. 5). At the very beginning, our feature vectors in the lookup table are randomly initialized. These vectors are then trained on the available training data via backpropagation in order to become optimal task-specific representations. In tangible terms, for our case, this means that we give our model the freedom to find the optimal semantic location representation by itself. The resulting representations will be used as input for our core model. A similar idea was proposed by Collobert et al. in [3] to learn feature vectors that represent words in a text corpus for solving NLP problems. After the hash table operation, our semantic location set, initially represented by a $n \times 1$ vector, becomes $n \times k$ matrix. This can be seen on the left in Fig. 2 and as *self.embedded_locs_expanded* in Listing 1.1.

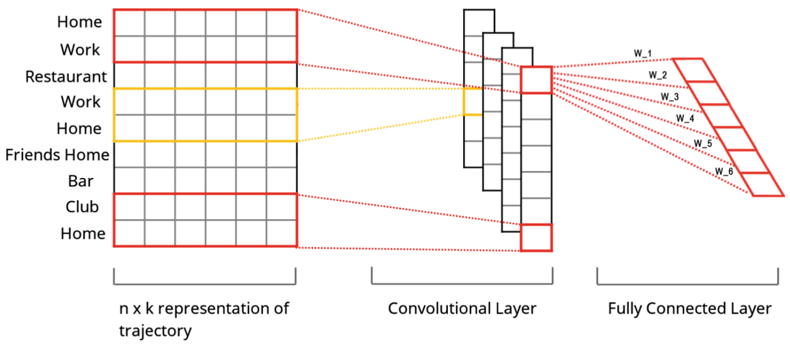

Fig. 2. An abstracted view on the core layers of our CNN.

In the next step, a set of convolutional filters is applied on the resulting matrix. These filters span along the entire feature vector dimension and across multiple locations of the trajectory as can be seen in Fig. 2. The number of filters is a hyperparameter that can be also set by the user. Like the size k of the embeddings dimension described above, it can affect the performance of the prediction.

The outputs of the filters are then concatenated and flattened (*self.h_pool_flat* in Listing 1.1) to make up a fully connected layer, linked to a Softmax output layer, which provides the final prediction about the next semantic location to be visited by a user. We decided against using pooling layers on the filter ouputs, since this led to the loss of significant feature information (e.g., locations in the latter part of the trajectory being more important to location prediction as the older ones).

In order to train our model, we used backpropagation with the Adam optimizer. The Adam optimizer maintains an individual learning rate for each network weight and adapts them separately. This is especially effective since our data is quite sparse compared to other more typical problems addressed by CNNs such as image recognition. We used Python and the Tensorflow[1] library to implement our model. To prevent overfitting, dropout is used on this flattened vector as shown in Listing 1.1 in line 14.

Listing 1.1. Convolution output and flattened layer.

```
# Convolution Layer
self.conv1 = tf.layers.conv2d(
    inputs=self.embedded_locs_expanded,
    filters=num_filters,
    kernel_size=[filter_size, embedding_size],
    padding="VALID",
    name="conv1")

# Combine all the features
filter_outputs_total = num_filters * ((sequence_length -
    filter_size) + 1)
self.h_pool_flat = tf.reshape(self.conv1, [-1,
    filter_outputs_total])

# Add dropout
self.h_drop = tf.nn.dropout(self.h_pool_flat, self.
    dropout_keep_prob)
```

Listing 1.2 illustrates the implementation of the fully connected layer. W and b represent the weights and the offset respectively. Furthermore we used Tensorflow's *nn.softmax_cross_entropy_with_logits* and *reduce_mean* functions to calculate the loss. The calculated loss is used by the Adam optimizer to adjust the weights of the Tensorflow graph, and as such to complete a single training step.

[1] https://www.tensorflow.org.

Listing 1.2. Fully connected layer and loss calculation.

```
# Final (unnormalized) scores and predictions
W = tf.get_variable(
    "W",
    shape=[filter_outputs_total, num_classes],
    initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.constant(0.1, shape=[num_classes]), name="b
    ")
self.scores = tf.nn.xw_plus_b(self.h_drop, W, b, name="scores"
    )
self.predictions = tf.argmax(self.scores, 1, name="predictions
    ")

# Calculate mean cross-entropy loss
losses = tf.nn.softmax_cross_entropy_with_logits(logits=self.
    scores, labels=self.input_y)
self.loss = tf.reduce_mean(losses)
```

5 Evaluation

In order to evaluate our approach, we used the MIT Reality Mining dataset [4], which contains the semantically enriched tracking data of approximately 100 users over a period of 9 months. Filtering the inconsistencies out and keeping the most consistent annotators left us with the two-semantic-level evaluation dataset of 26 users of [7]. Figure 3 illustrates the overall location distribution. We then extracted trajectories of a fixed length and considered the subsequent location to be the ground truth prediction label (see Fig. 4). We shuffled the resulting (trajectories, label) pairs and took 90% of them for training and 10% for testing. We trained and evaluated both the separated single-user models, as well as a multi-user model that contained the trajectories of all users. In the

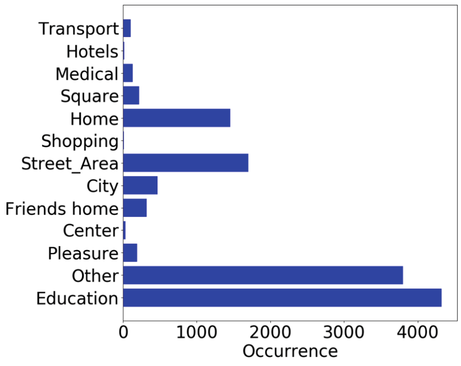

Fig. 3. Distribution of high-level semantic locations.

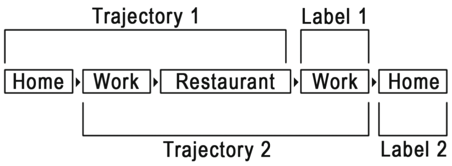

Fig. 4. Data Extraction exemplified with a trajectory length of 3.

case of the multi-user model, a single trajectory composed of all the available single-user trajectories was fed into the model as if it came from a single user. We further used the FFNN, the RNN and the LSTM from [7] as our baseline. In addition, since there is a timestamp present for every location visit in the Reality Mining data, we also tested the performance of our model when we include time as an extra feature. For this purpose we aggregated the available timestamps into hourly time slots. Finally, we evaluate a version of our model with the embedding layer missing. All models were evaluated in terms of *Accuracy*, *Accuracy@k*, *Precision*, *Recall*, and *F-Score*.

We tested several trajectory lengths (2, 5, 10 and 20) on different configurations of the following hyperparameters:

Filter Size: Width of the filter kernel, i.e. how many trajectories it encompasses.
Number of Filters: The number of different filters the CNN learns.
Embedding Dimension: The dimension of the learned location features.
Dropout Probability: The percentage of neurons in the fully connected layer that are dropped (used to minimize overfitting).

At the same time, we did a grid search to find the following optimal parameters as well: **Learning Rate**, **Number of training Epochs** and **Batch Size**. Both the results and the corresponding optimal parameter set can be found in Fig. 5.

In general, it seems that the longer the trajectory the better our model performs with regard to almost all of our metrics, e.g., accuracy, precision, recall and F-Score. However, if they get too long, e.g., >10, the performance drops. Especially in terms of recall and F-Score. This could be attributed to the fact that human movement is characterized, up to a certain length, by a long-term behaviour and thus raising the considered trajectory length in the model leads to an improved predictive performance.

In Fig. 6 we can see the results of our model, with and without an embedding layer. Both CNNs, with and without embedding layer, outperform the FFNN of [7] (used here as reference) with regard to all of our metrics. Additionally, the Embedding Layer seems to be giving a slight performance boost. Figure 7 contains the average outcome (over all users) of our model in the single-user model case in contrast to the FFNN, RNN and LSTM architecture. Our CNN outperforms the other ANNs in terms of accuracy by 7–8%, but falls a bit short in terms of precision, recall and F-Score. This could be interpreted as an indication that the CNN is worse at predicting location transitions that show up sparsely

Trajectory Length	Accuracy	Accuracy@4	Accuracy@10	Precision	Recall	F-Score
2	0.783	**0.976**	0.994	0.455	0.433	0.443
5	0.790	0.973	0.995	0.466	**0.439**	**0.451**
10	**0.792**	0.971	0.994	**0.467**	0.435	0.45
20	0.788	0.968	0.993	0.454	0.425	0.438

Fig. 5. Impact of trajectory length. Filter Size: **2**, Embedding Dimension: **100**, Number of Filters: **50**, Dropout Probability: **0.4**, Batch Size: **100**, Learning Rate: **0.001**, Number of Epochs: **10**.

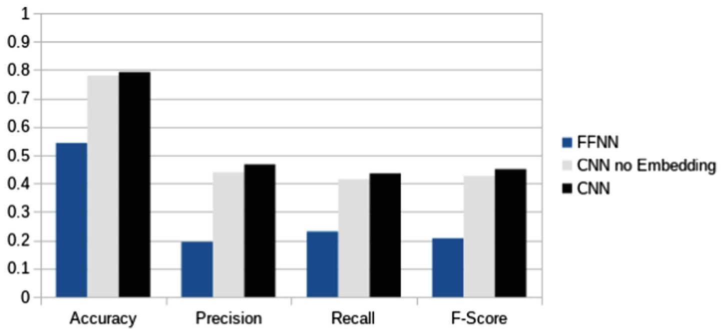

Fig. 6. Comparison of evaluation results of our architecture with and without embedding layer vs. FFNN.

in a dataset (in our case the respective single-user datasets) compared to the other ANNs. On the higher semantic level the accuracy discrepancy between the various models is similar to the low semantic version. However, in terms of precision, recall and F-Score the CNN seems to perform much worse than on the lower semantic version. It seems to disregard locations that occur relatively seldom in the dataset almost completely, which leads us to this result. In both

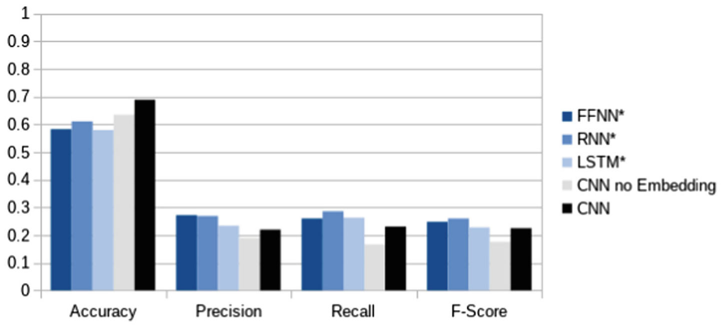

Fig. 7. Comparison of evaluation results of our architecture (CNN) vs. Karatzoglou et al. [7] (*) on the low semantic level (single user).

versions of the dataset (low- and high semantic level) the embedding layer seemed to make a small, but still significant difference.

Figure 9 contains the comparison results between the single-user and the multi-user modeling method. While the multi-user evaluation achieves much lower accuracies (as expected), it outperforms by far the single-user dataset in terms of precision, recall and F-Score. This can be attributed to the fact that the additional user information in the multi-user model fills the gap of missing locations and trajectories that can be often found in the single-user models (Fig. 8).

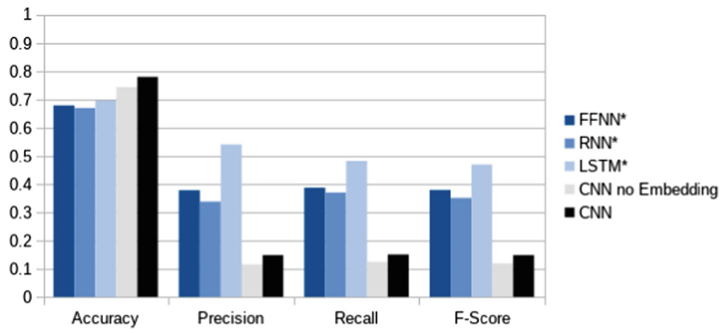

Fig. 8. Comparison of evaluation results of our Architecture (CNN) vs. Karatzoglou et al. [7] (*) on the high semantic level (single user).

Dataset 2 type	Accuracy	Accuracy@2	Accuracy@5	Precision	Recall	F-Score
Multi User	0.688	0.885	0.969	**0.53**	**0.428**	**0.474**
Single User	**0.78**	**0.919**	**0.993**	0.149	0.151	0.149

Fig. 9. Comparison of our multi- and single-user CNN models.

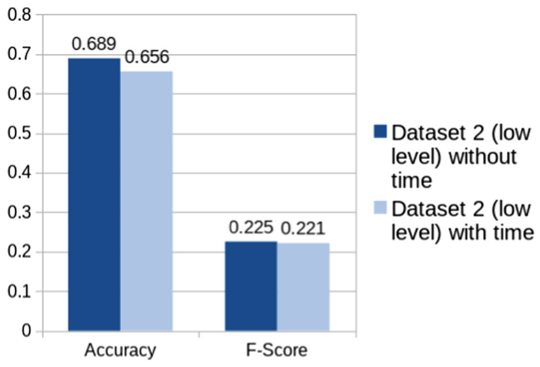

Fig. 10. Impact of time in the case of the low-level semantic representation.

Finally, in Fig. 10 it can be seen how adding time as an additional training feature affects the behaviour of our models. Similar to the results of [7], time seems to be having a negative influence on the prediction performance of our CNN model, both in terms of accuracy and F-Score.

6 Conclusion

In this paper, we investigate the performance of CNNs and embeddings in terms of modeling semantic trajectories and predicting future locations in a location prediction scenario. We evaluate our approach on a real-world dataset, using a FFNN, a RNN and a LSTM network as a baseline. We show that our CNN-based model outperforms all the above reference systems in terms of accuracy and is thus capable of modeling semantic trajectories and predicting future human movement patterns. However, our approach seems to be sensitive to sparse data. In addition, we show that, similar to the outcomes of [7], both the semantic representation level and the overall number of users considered for training the model can have a significant impact on the performance, especially with regard to precision and recall. In our future work, we plan to explore further the use of CNNs in the location prediction scenario by feeding additional semantic information into the model such as the users' activity and their current companion.

References

1. Alvares, L.O., Bogorny, V., Kuijpers, B., de Macedo, J.A.F., Moelans, B., Vaisman, A.: A model for enriching trajectories with semantic geographical information. In: Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, p. 22. ACM (2007)
2. Bogorny, V., Renso, C., Aquino, A.R., Lucca Siqueira, F., Alvares, L.O.: Constant-a conceptual data model for semantic trajectories of moving objects. Trans. GIS **18**(1), 66–88 (2014)
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
4. Eagle, N., Pentland, A.S.: Reality mining: sensing complex social systems. Pers. Ubiquit. Comput. **10**(4), 255–268 (2006)
5. Elragal, A., El-Gendy, N.: Trajectory data mining: integrating semantics. J. Enterp. Inf. Manag. **26**(5), 516–535 (2013). https://doi.org/10.1108/JEIM-07-2013-0038
6. Gao, Q., Zhou, F., Zhang, K., Trajcevski, G., Luo, X., Zhang, F.: Identifying human mobility via trajectory embeddings. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1689–1695. AAAI Press (2017)
7. Karatzoglou, A., Sentürk, H., Jablonski, A., Beigl, M.: Applying artificial neural networks on two-layer semantic trajectories for predicting the next semantic location. In: Lintas, A., Rovetta, S., Verschure, P.F.M.J., Villa, A.E.P. (eds.) ICANN 2017. LNCS, vol. 10614, pp. 233–241. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68612-7_27
8. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. In: The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10 (1995)

9. LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 253–256. IEEE (2010)
10. Lv, J., Li, Q., Wang, X.: T-CONV: a convolutional neural network for multi-scale taxi trajectory prediction. arXiv preprint arXiv:1611.07635 (2016)
11. Mathworks: Convolutional neural network (2018). https://www.mathworks.com/discovery/convolutional-neural-network.html. Accessed 19 Feb 2018
12. Spaccapietra, S., Parent, C., Damiani, M.L., de Macêdo, J.A.F., Porto, F., Vangenot, C.: A conceptual view on trajectories. Data Knowl. Eng. **65**(1), 126–146 (2008). https://doi.org/10.1016/j.datak.2007.10.008
13. Ying, J.J.C., Lee, W.C., Weng, T.C., Tseng, V.S.: Semantic trajectory mining for location prediction. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 34–43. ACM (2011)