Learning Citywide Patterns of Life from Trajectory Monitoring

Mark Tenzer
mtenzer@novateur.ai
Novateur Research Solutions
Ashburn, Virginia, USA

Zeeshan Rasheed
zrasheed@novateur.ai
Novateur Research Solutions
Ashburn, Virginia, USA

Khurram Shafique
kshafique@novateur.ai
Novateur Research Solutions
Ashburn, Virginia, USA

ABSTRACT

The recent proliferation of real-world human mobility datasets has catalyzed geospatial and transportation research in trajectory prediction, demand forecasting, travel time estimation, and anomaly detection. However, these datasets also enable, more broadly, a descriptive analysis of intricate systems of human mobility. We formally define patterns of life analysis as a natural, explainable extension of online unsupervised anomaly detection, where we not only monitor a data stream for anomalies but also explicitly extract normal patterns over time. To learn patterns of life, we adapt Grow When Required (GWR) episodic memory from research in computational biology and neurorobotics to a new domain of geospatial analysis. This biologically-inspired neural network, related to self-organizing maps (SOM), constructs a set of “memories” or prototype traffic patterns incrementally as it iterates over the GPS stream. It then compares each new observation to its prior experiences, inducing an online, unsupervised clustering and anomaly detection on the data. We mine patterns-of-interest from the Porto taxi dataset, including both major public holidays and newly-discovered transportation anomalies, such as festivals and concerts which, to our knowledge, have not been previously acknowledged or reported in prior work. We anticipate that the capability to incrementally learn normal and abnormal road transportation behavior will be useful in many domains, including smart cities, autonomous vehicles, and urban planning and management.

CCS CONCEPTS
- Information systems → Geographic information systems;
- Computing methodologies → Bio-inspired approaches; Anomaly detection; Online learning settings;
- Applied computing → Transportation.

KEYWORDS
Patterns of life, geospatial analysis, biological neural networks, self-organizing feature maps, anomaly detection

ACM Reference Format:
Mark Tenzer, Zeeshan Rasheed, and Khurram Shafique. 2022. Learning Citywide Patterns of Life from Trajectory Monitoring. In The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL ’22), November 1–4, 2022, Seattle, WA, USA. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3557915.3560978

1 INTRODUCTION

What can we learn from observing the mobility of human populations? The study of human movement and its properties, causes, motivations, and constraints has a long and storied history, dating to at least the 19th century (see [7] for a review). In its most modern...
form, the field increasingly uses a proliferation of positioning systems, including GPS, cell towers, and WiFi, to mine individual- and population-level patterns-of-interest [85]. Geospatial professionals such as city planners, transportation managers, and ride-sharing developers have a compelling need to understand patterns of normal behavior and detect deviations from them in real time.

However, incrementally learning patterns from data streams poses unique challenges for machine learning (ML) systems. Over time, many numbers of factors may alter the statistical distribution of the data stream. Some neighborhoods grow, and others decline; new businesses open, and old ones close their doors; a tourism boom may boost traffic, or an unexpected pandemic may shrink it rapidly. These concept drifts are a significant danger to ML systems. Models pre-trained on historical data are unable to adapt to the altered patterns, but conversely, ML systems that naïvely train on new data will instead forget older information which may be useful in the future. Since each gradient update optimizes only for the current mini-batch, and not previous historical data, any learned knowledge is rapidly forgotten in subsequent weight updates. This problem is commonly known as catastrophic forgetting.

A second issue for many ML methods, especially in deep learning, is the difficulty of explaining or interpreting their internal reasoning. As a result, they are often described as “black box” models. For anomaly detection, this is particularly problematic: with neither ground-truth labels nor interpretable reasoning, end users have little basis to trust the model’s decisions. Taken together, these distinct challenges combine into a much greater problem. We know that any “black box” system will encounter concept drifts, but because we cannot inspect the model’s reasoning or compare its predictions with ground truth, we cannot easily assess these drifts and their harmful effects on the model.

Instead, we propose an approach which borrows a modeling framework from computational biology and neurorobotics, two disciplines which have long researched how real-life organisms operate under dynamically changing environments and applied these insights to artificial systems. The proposed approach uses Grow When Required (GWR) networks of bio-inspired episodic memories, which are explicitly designed to acquire unsupervised knowledge over time and protect it from subsequent degradation. We ensure that all encoded knowledge is accessible throughout and after the learning process: a set of normalcy patterns, or patterns of life, enabling the model to detect and explain any deviations from typical aggregate mobility behavior. As an example, the proposed system detects multiple culturally important events in the classic Porto taxi dataset which, to our knowledge, have not been reported or modeled in previous work. We envision that these patterns-of-life models will be very useful in continually mining unsupervised GPS streams, due to their natural robustness to concept drifts, especially in domains where human-machine trust is paramount.

2 RELATED WORK

2.1 Machine learning and human mobility

The growth of large-scale public mobility datasets has encouraged a parallel development of machine-learning methods to generate predictions and mine patterns of interest from data (as reviewed by Toch et al. [85]). In the unsupervised setting most relevant to this work, researchers often use clustering methods, especially k-means [4, 6, 12, 73], and density-based [3, 12, 45, 58, 67] clustering. Hierarchical [49, 92] and spectral clustering [76] may also be used to group related users, places, or trajectories, and generative topic models such as latent Dirichlet allocation may reveal underlying routines or lifestyles [18, 25, 99]. Some recent works have incorporated variational autoencoders [29], graph convolutional networks [36], and spectral graph wavelets from manifold learning [89].

Recently many authors have used taxicab data sets for targeted prediction problems, such as prediction of GPS points within the trajectory (including the final destination and/or arbitrary points) [16, 22, 46, 48, 77, 96, 97], passenger demand [44, 75, 80], and travel time [1, 15, 27, 30, 43]. A smaller community of authors has also used these data-sets to develop unsupervised methods which aim to detect unusual or anomalous trajectories [34, 38, 42, 50, 81, 90]. Lam [42] employs a nearest-neighbor search to determine anomalous driving behavior, Irvine et al. [34] compute a probability distribution over learned sequences from tokenized trajectories; Wu et al. [90] use a reinforcement model of driver preferences and behavior. Other authors train recurrent neural networks in both supervised and unsupervised settings. Song et al. [81] label a set of trajectories as anomalous and train a model to classify them, while Liu et al. [50] propose a deep recurrent variational autoencoder.

2.2 Approaches to lifelong learning

The problem of catastrophic forgetting in neural networks was first reported more than 30 years ago [55, 72] and has been a highly active research domain ever since. The resulting literature of lifelong learning (a.k.a. continual learning, incremental learning) has largely crystallized into three distinct approaches (see [64] for a thorough review). First, replay (a.k.a. rehearsal) simply stores a small set of training inputs (such as, but not always, an online random sample) and periodically re-trains the model on it [13, 35, 39, 74]. Second, more complex regularization approaches such as [40, 47, 95] all add additional loss functions to encourage the model to preserve some previous knowledge while training on new data. Finally, and most relevant to our work, dynamic architectures change structural properties of the network itself to accomodate the incremental accumulation of new knowledge. Neurons can be inserted or removed; their connections, rewired into entirely new synaptic organizations; entire sets of weights, “frozen” or trained depending on the input stream. The scale of changes could be as small as a single neuronal unit or can expand to mutate large sub-networks of the model, e.g., [20, 79, 91, 93, 98].

One of the most well-studied families of dynamic neural architectures is the self-organizing map (SOM), developed by Kohonen in the early 1980s (e.g., [41]). SOMs consist a set of neurons, or units, undergoing competitive learning: each training example is matched to a unit with the most similar features (the best-matching unit, or BMU), and used to train the BMU and other nearby units within a predefined neighborhood function. A later, but similar, alternative is the neural gas, which introduced connections between related neurons [54], i.e., the famous Hebbian learning principle that “cells that fire together, wire together.” Neither of these formulations allows for neurogenesis, or the creation of new neurons. However,
they inspired a vast array of new “growing” networks, many explicitly intended for lifelong learning applications [9, 26, 53, 70]. Most importantly for this work, the grow when required (GWR) network [53] introduced a biologically-inspired habituation mechanism for a network to adjust its size in response to inputs.

These networks have been especially well-studied for autonomous agents and neurorobotics [14, 21, 24, 37, 60, 66, 69], including by the original authors of GWR [52], because these kinds of networks coincide with a key intrinsic motivation of any autonomous system: to progressively learn to ignore sensory inputs which are already well-known, and to identify anything else [59]. This principle is not unique to robotics, also finding applications as diverse as retail sales [19], action and emotion recognition [8, 56, 65], video surveillance [2, 84], medical imaging [5, 17, 61], and genomics [86]. However, few authors have applied any SOM or related method to transportation applications (rare exceptions include [10, 51, 78]), and to our knowledge, we are the first to apply either GWR or a lifelong-learning SOM to this domain.

3 PROBLEM DESCRIPTION

Let \( T = T_1, T_2, \ldots \) be the data stream of all trajectories in the dataset, in order of occurrence, where a trajectory is a variable-length series of latitude-longitude points \( T_i = (\phi_1, \lambda_1), \ldots, (\phi_n, \lambda_n) \). For computational convenience (to calculate distances in meters and kilometers simply), we transform these into UTM coordinates \( T_i = (x_1, y_1), \ldots, (x_n, y_n) \).

We assume that for each trajectory \( T_i \), there are some number of latent factors \( \Psi \) which influence \( T_i \), most notably the geospatial probability distribution of where \( T_i \) originates. Examples might include city-wide factors such as tourist volume, weather, or academic schedules and geospatially-local factors such as new construction or road closures. The identity of all the factors \( \Psi \), or even the number of factors \( |\Psi| \), is itself impossible to observe and intractable to learn. However, we know that some (subset of) \( \Psi \) may vary over time. Therefore, even without observing \( \Psi \) directly, we aim to detect changes in \( \Psi \) by observing changes in the trajectory stream \( T \).

Our problem, then, is to continually monitor a GPS trajectory stream \( T \), here from the Porto taxi dataset, and learn to construct a model \( f \) of \( T \)’s normal behavior without supervision. When an interval of trajectories \( T_i, \ldots, T_j \in T \), taken in the aggregate, deviates from ordinary behavior, \( f(T_i, \ldots, T_j) \) must detect the anomaly and provide enough spatiotemporal explanation of its own reasoning that a human observer can interrogate the purported anomaly (often called a “white box” model). We consider an anomaly detection correct if, and only if, both of the following conditions are met: (1) the model provides sufficient explanation of its reasoning that a human user can quickly search for sociocultural information to confirm or reject the detection, and (2) the detection is in fact accurate: additional pieces of evidence, such as contemporaneous sources, confirm a plausible explanation for unusual activity in the specified location and time.

The model \( f \) can detect changes in the data stream \( T \); the human-model team can detect changes in the cultural factors \( \Psi \). This is a notably more difficult problem than ordinary anomaly detection, which is typically only concerned with detection accuracy relative to a provided ground truth. Instead, our problem formulation mandates explanation and interaction between \( f \) and a human partner: we only count an anomaly detection as correct if it has a human-readable explanation. However, we argue that in the setting of fully unsupervised learning, requiring human-in-the-loop explanation and validation is essential in order to establish trust in the final model.

4 METHODS

4.1 Interpretation of a trained GWR network

One of the advantages of a GWR network, as opposed to a “black box” model such as a deep neural network, is that each component provides a convenient interpretation. The GWR can be viewed as an online clustering of the data stream. Each neuron has a weight vector \( w_i \) which represents a prototype input (such as a GPS location or citywide traffic pattern), and the model seeks to find a limited set of prototypes which are adequate to match all observations. We will show in Sections 5.2 and 5.3 that a properly-constructed set of prototypes \( \{w_i\} \) can be viewed as a model of the trajectory stream \( T \)’s normal behavior, with each \( i \) representing a different sub-type of “behavior.” We refer to these prototypical behaviors, operationalized as GWR weight vectors, as the patterns of life extracted from the data stream. Unlike some techniques such as k-means or Gaussian mixtures, the number of prototypes need not be known in advance or tuned as a hyperparameter: it will be learned during model training.

Each neuron \( i \) also has a habituation, a scalar value \( \eta_i \) which can be viewed as an inverse of the probability mass or support of each prototype. It is possible to recover from \( \eta_i \) an approximate measure of how frequently neuron \( i \) was activated, that is, the proportion of observations with which neuron \( i \) is associated.

The GWR can also be viewed as a one-layer autoencoder, a type of (typically deep) neural network which accepts an input \( x \) and attempts to learn a function to reconstruct it, \( f: x \rightarrow x \) given information constraints. These autoencoders are frequently used for unsupervised anomaly detection, where a sudden increase in reconstruction error implies an input \( x \) which does not match the model’s training set. Note that the error \( d_b \) described below is, essentially, an \( \ell_2 \)-based reconstruction loss (Eq. (2)), and the update formula for \( w_i \) (Eq. (7)) is a gradient descent step to minimize this loss. As in many unsupervised anomaly detections, an increase in the reconstruction loss \( d_b \) (or equivalently, drop in \( a \) implies an anomaly, but instead of a black-box result, the input \( x \) can be compared element-wise with the most relevant prototype, \( w_b \) (see Figs. 5 and 6).

4.2 Detailed training procedure

We describe the GWR algorithm formally in this section. For hyperparameter values and a graphical depiction of the algorithm, please refer to the Appendix.

4.2.1 Initialization. We begin with a set of two neurons represented by their weight vectors, \( A = \{w_1, w_2\} \). We also require a representation of whether two neurons are connected; we use a 2x2 (sparse) adjacency matrix \( E = 0 \). Note that this graph is undirected and \( E \) must remain symmetric. For brevity, we will omit duplicate
assignments to $E$, but any equality or assignment $E_{ij} = k$ also implies $E_{ji} = k$. We will use the notation $N(i)$ to refer to the neighbors of neuron $i$, that is, the set $\{ j \mid E_{ij} > 0 \}$.

4.2.2 Finding matching units. We begin iterating over each input $x$ in the data stream. We find the best matching unit (BMU) $b$, the neuron $w_i$ with the least distance between the input $x$. We also find the second-best matching unit $s$ similarly, by finding the BMU of all neurons except $b$.

\[
b = \arg \min_{i \in A} \|x - w_i\|^2 \quad s = \arg \min_{i \in A \backslash b} \|x - w_i\|^2 \quad (1)
\]

A connection between $b$ and $s$ is made, if it does not already exist, by setting $E_{sb} = 1$. This forms a synapse between the best two neurons for the given input, analogous to Hebbian learning principles from neurobiology [54].

4.2.3 Network activity. We calculate the activity of the network, $a$, as a simple transformation of the network’s reconstruction error $d_b$.

\[
d_b = \|x - w_b\|^2 \quad a = \exp(-d_b) \quad (2)
\]

Note that this provides a performance metric $a \in (0, 1]$ which the network seeks to optimize. A perfect reconstruction results in $a = 1$, whereas $\lim_{d_b \to \infty} a = 0$.

4.2.4 Neurogenesis. More importantly, however, the activity $a$ provides us a criterion to control the creation of new neurons. Note that $a$ represents the performance of the best matching unit: by definition, all the other neurons perform worse for this particular input $x$. Therefore, if $a$ is low, then none of the units matched the input well.

If $a$ is below an activity threshold $a_T$, we add a new neuron $r$.

\[
w_r = (x + w_b)/2 \quad A \leftarrow A \cup w_r \quad (3)
\]

\[
E_{rb} = E_{rs} = 1 \quad E_{sb} = 0 \quad (4)
\]

In other words, the weights of the new neuron $w_r$ are the average of the input $x$ and the best-matching unit $w_b$. In the adjacency matrix $E$, $r$ is inserted between the two neurons with the best reconstruction $b$ and $s$, so that it is connected to each of them, but they no longer connect to each other.

4.2.5 Neuron update. If, and only if, the activity threshold above is not reached, we instead update the weights for the BMU $b$ and any other neuron attached to it $N(b)$. A simple update rule would be,

\[
\forall i \in b \cup N(b), \quad \Delta w_i = \epsilon_i (x - w_i) \quad (5)
\]

where $\epsilon_i$ is a learning rate for neuron $i$. In practice, it is typical to use a relatively large learning rate for the BMU, $\epsilon_b$, and a smaller learning rate $\epsilon_n$ for its neighbors.

4.2.6 Final steps. The iteration concludes by altering $b$’s connections to neurons other than $s$ in the adjacency matrix $E$. Until this point, we have only inserted zeros and ones into $E$. However, the elements of $E$ actually represent synapse “ages,” that is, $E_{ij}$ is the number of examples observed since $i$ and $j$ matched together as the two best neurons. We increment the age of all $b$’s synapses, other than $s$.

\[
\forall i \in N(b) \backslash s, \quad E_{ib} \leftarrow E_{ib} + 1 \quad (5)
\]

Recall that we already reset the age of the $s$–$b$ synapse to $E_{sb} = 1$ previously (Section 4.2.2). If a synapse exceeds a certain age threshold $\mu_{\text{max}}$ it may be safely pruned, and if a neuron $i$ loses all connections $N(i) = \emptyset$ then it may also be removed from the GWR network. This concludes the processing of the input $x$, and we loop back to Section 4.2.2 and repeat for each input $x$ in the data stream.

4.2.7 Habitation. The above GWR implementation is nearly complete, but one issue remains. Note that neurogenesis (Section 4.2.4) and neuron updates (Section 4.2.5) are mutually exclusive. Since the network is initially untrained, the activity $a$ will likely be low and a new neuron will be created—precluding any training of existing neurons. For the next input, the existing neurons will remain poor, creating another new neuron—and preventing training. This loop of repeated neurogenesis leads to a network much larger than it needs to be; a smaller set of well-trained neurons would have been adequate. After all, the GWR network should only grow when required.

The solution is habituation, a computational model of how synapses lose efficacy as they are repeatedly activated [53, 83]. We modify the definition of a neuron $i$ to include both its weight vector $w_i$ and a scalar firing counter, which is initialized to $\eta_i = 1$. Crucially, $\eta_i$ decays each time $w_i$ is trained, as shown below. Therefore, during training $\eta_i$ will give us a measure of whether neuron $i$ is old and well-trained, or newly-formed and in need of tuning.

Habitation provides the network with two desirable properties. First, adding a new neuron requires not only that $a < a_T$ (see Section 4.2.4), but also that the best-matching unit have $\eta_r < \eta_T$ for some firing threshold $\eta_T$. If $\eta_r$ is close to 1, then $w_r$ has not yet been adequately trained, and the network will choose to improve its weights $w_b$ rather than adding a new neuron.

Second, we modify the rules for updating neurons presented in Section 4.2.5:

\[
\forall i \in b \cup N(b), \quad \Delta \eta_i = \tau_i (1 - \eta_i) - \tau_i \quad (6)
\]

\[
\Delta w_i = \epsilon_i \eta_i (x - w_i) \quad (7)
\]

where $\kappa$ and $\tau_b$, $\tau_n$ are additional hyperparameters controlling the rate of habituation. Like the learning rates $\epsilon_b, \epsilon_n$, the rate $r$ is larger for the BMU $b$ than its neighbors $n$.

Note that $\eta_i$ decreases whenever $w_i$ is updated (Eq. (6)) and also controls the size of $w_i$’s update (Eq. (7)). Therefore, the more neuron $i$ is trained, the more resistant it becomes to training. This helps to prevent catastrophic forgetting when the data stream changes, by explicitly preventing the degradation of knowledge stored within well-trained neurons.

4.2.8 Summary. Suppose example $x$ is best-matched with BMU $b$. The network’s performance is the activity $a$. There are three cases to consider:

1. The activity $a$ is close to 1: The GWR network performs well. The weight $w_b$ will be updated, but the magnitude of the term $x - w_b$ is small. As a result, $w_b$ does not change much (Eq. (7)).
2. The activity $a$ is below the threshold $a_T$, but the firing counter $\eta_b$ is not yet below its threshold $\eta_T$: The best-matching unit performs poorly, but its weights are based on relatively little
which would activate neuron with higher neuron density have smaller receptive fields, and would be the best-matching unit (BMU). Note that regions which connect similar neurons. (c): Receptive fields of each neuron produced via a Voronoi decomposition. Each region \( R_i \) is a convex polygon containing all points \( x \) which would activate neuron \( i \), i.e., all points for which \( i \) would be the best-matching unit (BMU). Note that regions with higher neuron density have smaller receptive fields, encoding finer spatial detail.

training data. Therefore, we train the weights \( w_b \). The magnitude of \( x - w_b \) (Eq. (7)) is large, and \( w_b \) will change more significantly.

The activity \( a \) is below the threshold \( a_T \), and the firing counter \( \eta_b \) is below threshold \( f_T \). The best-matching unit performs poorly, but its weights have been trained previously on large amounts of data. Rather than degrade that pre-existing knowledge with a weight update (causing catastrophic forgetting), we preserve it and insert a new neuron \( r \) instead.

5 RESULTS

Taxicabs are a particularly rich source of publicly-accessible research data, including from Porto [57], San Francisco [68], New York City [63], Rome [11], and Beijing [94]. Of these, the Porto dataset has been especially thoroughly researched since its 2015 release for an ECML/PKDD machine learning competition. Here, we present the results of our model using the Porto dataset.

5.1 Partitioning Porto with GWR

Our approach uses a hierarchical GWR with two levels. In the first level, we spatially partition the Porto area into sub-regions learned alongside the second-level anomaly detector. Let \( T = T_1, T_2, \ldots \) be the stream of all trajectories in the dataset, in order of occurrence. Let \( o_t \) be the originating point of the trajectory at time \( t \), that is, the point at which the driver picks up the passenger. We input each \( x := o_t \) into a GWR where each neuron learns a two-dimensional weight vector and iterate over all trajectories in the dataset. The hyperparameters for this GWR can be found in the Appendix.

We display the results of this origin-point GWR in Fig. 2a. On the left, we depict only the prototype origin points \( w_i \). Note that regions with high traffic density (e.g., downtown) have a high density of neurons compared to suburban and rural areas. This illustrates the effects of habituation (Eq. (6)) on neurogenesis; the GWR is only allowed to create new neurons when \( \eta_i < f_T \), which occurs in regions with high data density where neurons are frequently activated. Note also that there seems to be a minimum distance between neurons even in high-density regions. This is a consequence of the activity threshold: new neurons can only be created if \( a < a_T \). Because of the relationship between \( d_b \) and \( a \) (Eq. (2)), there is a one-to-one mapping between an activity threshold \( a < a_T \), such that neurogenesis only occurs for low \( a \), and a distance threshold \( d_T \) such that neurogenesis only occurs when the best neuron’s error \( d_b > d_T \). This allows us to control the distance between neurons; for example, to only create neurons when an input point is at least \( d_T = 1 \) km from all the network’s neurons, we set \( a_T = \exp(-1) \). In Fig. 2b, we further display the synapses \( E \). Recall that since synapses are added between two neurons \( b \) and \( s \) which are the closest to some point \( x \); thus, \( E \) tends to connect neurons which are spatially proximate.

Although our data come from GPS rather than visual data, the structure of GWR results in a close analogy to the early processing stages of biological vision (for a review, see [82]). In biological vision, any given optic nerve fiber will only respond to inputs within a restricted sub-region of the overall visual field, known as that fiber’s receptive field [31]. Evidence suggests that each receptive field maps to an equal amount of space in the brain’s cortex, i.e., an equal amount of processing and subjective “perception” [33, 71]. However, each field represents a differently-sized input region. In the fovea (the visual field’s center), many small receptive fields are packed closely together; in the periphery, larger fields are spread apart. The relationship between the size of a receptive field and its distance from the fovea is approximately linear [33]. Because receptive fields are more densely packed in the fovea, and each field receives equal cortical processing, this structure gives rise to a subjective experience of detailed high-resolution visual experience in the center of the visual field, combined with blurry low-resolution peripheral vision.

In our system, the computation of the receptive fields is a Voronoi decomposition (Fig. 2c). Just as we can determine a best-matching unit \( b \) given some input point \( x \), we can choose a region of points \( \{ x \} \) for which a specific neuron \( i \) would be the BMU,

\[
R_i := \{ x \mid \| x - w_i \|^2 \leq \| x - w_j \|^2 \forall j \neq i \} \tag{8}
\]

which forms a partition of the input space. In areas of the input space densely packed with neurons (e.g., downtown), receptive fields are quite small; conversely, in areas with a low density of neurons, receptive fields are larger—closely mimicking biological vision. The connection between GWR (and related methods) and Voronoi decompositions has long been key to their development [26, 53]. In practice, these Voronoi regions provide a better tokenization of the GPS input space than many popular alternatives: a set of regions which can be constructed incrementally, with guarantees that they (i) partition the full input space (any \( x \) can be assigned to a region), (ii) always are convex polyhedra, and (iii) adapt to the statistical density of the underlying data stream. As in biological vision, we ensure that each receptive field receives equal processing in higher-level analysis regardless of its size, by treating each as a separate feature with equal scale (as described below).
Figure 3: Selected city-wide prototypes, constructed incrementally by a GWR model. The color of each cell indicates the density of taxi trajectories expected to originate in that region. On any given day, the city’s traffic pattern is compared to the full set of prototypes; if none of them are a good match (i.e., $a'$ is low), the model determines that an anomaly occurred.

Figure 4: Activity score $a'$ for the second-level, city-wide GWR, with one value per day. The network takes roughly one month of data to reach a steady, high $a'$ value (e.g., August 2013); after that point, any sudden drop in $a'$ indicates the presence of an unusual event in taxi behavior. We confirm that culturally-well-known events such as Queima das Fitas and Festa de São João do Porto appear clearly as anomalies (red; see Section 5.2) but also uncover new anomalies not previously discussed in research on the Porto dataset (Section 5.3). (a): Queima das Fitas; (b): Serralves em Festa; (c): NOS Primavera Sound; (d): Festa de São João do Porto.

5.2 Modeling normal and anomalous dates
As alluded to above, our anomaly-detection framework incorporates two levels of GWR. In the second, we consider all trajectories $T_1, T_2, \ldots, T_D$ from a given day. Each one has a BMU $b$, a single neuron in the first-level GWR described above, implying an ordered list of neuron indices $b_1, b_2, \ldots, b_D$.

At the end of each day, we calculate a new input vector $x'$ by finding the proportion of trajectories $T_i$ which occurred in each region. This forms the input of a second GWR. The Porto dataset, which is one year long, thus defines a sequence of 365 input vectors $x'$, each of which represents a density of taxis originating in each region that day. We represent the properties of this second GWR as $w'_i, \eta'_i, E', a'_T, \text{etc.}$ to distinguish them from the first GWR. All hyperparameters can be found in the Appendix.

An unusual trait of this approach is that the size of the input $x'$ can change, since it has one entry for each neuron in the first GWR, which is itself growing. GWR implementations with input vectors of varying size are rare (although [69] is an exception). However,
Figure 5: Comparing the true taxi traffic $x'$ with the best-matching unit’s estimate, $w'_b$, for a series of known events (or a single day selected from a multi-day event). Red: more traffic than expected, blue: less. (a): Behavioral shifts on New Year’s Day 2014, part of a sustained period of slightly-depressed $a'$ during the winter holidays. Traffic increases at beaches in the west, decreases in Campanhã (a major local train station) in the east, and shifts within downtown. (b): Tuesday, May 6, 2014, a selection from Queima das Fitas 2014 and the date of the Cortejo Académico (Academic Procession), the festival’s peak. We see increases in traffic near Parque da Cidade do Porto (Porto City Park) in the northwest; many of the celebrations take place adjacent to this park [88]. We also observe increases downtown near the parade’s typical route [62]. (c): The night of June 23–24, 2014 is associated with a drop in activity in Campanhã and a large spike in taxi activity due to the Festa de São João do Porto (St. John’s festival), especially downtown and along the Douro river, where the festivities and fireworks take place, respectively [87].

Figure 6: Comparing the true taxi traffic $x'$ with the best-matching unit’s estimate, $w'_b$. Red: more traffic than expected, blue: less. (a): June 1, 2014. The citywide activity score $a'$ drops sharply to $a' = 0.883$, lower than any other date (other than the first training example provided to the model). A single region near Parque Serralves (Serralves Park) has more than quadruple the taxis expected by the GWR, caused by a massive festival hosted there by a local arts and cultural institution. (b): June 6, 2014. The citywide activity score $a'$ drops suddenly ($a' = 0.935$) due to a much greater-than-expected density of taxi traffic originating in two receptive fields near the Atlantic coast, which comprise the Parque da Cidade do Porto (Porto City Park). This anomaly is caused by the NOS Primavera Sound music festival.

we note that if a new neuron (i.e., Voronoi region) is created in the first GWR at time $t$, then by assumption the previous traffic in that region should be 0 (it did not exist previously). Therefore, a simple implementation is to concatenate a 0 into previous memories for the new region, $\forall i \ w'_i \leftarrow [w'_i, 0]$.

In Fig. 3, we visualize a selection of neurons from this higher-level GWR after training on the Porto dataset. Each neuron represents a prototype of an aggregated daily traffic distribution, and each actual observed day can be matched to a single prototype pattern, the best-matching unit $b'$. A large distance on day $t$ between an input $x'$ and its BMU $w'_b$ implies a low activity score $a'$. Thus, we can calculate this metric once per day and monitor it for any sudden drop in value, hinting at the existence of an unusual event or anomaly.

Based on our cultural research, we anticipated that we would find at least two major anomalous dates: (i) Queima das Fitas (“Burning of the Ribbons”), an eight-day festival from May 4–11, 2014, and (ii) Festa de São João do Porto (St. John’s festival) on June 23, 2014. As can be seen in Fig. 4, we can observe significant and sudden drops in the activity value $a'$ which coincide with these events. The activity score also is slightly depressed during the holiday season, from the weekend before Christmas through New Year’s Day.

The BMU weights $w'_b$ represent a predicted relative taxi volume in each Porto region for the given day, which can be compared directly to the actual observed pattern $x'$ to see which regions had more (or less) traffic than expected. In Fig. 5, we illustrate region-wise deviations from typical behavior in detail.
returns to its high plateau quickly after the cessation of the unusual event (Fig. 4, a–d). This remains true even when the unusual pattern is processed by the network many times in succession. Consider the eight-day Queima das Fitas festival (Fig. 4a): the activity $a'$ drops suddenly at the beginning of the festival, but returns to a high value immediately afterward. If forgetting had occurred, we would observe a gradual return to higher performance as the model reacquired the lost knowledge.

This follows from two key properties of the GWR update rule (Eqs. (6) and (7)). First, updates are only applied to a BMU and its neighbors, $b \cup N(b)$, preventing updates from propagating throughout the full network and restricting them to only the most relevant neurons. (The GWR synapse graph $E$ tends to be relatively sparse, as visualized for the first-level GWR in Fig. 2b). Second, the more often a neuron $i$ is activated, the lower its habituation $\eta_i$, and thus the smaller its weight update $\Delta w_i$. Therefore, even when weight updates do propagate to a neuron, the network selectively preserves those which are already well-trained.

In Fig. 7 we present additional evidence: one of the last neurons $w'_i$ created by the second-level GWR. Repeated anomalies with high activity near Parque da Cidade do Porto (Porto City Park), such as Queima das Fitas (Fig. 5b) and the NOS Primavera Sound music festival (Fig. 6b) have created a newly-formed neuron, describing a traffic pattern with unusually-high taxi density in these two regions. Once this neuron is created, new inputs with high density in this area (e.g., subsequent days of Queima das Fitas) match most closely with this neuron, making it the BMU $b$ and concentrating the weight updates on this neuron (Eqs. (6) and (7)). This protects other neurons from a poor weight update, automatically “freezing” their weights until normal traffic behavior resumes, as observed in Fig. 4.

5.3 Uncovering new events within Porto

However, there are other drops in June 2014 which we did not anticipate, and which do not coincide with any official holiday. In Fig. 6, we show the results for this region-wise comparison on June 1 and June 6. On June 1 (Fig. 6a), we observe a massive surge in activity in a single Voronoi region near Parque Serralves (Serralves Park), a 18-hectare park and part of a local art and cultural institution, Fundação de Serralves (Serralves Foundation). This combined geospatial and temporal localization provides enough contextual clues to research what happened on that date: Serralves em Festa, one of the largest festivals in Europe. The 2014 edition in particular marked the 25th anniversary of the Foundation and the 15th anniversary of its museum, Museu de Serralves, leading to an especially significant celebration (archived at [28] in Portuguese).

In Fig. 6b, we show a similar result for June 6, 2014. This anomaly can be traced to an unexpected surge of activity in two adjacent regions comprising Parque da Cidade do Porto (Porto City Park) just south of the beaches of Matosinhos. In fact, researching this date and Parque da Cidade uncovers contemporaneous reports of a major music festival that year, NOS Primavera Sound (archived at [23] in Portuguese). (Note that the festival takes place June 5–7, but we model taxis originating at regions, and empirically most traffic left the festival after midnight. As a result, the anomalies are detected on June 6–8.)

5.4 Evidence against catastrophic forgetting

Recall that lifelong learning problems such as this one, where knowledge must be acquired incrementally from a data stream, often struggle with catastrophic forgetting: the incorporation of new knowledge degrades information acquired in the past. In that case, the model would need to reacquire its old knowledge by redoing the training process, resulting in a slow increase in $a'$ mimicking the start of training (Fig. 4, July 2013). However, when anomalies are presented to a well-trained GWR, we observe that $a'$ instead
A APPENDIX

GWR ILLUSTRATION ON TOY PROBLEM

To provide a geometric intuition and visual illustration of the GWR algorithm, as outlined in the main text, we include Figs. 8 and 9 here. We illustrate the algorithm on a toy problem, using random samples from a two-dimensional uniform distribution \( U[0,1] \) with increased learning rates to better highlight the model’s behavior in each iteration.

Figure Fig. 8 demonstrates, at a high level, the GWR’s behavior during a weight update (Section 4.2.5). Figure Fig. 9 demonstrates, at a high level, the GWR’s behavior during neurogenesis (Section 4.2.4).

FIRST-LEVEL GWR (REGIONS)

Note that these hyperparameters are largely the same as in previous work, although with different notational convention [65, 66]. We make only narrow changes. First, we alter the activity threshold \( a_T \) to only add neurons when no existing neuron is closer than 1 km. Second, we reduce the habituation rates \( \tau_b, \tau_n \) to slow neurogenesis due to the immense number of trajectories (over 1 million) to be processed. Third, and finally, we disable neuron death (the removal of a neuron \( i \) if \( N(i) = \emptyset \)), because removing a neuron from the first-level GWR removes a learned region, deleting information from the second-level GWR that is essential for city-wide anomaly detection (Fig. 3). At the end of training, this GWR contains 531 neurons total, each representing a region of Porto or the surrounding area.

Hyperparameter values

- \( a_T \) threshold for neurogenesis = \( \exp(-1) \)
- \( f_T \) threshold for neurogenesis = 0.1
- \( \epsilon_b \) learning rate (BMU) = 0.5
- \( \epsilon_n \) learning rate (neighbors) = 0.005
- \( \kappa \) controls habituation = 1.05
- \( \tau_b \) controls habituation (BMU) = 0.0133
- \( \tau_n \) controls habituation (neighbors) = 0.00133
- \( \mu_{\text{max}} \) threshold for synapse pruning = 200

SECOND-LEVEL GWR (CITY-WIDE PATTERNS)

The second-level GWR uses values previously published in the literature [65, 66] for all hyperparameters except \( a'_T \), which we set to 1. As a result, neurogenesis is limited only by habituation \( \eta'_i \), not by the activity \( a'_T \), encouraging the formation of more neurons. At the end of training, this GWR contains 69 neurons total, each representing a prototypical traffic pattern throughout Porto’s regions.

Hyperparameter values

- \( a'_T \) threshold for neurogenesis = 1
- \( f'_T \) threshold for neurogenesis = 0.1
- \( \epsilon'_b \) learning rate (BMU) = 0.5
- \( \epsilon'_n \) learning rate (neighbors) = 0.005
- \( \kappa' \) controls habituation = 1.05
- \( \tau'_b \) controls habituation (BMU) = 0.3
- \( \tau'_n \) controls habituation (neighbors) = 0.1
- \( \mu'_{\text{max}} \) threshold for synapse pruning = 200
Figure 8: Illustrating the GWR algorithm on a toy problem, random points sampled from a uniform distribution $\mathcal{U}[0,1]$. For this visualization, learning rates have been exaggerated for clarity. (a): The state of the GWR network at the beginning of the training iteration. The neuron weight vectors $\{w_i\}$ are shown in black. A new point $x$ (red) has been input to the network. (b): the network finds the best-matching unit $b$ and second-best $s$ (blue). No edge exists yet between these neurons. Other neighbors of $b$, $N(b)$, are shown in lime green. (c): An edge is inserted between the neurons, and the neurons’ weights are updated (with a larger learning rate for $b$ than $N(b)$). For clarity, previous positions are shown in orange.

Figure 9: Illustrating the GWR algorithm on a toy problem, random points sampled from a uniform distribution $\mathcal{U}[0,1]$. For this visualization, learning rates have been exaggerated for clarity. Colors are the same as in Fig. 8. (a): The state of the GWR network at the beginning of the training iteration. (b): The network finds $b$ and $s$. An edge already exists here, which will be removed and replaced with a new neuron. (c): A new neuron is inserted between $x$ and $b$, with connections to both $b$ and $s$. Additional edges may be added during subsequent training iterations.