Applying Weak Supervision to Mobile Sensor Data: Experiences with Transport Mode Detection

Jonathan Fürst,[1] Mauricio Fadel Argerich,[1] Kalyanaraman Shankari,[2] Gürkan Solmaz,[1] Bin Cheng[1]
[1]NEC Labs Europe, [2]UC Berkeley
{jonathan.fuerst, mauricio.fadel, gurkan.solmaz, bin.cheng}@neclab.eu, shankari@eecs.berkeley.edu

Abstract

Machine learning (ML) applications in Internet of Things (IoT) scenarios face the issue that labeled data is expensive and hard to obtain. In addition, the performance of the trained models usually depends on a specific context: (1) location, (2) time and (3) data quality. In this work, we propose a weak-supervision approach for the IoT domain to *auto-generate labels* based on external knowledge (e.g., domain knowledge) through simple labeling functions. Our approach enables quick re-training of ML models for new contexts by removing the labeling bottleneck. We evaluate our approach in a smart transportation scenario, where we classify transport modes using mobile sensor data. Our weakly-supervised model achieves a micro-F1 score of 80.2%, with only seven labeling functions. This is close to the 81% of a fully supervised model, which requires manually labeled data.

Introduction

The Internet of Things (IoT) is expanding rapidly in various sectors such as smart buildings, smart cities, or smart transportation. At the same time, machine learning (ML) supported systems are increasingly used to provide meaningful insights by performing classification or prediction tasks on top of IoT data. Based on our practical experience, the IoT domain poses various challenges for generalizing traditional data driven ML approaches. We summarize these challenges along three dimensions, as depicted in Figure 1:

1. **Location.** Different locations (e.g., separate sensor deployments in a single city or deployments in multiple cities) behave differently. For example, the radio-frequency (RF) signal propagation depends heavily on the deployment location (e.g., through varying signal attenuation) (Fürst et al. 2018).
2. **Time.** Deployment characteristics as well as data patterns change through time. I.e., a ML model that works well when training data is collected, might degrade its accuracy throughout the lifetime of a deployment. Further, it is hard to quantify this degradation, because ground truth is expensive to obtain and deployment times comprise often several years.
3. **Data Quality.** In IoT deployments, data is frequently noisy, sparse and heavily imbalanced. E.g., an application might require the classification of relatively rare events such as road accidents or infrequent transport modes; a sensor network deployment might result in high variances in sampling frequency and missing data due to packet loss. As IoT devices have limited power and computation, collecting highly dense data is often not possible.

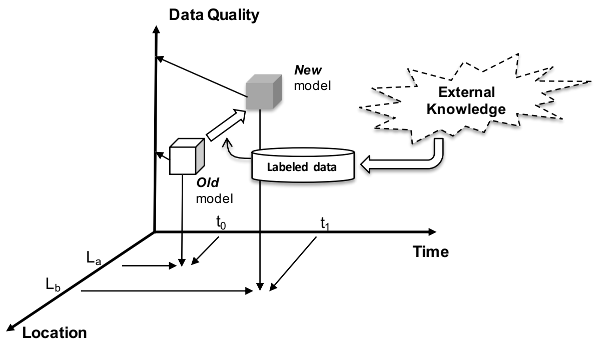

Figure 1: Challenge-Dimensions of Applying ML in IoT

The common factor in above challenges is a lack of training data (i.e., labeled data). Available training data would help to re-train a ML model for a different location, time or data quality context. However, training data collection is a key bottleneck for IoT applications (Mohammadi and Al-Fuqaha 2018). It often requires a human to manually annotate events in a data stream by observing the events in the real world. As we have seen in practice, this effort might need to be repeated for different locations and over the lifetime of a deployment. Fortunately, sensed phenomena in IoT deployments typically follow an underlying model. For instance, car acceleration follows a well understood physical model of engine torque that results in a positive traction force moving the car forwards when accelerating, and negative forces based on the car's inertia and frictional resistance of the road. To be practically useful, models do not need to be as complex and explicitly linked to physical models. Often *external knowledge* in form of much simpler and more abstract models (e.g., based on human domain knowledge or on past experiences) can already provide valuable input to a traditional purely data-driven ML process for IoT applications.

In this work, we include external knowledge for the ML training phase by auto generating labels for unlabeled mobile sensor data using weak supervision. Through automating the labeling process, a ML model can quickly be retrained for a different deployment context (see Figure 1). We demonstrate our approach with sensor data that we collect from user smartphones in order to perform transport mode classification (an essential problem in smart transportation see Section 2). Our approach avoids the need for large, manual training data collection and annotation. Instead, we leverage weak supervision sources that we apply automated to label sparse sensor data. Each of these sources on its own is noisy; the resulting labels are insufficiently accurate and provide limited coverage. However, we show that by combining multiple noisy knowledge sources, we are able to successfully train commonly used ML models for transport mode detection. To this end, we adopt the recently developed *data programming* concept (Ratner et al. 2017) used in Natural Language Processing (NLP) and knowledge base construction, to the transport mode detection problem, and integrate our method into an open-source mobility data collection and analytics framework (K. Shankari et al. 2018).

Compared to existing transport mode detection work, our method uses weak supervision to label fully unlabeled data, removing the need for hand-label training data. Our method works with sparse sensor data ($\approx 30\,\mathrm{s}$ sampling interval) and smartphone OS optimized APIs (iOS/Android) to query accelerometer-based activity data. This enables us to continuously collect data on user smartphones without heavily impacting battery consumption, thus making our method directly applicable in practice. We summarize our contributions as follows:

- *Practical application of weak supervision to IoT domain.* We successfully adopt weak supervision for the automatic, multi-class labeling of sensor data in a practical IoT system. We find obstacles to address, such as the easier accessibility of external knowledge and a general method for sensor data segmentation.
- *Validation on sparse dataset with 300k data points (ca.* $2500\,\mathrm{h}$*).* We validate our method with an in the wild deployment. Overall, our weakly-supervised method achieves a micro-F1 score of 80.2 % over four transport classes. The baseline supervised learning approach, which uses manually labeled data, achieves an only slightly higher result of 81 %.

Target Application & Problem

We now discuss our application scenario, its requirements and describe the transport mode detection problem.

Application Scenario

Currently, we implement a solution together with the local city, where we collect sensor data from end-users in an urban environment to create insights for both citizens and city administration. The main need of our stakeholders is to collect location trajectories and transport modes to: (1) provide recommendations to commuters of potential transport options (e.g., train, bike) and (2) provide an overall, aggregated view of transportation to the local government to inform policy decisions (e.g., new infrastructure, bus line change).

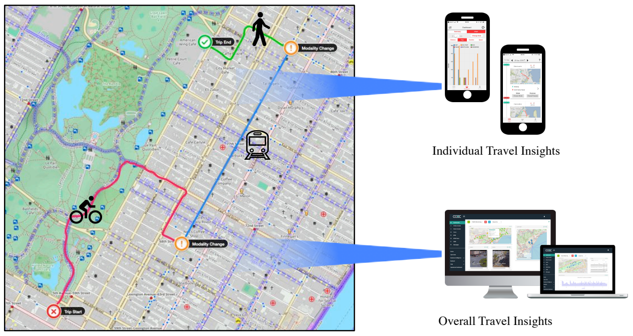

Figure 2: Application Scenario. Based on collected and analyzed data, we provide insights to individual commuters (e.g., their travel times for different transport modes or CO_2 footprint) and overall insights to the local government.

Transport Mode Detection Problem

Transport mode detection is fundamental to optimize urban multimodal human mobility (Gallotti and Barthelemy 2014) and for our application. It requires a two step segmentation of sensor data to *trips* and *sections* (based on transport modes), as well as accurate classification of these sections (e.g., walking, riding a bike, driving a car).

Other have proposed transport mode detection using smartphone sensors such as GPS (Zheng et al. 2008), accelerometer (Hemminki, Nurmi, and Tarkoma 2013), barometer (Sankaran et al. 2014), or combinations of these (Reddy et al. 2010) as well as fusing GIS data (Stenneth et al. 2011) to improve accuracy. In these works, the proposed methods leverage supervised ML using labeled data points as training data. This labeled data is provided by users/participants of the studies. The problems with these supervised approaches link to the challenges of applying ML to IoT systems in practice that we identified in Section 1:

1. **Manual labeling by users.** Data needs to be manually annotated with labels. Only persons for which this data was collected can annotate, i.e., only they know their method of transport for each section of their trips. This requires much user effort, reliance on the users willingness, and trust in their labeling.
2. **Limited generalization.** The trained ML classifier might be biased to the annotated dataset. Transport mode data is usually collected by relatively few people under constrained conditions (e.g., a set of volunteers from university travelling mostly inside a single city or region).
3. **Data Quality: Smartphone battery use and OS limitations.** Data collection is constrained by user requirements

of low battery impact and data use. Published datasets often have small, uniform sampling intervals. However, deployments in practice face challenges such as power-saving smartphone OS limitations for background services to reduce constant GPS data collection (e.g., geofencing, sensor batching and sensor fusion).

To overcome these problems, we improve the availability of labeled data by applying and adapting weak supervision techniques to the transport mode detection problem. Our approach, which lets us generate labeled training data in a quick and effective manner, is outlined in the following sections.

Applying Weak Supervision to Transport Mode Detection

In this section, we describe our approach of applying weak supervision to mobile sensor data, specifically to the transport mode detection problem. Our approach addresses the issues of (1) Mobile sensor data collection, (2) trip and section segmentation and (3) transport mode training/classification. Figure 3 depicts the main building blocks and flow of our approach. Sensor data (location and activity data) is sensed from user smartphones. In the *Segmentation Phase*, we then align time series from multiple sensors to the same sampling time and segment time series first into trips (by applying a dwell time heuristics) and section candidates (using a developed activity supported walk point segmentation algorithm). The outcome of this step is a set of section candidates, each candidate contains a time-series of location and activity data points. In the *Label & Training Phase*, we apply a set of labeling functions to these candidates. Each function encodes human heuristics and/or external knowledge (e.g., from OpenStreetMap) to "vote" on the transport class of a candidate section. We feed the resulting label matrix into Snorkel (Ratner et al. 2017), which learns a generative model from all labeling functions and their votes on each candidate section. Finally, we employ the generative model to label all candidate segments and train a discriminative ML model with the probabilistic labels. In the *Classification Phase*, this model is then used to classify incoming candidate section into transport modes. Based on this classification, we then re-segment by merging adjacent sections of the same mode and return the results (classified trips and sections) to users. We now describe the detailed working of the main phases of our method.

Mobile Sensor Data Collection

Our application requires that we collect location trajectories of trips. A major problem with location data collection is its high power impact (e.g., GPS uses $30\,\mathrm{mA}$ when active (Yu et al. 2014)). Internal sensors that have shown value for transport mode detection (accelerometer, gyroscope) require a high sampling frequency to be useful. For example, accelerometer sampling frequencies need to be usually above $10\,\mathrm{Hz}$ on three axial directions and not only depend on the phone's orientation and position, but also on the specific user and vehicle (Sankaran et al. 2014). This makes transport mode detection complex and power intensive. Instead, in our approach, we use the fused Location APIs of Android and iOS, which reduces battery consumption compared to native GPS sampling due to OS features such as sensor batching, sensor fusion and geo-fencing. Likewise, to gather transport mode relevant data from internal sensors, instead of sampling raw data for each available sensor, we query the Acivity API available on Android and iOS. This approach greatly improves battery consumption and reduces network traffic, but comes at the cost of sparse sensor data with a dynamic, context-dependent sampling interval.

Time Series Segmentation

In this phase we filter and re-sample incoming sensor time series and segment into *(1) Trips* and *(2) Candidate Sections* (see Algorithm 1 for a brief overview of the steps). Our sampled data is sparse, the time series of location updates and activity detection updates are not aligned, and without a fixed sampling interval. Therefore, we filter and re-sample both data-streams in `CleanTrip(trip)`, aligning the activity detection time series $S_A = (\{a_1, t_1\}, \{a_2, t_2\} \ldots \{a_n, t_n\})$ with the location time-series $S_L = (\{l_1, t_1\}, \{l_2, t_2\} \ldots \{l_m, t_m\})$:

$$S_{(A,L)} = (\{a_1, l_1, t_1\}, \{a_1, l_1, t_1\}, \ldots \{a_n, l_n, t_n\}) \quad (1)$$

Algorithm 1 Candidate Section Segmentation

```
1: trip = [$p_0, p_1 \ldots p_n$]    ▷ A trip consisting of a timeseries of N points
2: CLEANTRIP(trip)    ▷ resample data-points to location timestamps and map between Android and iOS activity modes
3: GETWALKPOINTS(trip)
4: CREATESEGMENTDRAFTS(trip)    ▷ Create draft segments based on assigned walk points
5: FILTERSEGMENTS(trip)    ▷ Filter and merge segments
```

After aligning both time series, we segment them into trips, using a dwell-time heuristics, and in into candidate segments, using a developed activity detection supported walk-point segmentation algorithm. We define a trip when a person travels/commutes between two locations A and B, while the person dwells in both places a substantial, configurable time period T. Currently, we have set $T = 600\,\mathrm{s}$. A trip itself might contain multiple sections $\{S_1, S_2, \ldots S_n\}$ with different transport modes (e.g., WALK $\rightarrow$ TRAIN $\rightarrow$ WALK). Detecting these sections is more complex than detecting the start and end of a trip as users can quickly change between two transport modes (e.g., from walk to train). To approach this problem, we follow the same line of thought as in (Zheng et al. 2008): between two modes, there must be a (small) walking segment. Based on this insight, we find candidate segments through a two step process on a point granularity and on a segment granularity. First, we find certain walk points, certain other points (i.e., non-walking), probable walk points and probable other points. The reason behind classifying points into "certain" and "probable" is

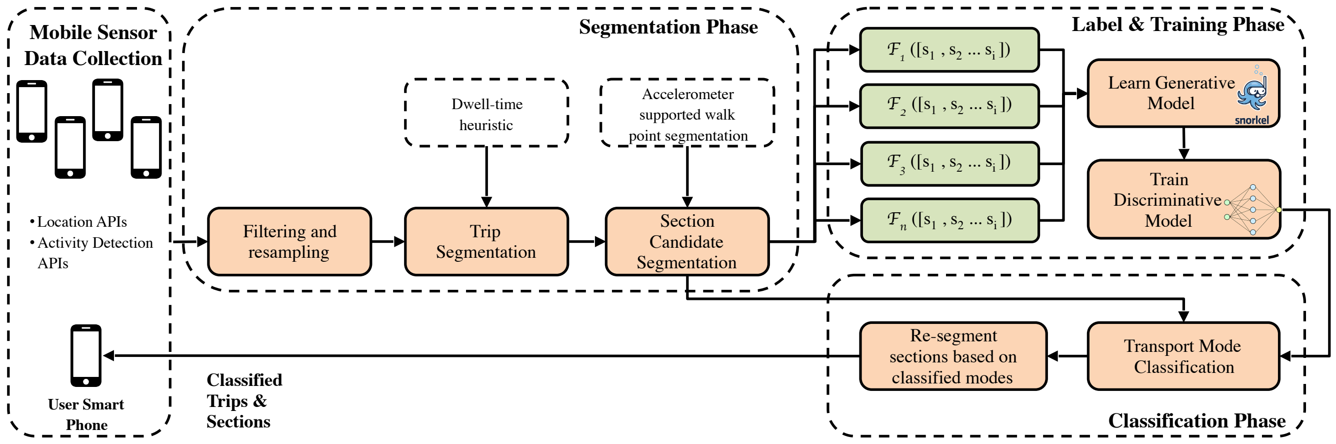
Mobile Sensor Data Collection
• Location APIs
• Activity Detection APIs
User Smart Phone
Segmentation Phase
Dwell-time heuristic
Accelerometer supported walk point segmentation
Filtering and resampling
Trip Segmentation
Section Candidate Segmentation
Label & Training Phase
$\mathcal{F}_1$ ([s$_1$, s$_2$ ... s$_i$])
$\mathcal{F}_2$ ([s$_1$, s$_2$ ... s$_i$])
$\mathcal{F}_3$ ([s$_1$, s$_2$ ... s$_i$])
$\mathcal{F}_n$ ([s$_1$, s$_2$ ... s$_i$])
Learn Generative Model
snorkel
Train Discriminative Model
Transport Mode Classification
Re-segment sections based on classified modes
Classification Phase
Classified Trips & Sections

Figure 3: Overview over Transport Mode Detection Steps. Data collection occurs on user smartphones, while segmentation, labeling, training and classification is executed in the cloud.

to not filter out small segments that are still highly likely to represent a different transport mode. In our experience, this is often the case when the walk distance between two modes of transport is short or when we do not receive location updates for a period of time due to no available signal (e.g., during an underground metro ride). The output of this step allows us to create initial draft segments. In the `FilterSegments(trip)` step we then use the draft segments and the assigned point classes to filter and merge candidate segments (e.g., we merge adjacent non-walking or walking segments).

Labeling, Training & Classification Phase

We use weak supervision sources to compute probabilistic labels for the generated candidate segments and use these labels to train a discriminative ML model that can classify unseen segments. With weak supervision we can generate training labels for—potentially large amounts of—unlabeled data based on imprecise or limited supervision signals. These signals can be modeled in a generative process that de-noises the labels based on their agreements and disagreements for each data point. In *Data Programming* (Ratner et al. 2017), the framework we use in our implementation, supervision sources are abstracted to so called *labeling functions*. Each function $f : X \to Y \cup \emptyset$ takes a data point $x \in X$ as input and outputs the label $y \in Y$ or abstains from casting a vote. Let us consider n labeling functions and m data points. This results in labeling matrix 2. This matrix, together with the three factor types of labeling propensity, accuracy, and pairwise correlation is the input for creating the generative model, resulting in probabilistic labels for each data point (for more details see (Ratner et al. 2017)).

$$M_L = \begin{bmatrix} y_{11} & y_{12} & y_{13} & \cdots & y_{1n} \\ y_{21} & y_{22} & y_{23} & \cdots & y_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & y_{m3} & \cdots & y_{mn} \end{bmatrix} \tag{2}$$

Through this generative process, the noise and variance in accuracy and coverage of each single labeling function is taken into account, putting more weight on high accuracy labeling sources and less on low accuracy ones. We then use these probabilistic labels to train a discriminative model (i.e., any common supervised ML model). This model generalizes beyond the information encoded in the labeling functions and can improve further when we generate more training data from unlabeled data. This is especially important for an IoT scenario, where new data is constantly generated in a streaming fashion. We discuss the specific implementation of our labeling functions and the discriminative model in the following section.

Experimental Evaluation

We validate our method against a dataset that we collect in the wild over a period of $4\,\mathrm{months}$, containing 300k datapoints from 8 end-users. We sense GPS location through iOS and Android Location API and accelerometer based activity data through the Activity API. Users have partially labeled data with a developed visual labeling tool. We use this data to evaluate our method, splitting our data in training (1/2) and test data (1/2). We classify four transport modes: walk, bike, car and train.

Labeling Functions & Generative Model

We implement seven labeling functions that combine external knowledge with sensor data to vote on the transport classification of a section. For example, we implement functions that use the sensed speed together with human heuristics on common speeds for different modalities to vote on the transport mode (`LF_median_sensed_speed` and `LF_quantile_sensed_speed`). We also integrate OpenStreetMap (OSM) and use the provided annotations of public transport stops (`LF_osm`). Table 1 summarizes our labeling functions and depicts their individual performance. Overall we achieve a micro-F1 score of 74 % for the generated labels. Note, all following F1 scores are micro-F1.

Figure 4 depicts the normalized confusion matrix for the generated transport mode labels. Walk, bike and car can be distinguished well, while our labeling functions have problems to differentiate car and train sections. After further investigating the problem, we find two main reasons: (1) car and train have similar speed characteristics and (2) the external knowledge in form of OSM data is not precise enough to differentiate well between both classes (37% accuracy). Last, we test how added knowledge in form of additional labeling functions and knowledge sources influences the overall labeling F1 score. Table 2 shows that we gradually improve the overall performance when we add additional knowledge sources from 64% to 74%. This indicates that further knowledge sources will improve performance.

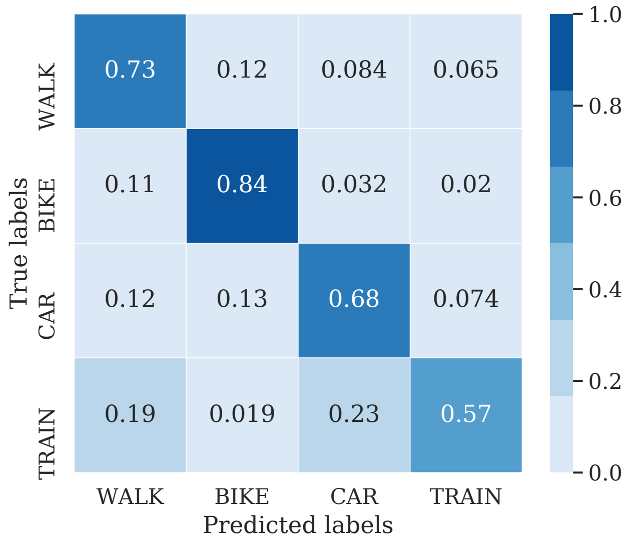

Figure 4: Weakly-Supervised Labels. Labeling confusion matrix for different modes. Overall F1 score: 74 %.

Discriminative End Model

Next, we train ML models using the data labeled by the generative model. Since we are dealing with an imbalanced dataset, we oversample the underrepresented classes using SMOTE (Chawla et al. 2002). This technique generates new synthetic data points from the underrepresented classes by cloning the original data points and altering their features slightly. We use SMOTE's default parameters (all classes but the majority one are sampled to balance the dataset).

We test several classifiers, from simple linear models to neural networks and observe the best performance in F1 score for Random Forests (RF). We have implemented our RF using Scikit learn with 200 trees and a maximum depth of 6 levels. RF mode detection F1 score is 80.2 %. As for our Neural Network (NN) approach, we use Tensorflow (Abadi et al. 2016) with Keras. After several tests with different architectures, we achieve the best accuracy with 3 hidden layers of 24 units each. We feed our balanced dataset and train the neural network for 100 epochs. NN mode detection F1 score is 78.4 %.

We also compare our results against a traditional supervised approach, using the hand-labeled data of the training split, to train the same RF. This results in a only marginally better F1 score of 81.0 %. Table 3 summarizes our overall results. Finally, Figure 5 depicts the confusion matrix for our end model (RF). Comparing these results to labels (Figure 4), we see that the trained model is able to remove noise and generalize beyond the information encoded in the labeling functions. Interestingly, RF performs better than the NN. We suspect that this might change when evaluated on a larger dataset, which we leave for future work.

Lessons Learned

Our results show that extensive hand-labeled data might not be necessary for some classification problems in the IoT domain if we are able to encode external knowledge based on domain experts, external knowledge bases or physical models efficiently. We believe that a key issue to be addressed is to make external knowledge easier accessible to encode it in weak-supervision signals. For instance, OpenStreetMap contains useful information for many smart city classification tasks (e.g., in form of user annotations, geometric information etc.), but it is relatively hard to access this information, also from a performance perspective (e.g, we needed to create a layered caching architecture to enable a fast, iterative development of our labeling functions).

Another issue is the granularity in which IoT time-series data should be labeled. Often, we can write labeling functions on multiple abstraction levels (e.g., a single sensor point or a sequence of sensor points). Recently, (Sala et al. 2019) have made similar observations for video streaming data (frame, sequence abstraction). In our application scenario, we have developed the trip and segment abstraction.

Table 1: Labeling Function Evaluation Analysis

Labeling Function	**Classes**	**Coverage**	**Accuracy**
`LF_max_velocity`	[0, 1, 2, 3]	100 %	72.1 %
`LF_median_velocity`	[0, 1, 2, 3]	99.6 %	68.5 %
`LF_motion_activity`	[0, 1, 2]	82.9 %	80.5 %
`LF_std_velocity`	[0, 1, 2, 3]	100 %	46.8 %
`LF_osm`	[3]	10.4 %	37.0 %
`LF_median_sensed_speed`	[0, 1, 2, 3]	100 %	73.6 %
`LF_quantile_sensed_speed`	[0]	65.7 %	78.3 %

Table 2: Labeling performance gain with additional knowledge. Velocity (V), Sensed Speed (S), Accelerometer (A), OpenStreetMap (OSM).

V	**V + S**	**V + S + A**	**V + S + A+ OSM**
64.3 %	70.35 %	72.4 %	74.1 %

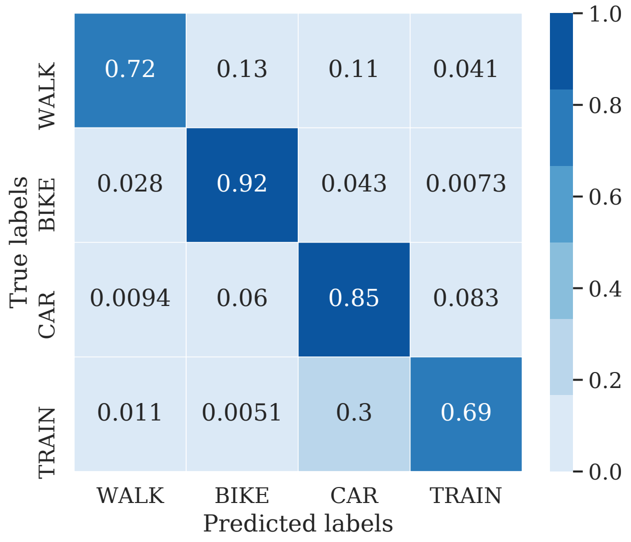

Figure 5: End model. Classification confusion matrix for different modes. Overall F1 score: 80.2 %.

Table 3: F1 score of weakly-supervised labels and end models against hand-labeled ground truth. Auto-generated labels (Generative Model - GM), Random Forest trained with auto-generated labels (Weak-RF), Neural Net trained with auto-generated labels (Weak-NN), Random-forest trained with hand-labeled ground truth (SUPV-RF).

GM	**Weak-RF**	**Weak-NN**	**SUPV-RF**
74.1 %	80.2 %	78.4 %	81.0 %

In future, we might see a general pattern evolve across different applications that can become the functionality of a data programming framework.

Conclusion and Future Work

We have demonstrated that weak supervision can be applied successfully to sensor data in a smart transportation application, close in performance to a traditionally supervised trained model, which requires more human effort for labeling. With additional data, we expect that weakly-supervised models will surpass traditional supervised approaches, as we have seen in other domains (Ratner et al. 2017). As next steps, we are deploying our approach in context of a larger smart transportation project together with the city. This will enable us to collect more data, expanding our classification to other transport modes and validating classification performance through time. We further will evaluate our method with deployment data from another city in order to test how general applicable our labeling functions are and how we can automatically adapt them (e.g., to different speed limits and public transport infrastructure in different countries).

Acknowledgements

This work has been funded in part by the EU Horizon 2020 Programme under grant agreements AUTOPILOT (AUTOmated driving Progressed by Internet Of Things, No. 731993), LOCUS (LOCalization and analytics on-demand embedded in the 5G ecosystem, for Ubiquitous vertical applicationS, No. 871249) and by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation program sponsored by DARPA.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16:321–357.

Fürst, J.; Chen, K.; Kim, H.-S.; and Bonnet, P. 2018. Evaluating bluetooth low energy for iot. In *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, 1–6. IEEE.

Gallotti, R., and Barthelemy, M. 2014. Anatomy and efficiency of urban multimodal mobility. *Scientific reports* 4:6911.

Hemminki, S.; Nurmi, P.; and Tarkoma, S. 2013. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM conference on embedded networked sensor systems*, 13. ACM.

K. Shankari; Mohamed Amine Bouzaghrane; Samuel M. Maurer; Paul Waddell; David E. Culler; and Randy H. Katz. 2018. E-mission: An open-source smartphone platform for collecting human travel data. *Transportation Research Record: Journal of the Transportation Research Board.*

Mohammadi, M., and Al-Fuqaha, A. 2018. Enabling cognitive smart cities using big data and machine learning: Approaches and challenges. *IEEE Communications Magazine* 56(2):94–101.

Ratner, A.; Bach, S. H.; Ehrenberg, H.; Fries, J.; Wu, S.; and Ré, C. 2017. Snorkel: Rapid training data creation with weak supervision. volume 11, 269–282.

Reddy, S.; Mun, M.; Burke, J.; Estrin, D.; Hansen, M.; and Srivastava, M. 2010. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)* 6(2):13.

Sala, F.; Varma, P.; Fries, J.; Fu, D. Y.; Sagawa, S.; Khattar, S.; Ramamoorthy, A.; Xiao, K.; Fatahalian, K.; Priest, J.; et al. 2019. Multi-resolution weak supervision for sequential data. *arXiv preprint arXiv:1910.09505*.

Sankaran, K.; Zhu, M.; Guo, X. F.; Ananda, A. L.; Chan, M. C.; and Peh, L.-S. 2014. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM.

Stenneth, L.; Wolfson, O.; Yu, P. S.; and Xu, B. 2011. Transportation mode detection using mobile phones and gis information. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM.

Yu, M.-C.; Yu, T.; Wang, S.-C.; Lin, C.-J.; and Chang, E. Y. 2014. Big data small footprint: the design of a low-power classifier for detecting transportation modes. *Proceedings of the VLDB Endowment* 7(13):1429–1440.

Zheng, Y.; Li, Q.; Chen, Y.; Xie, X.; and Ma, W.-Y. 2008. Understanding mobility based on gps data. In *Proceedings of the 10th international conference on Ubiquitous computing*, 312–321. ACM.