Appendix

A Methods

In the following, we provide further details on the data preprocessing and model training. First, the number of visits ($\nu(l)$) is power-law distributed, with outliers such as the home and work location that are visited frequently, which can lead to problems when serving as input to the model. We, therefore, log-transform the number of visits. Furthermore, it was mentioned that features were extracted for both the historic and the new location. The location purpose was thereby encoded as a one-hot vector with one entry for each category home/work/leisure/shopping. If the purpose is different or unknown, all four entries are zero. The POI features were derived by applying Latent Dirichlet Allocation (LDA) on POI data from OpenStreetMaps (OSM). Specifically, all POIs within a distance of 50m from a location were collected. We fit an LDA model on all location-POI pairs to encode them as fixed-sized vectors (i.e., POI topics), where the vector dimension is set to 16. Together, this leads to a feature vector of size 23, consisting of the coordinates (2), the start time (1), the purpose (4), and POI features (16). Finally, we add the visit frequency $\nu(l)$ for the locations where it is available (all except for l_θ). We do not include further information such as the mode of transport used to reach a location, since this information is not generally available for tracking datasets and would therefore limit the model's applicability. The following sections provide further details about each model.

A.1 Fully-connected feed-forward neural network

Neural networks have successfully detected patterns in spatio-temporal data for next-location prediction. The most basic version to be used is a fully-connected network. Here, a model with two layers of 128 and 56 neurons is trained with an SGD optimizer and a mean squared error loss. As explained above, the m locations with the highest visit frequency were fed in to the model. The parameter m was tuned to 10, and providing more locations as input did not improve performance. The architecture leads to 43k parameters in total. The learning rate was tuned to 0.0001.

A.2 Multi-head self-attention neural network

The multi-head self-attention (MHSA) network is the basis of all modern natural language processing models and has also become a main approach for time-series prediction tasks including next location prediction [5]. We implement an MHSA network for location prediction following Hong et al. [5], comprising N layers of masked multi-head attention and fully-connected feedforward network for learning the dynamics of individual mobility. We tune the number of layers N to 6, the number of heads within the multi-head attention to 8, and the dimension of the feedforward network to 256. Similar to the fully-connected model, we input a sequence of feature vectors corresponding to the m most visited locations, sorted by frequency.

A.3 Graph Convolutional Neural Network

We build up on previous work that proposed graphs as compact representations of human mobility that explicitly model dependency structures between locations [16, 20]. Here, we test whether GCNs can also leverage the topology of historic mobility to predict the visitation frequency of new locations. For this purpose, we implement a Graph-Resnet following Martin

et al. [13], comprising three graph convolutional layers of 42 neurons each with a kernel size of 4, and skip connections, as shown in Figure 2. The graph is passed through the Graph-Resnet, and the node embeddings are combined with average pooling, yielding a single vector. A major advantage of this processing is that the number of locations (nodes) can vary between users. The graph embedding, which resulted from node-pooling, is then concatenated with the embedding of the new location l_θ^u that was passed through a single layer of 32 neurons. Together, it is passed through two fully-connected layers of 64 and 32 neurons respectively, before yielding a single output corresponding to the normalized visit frequency.

B Ablation of kNN models

Finally, we tested how the kNN approach performs with varying k. Figure 6 shows the comparison. The kNN approach is better when more neighbors are considered, converging at $k = 21$. It was further tested whether it is better to aggregate with the median instead of the mean; i.e., $\hat{\nu}(l_\theta^u) = median(\{\nu(l^u) \mid l^u \in N(l_\theta^u)\})$, or to use a weighted average with the weight disproportional to the distance in feature space, formally $\hat{\nu}(l_\theta^u) = \frac{1}{W} \sum_{l \in N(l_\theta^u)} \frac{1}{d(l_\theta^u, l)} \nu(l)$, where d is the distance in the feature space ($d(l_1, l_2) = \|f(l_1) - f(l_2)\|$). W is a normalization factor that ensures the weights add up to one ($W = \sum_{l \in N(l_\theta^u)} \frac{1}{d(l, l_\theta^u)}$). However, the aggregation strategy only has a minor influence on the results.

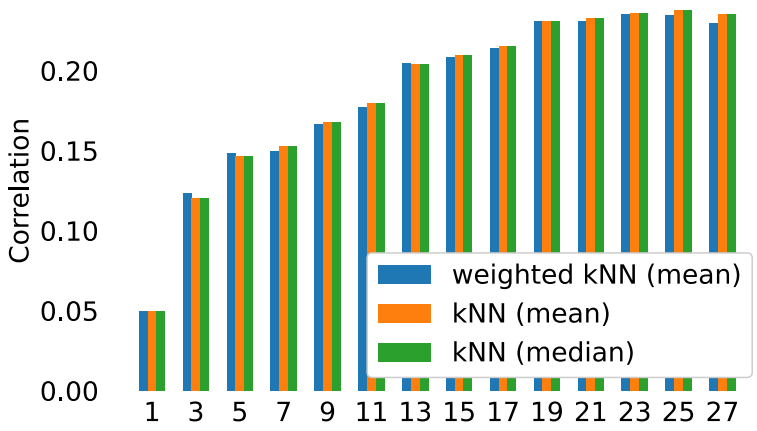

Figure 6 Ablation study of the k-parameter in a kNN approach: The more neighbors are included, the better the performance. Weighting the neighbors or aggregating with the median only changes the accuracy marginally.