Integration of GPS and Satellite Images for Detection and Classification of Fleet Hotspots

Francesco Sambo, Samuele Salti, Luca Bravi, Matteo Simoncini, Leonardo Taccari and Alessandro Lori
Fleetmatics Research
via Paisiello 16-20, 50144 Florence, Italy
Email: francesco.sambo@fleetmatics.com

***Abstract*—Hotspot detection and classification for a fleet of vehicles is usually performed based on GPS data sampled from the vehicles. In this paper, we explore how the integration of satellite images can improve GPS-based hotspot classification. We propose a system composed of a deep Convolutional Neural Network (CNN) for image classification and a Random Forest classifier that combines GPS-based features with the CNN output for hotspot classification. We introduce also a novel metric for scoring place detection and classification systems, able to account for both detection and classification errors. The new metric is used to assess experimentally the effectiveness of our system in combining the two sources of information.**

I. Introduction

Fleet management solutions, such as Fleetmatics REVEAL, rely on on-board GPS tracking devices to monitor the position of all vehicles of a fleet in real-time; GPS data is used to provide additional services, such as replay of meaningful routes, generation of driving reports and raise of alarms. Automated detection and classification of recurrent stop locations, or *hotspots*, is crucial for fleet management, as it provides invaluable insights into drivers behaviour and fleet operations; these, in turn, can be translated into a product more tailored to the customer and smarter in responding to his/her needs.

The discovery of relevant hotspots for a fleet is actually a two-step problem: *detection*, i.e. identification of hotspot location and size, and *classification*, i.e. association to the hotspot of a semantic label like home or depot. In the literature, place detection and semantic classification are mainly studied for human GPS trajectories [1], [2], [3] and, more rarely, for commercial vehicles [4]. Hotspot detection is usually accomplished by first aggregating nearby subsequent points from the same track and then clustering nearby points from different GPS tracks. Hotspot classification, on the other hand, is often based on a combination of different criteria: ad-hoc rules, such as the longest stop of the day classified as home or the most visited location as depot [2], [4], machine learning algorithms trained on statistics like average duration and arrival time, computed on single stops or on all the stops that belong to a hotspot [1], [3], or proximity to points of interest available in external databases, such as public transportation stops or shops and restaurants [2], [3].

Given the location of a hotspot, another powerful source of information on its category is the satellite image of its close surroundings. The literature on object detection from satellite images exhibits a growing interest in machine learning approaches [5] and in the rising paradigm of deep learning [6], where rich feature representations are automatically learned from images by means of Convolutional Neural Networks (CNN, [7]).

In this paper, we explore the integration of GPS data with satellite images for the classification of the hotspots of a fleet. As far as we are aware, most of the literature on the integration of images with GPS data has focussed on increasing the precision of GPS navigation with on-board vehicle camera images [8], [9], [10] and no work has yet considered the integration with satellite images.

To develop our solution, we use a dataset of continuous GPS monitoring of the whole fleet for 1025 Fleetmatics customers. To obtain a ground truth on which our models could be trained, we manually labelled the hotspots of each customer and classified them among the three classes *Home*, representing employees homes, *Depot*, representing a company common place such as a vehicles depot or an office, and *Other*, comprising recurrent clients of our customers and other special locations, such as an airport, a school, or a gas station. Moreover, we also independently labelled the land use of each hotspot as belonging to two classes: *Residential* and *Non-residential*. Among the possible land uses, we chose residential use since it can be generally identified from a satellite image with little ambiguity. For each hotspot, we also included in the dataset a satellite image of its location and surroundings.

Our classifier consists of two main components: a deep Convolutional Neural Network, which provides an initial classification of satellite images into the two land use classes, and a Random Forest classifier (RF, [11]), which uses the CNN output, together with several features extracted from GPS data, to compute the final classification of the hotspots. An independent subset of fleets is extracted from the whole dataset before training the system and is used as test set for performance assessment. Hotspot detection on test data is obtained by a grid-based partitioning of all the stops of a fleet, followed by the aggregation of neighbouring cells.

Our algorithm is meant to be highly parallelizable and able to scale to the size of the Fleetmatics customer base: we thus resort to Apache Spark [12] as parallel processing engine for data preprocessing, hotspot detection and Random Forest classification, and we use the Google Tensorflow library ([13])

and its python wrapper Keras[1] for CNN classification.

Another major contribution of the present paper is an end-to-end performance measure for place detection and classification systems, based on the Matthews Correlation Coefficient [14] and meant to penalise both misclassification errors and detection errors, such as imperfect overlaps between ground truth and detected hotspots. The score is designed to cope with the limitations of the other approaches to the evaluation of place detection and classification systems, such as separate evaluation of detection and classification [3] or evaluation of the entire system in terms of classification accuracy alone [1].

Experimental results show that our system is indeed able to effectively combine the information originating from GPS data and satellite images, outperforming the classifier built on GPS data alone. Furthermore, we demonstrate the effectiveness of our new performance measure by studying different variants of our hotspot detection procedure and by comparing them with the ideal case of ground truth detection.

The paper is organised as follows: Section II presents the dataset we used to build and validate our system, Section III describes in detail the different components of the system, Section IV introduces our novel performance measure, Section V shows experimental results and Section VI draws conclusions and future directions.

II. Dataset

To develop our system, we collected 15 days of GPS data for 1,025 Fleetmatics US customers, for a total of 11,668 vehicles and 869,574 GPS *stops*, i.e. time intervals between an engine off event and the subsequent engine on. Each stop consists of the vehicle position, as a longitude-latitude pair, and of the time stamps of the engine off and on events.

To define ground-truth data, we automatically detected candidates hotspots, for each fleet, as all places with a cumulative stop duration of at least 20 hours across the fleet in the two weeks. We then manually classified candidates among the classes Home, Depot and Other by visual inspecting the stop patterns, the vehicles routes and the location via Google Maps and Google Street View. The total number of hotspots for each class is 4,394 for Home (49%), 1,311 for Depot (15%), and 3,238 for Other (36%).

The set of fleets was then randomly split, excluding 205 fleets (20%) from further analyses and keeping them as an independent test set, meant to assess the performance of our system.

III. Methods

Our procedure for hotspot detection and classification is represented in Figure 1. For each ground truth hotspot in the training set, we download a satellite image using the Google Static Maps API, centered in the hotspot center and with 256×256 pixels at zoom level 18. Images are further split into a train set (615 fleets) and a validation set (205 fleets) and used to train a CNN.

[1]https://keras.io/

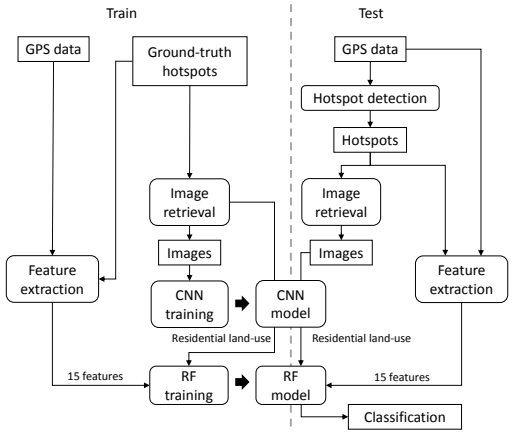

Fig. 1. Architecture of the hotspot detection and classification procedure.

For the test set, we run the hotspot detection procedure described in the next section and then retrieve the satellite images centered in the detected hotspots. The CNN model is used to compute, both for ground truth hotspots in the train set and for detected hotspots in the test set, the probability of being to the *Residential* land use class. In parallel, the ground-truth hotspots and the GPS data train set are used to extract a set of GPS-based features, as described in Section III-C. The features are used, together with the output probability of the CNN on the training images, to train a Random Forest classifier, which then processes the test set and outputs the final hotspot classification. All the components are explained in more detail in what follows.

A. Hotspot detection

To detect the hotspots of a fleet starting from the GPS stops of all the fleet vehicles, we designed the following procedure.

1) Group stops into cells, defined by a regular longitude $\times$ latitude grid with spacing s degrees.
2) For all cells, compute the cumulative stop time, i.e. the total stop time spent in the cell by all the vehicles of the fleet, and retain all cells with cumulative stop time larger than or equal to t_{cell} as initial hotspots.
3) For a number of iterations it_{aggr}, aggregate pairs of adjacent hotspots into larger hotspots, of rectangular shape and with boundaries defined by the minimum and maximum latitude and longitude of the two aggregated hotspots.
4) For all resulting hotspots, compute again the cumulative stop time and retain as final hotspots the ones with cumulative stop time larger than or equal to t_{aggr}.

We choose to adopt this solution, similar to [4] and as opposed to more complex clustering approaches [1], [3], because it is highly parallelizable and it yields sufficiently good results on our data, as shown in Section V. Based on an exploratory analysis of their effect on the final classification performance (not reported), the parameters were set to $s = 0.0008$ degrees (yielding initial cells of about 80×80 meters at US latitudes), $t_{cell} = 8$ hours, $it_{aggr} = 3$ and $t_{aggr} = 20$ hours.

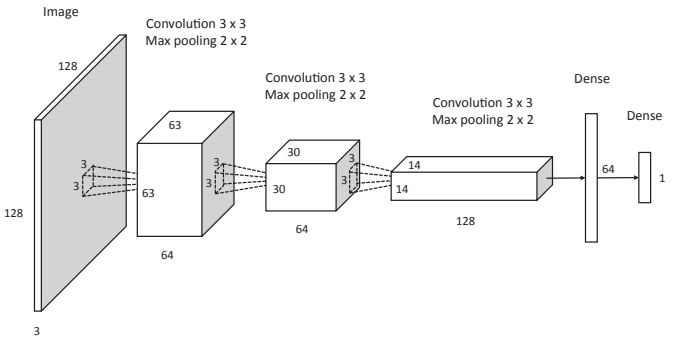

Fig. 2. Layers and parameters of the convolutional neural network.

B. Image classification with convolutional neural networks

We perform satellite image classification by means of a convolutional neural network. Many of our architectural and algorithmic design choices follow closely the ones suggested in [6], which reports best practices and state of the art results for satellite image classification with Convolutional Neural Networks. However, we add to the model suggested in [6] some further improvements, drawn from the recent literature on deep learning, such as PReLU activation functions [15] and the Adam training algorithm [16].

The architecture we choose, represented in Figure 2, consists of three 2D *convolutional blocks*, followed by a dense hidden layer and a dense output layer. The input to the networks is a 128×128 color image with the 3 RGB channels. Each convolutional block consists of:

1) A convolutional layer [17], with kernels of 3×3 pixels, stride of 1 pixel and multiple channels (64 in the first two blocks, 128 in the third);
2) A non-linear activation function, in the form of a Parametric Rectified Linear Unit (PReLU, [15]);
3) A max-pooling layer, of size 2×2 and stride 2.

The output of the third convolutional block is flattened and fed to a dense hidden layer, with 64 neurons and PReLU activation function, and this in turn is the input to an output layer with one neuron and sigmoid activation function.

The activation function we choose, the Parametric Rectified Linear Unit, an extension of the most commonly adopted ReLU, has recently allowed deep networks to surpass human level performance on the ImageNet 2012 classification dataset [15]. The PReLU activation function is defined as:

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases} , \qquad (1)$$

where y_i is the input of the nonlinear activation f on the $i-$th channel and a_i is a coefficient controlling the slope of the negative part, whose value is learned during training. The original ReLU, which outputs a flat 0 for $y_i \leq 0$, is indeed made parametric by the a_i coefficients. As suggested in [15], we initialize all a_is to 0.25 before training.

In line with the literature on binary classification problems, we choose binary entropy as loss function. To train the network, we adopt the common approach of Stochastic Gradient Descent (SGD), which partitions the training data in random sub-samples, known as *minibatches*, and iteratively adjusts the network weights according to the gradient of the loss function in each minibatch, cycling through the entire dataset for several times, or *epochs*. We choose a variant of the SGD algorithm known as Adam, which maintains different learning rates for the network weights and adjusts them adaptively, based on estimates of the first and second moments of the gradients [16]. Two free parameters of the Adam algorithm, namely the step size and the minibatch size, were tuned based on classification performance in the validation set and then fixed to 5×10^{-5} and 8, respectively; all the other parameters were left as default. We also use dropout, which reduces the risk of overfitting by ignoring a random sample of 50% of the nodes in each iteration [18].

Before being fed to the network, images undergo several pre-processing and augmentation steps:

1) rotation with a random angle, uniformly sampled between ± 45 degrees,
2) shift of height and width with random ranges, uniformly sampled between $\pm 20\%$ of the image size,
3) zoom with a random range, uniformly sampled between ± 0.2,
4) random horizontal flip, with probability 0.5,
5) rescaling of all pixels between 0 and 1,
6) subtraction of the mean value of each pixel across the training dataset,
7) resizing to 128×128 pixels.

Random transformations 1 to 4 are augmentations of the original image set, meant to increase the generalization ability of the trained classifier. Rescaling (5) and mean shift (6) yield a more stable gradient descent and resizing to a smaller image (7) leads to a model with fewer parameters. Several configurations of random perturbations have been tried in an exploratory analysis (not reported) but these are the ones leading to the best generalization performance on the validation set.

The validation set is also used to control overfitting, a known Achilles' heel of neural networks: the network is trained for a total of 200 epochs and, at the end of each epoch, it is used to process the validation set and to assess classification performance by means of the Matthews Correlation Coefficient (MCC, see Experimental Results for the definition). The best network is chosen as the one with the maximum MCC on the validation set.

C. Feature extraction from GPS data

For all the fleets in the training and test sets, we aggregate all stops falling within the ground truth hotspots (for the training set) and the detected hotspots (for the test set) and use them to compute the following features of each hotspot:

- average number of stops per day,
- percentage of fleet vehicles that stop at least once in the hotspot,
- mean and standard deviation of stop durations,
- maximum and minimum stop duration,

- average cumulative stop duration per day,
- percentage of overnight stops,
- area and aspect ratio (height/width) of the hotspot bounding box,
- mean spatial density (number/area) of stops per day,
- percentage of stops starting or ending in one of two specific times of the day (morning, i.e. from 5 AM to 2 PM, and afternoon/evening, i.e. from 2 PM to 11 PM),

for a total of 15 GPS-based features.

We have been inclusive in the choice of GPS features since we use them to train Random Forests, which are automatically able to select the most relevant features for the classification task. In Section V, we use the learned Random Forest to provide an *a posteriori* ranking of the features in term of their usefulness for classification.

D. Random Forest classification

A Random Forest classifier [11] is trained on the hotspots of the fleets in the the train set, with the 15 GPS-based features plus the Residential land use probability estimated by the CNN, as explained in the previous sections. We choose the Random Forest implementation from the Apache Spark MLlib library[2], which uses Gini impurity to estimate the gain of possible splits during classification tree growth and samples $\sqrt{m}$ features for each tree, where m is the total number of features.

We choose to not limit the number of instances per node and the minimum gain required for a split, using only the maximum tree depth and the number of trees in the forest to control the model complexity. Five-fold cross-validation on the train set is used to tune the two parameters, choosing the combination of values which yields the highest macro-average [19] MCC on the five test folds. The optimal parameter values are thus estimated as 200 trees with maximum depth 20 for the complete RF and 300 trees with maximum depth 10 for a simpler version that uses only the GPS-based features.

IV. Performance measure

Due to the high unbalance between the classes in our dataset, we choose as performance measure the Matthews Correlation Coefficient (MCC, [14]). Originally defined for binary classification problems, the MCC requires the indication of a positive and a negative class and, starting from the counts of true positives (*tp*), true negatives (*tn*), false positives (*fp*) and false negatives (*fn*), it is computed as:

$$\text{MCC} = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}} \quad (2)$$

The MCC is especially useful for classification problems with high class imbalance: it lies between -1 (worst performance) and +1 (best performance) and equals 0 in case of majority classification (a particularly undesired but frequent pitfall of training a classifier on imbalanced data). Compared to other measures that are frequently used with class imbalance, such as Precision and Recall, MCC has the added value of condensing all four counts *tp*, *fp*, *tn* and *fn* in a single aggregate measure [14]. A common extension of the MCC to the multi-class case is the macro-average approach [19], where each class in turn is considered as positive and all the other classes as negative, MCC is computed in each case and then averaged.

[2]https://spark.apache.org/mllib/

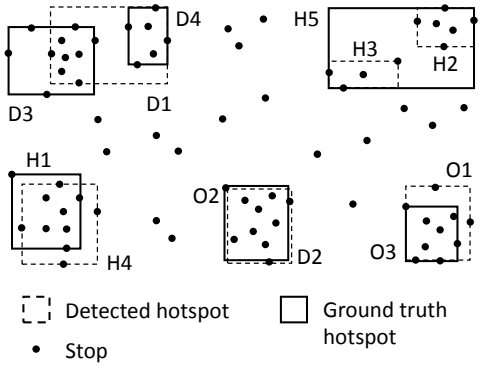

Fig. 3. Examples of detected and ground-truth hotspots.

TABLE I
EXAMPLE OF CONFUSION MATRIX

Classification / Ground truth	H1	H2	H3	D1	D2	O1	out
H4	8	0	0	0	0	0	2
H5	0	6	4	0	0	0	0
D3	0	0	0	8	0	0	3
D4	0	0	0	6	0	0	0
O2	0	0	0	0	10	0	0
O3	0	0	0	0	0	8	0
out	1	0	0	0	0	2	17

For the training phase of both the CNN and the RF, we consider hotspots as atomic samples to be classified, i.e. when computing the MCC, every hotspot can be a true positive, false positive, etc. For the testing phase, which involves also hotspot detection starting from GPS stops, we consider each stop as an atom and design a metric on top of the MCC that can capture and penalize mistakes in hotspot detection. We dub this measure *Detection and Classification MCC (DCM)*. To introduce the measure, we will refer to the confusion matrix in Table I as a running example, where each row represents a ground truth hotspot, belonging to class home (H), depot (D) or other (O), plus *out* for the stops falling out of each ground truth hotspot, each column represents the detected and classified hotspots and each cell counts the number of stops falling in the intersection between the ground truth and the detected hotspot. A graphical representation of the same running example is given in Fig. 3.

For each class C, we define as true positives all stops that lie both in a detected hotspot classified as C and in a ground truth hotspot of class C, false positives all stops within detected hotspots of class C but not in ground truth hotspots of class C, and so forth for false negatives and true negatives. In our running example for class O, 8 stops lie in the intersection

between detected hotspot $O1$ and ground truth hotspot $O3$ (element at column $O1$ and row $O3$ in Table I), so *tp* equals 8; 2 stops are in $O1$ but not in $O2$ or $O3$ (sum of all elements in column $O1$ excluding rows $O2$ and $O3$), so *fp* is 2; all stops in $O3$ are also in $O1$, but $O2$ is misclassified ad Depot, so the 10 stops in it count as *fn*; all the remaining 55 stops do not lie in hotspots of class O (sum of all elements not in column $O1$ or rows $O2$ and $O3$), so *tn* is 65.

In order to penalize multiple detected hotspots intersecting the same ground truth hotspot, like detected homes $H2$ and $H3$ with ground truth home $H5$ in Fig. 3, in each row of the confusion matrix we divide the *positive counts*, i.e. the counts of hotspots belonging to the correct class, by the number of nonzero positive counts in the confusion matrix before summing true positives. In the example for class H, tp will then be the sum of 8 for hotspot $H1$, $6/2$ for $H2$ and $4/2$ for $H3$. A similar penalty is applied to detected hotspots intersecting multiple ground truth hotspots. As an example, the detected depot $D1$ intersects two ground truth depots $D3$ and $D4$, so *tp* for the class D will be $8/2 + 6/2 = 7$.

Given the confusion matrices computed according to the outlined algorithm, we define the DCM as the macro-averaged MCC across the confusion matrices.

V. Experimental results

Figure 4a reports boxplots of the macro-average DCM on the 205 fleets in the test set, comparing simple Random Forest with just the GPS-based features and complete Random Forest including also the Residential land use estimated by the CNN. As it is clear from the figure, the simple Random Forest already exhibits a quite good performance, with median macro-average DCM of 0.854. However, when information on the land use from the CNN is taken into account in the complete Random Forest, the performance further improves, reaching a median value of 0.898 for the macro-average DCM while the DCMs distribution shifts towards higher scores. The improvement is statistically significant, with a p-value of 0.009 for a Wilcoxon signed rank test.

Figure 4b further explores the test DCM obtained by the complete Random Forest on each separate class. The best and most consistent classification performance is obtained on the Depot and Home classes, with median DCMs of 0.982 and 0.969 and inter-quantile ranges of 0.094 in both cases. For the Other class, the median DCM is equal to 0.825, but the distribution is much more spread towards lower DCMs, with and inter-quantile range of 0.44.

A posteriori feature ranking from the learned random forest is accomplished as suggested in [20], accumulating for each feature the improvements in Gini impurity obtained during training by splitting on it. The most useful features emerged as the average cumulative stop duration per day and the fraction of overnight stops, followed by the average number of stops per day, and the percentages of stops starting in the morning and in the afternoon. The CNN output (Residential land use) ranks 6th, before all the remaining GPS features: this confirms its usefulness in providing complementary evidence to GPS data to correctly classify the hotspot.

TABLE II
AWS instances and execution time for the different components of our system

Phase	Instance	Execution time
Hotspot detection	m4.2xlarge	18s
CNN training	p2.xlarge	2h 40m
CNN prediction	p2.xlarge	1m 56s
GPS feature extraction	m4.2xlarge	25s
Random Forest training	m4.2xlarge	61s
Random Forest prediction	m4.2xlarge	5s

By leveraging the ability of the proposed metric to compare end-to-end performance of different pipelines, Figure 4c compares our hotspot detection procedure (central boxplot), consisting in grid-based partitioning of stops into rectangular cells followed by aggregation of adjacent cells, with a simpler approach comprising only grid-based clustering (left boxplot) and with the theoretical maximum DCM obtainable with our classifier by using ground truth for hotspot detection. As it is clear from the figure, aggregation of neighboring cells greatly improves simple grid-based clustering, raising the median DCM from 0.749 to 0.898, but additional improvements of the clustering procedure could allow it to grow further and reach its theoretical limit of 0.956.

All the experiments were run on two machines, an Amazon EC2 m4.2xlarge instance with 8 2.3GHz Intel Xeon E5-2686v4 processors and 32GB of RAM, for the computations under the Apache Spark environment (hotspot detection, GPS feature extraction and Random Forest classification), and an Amazon EC2 p2.xlarge instance with 4 Intel Xeon E5-2686v4, an NVIDIA K80 GPU and 61GB of RAM, for image processing and CNN classification. Table II reports the machine used and the execution time for the different components of our system: the vast majority of computational time is taken by training the CNN (2h 40m), but once training is over the complete process of hotspot detection and classification for the 205 test fleets takes a total of 2m 44s.

VI. Conclusion

In this paper, we introduced a novel approach to the detection and classification of hotspots for fleets of vehicles, based on the integration of GPS data sampled from the vehicles with satellite images of the hotspot. As components of our system, we chose two state-of-the-art classifiers and combined them effectively, indeed improving the classification performance on our test data.

Experimental results, however, show that classification performance is quite different across the three classes Home, Depot, and Other, with the classifier being much more effective on the former two than on the latter. The reason for this probably lies in the different levels of homogeneity within each class: employees homes and fleet depots tend to be better defined and less heterogeneous, in terms of both visit patterns

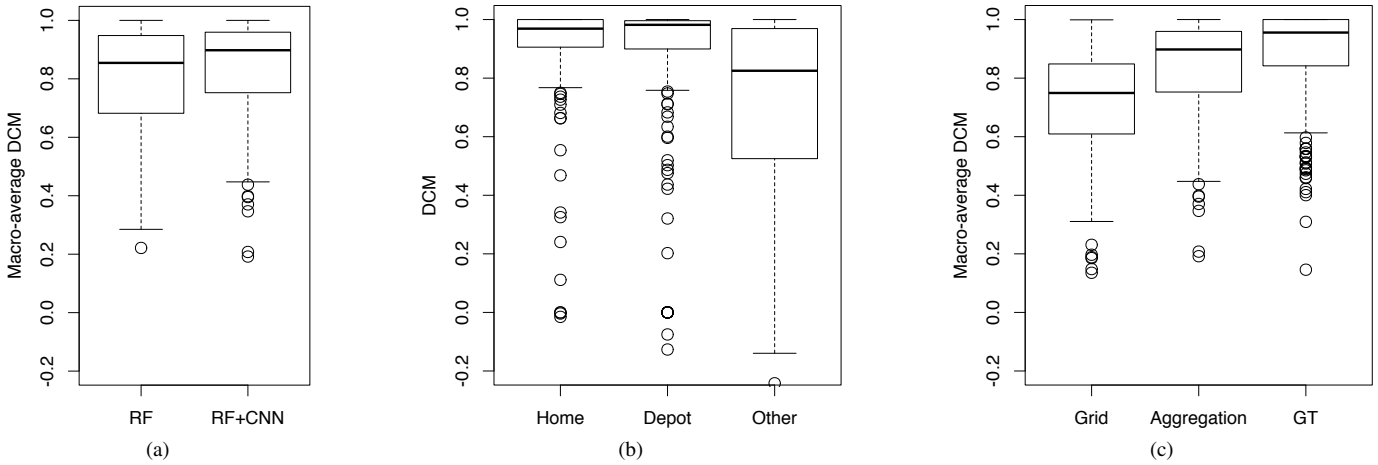

Fig. 4. Boxplots of DCM across the 205 fleets in the test set. 4a: macro-average DCM of simple RF without the CNN output and complete RF including the CNN output. 4b: DCM on the three separate classes of the complete RF model. 4c: macro-average DCM of the complete RF model with three types of hotspot detection, namely simple grid partitioning in cells, grid partitioning followed by adjacent cell aggregation, and ground truth detection.

and visual appearance, than all the remaining hotspots falling in the Other class.

A further contribution of the paper is the introduction of a novel performance measure, based on the Matthews Correlation Coefficient, which allows us to account for both detection and classification errors in the evaluation of our system and which could effectively be used in several other problems of object detection and classification. By means of our new performance measure, we assessed the effectiveness of our simple hotspot detection procedure, but also evidenced that there is room for improvement.

Indeed, one of the possible future directions of this work is to explore the best way to apply state-of-the art clustering approaches, such as DBSCAN [21], in our parallel processing framework, while retaining the high level of scalability of the present solution. We also plan to explore alternative ways to integrate the Random Forest and CNN classifiers and assess the potential of adding other sources of information to the system, such as additional land use classes [6] or places of interest [3], gathered from public databases such as Foursquare, Open Street Map or Google Maps.

REFERENCES

[1] L. Montini, N. Rieser-Schüssler, A. Horni, and K. Axhausen, "Trip purpose identification from GPS tracks," *Transportation Research Record*, no. 2405, pp. 16–23, 2014.

[2] K. Siła-Nowicka, J. Vandrol, T. Oshan, J. A. Long, U. Demšar, and A. S. Fotheringham, "Analysis of human mobility patterns from gps trajectories and contextual information," *International Journal of Geographical Information Science*, vol. 30, no. 5, pp. 881–906, 2016.

[3] M. Lv, L. Chen, Z. Xu, Y. Li, and G. Chen, "The discovery of personally semantic places based on trajectory data mining," *Neurocomputing*, vol. 173, Part 3, pp. 1142–1153, 2016.

[4] K. Gingerich, H. Maoh, and W. Anderson, "Classifying the purpose of stopped truck events: An application of entropy to GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 17 – 27, 2016.

[5] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 117, pp. 11 – 28, 2016.

[6] Y. Zhong, F. Fei, Y. Liu, B. Zhao, H. Jiao, and L. Zhang, "SatCNN: satellite image dataset classification using agile convolutional neural networks," *Remote Sensing Letters*, vol. 8, no. 2, pp. 136–145, 2017.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444.

[8] C. Rose, J. Britt, J. Allen, and D. Bevly, "An integrated vehicle navigation system utilizing lane-detection and lateral position estimation systems in difficult environments for GPS," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2615–2629, 2014.

[9] J. Bao, Y. Gu, L.-T. Hsu, and S. Kamijo, "Vehicle self-localization using 3D building map and stereo camera," in *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2016-August, 2016, pp. 927–932.

[10] J. Lim, K. Choi, J. Cho, and H. Lee, "Integration of GPS and monocular vision for land vehicle navigation in urban area," *International Journal of Automotive Technology*, vol. 18, no. 2, pp. 345–356, 2017.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust *et al.*, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[14] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, p. 412, 2000.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Comp. Soc., 2015, pp. 1026–1034.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[19] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information retrieval*, vol. 1, no. 1-2, pp. 69–90, 1999.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, 2nd ed. Springer, 2009.

[21] L. Gong, H. Sato, T. Yamamoto, T. Miwa, and T. Morikawa, "Identification of activity stop locations in GPS trajectories by density-based clustering method combined with support vector machines," *Journal of Modern Transportation*, vol. 23, no. 3, pp. 202–213, 2015.