Traffic4cast at NeurIPS 2022 – Predict Dynamics along Graph Edges from Sparse Node Data: Whole City Traffic and ETA from Stationary Vehicle Detectors

http://traffic4cast.ai – https://github.com/iarai/NeurIPS2022-traffic4cast

Moritz Neun[*,a] MORITZ.NEUN@IARAI.AC.AT
Christian Eichenberger[*,a] CHRISTIAN.EICHENBERGER@IARAI.AC.AT
Henry Martin[*,†] HENRY.MARTIN@IARAI.AC.AT
Markus Spanring[*] MARKUS.SPANRING@IARAI.AC.AT
Rahul Siripurapu[*] RAHUL.SIRIPURAPU@IARAI.AC.AT
Daniel Springer[*] DANIEL.SPRINGER@IARAI.AC.AT
Leyan Deng[‡,§], **Chenwang Wu**[‡], **Defu Lian**[‡], **Min Zhou**[§] DLEYAN@MAIL.USTC.EDU.CN
Martin Lumiste[¶], **Andrei Ilie**[¶,‖] MARTIN.LUMISTE@BOLT.EU
Xinhua Wu[**], **Cheng Lyu**[††], **Qing-Long Lu**[††],
Vishal Mahajan[††] WU.XINH@NORTHEASTERN.EDU
Yichao Lu[‡‡] YICHAO@LAYER6.AI
Jiezhang Li[§§], **Junjun Li**[§§], **Yue-Jiao Gong**[§§] GONGYUEJIAO@GMAIL.COM
Florian Grötschla[¶¶], **Joël Mathys**[¶¶] FGROETSCHLA@ETHZ.CH
Ye Wei[***], **He Haitao**[†††], **Hui Fang**[***] Y.WEI@LBORO.AC.UK
Kevin Malm[‡‡‡] KEVIN.MALM@HERE.COM
Fei Tang[§§§] FEI.TANG3@GMAIL.COM
Michael Kopp[*] MICHAEL.KOPP@IARAI.AC.AT
David Kreil[*] DAVID.KREIL@IARAI.AC.AT
Sepp Hochreiter[¶¶¶,*] SEPP.HOCHREITER@IARAI.AC.AT

Editors: Marco Ciccone, Gustavo Stolovitzky, Jacob Albrecht

[*] Institute of Advanced Research in Artificial Intelligence (IARAI), Vienna, Austria
[†] Institute of Cartography and Geoinformation, ETH Zurich, Switzerland
[‡] School of Data Science, University of Science and Technology of China
[§] Huawei Noah's Ark Lab
[¶] Bolt Technology, Tallinn, Estonia
[‖] University of Bucharest, Bucharest, Romania
[**] Department of Civil and Environmental Engineering Northeastern University Boston, MA, USA
[††] Chair of Transportation Systems Engineering, Technical University of Munich, Germany
[‡‡] Layer 6 AI, Toronto, Canada
[§§] School of Coumpute Science and Engineering, South China University of Technology, Guangzhou, China
[¶¶] ETH Zurich, Switzerland
[***] Department of Computer Science Loughborough University Loughborough, UK
[†††] School of Architecture, Building and Civil Engineering Loughborough University Loughborough, UK
[‡‡‡] HERE Technologies, Chicago, IL, USA
[§§§] Kaiko, Zurich, Switzerland
[¶¶¶] Machine Learning Institute, Johannes Kepler University Linz, Austria

[a] Equal Contribution

Abstract

The global trends of urbanization and increased personal mobility force us to rethink the way we live and use urban space. The *Traffic4cast* competition series tackles this problem in a data-driven way, advancing the latest methods in machine learning for modeling complex spatial systems over time. In this edition, our dynamic road graph data combine information from road maps, 10^{12} probe data points, and stationary vehicle detectors in three cities over the span of two years. While stationary vehicle detectors are the most accurate way to capture traffic volume, they are only available in few locations.

Traffic4cast 2022 explores models that have the ability to generalize loosely related temporal vertex data on just a few nodes to predict dynamic future traffic states on the edges of the entire road graph.

In the core challenge, participants are invited to predict the likelihoods of three congestion classes derived from the speed levels in the GPS data for the entire road graph in three cities 15 min into the future. We only provide vehicle count data from spatially sparse stationary vehicle detectors in these three cities as model input for this task. The data are aggregated in 15 min time bins for one hour prior to the prediction time. For the extended challenge, participants are tasked to predict the average travel times on super-segments 15 min into the future – super-segments are longer sequences of road segments in the graph.

The competition results provide an important advance in the prediction of complex city-wide traffic states just from publicly available sparse vehicle data and without the need for large amounts of real-time floating vehicle data.

1. Introduction

Going beyond the *Traffic4cast* challenges at NeurIPS 2019, 2020, and 2021 (Kreil et al., 2020; Kopp et al., 2021; Eichenberger et al., 2022), *Traffic4cast* 2022 explores models that have the ability to generalize loosely related temporal vertex data on just a few nodes to predict dynamic future traffic states on the edges of the entire road graph. In our **core challenge**, participants are asked to predict the likelihood of three congestion classes derived from for three cities for the entire road graph 15 min into the future. A schematic overview is shown in Figure 1, illustrating the common red, yellow, or green coloring of roads on a traffic map. We provide vehicle count data from spatially sparse stationary vehicle detectors in these three cities in 15 min aggregated time bins for one hour prior to the prediction time slot. Stationary vehicle detectors are often inductive loops (electrically conducting loops) installed in the road pavement (Wikipedia, 2023) or cameras at traffic lights, and providing vehicle counts and sometimes also speed and occupancy data – *Traffic4cast* 2022 only uses the vehicle count data. See Figure 2 for the location of these stationary vehicle detectors in relation to the rest of the road graph. For our **extended challenge**, participants are tasked to predict the average travel times on super-segments 15 min into the future. Average travel times correspond to ETAs (Expected Time of Arrival), and we loosely use both terms synonymously. Super-segments are longer sequences of road segments in the graph.

Solving these challenges has direct substantial implications for our ability to forecast, plan, and analyze urban traffic leading to a considerable impact on society, environment, and health. Our competition addresses the two main challenges identified in (Manibardo et al., 2020) for the application of deep learning methods for road traffic forecasting by formulating the task in an actionable (*i. e.* application-driven) setting and by providing the dataset and leaderboard. Its challenges directly allow for ETA (estimated time of

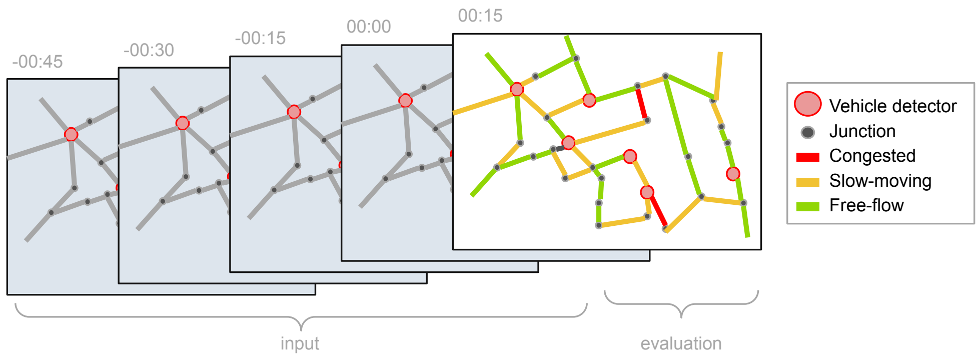


Figure 1: *Traffic4cast* 2022: Traffic Dynamics Prediction on Graphs for the Congestion Classification Task. Given one hour of sparse vehicle count data only (red disks) the task is to predict the congestion classification for all road segments 15 min into the future (green, yellow, and red lines). For the extended challenge, the prediction targets are average travel times along super-segments.

arrival) prediction (see (Hu, 2022; Schleibaum et al., 2022)) from publicly available sparse data. Therefore, without the need for large amounts of real-time floating vehicle data, it dramatically lowers the technical and financial entry barriers for performing such predictions. This is an active area of research (Derrow-Pinion et al., 2021; Hu, 2022; Elmasri, 2019; Schleibaum et al., 2022), which our competition may directly impact. The vast city-wide scope of our competition's underlying data directly boosts the active field of research on full-field traffic state identification. Currently, the traffic state identification is often limited to particular road sub-systems for which data is available, *e. g.* (Zhang et al., 2022a).

Moreover, the ability to generally predict temporal edge features in graphs from sparse vertex data is crucial for many real-world applications, many of which are critical to the functioning of our society. In (Kapoor et al., 2020) the spread of COVID-19 can be modelled as predicting temporal edge features in a large spatio-temporal graph. Malicious software can be identified as anomalous patterns in temporal sequences of evolving multiscaled API graph (Zhang et al., 2022b). Solutions to traffic prediction on graphs are also relevant to Wide Area Network traffic prediction (Mallick et al., 2020) and to learning the temporal dynamics of crypto-currency networks (Pareja et al., 2019).

1.1. Data

We provide a unique data set derived from massive industry-scale GPS data, large openly available stationary vehicle detector data, and road graph information from three diverse metropolitan areas.

The GPS data has been aggregated and made available by HERE Technologies (HERE Technologies, 2020) as in the previous instances of *Traffic4cast* (Kreil et al., 2020; Kopp et al., 2021; Eichenberger et al., 2022). The data originates from a large fleet of probe vehicles which recorded their movements in the years 2019 to 2021. For this year's competition we focus on three cities: London, Madrid, and Melbourne (see Figure 2). The underlying raw data is similar to the input for commercial traffic maps and routing products. Unlike previous years, this data is not available as test input, but serves to derive the ground truth labels

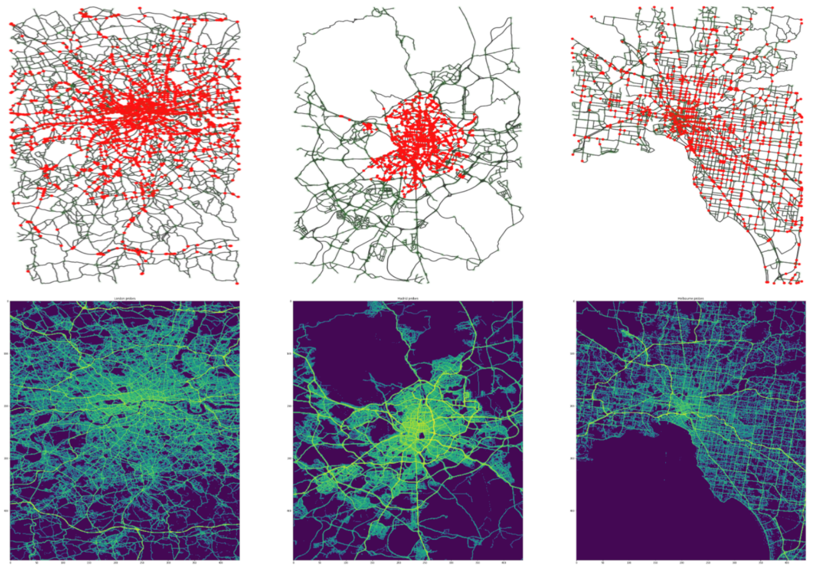


Figure 2: From left to right we show London, Madrid, and Melbourne. Top panels: distribution of stationary vehicle detector locations (red dots) in comparison to the sampled road networks. Bottom panels: GPS probe data covering a much finer and larger road network. The road graph sampling removed road segments with very few data points. Sources ©HERE Technologies, ©OpenStreetMap Contributors, ©TfL and Highways England, ©Madrid City Council, ©Victoria State Government

for our core and extended challenge, see (Neun et al., 2023) and Appendix D. In addition, participants can use the speed data on the graph edges for jointly modeling traffic or as labels for intermediate tasks. A selected set of time intervals for all three cities are held back for the test set. Specifically, we split on alternate weeks into training data and data held back for sampling 100 test slots.

Stationary vehicle detector data is the new additional data source for *Traffic4cast* 2022. Stationary vehicle detectors are often made publicly available by city or motorway authorities. Furthermore, an increasing collection of crowd-sourced data through alternative sensor networks (Vanherle, 2021; Sardo et al., 2022) is becoming widely available. Stationary vehicle detectors are spatially sparse (up to $\sim$ 4000 per city, see Table 1 and Figure 2) but capture all traffic at their locations. We focus on the large, publicly available, stationary vehicle detector data for London, Madrid, and Melbourne. The stationary vehicle detector measurements are represented as nodes with measured volumes per 15 min time bin.

We are providing a pre-compiled version of the derived data sets as well as tools to convert or easily load the data in our GitHub repository.

1.1.1. Input and Static Data

The input data for our core and the extended challenge is derived from publicly available stationary vehicle detector data sets from Madrid, Melbourne, and London (see Table 1). See Appendix A for more details. The stationary vehicle detector measurements can be attached to the nodes in the provided road graph (see also Figure 1).

city	# vehicle detectors / # nodes with detector data	# nodes (incl. detectors)	# edges	# super-segments
London (TfL, 2022; HE, 2022)	3751 / 3751	59110	132414	4012
Madrid (Madrid, 2022)	3840 / 3875	63397	121902	3969
Melbourne (VicGov, 2022b)	2589 / 3982	49510	94871	3246

Table 1: Number of stationary vehicle detectors and road graph nodes, edges, and super-segments for the three cities London, Madrid, and Melbourne. Vehicle detector data may be split across several nearby nodes.

We are providing a road graph, represented by a directed graph, for each city with stationary vehicle detector data. The road graph is derived from OpenStreetMap (OpenStreetMap contributors, 2022), simplified by filtering out local roads with no or almost no GPS ground truth and collapsing nearby nodes that are not relevant for the traffic flow and connectivity. See Table 1 for an overview of the graph vertex and edge sizes and Appendix B for more details on the derivation.

In addition, we are providing a choice of super-segments (Derrow-Pinion et al., 2021) represented as lists of segment IDs. The use of super-segments is motivated by contraction hierarchies (Geisberger et al., 2008), which are a common strategy in most commercial routing engines. Super-segments are sampled from the road graph and validated using a commercial routing engine. See Appendix C for details.

1.1.2. Output Labels

The aggregated GPS probe data are used to generate our ground truth labels. As the stationary vehicle detector data comes at 15 min intervals, the 5 min time bins in the spatio-temporal data format that has been used in previous editions of the competition, see (Eichenberger et al., 2022), are aggregated to 15 min time bins (Neun et al., 2023). We then derive the following ground truth labels:

- Congestion Class (**CC**; red/congested, yellow/warning, green/uncongested) for each segment in the road graph. The class is derived from the aggregated GPS probe data; if not enough data is available to derive the congestion class, it will output missing value. The code to derive the labels from the speed data is available to participants.
- Travel Time (**ETA**) for each super-segment. This is calculated using the the speeds in the segments in the super-segment weighted by their length. The code to derive the labels from the speed data is available to participants in our GitHub repository.

More details can be found in (Neun et al., 2023) and Appendix D.

1.2. Metrics

In order to give more weight to the less frequent but nonetheless important red class, we use weighted masked cross-entropy loss (PyTorch, 2022) on congestion classes. The class weights are derived through macro-averaging (scikit-learn, 2023) the labels in the training set for each city separately and published in our GitHub repository. Missing congestion class values are masked out. The overall score is computed as the average of the 3 city scores. The metric is formally defined in Appendix F.

We use L1 Loss for the extended competition as a simple domain-specific evaluation score known in the ETA literature (Elmasri, 2019; Hu, 2022; Derrow-Pinion et al., 2021; Schleibaum et al., 2022). Notice that in contrast to the core competitions, there is no "evaluation mask", *i. e.* all super-segments have an ETA label. As in the core competition, we take the average of the 3 city L1 losses. In addition, sampling super-segments from important road junctions results in oversampling areas with more data, potentially allowing for more accurate predictions than learning historic defaults in situations with temporally and spatially sparse data.

2. Standout Solutions

Traffic4cast 2022 saw over 80 submissions. Here, we briefly describe the top contributions.

2.1. ustc-gobbler: Transposed Variational Auto-Encoder and Graph-Attention Networks

Deng et al. (2022) use a variational auto-encoder to reconstruct missing vehicle count data for nodes without a stationary vehicle detector on the transposed matrix to get different values at different nodes. They engineered a lookup of exact time information from the stationary vehicle detector data as stationary vehicle detector data is publicly available. They embed static edge features to get weights of a GATv2 layer (Brody et al., 2021), which is applied on node embeddings and the reconstructed stationary vehicle detector data. In addition to the pairs of node features from node embedding and stationary vehicle detector data, embeddings of a k-means volume cluster, time information, and edge index are fed into final dense layers to produce the final congestion classes. In the extended competition, additionally, the super-segment-edge adjacency matrix is applied on the intermediate edge features before feeding into the final MLPs. They use the sum of reconstruction and cross-entropy (resp., L1 loss in the extended competition) as loss function.

2.2. Bolt: LightGBM leveraging PCA-based feature extraction

(Lumiste and Ilie, 2022) use PCA-based feature extraction to encode the global traffic state and as proxy for time information. In addition to PCA-based features, they use further city global features, feed target-encoded features, static edge/super-segment features and positional features into a GBM model both for the core and extended competition. They also use target encodings as score initialization. Global features dominate the predictions, closely followed by positional and target encodings.

2.3. TSE: LightGBM with Similarity-Based Feature Extraction

Wu et al. (2022) use a k-NN approach to derive super-segment features from the input vehicle count data in the extended competition. The similarity-based feature extraction is inspired by the assumption that vehicle count data are good encoders of traffic state. Multiple statistics and multiple neighbor sets are used. Manhattan distance is found to perform best. A Gaussian process-based approach is used to impute temporally missing data at stationary vehicle detector locations at two periods, both going into the similarity-based features. In addition to similarity-based features, static super-segment features (incl. historical features) and volume-based features from the underlying stationary vehicle detectors, as well as additional combined features are fed into a GBM.

2.4. oahciy: Two-Stage GBM (XGBoost and LightGBM)

Lu (2022) passes all 4 dynamic vehicle count data of all nodes to a first-stage XGBoost and LightGBM ensemble to extract month, day of the week, and time of day. Target encodings are computed at three levels: for time of day, weekday vs. weekend, and day of the week. The time information is passed to the second-stage XGBoost and LightGBM ensemble along with static features (incl. edge and node indices), dynamic features, and smoothed target encodings at all three levels. In the extended competition, the extracted time information is passed along with smoothed target encodings (the dynamic vehicle count information is not used at all). The target encoding features achieve the highest feature importance scores in both competitions; in the extended competition, the super-segment ID is particularly important as well.

2.5. GongLab: Multi-Task Learning GNN

Li et al. (2022) use embeddings of the dynamic vehicle count data as node features and embedding of static edge properties and a global volume-based target encoding as edge features to go into a GNN. After a final embedding of node features, edge labels are derived from the pair of neighboring node features. They not only derive the logits for the congestion classes but also use the speed labels provided in the competition and the volume class from the underlying GPS data in a multi-task setting. The loss function is a weighted sum of the three corresponding losses. The labels in the extended competition are derived by summing up the quotients of edge length and edge speed in the super-segments without additional training.

2.6. discovery: Hierarchical GNN

Grötschla and Mathys (2022) introduce additional nodes to reflect hierarchical graph structures: in the core competition, they add edges between supernodes (important road junctions that were used in the extended competition to derive super-segments), to support the information exchange between "important" intersections in dedicated interleaved layers; in the extended competition, additional super-segment nodes are added and connected to all the nodes of the corresponding super-segment, allowing for information pooling from all nodes lying on the same super-segment and then propagating it back to them in dedicated layers. In the core competition, congestion classes on edges are derived from the two neigboring node features. In the extended competition, they use the signal both from the congestion classes as well as the ETAs on super-segment nodes for training. Grötschla and Mathys (2022) use static edge features; in addition to the dynamic vehicle count data, node features contain positional embedding global mean and standard deviation of the vehicle counts. This combination allows to learn the global traffic state as well as to cover up for spatial and temporal sparsity.

2.7. ywei: Two-Stream GNN

Wei et al. (2022) base their solution on *LinkX* (Lim et al., 2021), first extracting node features from city topology (adjacency matrix) and from stationary vehicle detector data separately, and then combining the two to get node features from both sources. The node features of neighboring nodes are then concatenated and fed into an MLP to get the final congestion prediction on edges. Surprisingly, this approach does not use any feature engineering nor even static road information at all, only leveraging the road graph information and the dynamic stationary vehicle detector data as input. Using MLP-based embedding of node features and adjacency matrix, this approach is in principle able to learn global traffic state information as well as to cover up for spatial and temporal sparsity.

3. Synopsis and Discussion

Looking at the different solutions above, we see both Gradient Boosting Methods and GNN approaches. All participants used models trained on each city separately. We also see participants experiment with multi-task settings, trying to exploit feedback from both tasks, approaches avoiding feature engineering, and multi-level graphs, to name just a few. Table 2 highlights the key aspects and differences of the chosen architectures and informs the discussion below.

3.1. How Did Approaches Deal with Global State/Time Information?

As we were interested in extracting traffic state from the sparse vehicle detector data alone, we did not explicitly provide time information in the test set, whereas this kind of information would be available in a production setting. Many of the participants report the high relative importance of these features and found many different ways of retrieving such global state such as the time information, often combined with target encoding (see Table 2). (Deng et al., 2022) engineered a lookup of time information as stationary vehicle detector data is publicly available. (Lumiste and Ilie, 2022) used PCA to derive global features which identify traffic state well. (Wu et al., 2022) used kNN filters on stationary vehicle detector data to extract similarity-based edge features from the neighbor set. (Lu, 2022) used a first GBM to extract time information from the dynamic stationary vehicle detector data. (Li et al., 2022) used global volume clusters to look up target encodings. (Grötschla and Mathys, 2022) feed global volume means and standard deviations from the past hour into their model. (Wei et al., 2022) use an MLP to embed stationary vehicle detector data.

3.2. Dealing with Temporal and Spatial Sparsity in the Input Data

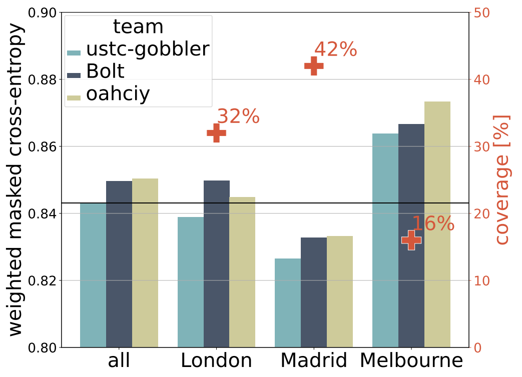
The input data was sparse, both temporally and spatially. Spatial sparsity means that only 6 %–8 % of nodes (see Table 1) in the road graph have stationary vehicle detector installed, and temporal sparsity means missing vehicle count values at stationary vehicle detector locations. The output labels still need to be predicted on the edges of the full graph. Temporal coverage varies significantly between cities and road types, see (Neun et al., 2023).

Team, rank (c./e.), approach	city spec.[a]	time information/ global state[b]	ensemble (p.city) (c./e.)[c]	avg. model size (c./e.)[d]	total model size (all cities) (c./e.)[e]	input dim edges (c./e.)[f]	input dim supersegments (c./e.) [g]	input dim nodes (c./e.)[h]	static road information[i]	target encoding (historic distribution)[j]	node index or position [k]	index or position (edges / supersegments) [l]	edge / supersegment label derivation[m]
ustc-gobbler (1/1) GNN (GATv2-Conv) (Deng et al., 2022)	yes	lookup	18.3 / 10.0	37.1M / 41.2M p.	2.0G / 1.2 G p.	10 / 8	–/ 1 + |E| + |V|	4/4	yes	no	yes	yes	MLP
Bolt (2/4) GBM (Lumiste and Ilie, 2022)	yes	PCA	1/1	23.5M / 3.41M n.	70.6M / 10.2M n.	38.0 / –	– /32.7	–	yes	yes	no	yes	–
TSE (6/2) GBM (Wu et al., 2022)	yes	kNN similarity	–/2.3	– / 249k n.	– / 1.74M n.	– / –	– /196.4	–	yes	yes	–	yes	–
oahciy (3/3) GBM (Lu, 2022)	yes	two-step GBM	1/1.7 (*)	18.6M / 2.78 M n.	55.7M / 13.9M n.	59.0 / –	– /11.7	–	yes	yes	yes	yes	–
GongLab (4/6) GNN (Li et al., 2022)	yes	global volume cluster	9=9	479M = 479M p.	12.9G = 12.9G p.	5=5	–	4=4	yes	yes (lookup based on volume cluster)	no	no	Link-Predictor / sum from predicted speeds
discovery (7/5) GNN (hierarchical) (Grötschla and Mathys, 2022)	yes	global volume mean/std (4 x 15 min)	1/1	6.8M / 6.8M p.	20.3M / 20.9M p.	14/14	–/–	4/4	yes	no	yes	no	Link-Predictor / pooling edges
ywei (7/–) GNN (LinkX) (Wei et al., 2022)	yes	MLP	1/–	15.4M / – p.	46.2M / – p.	|E|/–	–/–	4/–	no	no	no	no	Link-Predictor / –

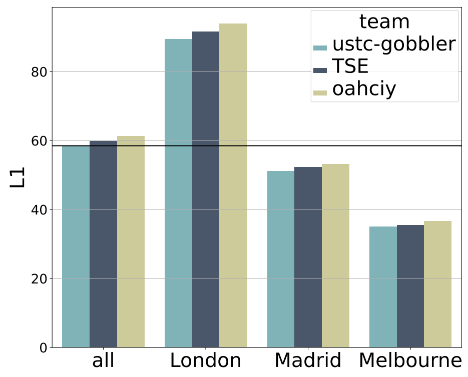
Table 2: Synopsis. [a] Was the model trained per city? [b] How does global state information enter the model? [c] How many trained models per city? [d] Average number of parameters (non-GBM) resp. number of nodes (GBM) per trained model. [e] Sum of number of parameters (non-GBM) resp. number of nodes (GBM) of all trained models of all cities. [f]/[g]/[h] number of features per edge/super-segment/node. [i] Were static road attributes? [j] Was target encoding used? [k]/[l]/[m] was the node/edge/super-segment index or position used as feature?

Tabular GBM approaches are a way to deal with spatial sparsity with appropriate features (Lumiste and Ilie, 2022; Wu et al., 2022; Lu, 2022). All three approaches use target encodings. On the other hand, many different techniques were used in GNN approaches: (Deng et al., 2022) uses TVAE and node embeddings, which can cover up for temporal and spatial sparsity; (Li et al., 2022) used global volume clusters to lookup target encodings; (Grötschla and Mathys, 2022) use node positional encoding and global features, which can cover up for temporal and spatial sparsity; (Wei et al., 2022) use an MLP to embed stationary vehicle detector data.

3.3. Where Was it Hard to Predict?



(a) Core competition



(b) Extended competition

Figure 3: Losses in the *Traffic4cast* 2022 in the leaderboard for the first three teams for core and extended competition: loss averaged over all cities and per city losses. The solid horizontal line shows the overall loss of the winner. The red pluses show the per-city label coverage in the core competition.

Figure 3(a) shows the overall and per-city losses for the first 3 participants in the core competition. We see the highest loss level in Melbourne and the lowest loss level in Madrid. The loss level shows an inverse correlation with the per-segment label coverage (London: 32%, Madrid: 42%, Melbourne: 16%) and a correlation with the imbalance of the label distribution. See Appendix E for more details.

The loss levels in the extended competition are shown in Figure 3(b). They show an inverse correlation with the narrowness of the label distribution – the Melbourne distribution with the highest peak and narrowest shape has the lowest loss level, whereas London with a flat peak and the fattest tail shows the highest loss level. See Appendix E.

3.4. Do Models Learn a Historic Distribution?

In Figure 4, we plot the edge-wise hourly historic empiric label distribution (not taking into account day of week, only the hour of day) against the re-weighted model outputs (submission by oahciy (Lu, 2022), whole test set for London). We see a strong correlation between the historic distribution and the predicted distribution after re-weighting. Details on the re-weighting and plots for other cities have a similar shape, see Appendix H.

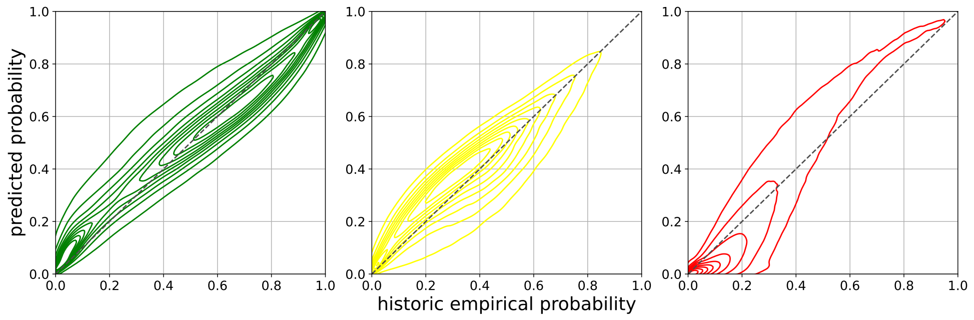


Figure 4: Probabilites from re-weighted logits vs. historic ground-truth distribution, submission by (Lu, 2022), test set for London for 14:00–18:00. The historic empiric distribution is computed on the training data.

4. Summary and Outlook

The *Traffic4cast* Traffic Map Movie datasets offer a unique source of massive floating car data, unprecedented in its scale and availability across different metropolitan areas. The real-world data has been provided by HERE Technologies and the spatio-temporal aggregation is privacy-preserving. This has already enabled successful short-term traffic predictions (Kreil et al., 2020; Kopp et al., 2021) as well as a demonstration of transfer learning in this context (Eichenberger et al., 2022).

This year *Traffic4cast*, for the first time, addressed directly application relevant challenges: taking as input only sparse traffic count data, can we for the whole city predict congestion classes, as we know them from the red-yellow-green maps in navigation apps, or even longer segment travel time (ETA)? In this new setup, the *Traffic4cast* traffic map movies provided the prediction labels by merging the aggregated speed-readings with a full road-graph. As a result also a general traffic segment speed graph dataset has already been made available for 10 cities (Neun et al., 2023), and we see this development as a first step towards a traffic graph benchmark dataset in the ML domain.

The recurring nature of general traffic patterns allows for good prediction of common situations but special attention and treatment of rarer or more anomalous congestion events is also needed (Eichenberger et al., 2022). We therefore used class weighting and super-segment sampling as discussed in Section 1.2.

The spatial sparsity of the stationary vehicle detectors and their data being associated with nodes (and thus undirected) obviously limits the amount of local information that can be extracted. Nevertheless, just from the counts input, the models seem to capture the historic distribution well. Interestingly, this made the task amenable to both GBM and GNN. Furthermore, a more imbalanced label distribution and lower coverage made the core competition more difficult in Melbourne than in London and in Madrid.

It would be interesting to use further input signals and even to reverse the task and predict the ground-truth like stationary vehicle detector information from the more fine-grained GPS data. In a similar vein, this could also inform better placement of vehicle detectors for an improved detection of the global traffic state.

Acknowledgments

We would like to thank HERE technologies for making our competition data available.

Author Contributions Statement

Following CRediT (Contributor Roles Taxonomy[*]), the authors have contributed as follows. *Writing – original draft, Software*: M.N., Ch.E. *Data Curation*: M.N., Ch.E, K.M. *Software (baselines)*: M.N., Ch.E., H.M., M.S., R.S., D.S. *Methodology (competition)*: L.D., C.W, D.L., M.Z., M.L., A.I., X.W., C.L., Q.L., V.M., Y.L., J.L, J.L., Y.G., F.G., J.M., Y.W., H.H., H.F. *Conceptualization*: M.N., Ch.E, F.T., M.K., D.K, S.H. *Writing – review & editing*: all.

References

Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Computers, Environment and Urban Systems, 65:126–139, 2017. ISSN 0198-9715. doi: https://doi.org/10.1016/j.compenvurbsys.2017.05.004. URL https://www.sciencedirect.com/science/article/pii/S0198971516303970.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2021. URL https://arxiv.org/abs/2105.14491.

Leyan Deng, Chenwang Wu, Defu Lian, and Min Zhou. Transposed variational auto-encoder with intrinsic feature learning for traffic forecasting, 2022. URL https://arxiv.org/abs/2211.00641.

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Veličković. ETA Prediction with Graph Neural Networks in Google Maps. Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pages 3767–3776, October 2021. doi: 10.1145/3459637.3481916. URL http://arxiv.org/abs/2108.11482. arXiv: 2108.11482.

Christian Eichenberger, Moritz Neun, Henry Martin, Pedro Herruzo, Markus Spanring, Yichao Lu, Sungbin Choi, Vsevolod Konyakhin, Nina Lukashina, Aleksei Shpilman, Nina Wiedemann, Martin Raubal, Bo Wang, Hai L. Vu, Reza Mohajerpoor, Inhi Kim, Luca Hermes, Andrew Melnik, Riza Velioglu, Markus Vieth, Malte Schilling, Alabi Bojesomo, Hasan Al Marzouqi, Panos Liatsis, Jay Santokhi, Dylan Hillier, Yiming Yang, Joned Sarwar, Anna Jordan, Emil Hewage, David Jonietz, Fei Tang, Aleksandra Gruca, Michael Kopp, David Kreil, and Sepp Hochreiter. Traffic4cast at neurips 2021 – temporal and spatial few-shot transfer learning in gridded geo-spatial processes. In Hugo Jair Escalante and Katja Hofmann, editors, Proceedings of the NeurIPS 2021 Competition Track, volume forthcoming of Proceedings of Machine Learning Research. PMLR, 2022.

[*]https://credit.niso.org/

Mohamad Elmasri. Beyond L2 Loss — How we experiment with loss functions at Lyft, December 2019. URL https://eng.lyft.com/beyond-l2-loss-how-we-experiment-with-loss-functions-at-lyft-51f9303f5d2d.

Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In Catherine C. McGeoch, editor, Experimental Algorithms, pages 319–333, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-68552-4.

Florian Grötschla and Joël Mathys. Hierarchical graph structures for congestion and eta prediction, 2022. URL https://arxiv.org/abs/2211.11762.

HE. Highways england WebTRIS., 2022. URL http://webtris.highwaysengland.co.uk/.

HERE Technologies. Sample data | here developer, September 2020. URL https://developer.here.com/sample-data. Last accessed 11 October 2022.

Xinyu Hu. DeepETA: How Uber Predicts Arrival Times Using Deep Learning, February 2022. URL https://eng.uber.com/deepeta-how-uber-predicts-arrival-times/.

Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. Examining covid-19 forecasting using spatio-temporal graph neural networks, 2020. URL https://arxiv.org/abs/2007.03113.

Michael Kopp, David Kreil, Moritz Neun, David Jonietz, Henry Martin, Pedro Herruzo, Aleksandra Gruca, Ali Soleymani, Fanyou Wu, Yang Liu, Jingwei Xu, Jianjin Zhang, Jay Santokhi, Alabi Bojesomo, Hasan Al Marzouqi, Panos Liatsis, Pak Hay Kwok, Qi Qi, and Sepp Hochreiter. Traffic4cast at neurips 2020 – yet more on the unreasonable effectiveness of gridded geo-spatial processes. In Hugo Jair Escalante and Katja Hofmann, editors, Proceedings of the NeurIPS 2020 Competition and Demonstration Track, volume 133 of Proceedings of Machine Learning Research, pages 325–343. PMLR, 2021. URL http://proceedings.mlr.press/v133/kopp21a.html.

David P Kreil, Michael K Kopp, David Jonietz, Moritz Neun, Aleksandra Gruca, Pedro Herruzo, Henry Martin, Ali Soleymani, and Sepp Hochreiter. The surprising efficiency of framing geo-spatial time series forecasting as a video prediction task – insights from the iarai *Traffic4cast* competition at neurips 2019. In Hugo Jair Escalante and Raia Hadsell, editors, Proceedings of the NeurIPS 2019 Competition and Demonstration Track, volume 123 of Proceedings of Machine Learning Research, pages 232–241. PMLR, 08–14 Dec 2020. URL http://proceedings.mlr.press/v123/kreil20a.html.

Jiezhang Li, Junjun Li, and Yue-Jiao Gong. Multi-task learning for sparse traffic forecasting, 2022. URL https://arxiv.org/abs/2211.09984.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods, 2021. URL https://arxiv.org/abs/2110.14446.

Yichao Lu. An efficient two-stage gradient boosting framework for short-term traffic state estimation, 2022. URL https://arxiv.org/abs/2302.10400.

Martin Lumiste and Andrei Ilie. Large scale traffic forecasting with gradient boosting, traffic4cast 2022 challenge, 2022. URL https://arxiv.org/abs/2211.00157.

Madrid. Portal de datos abiertos del ayuntamiento de madrid. tráfico. histórico de datos del tráfico desde 2013., 2022. URL https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=33cb30c367e78410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default.

Tanwi Mallick, Mariam Kiran, Bashir Mohammed, and Prasanna Balaprakash. Dynamic graph neural network for traffic forecasting in wide area networks. In 2020 IEEE International Conference on Big Data (Big Data), pages 1–10, 2020. doi: 10.1109/BigData50022.2020.9512748.

Eric L. Manibardo, I. Laña, and J. Ser. Deep Learning for Road Traffic Forecasting: Does it Make a Difference? ArXiv, 2020. doi: 10.1109/TITS.2021.3083957.

Moritz Neun, Christian Eichenberger, Yanan Xin, Cheng Fu, Nina Wiedemann, Henry Martin, Martin Tomko, Lukas Ambühl, Luca Hermes, and Michael Kopp. Metropolitan segment traffic speeds from massive floating car data in 10 cities, 2023. URL https://arxiv.org/abs/2302.08761.

OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2022.

Sophia Parafina. Faqs – uber movement: Let's find smarter ways forward, together, 2022. URL https://developer.here.com/blog/mapping-traffic-congestion. Last accessed 22 February 2023.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs, 2019. URL https://arxiv.org/abs/1902.10191.

Diwas Poudel. What are different color in google maps ?, 2022. URL https://ourtechroom.com/tech/color-in-google-maps/. Last accessed 22 February 2023.

PyTorch. Pytorch docs torch.nn.crossentropyloss, 2022. URL https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. Last accessed 27 February 2023.

Margarida Sardo, Sophie Laggan, Laura Fogg-Rogers, Elke Franchois, and Anke Bracke. WeCount Short Evaluation Summary: Citizen Science on urban mobility, March 2022. URL https://doi.org/10.5281/zenodo.6337283.

Sören Schleibaum, Jörg P. Müller, and Monika Sester. An Explainable Stacked Ensemble Model for Static Route-Free Estimation of Time of Arrival. arXiv:2203.09438 [cs, stat], March 2022. URL http://arxiv.org/abs/2203.09438. arXiv: 2203.09438 version: 1.

scikit-learn. Metrics and scoring: quantifying the quality of predictions, 2023. URL https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification. Last accessed 27 February 2023.

Joshua Stevens. Your favorite traffic map is lying to you, 2013. URL https://www.joshuastevens.net/design/your-favorite-traffic-map-is-lying-to-you/. Last accessed 22 February 2023.

TfL. Public TfL data (or 'open data'). TIMS., 2022. URL https://roads.data.tfl.gov.uk/.

Uber Technologies. Faqs – uber movement: Let's find smarter ways forward, together, 2023. URL https://movement.uber.com/faqs?lang=en-US. Last accessed 11 October 2022.

Kris Vanherle. Telraam sensor dataset api - www.telraam-api.net, August 2021. URL https://doi.org/10.5281/zenodo.5196100.

VicGov. Victorian government open data. traffic signal configuration data sheets., 2022a. URL https://discover.data.vic.gov.au/dataset/traffic-signal-configuration-data-sheets.

VicGov. Victorian government open data. traffic signal volume data., 2022b. URL https://discover.data.vic.gov.au/dataset/traffic-signal-volume-data.

Ye Wei, He Haitao, and Hui Fang. Spatial-temporal city-scale congestion prediction using a two-stream graph neural network, 2022. URL https://github.com/Ye-We1/Traffic4cast2022/.

Wikipedia. Induction loop. https://en.wikipedia.org/wiki/Induction_loop, 2023. Accessed 2023-03-021.

Xinhua Wu, Cheng Lyu, Qing-Long Lu, and Mahajan Vishal. Similarity-based feature extraction for large-scale sparse traffic forecasting, Oct 2022. URL https://arxiv.org/abs/2211.07031.

Zhao Zhang, Ding Zhao, and Xianfeng Terry Yang. A Hybrid Physics Machine Learning Approach for Macroscopic Traffic State Estimation. arXiv:2202.01888 [cs, eess], February 2022a. URL http://arxiv.org/abs/2202.01888. arXiv: 2202.01888.

Zikai Zhang, Yidong Li, Wei Wang, Haifeng Song, and Hairong Dong. Malware detection with dynamic evolving graph convolutional networks. International Journal of Intelligent Systems, n/a(n/a), 2022b. doi: https://doi.org/10.1002/int.22880. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/int.22880.

Appendix A. More Details on Stationary Vehicle Detector Data

We provide normalized counts with id, lat, lon, heading, day, 96 volume counts per day, using NaN for no data. Madrid and Melbourne are providing 15 min aggregates. We are normalizing the 5 min sliding windows in London to the same 15 min time bins for convenience and consistency. The data in Madrid contains the heading for each loop counter, whereas for London and Melbourne this is not readily available and could be derived from the associated road segment via parsing street designation; hence, we decide to assign the counter data to nodes without using the heading information.

Snapping As seen in Table 1, multiple detectors can be close to the same road graph node. In this case, their counts are added up. The reference to the city-specific counter IDs is tracked in the list in the `counter_info` attribute for road graph nodes.

Splitting For Melbourne, each detector can be composed of up to 24 individual detectors as detailed in (VicGov, 2022a). Unfortunately, these documents are not readily parsable. Hence, we take the sum of the individual detector readings and use only the common prefix as ID. This can lead to the situation that we need to split the detector value among multiple road graph nodes number of road graph nodes at large junctions in Melbourne. The number of nodes the value is split into is tracked in the `num_assigned` attribute.

The data specification can be found in our competition GitHub repo, and code for downloading and normalizing vehicle detector data can be found in the GitHub repo for (Neun et al., 2023).

Appendix B. More Details on Road Graph

We use the road graph derivation `OSMnx` (Boeing, 2017) as described in (Neun et al., 2023) with the following additions:

- **introduce nodes for vehicle detector locations** We assign the counter to the nearest node in the road graph if the node is closer than 40 m to the counter location. Else we project the counter location to the nearest edge (discarding the counter if the distance to the nearest edge is greater than 20 m); then, we split the nearest edge if the projection point is not close enough to either the start or end point of the edge. When an edge is split, attributes like length, travel times and geometry need to split proportionally as well. Note that an edge in the original OpenStreetMap road graph may be split multiple times in this approach.

- **filter out road segments with low volume** We use a $495 \times 436 \times 4$ volume heatmap per cell and heading; the heatmap is created as average of daily volumes from 30 randomly selected days in the training data.

- **remove unconnected components and dead-ends from the graph**

  - **clean edges with no access** We remove edges with restricted access, such as ‘no’, ‘private’, ‘official’, ‘permit’, ‘delivery’, ‘designated’, ‘emergency’

- **clean edges with low volume** We then remove residential and unclassified edges with low volume (highest daily volume as per heatmap below 10 among all intersecting cells, of both directions if not one-way). Here, in order to avoid deleting *e. g.* roundabouts upfront, we do not delete edges shorter than 50 m and keep nodes with a vehicle counter.
- **clean dead end edges** We then delete dead-end edges.
- **clean isolates** and remove isolates (*i. e.* nodes without neighbors).
- **clean self loops** + **clean isolates** + **clean dead end edges** Furthermore, we filter self-loops shorter than 300 m. As this can introduce new isolates and dead end edges, we remove those newly introduced.
- **clean no neighbors**
- **clean sub graphs** + **clean dead end edges** We keep only the largest connected component of the graph.
- **clean circle ramps** + **clean end circles** + **clean isolates** We remove clean short circle ramps (*i. e.* one node only connected to two other nodes which already have a direct connection).
- **clean multi edges** Finally, in order to avoid multi-edges in the competition, we split all multi-edges but the shortest one.

Appendix C. More Details on Super-Segment Sampling

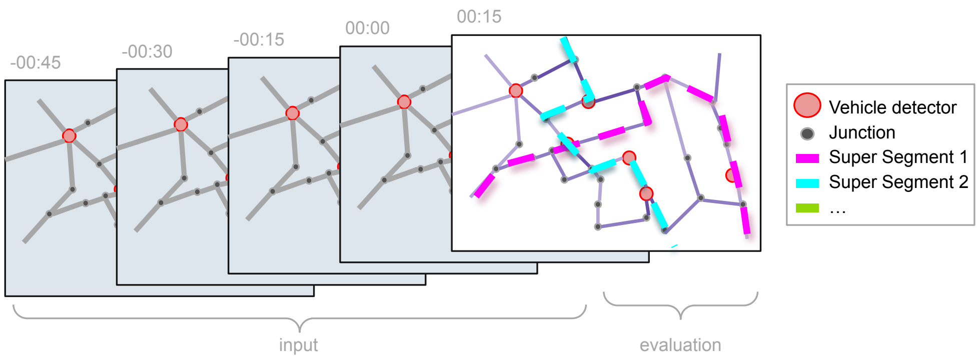


Figure 5: *Traffic4cast* 2022 Extended Task. Given one hour of sparse vehicle count data only (red disks) the task is to predict the average travel times along sampled super-segments in the city road-graph.

For the extended task (see Figure 5) we are providing a choice of super-segments (Derrow-Pinion et al., 2021) represented as lists of segment IDs. The super-segments for *Traffic4cast* 2022 are sampled from the full road graph with the aim to capture corridors used by most typical routes on the road graph. By construction, there is at most one super-segment between any pair of nodes.

We start by choosing 400 **connected key intersections**, which are those nodes with the highest maximum daily volume on any edge connected to it (incoming or outgoing), weighted by OSM importance, and which have at least 3 neighbor nodes in the graph. We then go through this candidate list, removing neighbors of chosen nodes from the candidate list. For each city, there is also a hand-curated whitelist of nodes which is added to this candidate list.

From these filtered connected key intersections, we first derive a list of **super-segments** for each of these key intersections. For each connected key intersection, we look at the other connected key intersections in increasing circles defined by radius of beeline distance, length of the shortest path in meters and number of segments in the super-segment candidate. Key intersections reached will not be considered again for the same source key intersection.

For the shortest path, Dijkstra algorithm is used with edge weights linear in the length of the edge (in meters) and inversely proportional to the edge importance: $weight(ed) = ((6.0 - importance(ed))/2) * length(ed)$, penalizing longer and unimportant edges. The search for super-segments from a connected key intersection will be aborted once a super-segment further away than 10 000 m is found or more than 3 super-segments have been found for a source and the search will be aborted after the 4th search circle.

For each city, there is also a hand-curated whitelist of super-segments which is added to the list of sampled super-segments.

Appendix D. More Details on Derivation of Output Labels

We refer to (Neun et al., 2023) for the derivation of dynamic edge speeds and edge free flow speeds. Here, we only describe the derivation of labels from there on.

D.1. Congestion Classes

To the best of our knowledge, there are only informal descriptions of how tech companies derive the congestion classes in their online maps, *e. g.* (Poudel, 2022; Uber Technologies, 2023; Stevens, 2013). Hence, we choose one of them, namely (Parafina, 2022), and derive the congestion classes in the following way:

```
def extract_cc(segment_speed_stats, t, freeflow_speed_kph):
    median_speed = segment_speed_stats.median_speeds[t]
    if median_speed == 0:
        return 0
    if median_speed == 255:
        return 0
    assert(freeflow_speed_kph > 0)
    probe_volume = segment_speed_stats.volumes[t]
    median_speed_kph = segment_speed_stats.median_speeds_kph[t]
    congestion_factor = median_speed_kph / freeflow_speed_kph
    if congestion_factor < 0.4 and probe_volume >= 5:
        return 3
    elif congestion_factor >= 0.4 and \
         congestion_factor < 0.8 and probe_volume >= 3:
        return 2
    elif congestion_factor >= 0.8 and probe_volume > 0:
        return 1
    else:
        return 0
```

The congestion factor reflects the percentage of free flow speed. If the median speed in the data is 0 or 255, we do not classify it as this hints at data corruption. We set 0.4 and 0.8 as boundaries to classify for congestion (red, 3), reduced (yellow,2), and uncongested (green,1). In addition, we impose volume limits for the classification, requiring more evidence to classify as red than for yellow. For US highways (speed limit 65 mph), green is moving at least 50 mph (congestion level 80%), orange approximately 25–50 miles per hour (congestion level 40–80%), red below 25 (congestion level 40%).

D.2. Super-Segment ETAs

ETAs are derived for all super-segments with a two-level defaulting mechanism: for each edge ed in the super-segment, a speed $v(ed)$ is derived, taking the current median speed from the dynamic data as first priority, free flow speed as second priority and signalled maxspeed as third priority. The speed is clipped below at $0.5\frac{km}{h}$. If ETA is longer than the 15 min slot and half of both neighbor time slots, use 30 minutes plus the mean speed of all three slots. The eta for super-segment with $edges$ is then derived by $\sum_{ed \in edges} \frac{length(ed)}{v(ed)}$:

```
# prec-computed for whole day
def compute_edge_speeds_for_one_day(...):
    for uv, maxspeed in edge_maxspeeds_kph.items():
        if uv in edge_free_flows_kph:
            free_flow = edge_free_flows_kph[uv]
            speeds = [free_flow for _ in range(96)]
            sources = ["free_flow" for _ in range(96)]
        else:
            speeds = [maxspeed for _ in range(96)]
            sources = ["maxspeed" for _ in range(96)]
            maxspeed_cnt += 1
        edge_speeds[uv] = {"speeds": speeds, "sources": sources}
    print(f"{maxspeed_cnt} / {len(edge_speeds)} edges only have maxspeed")
    for t in range(0, 96):
        tsc_df = sc_df[sc_df["t"] == t]
        for u, v, ms in zip(tsc_df["u"],
                            tsc_df["v"],
                            tsc_df["median_speed_kph"]):
            esp = edge_speeds[(u, v)]
            esp["speeds"][t] = ms
            esp["sources"][t] = "current"


def compute_eta(edges, edge_speeds, t):
    path_eta_s = 0

    for ed in edges:
        e = ed["edge"]
        esp = edge_speeds[e]
        edge_speed_kph = esp["speeds"][t]
        speed_used = esp["sources"][t]

        length_m = ed["length"]

        if edge_speed_kph < 0.5:
            edge_speed_kph = 0.5
        edge_speed_mps = edge_speed_kph / 3.6
        edge_eta_s = length_m / edge_speed_mps

        if edge_eta_s > 1800:
            # If ETA longer than the 15 min slot and half of
            # both neighbor time slots,
            # use 30 minutes plus the mean speed of all three slots.
            speeds = esp["speeds"][max(t - 1, 0) : t + 2]
            sources = esp["sources"][max(t - 1, 0) : t + 2]
            edge_speed_kph = mean(speeds)
            edge_speed_mps = edge_speed_kph / 3.6
            edge_eta_s = 1800 + (length_m / edge_speed_mps)

        path_eta_s += edge_eta_s

    return path_eta_s
```

Note that, upon congestion, there will be a higher chance of having data going into current median speeds, and in those cases the second and third line will be appropriate; the clipping is necessary since, in case of congestion, the edge speed can get very low as slow or standing vehicles will emit many more data points along the same edge. Furthermore, in a city environment (in contrast to a motorway locked down by a traffic), a full traffic breakdown is also less likely; as the ETAs are computed for full 15 min intervals, the ETA reflects on an expectation.

Such a heuristic approach is also behind industrial ETA routing machines – all companies basing their ETA products on floating car data will not have a full view of traffic and will have default values.

Appendix E. Ground Truth Label Distribution Core and Extended Competition

E.1. Core competition label and coverage distribution

Figure 6 shows that the labels are most equally distributed in Madrid, and most imbalanced in Melbourne. Figure 3(a) shows that the the per-segment label coverage is highest in Madrid and lowest in Melbourne. In addition, Figure 7 shows that Madrid has many segmenst with a very high coverage, whereas in Melbourne there are almost none with a very high coverage.

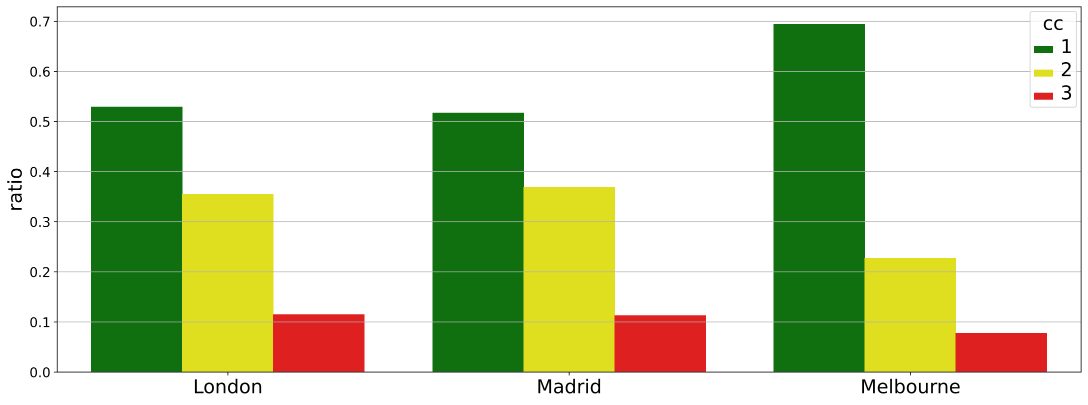


Figure 6: Label distribution in the test set of the core competition for the three cities London, Madrid, Melbourne.

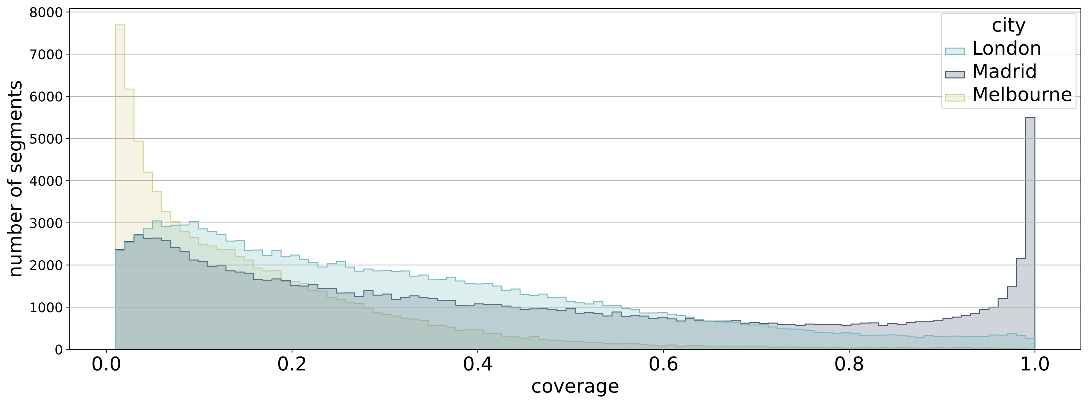


Figure 7: Distribution of per-segment label coverage in the test set of the core competition for the three cities London, Madrid, Melbourne.

E.2. Extended competition label distribution

Figure 8 shows that London has the longest tail in super-segment ETAs, and Melbourne has the most narrow distribution.

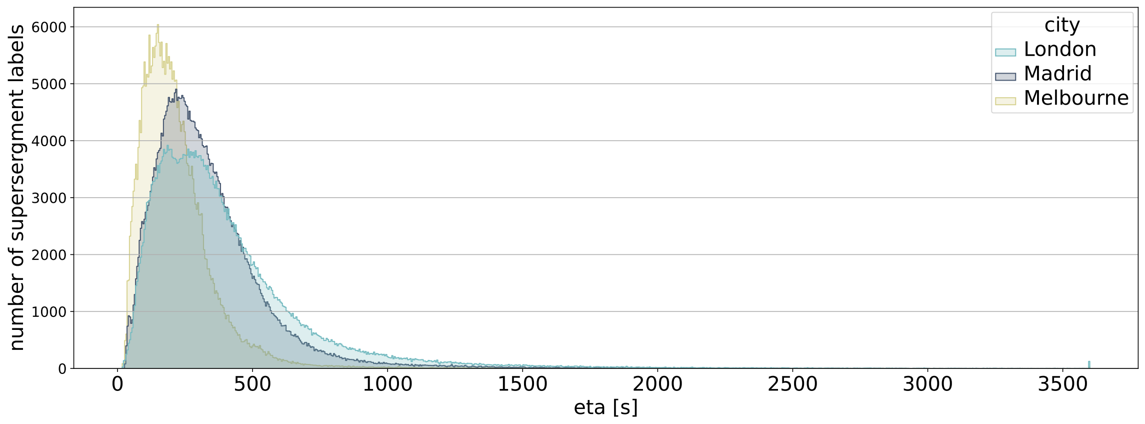


Figure 8: Histogram of super-segment labels in the test set of the extended competition for the three cities London, Madrid, Melbourne.

Appendix F. Notation Weighted Masked Cross-Entropy Loss

Following the notation of (PyTorch, 2022),

$$\ell(\hat{y}, y) = \frac{1}{k} \cdot \sum_{n=1}^{N} l_n, \quad l_n = -w_{y_n} \log \frac{\exp(\hat{y}_{n,y_n})}{\Sigma_{c=1}^{C} \exp(\hat{y}_{n,c})} \cdot \mathbb{1}\{y_n \neq \kappa\}, \quad k = \sum_{n=1}^{N} w_{y_n} \cdot \mathbb{1}\{y_n \neq \kappa\} \tag{1}$$

where $\hat{y} \in \mathbb{R}^{N \times C}$ are the input logits, $y \in \{0, ..., C\}^N$ is the target, $w \in \mathbb{R}^C$ is the class weight, $C \in \mathbb{N}$ is the number of classes to be predicted, $N \in \mathbb{N}$ is the number of samples, κ specifies a target class that is ignored (*i. e.* models must not predict no data). Cross-entropy loss will penalize small predicted probabilities disproportionately.

In our setting, $C = 4$ and $\kappa = 0$, *i. e.* , we always mask on unclassified edges in the ground truth (0 = unclassified, 1 = green (uncongested), 2 = yellow (slowed down/warning), 3 = red (congested)), N goes over edges and timestamps.

In problems where infrequent classes are nonetheless important, macro-averaging may be a means of highlighting their performance (scikit-learn, 2023). In our case, since we have more red than yellow than green in all cities and the congested situations are particularly important. We compute the macro-averaged class weights computed on the $\tilde{N}$ training labels $\tilde{y}$,

$$w_c = \frac{\tilde{N}}{|C| \cdot \Sigma_{n=1}^{\tilde{N}} \mathbb{1}\{\tilde{y}_n = c\}}, \quad c \in C - \{\kappa\}, \tag{2}$$

giving equal total weight to each class as can be seen from the decomposition

$$k = \sum_c N_c \cdot w_c = \sum_c \frac{N_c \cdot \tilde{N}}{|C| \cdot \tilde{N}_c} \approx \sum_c \frac{N_c \cdot N}{|C| \cdot N_c} = \sum_c \frac{N}{|C|}$$

if we assume assume the label fractions to be almost the same in the test and training set, *i. e.* if we assume $\frac{N_c}{N} \approx \frac{\tilde{N}_c}{\tilde{N}}$, where $N_c = \sum_{i=1}^{N} \mathbb{1}\{y_i = c\}$ the number of labels of class c in the test set and where $\tilde{N}_c = \sum_{i=1}^{\tilde{N}} \mathbb{1}\{\tilde{y}_i = c\}$ is the number of labels of class c in the training labels $\tilde{y}$ from which the weights w_c were computed.

Appendix G. Loss Distribution Core per Ground-Truth Label

Figure 9 shows that predicting green and yellow was similarly difficult in all three cities, whereas predicting red was harder in Melbourne and easiest in Madrid. This is strongly inversely correlated with the label distribution of Figure 6. The city-wise mean and summed losses per ground truth class c were computed as follows. Filtering out the ignored class in Equation (1), we have

$$k = \sum_{n=1}^{N} w_{y_n} = \sum_c N_c \cdot w_c = \sum_c N_c \frac{\tilde{N}}{|C| \cdot \tilde{N}_c} \approx N.$$

Then, we consider

$$\mathit{summedloss}(c) = \frac{1}{k} \cdot \sum_n l_n \cdot \mathbb{1}\{y_n = c\}, \quad \mathit{meanloss}(c) = \frac{1}{N_c} \sum_n l_n \cdot \mathbb{1}\{y_n = c\}.$$

The mean losses of Figure 10 reflect the summed losses scaled by the class weights as we have

$$meanloss(c)/(w_c{\cdot}|C|) = \frac{|C| \cdot N_c}{N_c \cdot N \cdot |C|} {\cdot} \sum_n l_n {\cdot} \mathbb{1}\{y_n = c\} \approx \frac{1}{k} {\cdot} \sum_n l_n {\cdot} \mathbb{1}\{y_n = c\} = summedloss(c).$$

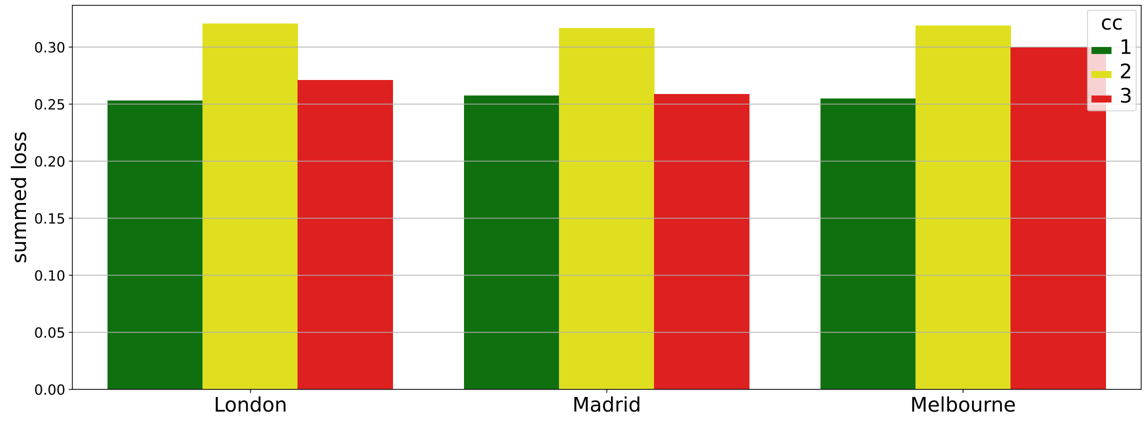


Figure 9: Summed loss per ground-truth in the test set of the core competition for the three cities London, Madrid, Melbourne for team oahciy. The sum of the green, yellow and red bar per city gives the per-city losses of Figure 3(a).

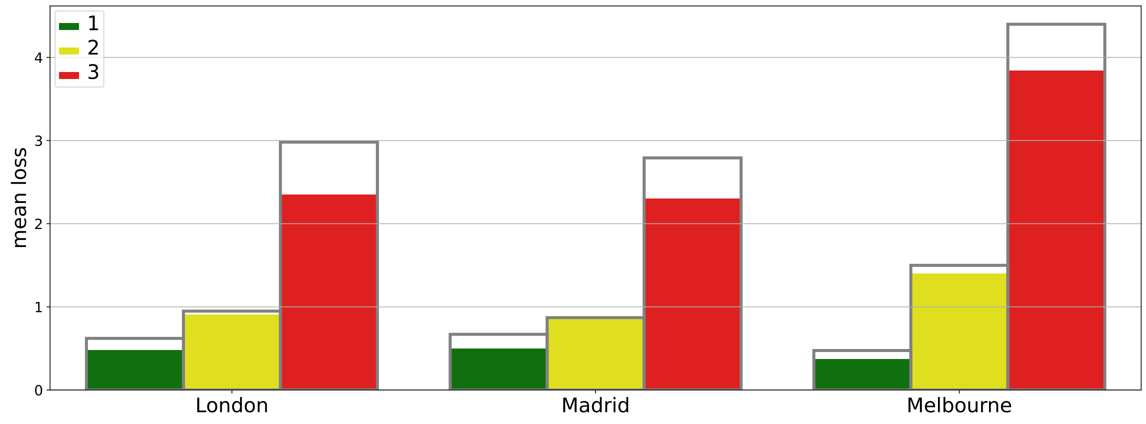


Figure 10: Mean loss per ground-truth in the test set of the core competition for the three cities London, Madrid, Melbourne for team oahciy. The sum of the green, yellow and red bar per city gives the per-city losses of Figure 3(a). The grey boxes reflect the class weights.

Figure 11 shows losses binned by per-segment coverage. The mean loss curve for Madrid is almost monotone decreasing, whereas London sees a slightly elevated plateau around 70% coverage and Melbourne sees a more accentuated plateau around in the are 50–60%; the mean loss curve reflects to traffic mix in those streets. Due to the steep decrease in mean loss in Madrid for higher coverage, the high number of data points for high coverage still results in an overall loss as seen in the cumulative plot.

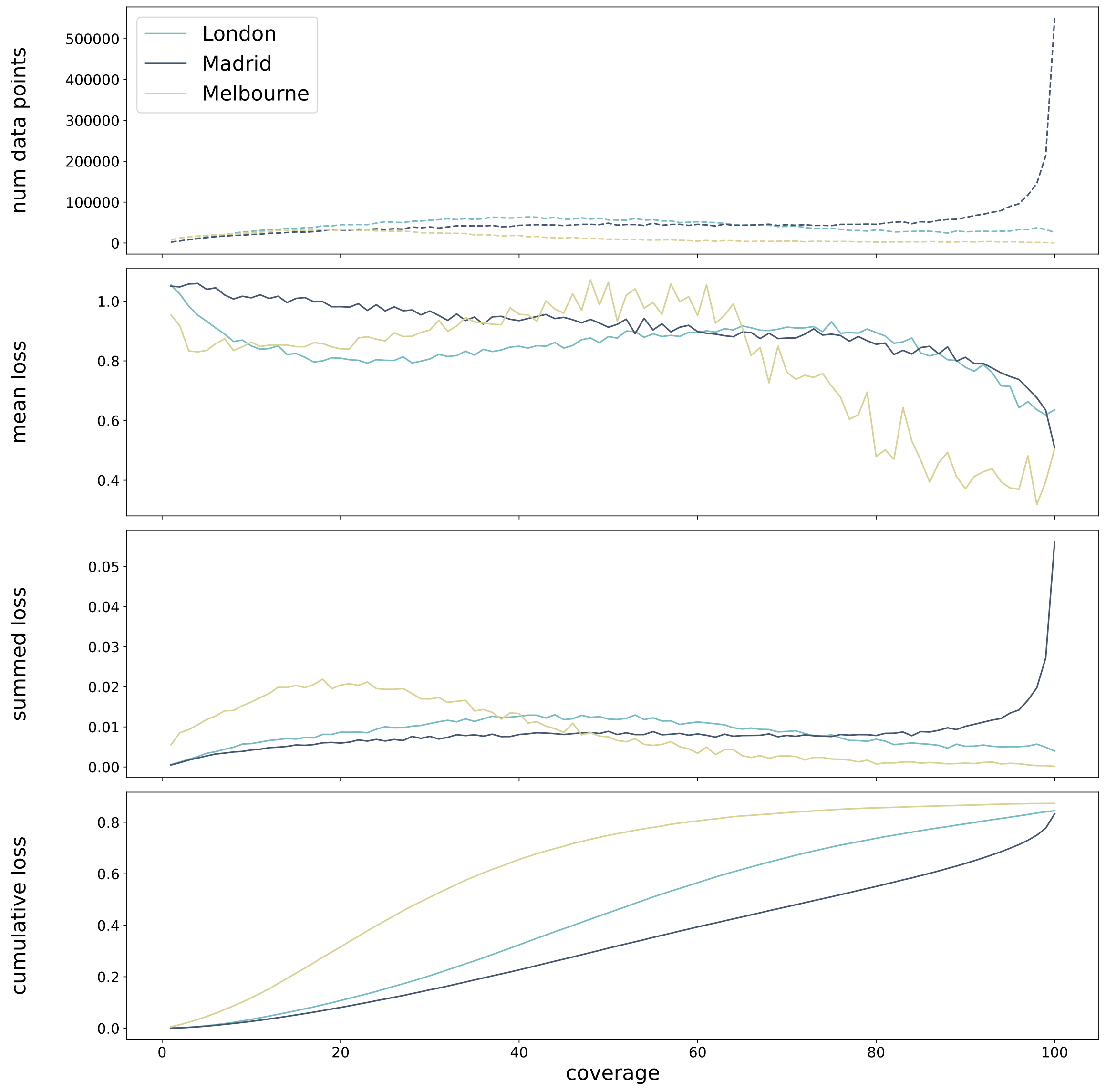


Figure 11: Number of data points and losses per coverage (mean per data point, summed per coverage, and cumulative).

Appendix H. More on Capturing the Historic Distribution

Here, we detail on the comparison of model outputs with the hourly, segment-wise historic distribution as discussed in Section 3.4 The per-city class weights from Section F will drive the models to overestimate the rare classes (red and yellow) and underestimate the non-rare classes (green), as shown by the plots without re-weighting. The re-weighted probabilities are derived as

$$\hat{p}_{n,c} = b_n \cdot \exp(\hat{y}_{n,c} \cdot w_c) \tag{3}$$

from the model outputs $\hat{y}_{n,c}$ and normalization factor b_n.

H.1. London

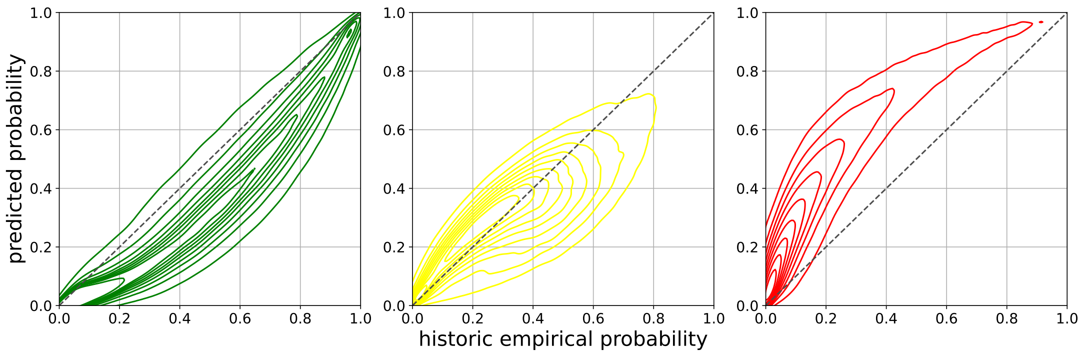


Figure 12: Probabilites from logits without re-weighting vs. historic ground-truth distribution, submission by (Lu, 2022), test set for London for 14:00–18:00. The historic empiric distribution is computed on the training data.

‘

H.2. Madrid

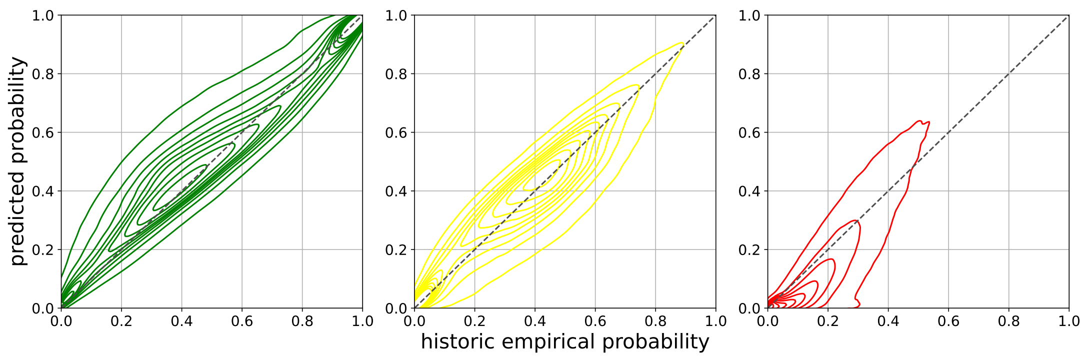


Figure 13: Probabilites from re-weighted logits vs. historic ground-truth distribution, submission by (Lu, 2022), test set for Madrid for 14:00–18:00. The historic empiric distribution is computed on the training data.

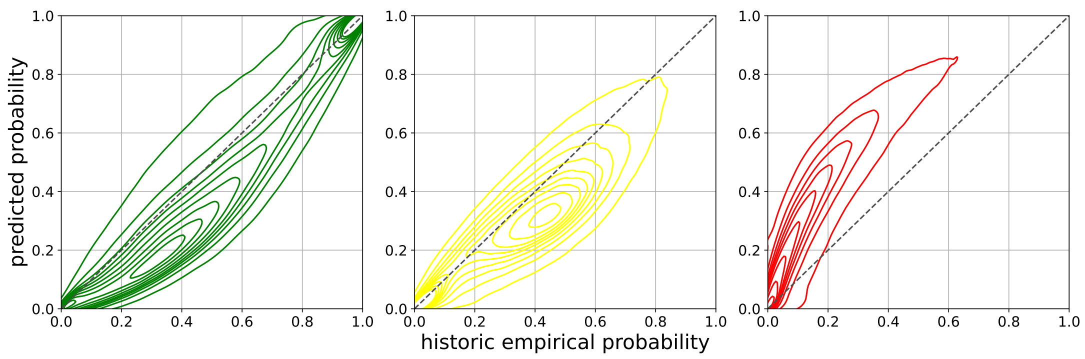


Figure 14: Probabilites from logits without re-weighting vs. historic ground-truth distribution, submission by (Lu, 2022), test set for Madrid for 14:00–18:00. The historic empiric distribution is computed on the training data.

‘

H.3. Melbourne

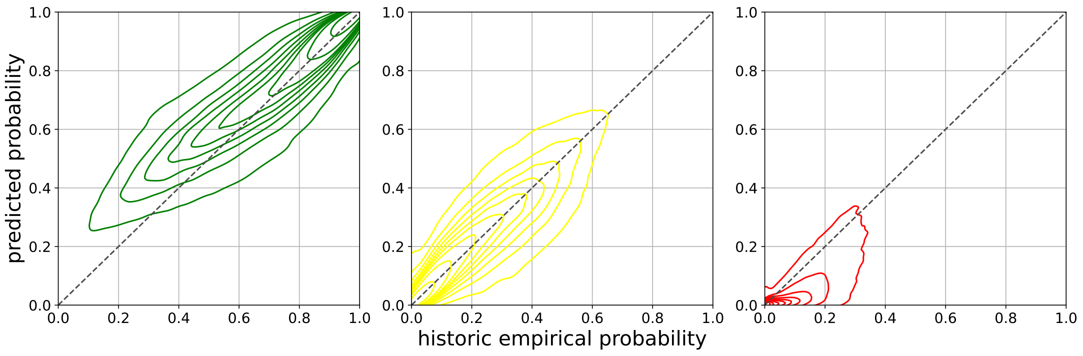


Figure 15: Probabilites from re-weighted logits vs. historic ground-truth distribution, submission by (Lu, 2022), test set for Melbourne for 14:00–18:00. The historic empiric distribution is computed on the training data.

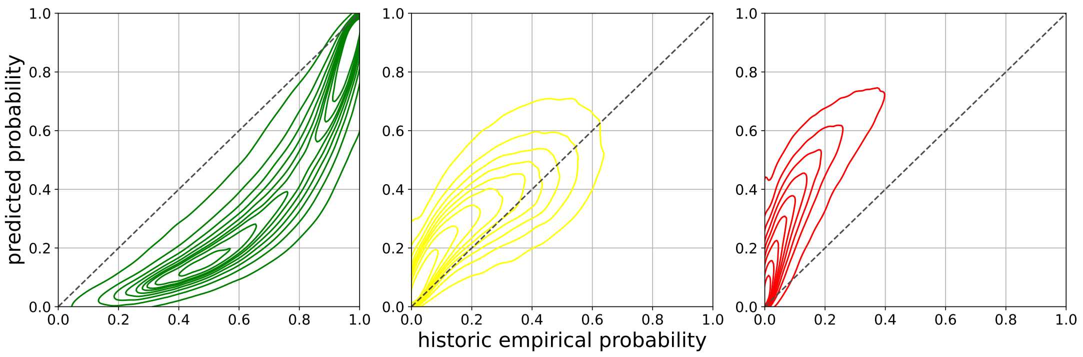


Figure 16: Probabilites from logits without re-weighting vs. historic ground-truth distribution, submission by (Lu, 2022), test set for Melbourne for 14:00–18:00. The historic empiric distribution is computed on the training data.

‘