Abstract

The technique of compressive sampling (network tomography) is demonstrated using traffic counts on a transporation network to reconstruct the origin-destination demand matrix. Compressive sampling is an underdetermined inversion problem with infinitely many possible solutions by which the task is to identify the most likely or best one that suits the data and the constraints. A linear solver, MCMC estimation, Multi-Layer Perceptron and an Autoencoder are evaluated here. The 6.5GB count data in .csv format requires a use license from the ETH Zürich. It contains the observations of cars passing detectors on road segments in bins of 15 minutes for roughly 2 weeks for 40 cities worldwide. In this paper, this data is analyzed, cleaned and georeferenced to a topological road network. The network is downloaded from Open Street Maps and adapted to the modelling purpose with the popular Python package OSMnX. The solution techniques are first tested and tuned on toy models from the literature. The data model is formatted to match the structure of the toy models and OD matrices are estimated from the real counts using the solvers. The model results are in rough agreement with one another, with systematic differences that depend on the model specification and approach. In general, the results are sobering, but this is consistent with the few published attempts at this challenge.

A Note on Terminology

As this paper was finished in the nick of time, some roughly synonymous terminology may be used uncarefully without sufficient explanation. “Sensor” and “detector” are the same thing here even if “sensor” is a more general term technically. “Route” is the same as “path” and consists of a series of edges (also nodes). “Edge” and “link” are the same, as well, and refer to directed connections between nodes. “Nodes” are the endpoint of edges but are not always intersections. In fact, network models of intersections are composed of many nodes and edges, but in this paper an intersection is simplified to a single node (unless it is a significantly complex spatial structure like a traffic circle). A road “segment” refers to a physical entity, not a topological one, and may correspond to a single edge or a series of edges without intersections. “Flow” is demand on an edge, i.e. vehicles in motion or in a queue, but it can also indicate the potential for flow, i.e. the total number of vehicles or travellers waiting to depart at an origin or expected to arrive at a destination in a specific time frame. Thus “demand” and “flow” may be used interchangeably in this paper, though it is not strictly an equivalence (the former being a potential and the latter being a flux). Finally, “graph” is used synonymously with “network” for the model of the road system, but “network” can also refer to a neural network!

Origin Destination Matrices in Transportation Planning

A matrix of travel demand from origins to destinations (“OD Matrix” or “OD”) is the staple of long-term transportation planning and is also used for real-time traffic management. The matrix consists of the number of trips, i.e. episodes of exiting from one geographic location and moving through space to arrive at another. The trips can be measured in various units such as people or vehicles and are aggregated into time slices and geographic zones that are represented by the term origin or destination. Sometimes only workday average and weekend average travel rates suffice for long-range planning. But matrices which resolve the morning and evening peak travel as well as off-peak, or which also reveal the mid-day peak on weekdays and the different patterns on Saturday versus Sunday are also common. Time slices of one hour on a representative “weekday” and “weekend” usually suffice for regional planning purposes.

The demand from the matrices is routed with mathematical models through a topological model of a road network that consists of nodes and directed edges. This analysis step is called “Assignment”. Frequently the assignment takes place after the mode (“Mode Choice”, i.e. the means of transportation) of the traveller has been determined by a model: thus planners can assign people to cars and buses, for example. It is therefore important to distinguish the demand of individuals for travel versus the demand of vehicles for space on the road, as these numbers can be quite different. The mathematical models, as well as the network models used, range from macroscopic and simple (only highways, only assigning blocks of demand to network edges) to microscopic and quite complex (mimicking driving styles, ride-sharing, traffic light cycles and queues and the search for parking spaces). Using such models, the planner or researcher can predict traffic flows, queues, and speeds on all the edges of the network model.

A demand matrix cannot be entirely observed. It is traditionally estimated by expensive and imprecise suveys of the travel habits of individuals and firms. The data collection may require participants to record a travel journal for a week, perhaps just a day, to carry a GPS device, and to offer personal demographic details. Additional scoio-economic information about the origin and destination zones of the traveller are collected from other sources. These surveys and compilations of geospatial data form the basis of models that extrapolate how many trips will be produced (generated) or attracted across the zones designated for the study. The models parametrize the number of trips generated or attracted on the basis of land use, type of activities in the sectors, population density, demographic statistics, etc.., generally with a type of regression, to explain the number of trips observed in the survey. The modelling step which allocates trips from origins to destinations is called, “Distribution” and like Assignment and Mode Choice, the models can range from simple models to very complex systems. The simplest type of distribution model is based on Newton’s formula for gravitational attraction. The effort required to produce the OD matrix means that it is often used for decades before it is renewed. It also means that the costly information is rarely made available for free, i.e. for research.

There has long been incentive to reduce the cost and keep existing OD matrices current by using alternative methods for estimating travel demand. The growth of digital data sources and machine learning algorithms offer a special opportunity. Location and speed are recorded by GPS transceivers in personal telephones and vehicle diagnostic systems. Antenna localization or Bluetooth-IDs can also be used to track mobile phones along a route, with appropriate measures to protect identities. Taxi fleets often track their vehicles and many Uber drivers’ tracks are also stored digitally, some online. Researchers can install GPS recorders in cars which are then driven along with the flow of traffic (“floating cars”), on routes which are hypothesized to be commonly chosen. Buses are also often GPS-located for performance monitoring and driver safety. Finally, counts of vehicles on the road are made by cameras (sometimes tracking-enabled with license plate recognition), in-road loops, and traffic management controllers. The marginal cost of collecting this data is nearly zero. Some of this information is not available for privacy reasons, some is protected by proprietary concerns, and some is simply not helpful for deriving the travel patterns of the majority of traffic. The literature on calculating OD matrices from digital measurements of the system is correspondingly partially accessible, partially transparent and illustrates a range of methods using a wide array of data inputs.

The UTD19 dataset of traffic loop-sensor counts was assembled at the ETH Zürich for the purpose of standardized modelling of speed-congestion dynamics on road links (Loder et al. 2020). The data provides counts on road segments for 40 cities for a continuous period of several weeks in 2018. The 6.5 GB files in .csv format are available upon application from the Insitute of Transportation Studies (IVT) at ETHz. This study intends to demonstrate the derivation of OD matrices from this data by combining it with free topological road networks from Open Street Maps (OSM).

Literature Review

The basic problem to solve is Y = A * X where dim(Y) << dim (X) and A is sparse and not invertible, Y is edge counts and X is OD demand. The mathematical technique belongs to the family of compressive sampling problems (network tomography when applied to data connected by a network topology). The technique is commonly used to calculate packet demand on the Internet from a sample of byte flows, or to add missing pixels to an undersampled image in medicine where a quick scan of a patient is required and time cannot be spent on collecting full images.

Methods to calibrate, update or to derive OD matrices on the basis of traffic counts were first published in the 1980s by academics with an engineering background in the area of transportation planning (e.g. Cascetta 1984). A thorough review is in Li et al (2022), Sanandaji and Varaiya (2014) or Bera and Rao (2011). The mathematical approach spans methods from generalized least squares (Bell 1991) to stochastic approaches like entropy maximization (Van Zuylen and Willumsen 1980) to find the most likely allocation of counts to routes and thus to the origins and destinations. The "bi-level" approach to solving the problem iteratively performs a traffic assignment to re-allocate the estimated OD demand to the network routes, which is repeated until a high agreement between input counts and output OD is reached (for example Syffen et al 2023).

Vardi(1996), a statistician, is credited by Hazelton(2015) of coining the term "network topography" to characterize the inverse problems of this type. Vardi introduced a Markov route choice matrix in place of *A*, and employed a Method of Moments with Poisson priors (and first and second derivatives) to solve for the OD matrix (Hazelton 2015, Dey et al 2020).

Hazelton (2015) casts the estimation of the OD matrix as a Bayesian inference problem which he solves using expectation maximization, MLE and various sampling methods for route choice based on work from Tebaldi and West (1998) and Airoldi and Blocker (2013). He compares the results to the method of moments (Vardi 1996), which is an attempt to tune the OD flow by constraining the mean, standard deviation, skew and kurtosis of the distributions. Hazelton also reviews efforts from the literature to better sample the extremely high number of possible routes (which presented difficulties for implementing the method of moments).

According to Hazelton, with expectation maximization and the assumption of a distribution of trip production, one need not have prior information about the OD matrix in order to calculate it from counts on the network, but the solutions are commonly sensitive to initial estimates and get caught in local expectation maxima, so the best-possible estimate of OD flows are desirable for initializing the model. Cascetta et al (2013) confirm this necessity for their method of applying a Kalman filter in OD estimation from counts. However acquiring a sufficient sample of the routes that are actually used is not always straightforward. One must find an old OD matrix and assign it to the network for an initial indication of route choice, or acquire floating car data, tracks of Uber or other vehicles, license plate recognition data, etc. The actual route flows are not important: knowing the appropriate probability distribution across the chosen routes is all that is needed. Estimating this probability distribution and sampling from it with a Gibbs sampler was the core of the Tebaldi and West (1998) approach.

Indeed it became apparent that none of the papers found in the literature search, apart from toy models, complete the estimation of OD matrix demand by using traffic counts and a road network model alone. All engage iterative assignment or simulation models, initial estimates of (or simply old) OD matrices, floating car datasets or other indications of travel time on routes, and other survey or measurement data to refine the allocation of counts to origins and destinations (Kurzhanskiy 2022, Dey et al. 2020, Mussone et al. 2010, Behara 2019, Li et al 2022, Bera and Rao 2011, Bauer et al. 2018).

Mussone et al. (2010) is an early adopter of NN approaches, modelling six junctions on the M42 motorway in Birmingham, England. The demand is divided into four classes ("low", "to destination 7", "to destination 6", "AM/PM peaks"), there are 14 OD pairs, 103 detectors. This is a very highly populated count matrix relative to the number of possible routes, which themselves are enumerable. It is not the sparse matrix encountered on larger networks. The model is an MLP with one hidden layer and tanh transfer function, with no further details of its structure. The authors' predominant concerns were the effective treatment of the heteroscedasticity of the input counts and choosing which lane of the multi-lane sensors to use in the model. However they made use of speed measurements from automated number plate recognition cameras.

In casting the problem as a deep learning model, a very helpful text is "Deep learning methods for inverse problems" from Shima Kamyab et. al (2022). Specific DNN model specifications for specific kinds of inverse problems are sketched and even compared quantitatively. The paper contains an overview of inverse problems, their definition, and some literature review. The paper indicates that an Autoencoder would suit the network tomography problem.

Graph convolution networks have been proposed and demonstrated in many forms to predict link flows on counted edges in the future from historical counts: for example predicting the next hour from the last one. The Synchronous Time Series Graph Convolution Network model (STSGCN, Song et al 2020), for example, simultaneously trains time and network (convolution) dimensions to account for time-space interactions in modelling the predictions. The Attention Based version (ASTGCN, Guo et al 2019) estimates flows over 3 time periods at once. Each of these papers has a github repository with a Python implementation in pytorch and a data sample. However this family of GCN models doesn't map counts to an OD matrix, or even impute flows on non-counted edges, so the imputation via an estimation of the OD matrix would have to occur first. STSGCN was tested on its California dataset and though a goal was set to also implement it for the UTD19 dataset, time ran out during this project and it couldn't be tried yet.

The CGAME model of Li, He and Wu (2022) is a bi-level model using the newest ML technology and purports to solve the problem of generating an OD matrix from counts on edges without prior knowledge of historic routes, historic OD matrices, and without an assignment model. The authors implement a cyclic graph attentive matching encoder neural network architecture which iterates the solution of an OD matrix from counts, using a graph matching neural network instead of the assignment model or microsimulation. The forward network encodes the counts into features which are decoded into and OD matrix. The backward network uses the same decoder model to encode an OD matrix into features which are decoded by the former encoder into counts, using the same graph-matching neural network. At the center in the latent feature space, the graph matcher plays the role of the assignment model to learn weights that map the count features to the OD features and vice-versa. Thus the above expectation maximization with prior distributions and the assignment models are both replaced by neural networks. The paper has no published code and the english is poor. Details are lacking on the method, however it is apparent that the authors do NOT avoid using an initial histiorical OD matrix and that they also employ a dynamic assignment model to initialize a set of plausible OD routes for their realistic scenario in the city of Haikou. Implementing this very interesting model was too ambitious for the scope and time allotted for this project unfortunately.

Mathematical Formulation

The calculation of an OD demand matrix on a network from counts on edges of the network is an "inversion problem", a problem of inverting an equation: Y = A * X.

The following derivation of the objective functions to solve is taken from Kurzhnaskiy(2022), Dey et al. (2020), Mussone et al. (2010), Behara(2019) and Hazelton(2015).

The assignment of demand onto a network (assuming equilibrium, i.e. no traveller can improve his cost by switching to another route) is a continuous relationship between the link (edge) flows *f* as a function of link costs *c*. In its most general form, the relationship can be expressed:

$$f = A * P(c(f) * d(c(f))$$ (Mussone eqn 1)

A is the link<->path (edge<->route) incidence matrix equal to 1 if link i is on path j and 0 otherwise;

P is the path choice probability matrix;

and $f > 0$.

The more detailed and accurate the *P* and *c* functions are, the better the model results will be. In general,

$$f = G(d)$$

Where G is very sparse, not invertible, and has to be estimated recursively. Often this entails assigning the OD matrix that is estimated and changing the parameters of *G* until the matrix matches an observed one.

Neglecting traffic dynamics such as slowing or queueing due to congestion, or changing route preferences due to perceived increases in travel time due to congestion (which would alter the distribution of traffic onto the count detectors without altering the OD demand), one can work with the adjacency matrix, *A*, as in Hazelton(2015). This is equivalent to a compressive sampling problem on a network in telecommunications:

$$Y = A * X$$

where Y is the vector of edge counts, *A* is the adjacency matrix of *edges with counts on them* and routes between origins and destinations, and *X* is the matrix of *route* flows. If a time series of counts is available, the *Y* matrix has dimension (N_t,I), *A* is (I,J) and *X* is (J, N_t).

The OD matrix is the sum of flow on the routes that lead from each O to each D and has dimension (K,N_t).

I is always smaller than or equal to *J* and is usually many factors of 10 smaller. *A* is therefore a "wide" matrix with many zeros. Any number of *X*'s can satisfy the equation and the solution must be chosen to maximize the probability that *X* matches *Y* and *A*.

By necessity, not all possible routes from O to D can be included in the *A* matrix. The number of possible routes through a network is simply too large to be computationally feasible. The choice of how many, and which, routes to include, is a topic of much research. This paper chooses the *b* time-shortest routes between origins and destinations and uses several choices of *b* to examine the effect. This is the technique of all of the published papers applying this method in practice.

The search for the most appropriate demand allocation with respect to realistic route choice can be focused by replacing the *A* matrix with the *P* matrix of route choice, as discussed by Hazelton(2015) and outlines step-by-step in Dey et al (2020), which presents a reformulation of the matrix used by Vardi(1996).

A is first recast as *A'*, a link-choice probability: *a'ik* = 1 if link *i* belongs to the only path on OD pair *k*; 0 if it is not on any path between OD pair *k*; and 0<*a'ik*<1 if link *i* belongs to a possible route of the OD

pair *k*. This is the zero-memory Markov matrix of route choice: in the choice of the next edge to take, it does not matter what edges were already chosen. This matrix is not used in this paper for calculation of OD demand, but as an intermediate step for the calculation of the *P* matrix.

P is derived from this *A'*, where active links *M* are those in *A'* with nonzero values. $Q^{(m)}$ is the m^{th} order state transition matrix of a Markov chain for the OD pair *k*. If the initial probability of the chain for OD pair *k* is π^k, the probability for an edge *i* on a route of OD pair *k* is $\pi_i^k = P_i^k$ and at the end of the route it is π_{end}=0:

$$P_i^k = \pi_i^k \sum_{state}^{M} [Q^k]^{state}$$

Pik is the probability of *i* over all possible routes on OD pair *k* that involve link *i* and represents the finite-memory Markov routing matrix of the network. (Note that in the notebooks, A is called aij, A' is called Pij and P is called P or P_mat).

Using this additional information from the network for the allocation of counts back onto the OD matrix: *Y = P * X* is still a highly underconstrained problem. Vardi (1996) introduced the Method of Moments to add constraints of higher-order moments of the prior distribution of the demand, assuming the counts were independently Poisson-distributed. His complete equation is found in Dey et al. (2020) in equations 13, 20, 21, 22 and 23 and were implemented as part of this study for a comparison of the Method of Moments as well as a comparison of using a [0,1]-valued edge-route adjacency matrix *A* versus a route choice adjacency matrix *P* that contains more information about the network flow.

It is important to note that *P* has dimenson (K, N_t), i.e. the number of OD pairs and not the number of routes.

The estimation of *X* in either case is a matter of iteratively minimizing the function:

$$\min\left(\left\|Y - \hat{Y}\right\|\right) = \|Y - AX \; or \; Y - PX\|$$

subject to x_j >=0. In practice this means the mean square error. The best sparse solution is guaranteed by adding the l_0 regularization term but this would be an NP hard problem to solve. The l_1 norm however converges to the best sparse solution. According to Kurzhanskiy (2022) this norm is built in to the problem and adding it to the least square term does not help the solution. Instead, a weighted l_1 norm is recommended:

$$\sum w_j x_j$$

and added to the least square error for route *j* Alternatively, the nonconvex infinity norm can be used in models with appropriate solvers.

This paper employs the weighted l_1 norm in the deep neural networks, the infinity norm in the CVXPY solver and no regularization in the MCMC solution.

Finally, an adjacency matrix of all network edges and routes containing them can easily be generated from the network to enable the calculation of flows on the non-counted links from the generated OD matrix, thus imputing edge flows for all edges in the network (this is called a2ij in the notebooks). At this stage, the network with a complete set of flows for all time slices can be used in other models for flow prediction, for example.

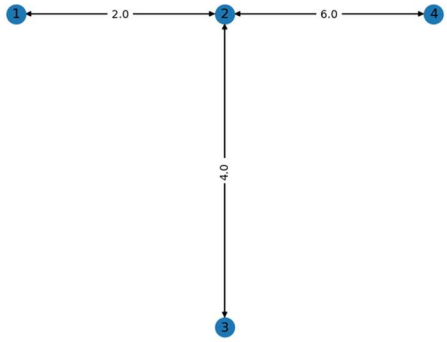
Toy Models to Study the Methodology

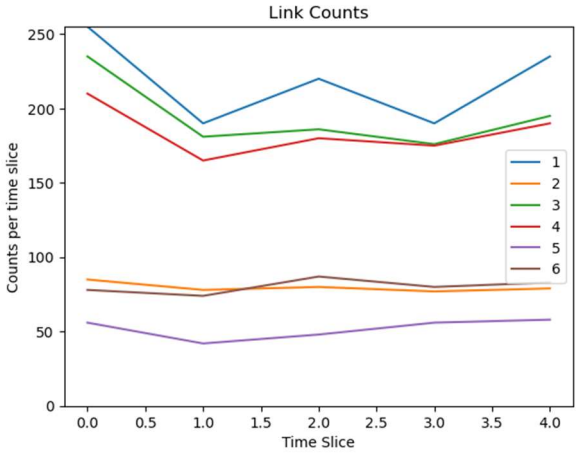
The modelling involves conceptual, mathematical and programming challenges that needed to be tested on a bench before using the compressive sensing/network tomography methods on noisy, large-scale real data. The data structures for the modelling needed to be tested and verified for all eventualities that may crop up in the data or the experiments. The effect of regularization efforts such as limiting the route choices or using normalization terms in the objective function were tested, as was the suitability of standard statistical solution methods as a reference for evaluating the performance of later ML methods. Runtimes of the various solution approaches could also be estimated using the toy models, as well as some hyperparameters. The toy models also provided the framework for analyzing the outputs of the models.

The methods were programmed and tested on examples from the literature, because some of these were also published with the expected solutions for the OD flows, or at least expressions for the intermediate steps of model preparation like the Markov route choice transition matrix *P*. The models tested aspects of the modelling challenge such as: graphs containing edges with no traffic counts on them; networks with varying numbers of routes per OD pair; time-varying counts vs. a single time slice of counts; graphs in which not all nodes are origins or destinations; and weighted edges with simulated travel times for implementing simulations of route preference:

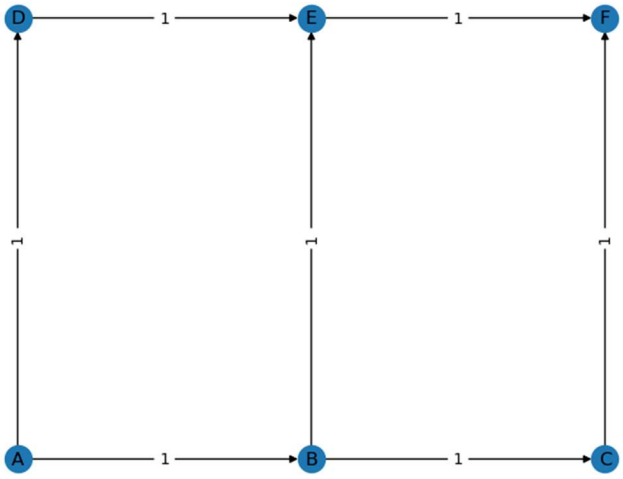
The following models were implemented:

- Hazelton (2015) model 1, a toy model of London Road and University Road in Leicester.
    - "T" intersection of 4 nodes but only 3 were origin or destination
    - 5 time slices of real data on 5 of the 6 edges
    - Two-way directed edges between nodes
    - One route per OD
    - Solved as in Hazelton (2015) with the Method of Moments from Vardi (1996) as presented in Dey, et al. (2020) (workbook CS 06)
    - Cvxpy convex solver and all results from Hazelton's paper in workbook CS_02
    - Cvxpy and Lasso fit analysis in workbook CS_01
    - Edge weights added to test *P* matrix route calculation
- Dey , et al.(2020) model 1
    - Abstract grid network with 6 nodes and 30 OD relations (6 x 5)
    - 7 one-way directed edges
    - Varying number of routes/OD: 16 routes in all
    - Set of 7 time-dependent counts made up (all links counted)
    - Test of the P matrix (finite memory Markov transition) calculation
    - Solved with Method of Moments and CVXPY in workbook CS_07
    - Solved with CVXPY and MLP in workbook DNN_2





Model 1 from Hazelton(2015). Plots from the Networkx package only show the edge weights ("travel cost") for one direction. The graph has 6 edges, node 2 is neither origin nor destination.



Model 1 from Dey, et al.(2020). Uses demand from previous model where one set of counts is duplicated for the seventh edge. Note the one-way edges.

Solvers

Pymc3 is a widely touted solver package for inversion problems but all attempts with it failed. Its efficient operation relies on the package Theano. Theano doesn't seem to exist anymore. The Pymc3 resource says to use Theano-Pymc. But Theano-Pymc indicates that it works for pymc 1.1. The link from pymc3 to this package goes to the github page of "aesara" and not to Theano-Pymc. There were many other errors in the pymc3 documentation and it was deemed not only a waste of time but also not trustworthy.

The solver CVXPY (Diamond and Boyd 2016) was used for a first quick solution to the inversion problem. Its mixed-integer solver ECOS-BB emits a warning that it may not produce correct answers. The recommended solver SCIP failed to install correctly in the environment. ECOS-BB worked for the simple cases but indeed sometimes it failed to converge, without explanation. Installing another free mixed-integer solver was risking wasting more time. The CVXPY float solver SCS succeeds in complicated models but delivers very small negative values despite the constraint for positives, however these could be rounded up to zero.

In addition, an Expectation Maximization solver was written which naively used the analytical derivative of a Poisson distribution, containing a factorial, which would obviously fail to calculate for any appreciably high flow value, so this notion was discarded.

A Markov Chain Monte Carlo simulation was programmed to enable the evaluation of different prior distributions (though in the time constraints only a Poisson with variable dispersion was tried).

Finally a Multilayer Perceptron was written to be tested with the second model.

Hazelton Model 1

Hazelton(2015) presents the time-average flows on the toy network from several calculation methods. The CVXPY result fits well within the spread.The method of moments is a clear outlier whether the original results from Hazelton's implementation of Vardi's paper or the implementation here of Vardi's method as modified by Dey et al.. Hazelton's MLE calculation based on Poisson or Negative Binomial priors match CVXPY. This is evidence that the model preparation and the solution method work as intended. It is however clear that the implentation of the method of moments as detailed in Dey, et al.(2020) did not succeed.

Route Flow = OD Flow Toy Model: Results from Literature and Implementation				
Poiss_MLE reported	Vardi MoM reported	NegBin_MLE reported	CVXPY calculated	Dey et al. MoM calculated
175.5	420	176.4	177.0	1740
41.5	37.5	41.0	41.0	254
183.9	155.4	183.5	181.4	362
10.1	39.8	10.6	10.6	457
61.9	49.7	63.1	62.4	144
14.5	7.0	12.8	14.0	-0.86

Dey et al Model 1

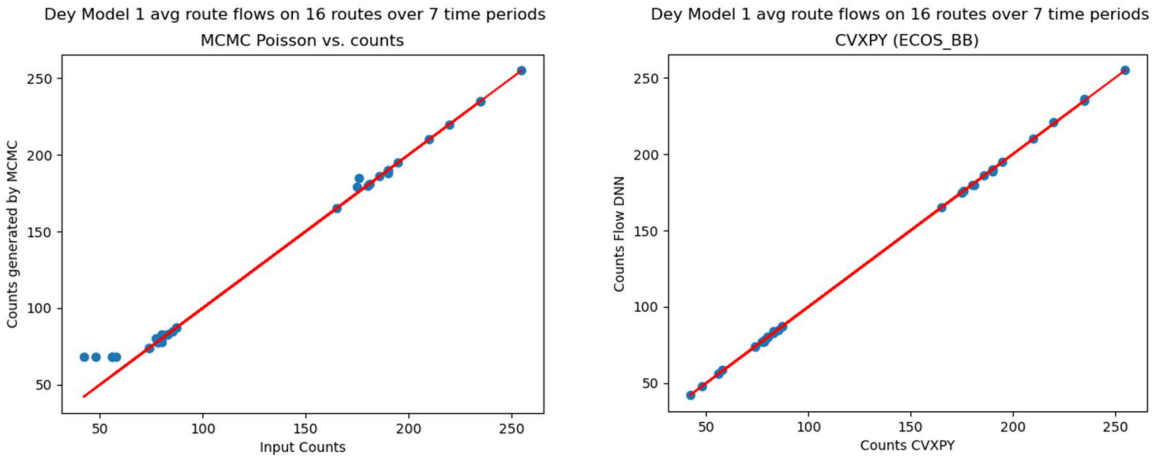
This toy model is used in the publication to demonstrate the calculation of the finite-memory Markov transition matrix as a representation of route choice and thus the basis of assigning the probability that flow on a particular measured edge will be assigned to a particular route and a particular origin. This regularization measure, providing additional information to the model for allocating the OD demand (from counted edges to the OD matrix), is an effort to reduce the number of routes per OD that are necessary for fitting the model, while providing reassurance that the most likely routes have indeed been chosen. Dey et al. use this method in their real-world application, though no details of its implementation on real data are provided. As mentioned in the figure caption, there were no counts and no solution offered for the toy model, as its purpose in the paper was different. Here, as a test, for the 7 edges of this model, the 6 sets of counts from the other model were adopted, plus a seventh set of counts that was just a copy of one of the other 6.

The P matrix implementation reproduced the matrix as in the paper. But the method of moments as applied to Hazelton's Model 1 did not seem to be correct. There is much doubt as to whether this method could be applied to a real-world scenario without more information from the authors.

MCMC

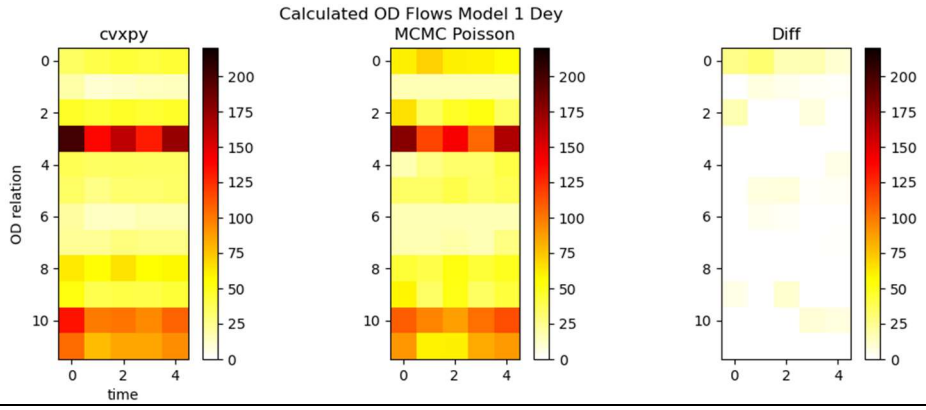
With the simulated edge counts, the second model was used to test a Markov Chain Monte Carlo estimation with Poisson sampling against the CVXPY solution. The MCMC was written to compare with the statistical papers, to provide the desired integer solution, and to provide a benchmark solution on the validation dataset for comparison with later ML approaches. The MCMC samples route flow

candidates from a Poisson distribution and assesses the probability of the candidate with a draw from a Gamma distribution in a Metropolis-Hastings sampler with tuned dispersion of the prior. The candidates and assessments occur in random order and individually on each route flow, i.e. the updates are made for each route (OD relation) individually and not all at once as a batch for each instance of count observations. The MCMC was initialized with route flows of 100 for each flow and was allowed to run for 5000 iterations, tuning the Poisson dispersion every 100 for 4000 iterations. The route flows were solidly fixed by then to single values. The input edge counts are reproduced fathfully by the CVXPY solver and the MCMC overestimates the counts on low-count edges (note that there are no edges in this model with missing counts).



MCMC solution with the second toy model: agreement for estimated edge counts vs. input edge counts compared to CVXPY solver, which shows perfect agreement.

The MCMC and CVXPY OD flows agree qualitatively and quantitatively, as well. Either solution seems like it would be a suitable benchmark, though the MCMC may be preferred as it is an integer solution.

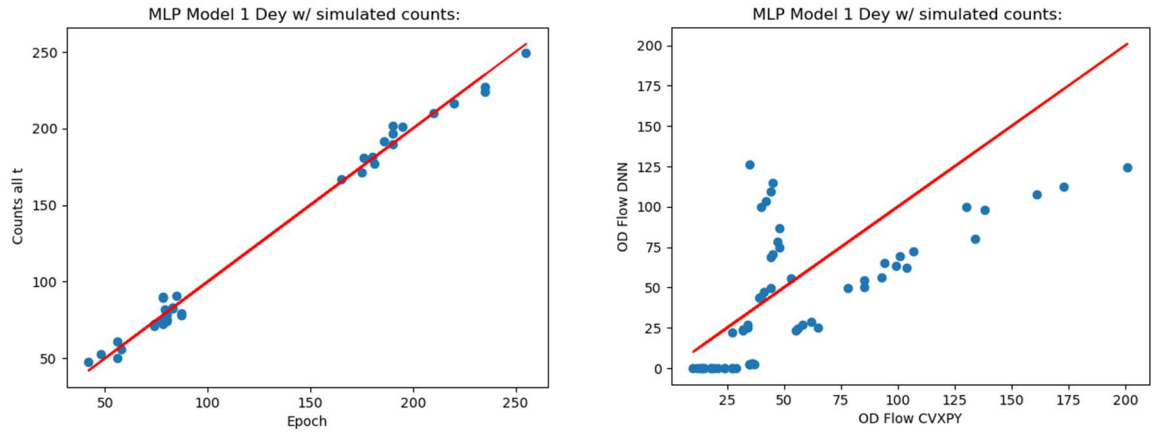


MCMC solution with the second toy model: OD flows plotted as OD relation vs. time, compared to CVXPY solver, and their difference.

With these solvers indicating that this toy example and the matrices derived from it are suitable for testing other solvers, a multi-layer perceptron (MLP) model using the [0,1] adjacency matrix was attempted and compared with the previous solvers. This offered the opportunity to become acquainted with a Deep Neural Network's (DNN) performance with such data and to choose suitable parameters and hyperparameters. Several configurations were tested with the final model having two hidden layers of 256 nodes. There are 7 input values (the counts) and 16 output values (the

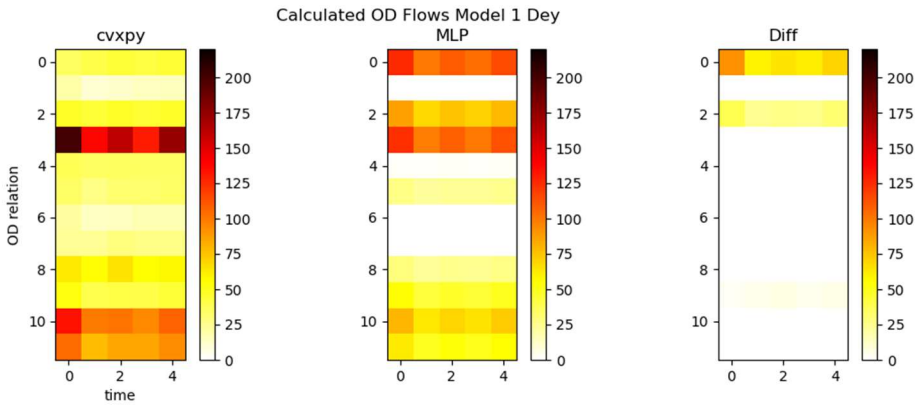
route flows). ReLU rectification ensures that only positive flow values come from the model, as all the inputs are also positive.

A sweep over weighted LASSO (*l*1) regularization coefficients yielded a best Lambda of 0.01, based on MSE loss between estimated and input edge counts. With this rate, the learning was basically complete after 1500 epochs, to an MSE loss of very nearly zero. The agreement of the count estimations with the CVXPY solver is very good, as well. However the distribution of the derived OD flows in each model diverges. The MLP produces five zero-valued OD flows, relations to which no demand is attributed. This is one sixth of the OD matrix. And on a few OD relations it allocates a large number of trips where the CVXPY allocated on the order of 4-5 times fewer trips. The rest of the OD relations in the MLP model have approximately 65-75% as many trips as in the CVXPY solver.



MLP experiments with the second toy model: agreement for estimated edge counts with CVXPY solver (the X-axis label is incorrect). However estimates of OD flows show many zero-valued OD and less agreement.

The heatmap plot of OD flows shows that there are very large differences between the MLP and the convex solver in this configuration. This is the best result from the MLP after attempts at tuning.



MLP solution with the second toy model: OD flows plotted as OD relation vs. time, compared to CVXPY solver, and their difference.

Much effort was spent trying to discover why this discrepancy occurred and how to resolve it. Using L1 loss instead of MSE does not reduce the zero-value OD's. Decreasing the learning rate to 1e-6 slows learning and gives the same number of zero-value OD relations if running to a flat loss curve, but interestingly has fewer zero-valued OD relations if stopped early. Increasing the learning rate to 1e-2 reaches a minimum loss after 3 epochs which becomes unstable and there are still zero-valued

OD flows. Setting the layer bias to True was an important improvement to the agreement with the counts and the flows and is included in the plots above. Shuffling the counts resulted in zero-valued counts and was a catastrophe, though there is no reason that the time series order need be maintained in this model. The cause is unknown. For this toy model, the MLP runs in a batch size of 1. The results could not be improved over those shown in the above plots.

Constructing the Real-World Scenario

The data-based model uses counts from UTD19 (IVT, ETH Zürich) and a street map from Open Street Maps (OSM).

OpenStreetMaps with OSMNX

OSMnx is a Python package from Univeristy of Southern California Professor G. Boeing (2017) to download and work with geospatial datasets from Open Street Maps (OSM). Supplying a "PLACE_NAME" downloads a graph from OSM that is cropped by a poly gon shapefile for that place name. The polygon is stored on a database on OSM and the precise spelling of the place name must be ascertained by trial and error or by consulting the database. Further OSMNX processing flags that were used for this work produce a streamlined topology for modelling: "simplify" removes intermediate nodes by combining the multi-segment edges into single, longer ones and it aggregates intersections into single nodes; "truncate by edge" assures that there are no edge-less singleton node subgraphs around the cropped area after an edge has been cropped out; "clean_periphery" simplifies the graph margins before cropping, and "network_type" restricts the edges to those serving the transportation mode in the keyword, for example, "drive" to indicate motor vehicles. The resulting graph is an OSMX wrapper of a DiGraph, or directed graph, object in the Python "Networkx" package (Hagberge et al. 2008). Each edge has an origin node, u, and destination node, v. A machine-cropped directed graph may still have topological inconsistencies resulting in effective dead-ends and circular paths. Preparing a graph to model realistic road networks would require careful review and alteration of the graph edges and nodes by hand such that the routes between origins and destinations are plausible. This check for realism was not done for this demonstration of the methodology and care had to be taken in the choice of origins and destination to avoid unreachable combinations, or else errors would be thrown in the route searches.

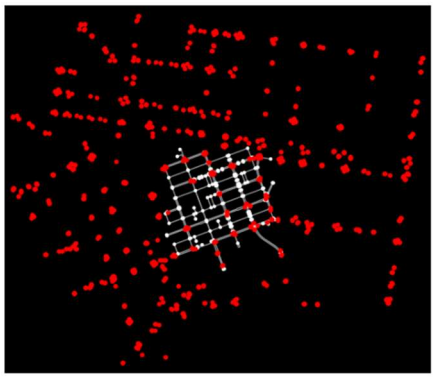
Preparing the counts data and road network model

The counts and detectors for the city of interest are extracted from the UTD19 database. All extraction from the original 6.5GB .csv database was done in advance and the counts for each city saved off separately into forty separate .csv files with UTF-8 encoding. Each city's counts can be read in separately, though some files are quite large still (e.g. Lucerne has over 1GB of counts).

First tests were performed on Birmingham, England because the counts file is a conveniently small size, but the distribution of these counters proved too dispersed relative to the road network density to be very useful. Also, a grid-type network more similar to the toy networks (with nodes of degree 4) was preferred. The center district of Melbourne, Australia proved to have a suitable combination of both grid-like roads and a higher concentration of sensors.

In a first round of data cleaning, only the detectors with valid counts are chosen. No indication of the "bad data flags" are given in the UTD19 dataset, so some manual tests have to be undertaken. The flags were identified and are different for each variable. Some detectors have no counts for the dates of collection. Some detectors have only a "bad data" flag for all time stamps and some have predominantly missing data. These lines are deleted from the counts data in the workbook.

The graph for the region (city) of interest is then extracted from OSM and the detector locations are superimposed on a plot of the graph as pandas geodataframes.



Downloading an entire city graph from OSM would show a tiny portion covered in red dots (traffic counter locations). The 2x2km square about Russel Street in Melbourne (left) is however small compared to the extent of the detectors available (right).

The graph is cropped (truncated, see above) to coincide with a selection of detectors to yield a tractable computational problem (in this case, the PLACE NAME for a central street in Melbourne, Russel Street, sufficed: a 2km x 2km square about a central point on the street is automatically used by OSMnX). The travel times in free-flow traffic for the edges of the road network are calculated by a function in OSMnX from the road speeds that are available as edge attributes in the OSM data.

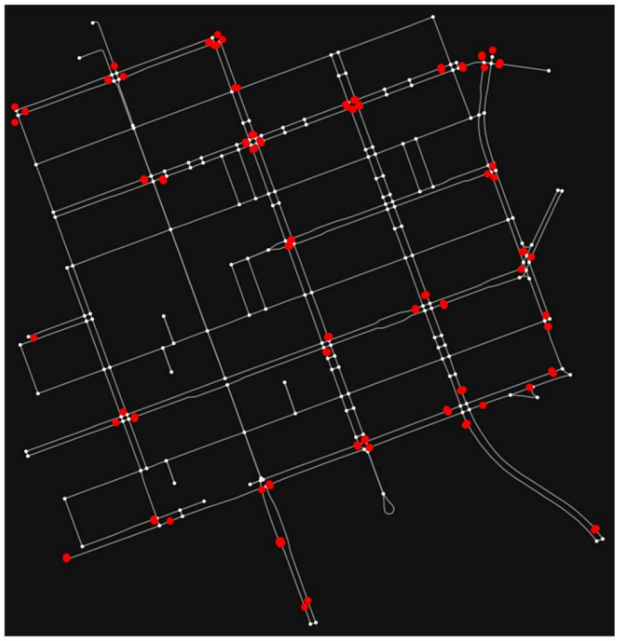
The detector dataset does not have useful topological embedding aside from a relative location on a series of waypoints along a road segment, which is used in a further data cleaning step outlined below. So the detectors are assigned to the graph edges by Euclidean distance. Only detectors within 0.01 degrees of latitude/longitude of a road edge are even considered and the rest are removed from the set.

Occasionally this results in assigning the detector to the wrong lane or the wrong lane direction, i.e. graph topological errors. The errors can occur for many reasons: incorrect or imprecise recording of the latitude and longitude of either the edge geography in the OSM network or the detector; a detector that is located next to a road and which senses from a short distance away; errors introduced in the graph projection used in the OSM file and/or the matching algorithm within OSMnX; uncertain assignment of the edge when two edges are equidistant to the detector; inconsistency in abstracting a geolocation of a road segment (where is the center of a road, where is the detector in relation to it?) or between different models, i.e. a city engineer, a transport consultant, the regional planning office. The OSM network is after all a Wiki product maintained by volunteers. These attributions have to be checked and corrected.

The UTD19 dataset contains a link file of waypoints (intermediate nodes) on edges between intersections on the road segment where the detector is located. This information was used in the original research on congestion/speed profiles for which the data was collected, to correct for queues that formed on sensors near intersections, whereas sensors far away from intersections did not measure this queueing. This waypoint file gives the topological direction of traffic flow over the detector and can be used to assign the detector to the correct lane and edge direction in combination with the cardinal direction of the road segment of the detector that can be calculated

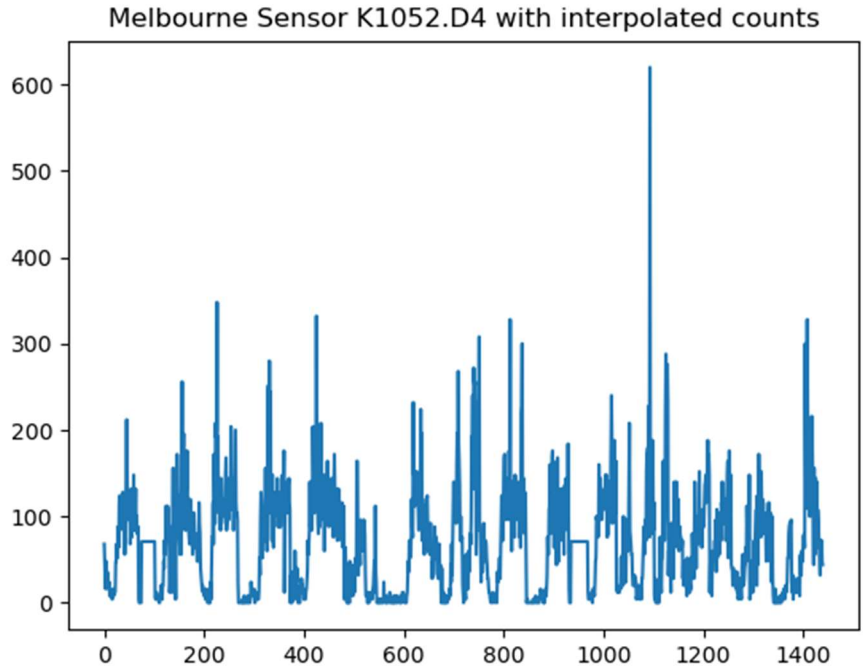
with OSMnX, thus bridging network topology and pure geospatial orientation. However many detectors do not have any links/waypoints recorded in the UTD19 dataset and these detectors cannot be reliably georeferenced to the study network, so they must be discarded at this point.

The counts of the detectors that are finally successfully geo-referenced to the graph are now re-analyzed. A pivot table of detector ID vs. time slice is necessary for the modelling. Detectors which lacked measurements on some time slices will have their counts for these time slices filled with "nan" when the matrix is broadcast in the pivot operation. These missing values can be interpolated over a small number of time slices. Some detectors have no counts for a large portion of time slices however, and these detectors are removed from the dataset at this point.

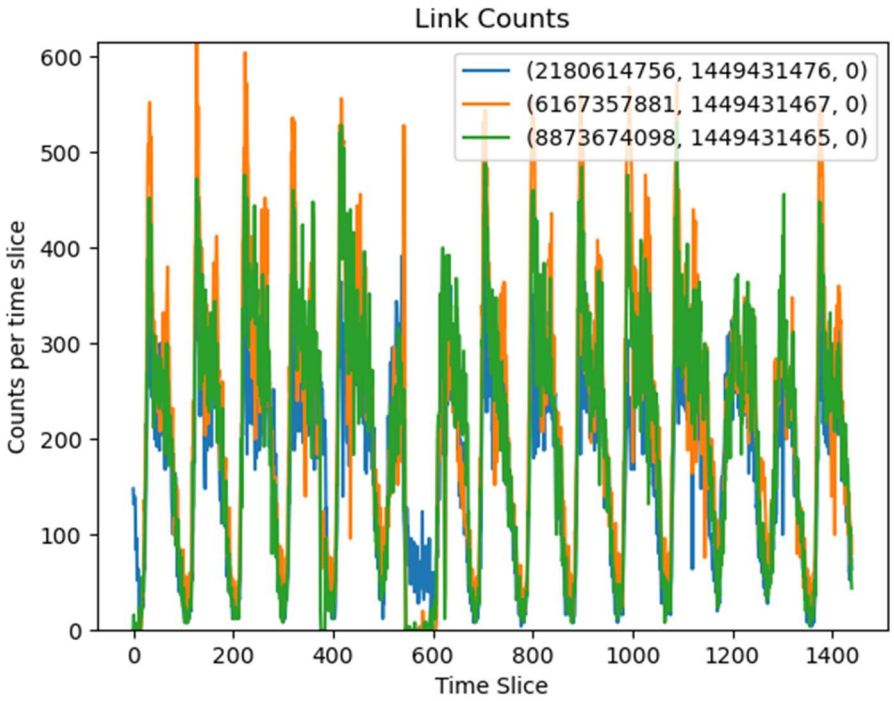


Final network of Russel Street, Melbourne with 67 counters

With this final set of detectors winnowed down, the dataset of counts vs. time is re-made and the missing counts are interpolated by filling with the detector's average value. The time slices in the plot refer to the specific time period February 12, 2018, 6:00 AM – February 26, 2018.



The horizontal flat sections of the trace indicate where missing values were filled with the average for the series of this detector (units are vehicles/15 minutes vs time slice, 15 minutes each).



Sample of link counts for 3 detectors over 1440 time slices = 15 days. The legend contains the node names of the ends of each edge corresponding to the counter, and the edge key (=0)

Selecting Origins and Destinations for the Data-Based Model

The origins and destinations of travel must be chosen. Each zone is a generator (origin) or attractor (destination) of travel (trips). These typically have a significance to spatial or transportation planners and are thus linked to geographic structures (locations of human activity) that generate travel demand. Municipalities around the world have different ways of defining their traffic planning zones depending on many political, economic and technical factors. In practice, all the nodes of the road network model would be linked to the central point of each planning zone with network edges which all have a constant cost, representing a slow local speed for "the last mile" of travel, including vehicle access, parking, etc. Thus the demand can be distributed to various points of entry on the transport network, i.e. the traffic can flow on multiple routes per origin and destination from many access points. There are many fewer origins and destinations than there are nodes in the model. It would be very useful for completing this modelling exercise to have a realistic set of origin and destination zones in a shapefile. A simple grid or other abstract set of zones and graph edges could have been generated for the purposes of this paper but as this was another unfamiliar task under time constraints, its attempt will have to wait.

To spare complication, nodes of degree less than or equal to two were chosen as origins and destinations for this demonstration in Melbourne: 29 of 211 nodes therefore provide access/egress to the network. Most of these nodes are located around the edge of the network. It is crucial to bear in mind that all calculations on the network flows scale with the number of zones squared, so keeping the number of OD small is important at this phase of feasibility testing. Choosing the origins and destinations like this, essentially at random, on a cropped graph that has not been carefully topologically analyzed and corrected results in many destinations which are unreachable from the origins. A depth-first search of reachable nodes from the origins yields 319 reachable OD pairs for the theoretical 29 x 28 = 812 OD's total on this graph. This set of OD pairs will suffice to demonstrate the calculation of network flows from sparse edge counts.

Routes

Finally, the routes between the origins and destinations are chosen, along which the modelled traffic will flow. The routes are series of edges which will contain the edges with counters and also many more edges without counters. The routes provide the structure for distributing the traffic flow from the origins onto the edges via the A and P matrix. The algorithms presented here will invert this process to attribute the measured signals of the flow onto the routes and back to the origins (and destinations). Ideally, all routes between OD's would be used to reconstruct the flow from the origins to destinations but this is usually prohibitively expensive computationally. The possible number of routes on realistic networks, even if limited to not repeating visits to nodes (so-called "simple" paths), can reach the order of the number of nodes in the network factorial. In practice, transportation planners observe that a large proportion of traffic uses the same handful of routes for a given OD pair: ca. 90% distributed across 6 or 8 routes is realistic.

All the information possible about the distribution of the trips across the edges would also help the model identify the most probable OD flows from the infinite possible combinations that can be constructed from the counts, for example information about the relative travel time (cost) across the routes. That is the role of the P matrix from Dey, et al. or the Gibbs sampling method of Tebaldi and West (1998). As a start, the [0,1]-valued adjacency matrix A of edges with counts versus the route they are on is the first structure used to constrain the calculation, and the P matrix will be tested in the methods.

By choosing the set of routes for the model based on the shortest cost (travel time) in a search with the Networkx package, the most-probable routes are already chosen. Successively adding more routes per OD in this fashion adds progressively less desirable, but still the subset of most desirable, routes in the assumption of no congestion. Congestion would alter the ranking of favorability of the routes. For this study, it will be assumed that this does not occur. Choosing one route per OD gives the most likely route; choosing 10 routes gives the 10 most likely routes. By using the A matrix however, there is not further information about the quality of or preferences for these routes that the model can make use of. That is the role of the *P* matrix, which takes into account the travel cost of the edges and weights the allocation of the demand across the routes accordingly.

For toy models, the adjacency matrix may be nearly square (similar number of edges with counts and routes) and filled with ones (a small selection of short routes over few nodes). However in realistic road or telecommunication networks, this matrix is characteristically extremely sparse and "wide", i.e. many routes with few edges and mostly zero entries. In the very simple Melbourne example here, with two routes per OD, the matrix has dimension 67 counted edges by 627 routes (not all OD's have two distinct simple paths between them) and contains 3.6% ones and the rest zeros.

The plot shows the 29 origins and destinations (319 OD relations) as dots and 627 routes between them (2 routes per OD relation except where there is only one route). The 67 counters are represented by blue dots. Choosing 2 routes per OD samples most of the network edges.

The Data Model in Comparison with Other Published Models

Dey et al. (2020) demonstrate their model (coincidentally also in Melbourne) on 484 OD pairs and 484 aggregate routes (*P* matrix), 54 traffic links, and 23 counting detectors, compared the the model here with 319 OD pairs and 377 links, 67 detectors with counts and 627 routes. Mussone et al (2011) study 14 OD relations with 103 sensors and 14 routes on 240 time slices.

The models on real data: MLP, Autoencoder, MCMC and CVXPY

MLP

The MLP is described above in the section on toy models and only the layer sizes were increased to 512 and 1024 for the larger dataset: 67 inputs and between 319 (one route per OD) and 7665 outputs (number of routes per OD = 25) per instance. Only the matrices (input counts, output flows) have to be transposed to accommodate the pytorch data loading standard.

Autoencoder

An Autoencoder was specified with hidden layers of 256 and 32 nodes in the encoder and the other way in the decoder, also using the weighted *l*1 regularizer as in the MLP. It benefitted from a smaller learning rate of 1e-7 instead of 1e-6 used in the MLP.

Both models reached plateaus in MSE loss quickly, so a learning rate scheduler (exponential) was employed to try to jog the solution away from a local minimum, which sometimes improved the result but which little change in the resulting demand.

MCMC

The MCMC implementation used with the Dey, et al. Model 1 was used for the data model with a modification to accommodate each time slice independently (the previous model was only able to run a single one-dimensional vector of flows). Tuning every 50 iterations with 600 iterations of draws up to 1000 iterations was found by trial and error to be more than long enough for a stable solution. The initial mean flow values were set to 10.

Convex Solver

And finally, the CVXPY solver with the “infinity norm” regularizer was re-employed on the validation dataset for this purpose, as well. The mixed-integer solver ECOS-BB diverged (result = “inf”) without error messages and ultimately delivered an output matrix of “None”, so the float solver SCS was used.

Train, Validation, Test Samples

The 67 sets of counts contain the number of vehicles passing in 1440 time slices in a time series. Each time slice is 0.25 hours or 15 minutes of accumulated vehicle counts. The time-series information in the dataset was unfortunately not utilized for this study, so the data is essentially a random set of counts, though in reality there could be causal influence between the counts registered between the counters during a time interval, and influences across the time intervals, due to vehicle interactions on the road network. The training portion is set at 75% (1080 instances = 270hrs = 11.25 days), validation at 20% (287 = 71.75 hrs = 2.98 days)and test at 5% (72 = 18 hrs = .75 days), and they are entered into the model by the pytorch DataLoader in chronological sequence and not shuffled. The test set was never used. The validation set was intended to be used to identify an early stopping point, but learning with the validation set followed the training set and no distinction between them could be made to indicate when overtraining might be occurring, so the validation result is used here as a test set to evaluate the model performance at prediction.

Batch vs. instance training

Either DNN model converges within 100 epochs with the Adam optimizer and the computation time for a batch size of one is 12 minutes on the GPU (Nvidia RTX 3080 8GB) for the training and validation samples in the Melbourne example. Batch training was tried with sizes up to 32 and this speeds up the computation immensely to under 2 minutes and does not affect the result.

Scaling

Scaling the inputs to a range of [0,1] by subtracting the mean and dividing by the standard deviation of the counts, which is commonly needed for preventing gradient extremes, was attempted in a CVXPY solution of the Hazelton Model 1 but was found to give very poor agreement with the published OD demand, after un-scaling. Small gradients could be the cause of the learning plateau. However I foresaw using the road capacity constraints in a next generation model to simulate traffic flow macroscopically, which would feed back into the neural network optimization. This would entail replacing the [0,1] adjacency matrix with a weighted matrix that would change according to the traffic demand on each edge. Scaling the variables in the neural network would then require un-scaling them for the out-of-neural-network loss calculations that took into account the changed *A* (or *P*) matrix. Scaling would have been an extra complication with uncertain benefits in view of the time constraints of the project.

Adjacency Matrix

Each type of DNN model was run with the [0,1] adjacency matrix and the finite-memory Markov *P* matrix for route optimization. The matrices are very sparse:

Adjacency Matrix for Russel Street, Melbourne: Dimension and Sparsity					
Routes per OD	1	2	5	10	25
Dim *A*	(67,319)	(67,627)	(67,1545)	(67,3075)	(67,7665)
% 1 in *A*	3.3	3.6	4.1	4.4	5.0
Dim *P*	(67,319)	(67,319)	(67,319)	(67,319)	(67,319)
Sum(*P*)	17.0	0.12	3.2e-7	2.7e-6	6.4e-19

The *P* matrix has worryingly very small values, as if it is not calculated correctly. It was verified to be correct with respect to the paper from which its derivation was implemented. Yet that paper had counts on all edges in the network, thus all edges in the network were in the P matrix rows. In the real data, a very small portion of edges have counts on them, as seen from the *A* matrix, and the others aren't accounted for in the *P* matrix. However the point of the *P* matrix is to calculate route choice probabilities, so it would seem that a travel time (cost) should be available for each edge in the network for this calculation, and not just for the edges with count detectors, or else the probability could never sum to one. This does not seem to work intuitively if substantial portions of each route are not present in the matrix. The approach needs more thought and perhaps discussion with its authors, or access to the original Vardi 1996 paper (which is behind a paywall) before it can be used. Despite the work put into deriving and testing the *P* matrix, and running models with it, these results won't be reported.

Model parameters and tuning

The experience of using the MLP on the toy models and on the training set of the Melbourne data led to some adaptations of the DNNs that were partially listed above. Other changes include:

- The number of nodes in the MLP layers was increased over the number used in the toy models in order to exceed the number of input counts and the number of output route flows. With fewer nodes the route flows had more zero-values (more concentration of the flows onto favored routes).
- The Autoencoder hyperparameters were adjusted by hand, like the MLP. It is run as the MLP with 256/32 nodes, all ReLu, and a lower learning rate.
- The DNN runs were shortened to 300 epochs (rather than 3000) because of this stagnation

- The MCMC and the CVXPY solvers were only run for the validation set as they are estimators and aren't parametrized models that could be trained first and used for inference on a different dataset.

Results

All of the models were run for a base case of 2 routes per OD relation (627 routes) with the *A* adjacency matrix. The validation dataset is used to compare the output across models.

The main outputs to evaluate are the counts generated by the models, relative to the input counts, and the OD flows (demand matrix) calculated from the route flows that were generated in the models. The latter have distributions in time, magnitude, and on the road network that need to be compared and explained, if possible.

The MLP and Autoencoder (AE) were run additionally with 1, 5, 10, and 25 routes per OD relation and with the P matrix. These results will be mentioned when it provides insight to the model performance.

Individually, each model appears to produce plausible traffic counts that reflect more or less the real traffic counts that were the model input. This is remarkable considering that the *A* matrix contains an entry in less than 5% of its elements, i.e. the whole was generated from 5% of its parts. This is the essence of compressive sampling/network tomography, that the network contains enough information to allow a model to reconstruct some semblance of the source of the flow.

Certain output OD flows seem implausible, however, in particular because some models tend to produce many OD flows of zero for extended time periods, which is unrealistic. The OD flows do not agree well across models.

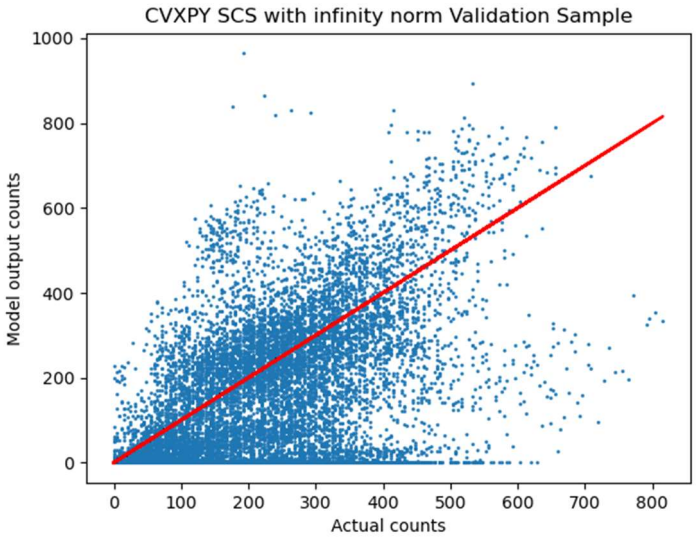
Information is either not available in the *A* matrix or is not utilized effectively by the models to distribute the OD flows. The deviations are linked to the model type however and are not entirely random. The marginal and total sums of the generated counts also depart from reality in some models. At the very least, the total counts should be maintained in a well-specified model.

The models have to be compared with each other and evaluated heuristically, until other data can be obtained. But this can still lead to insights.

CVXPY

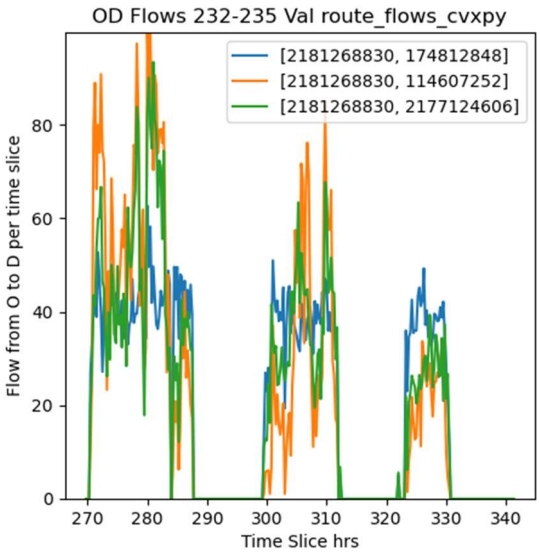
Runtime 6:40 minutes on CPU on the validation dataset. The input counts are reproduced with more fidelity than most models and the OD flows have no relations which carry zero vehicles(except at night), so the trips are distributed throughout the road network. The exact way in which the "conic" solver works is however still elusive to this author and a topic for another paper.

The counts and the OD flows output from the model are plotted for comparison with the toy models and other model types. Recall that in the toy model in which all edges had input counts, the estimated counts had zero MSE and lined up perfectly on the red line of true counts in the CVXPY model. Here, the CVXPY has estimated counts of zero for a portion of detectors which had real counts greater than zero (along x-axis):



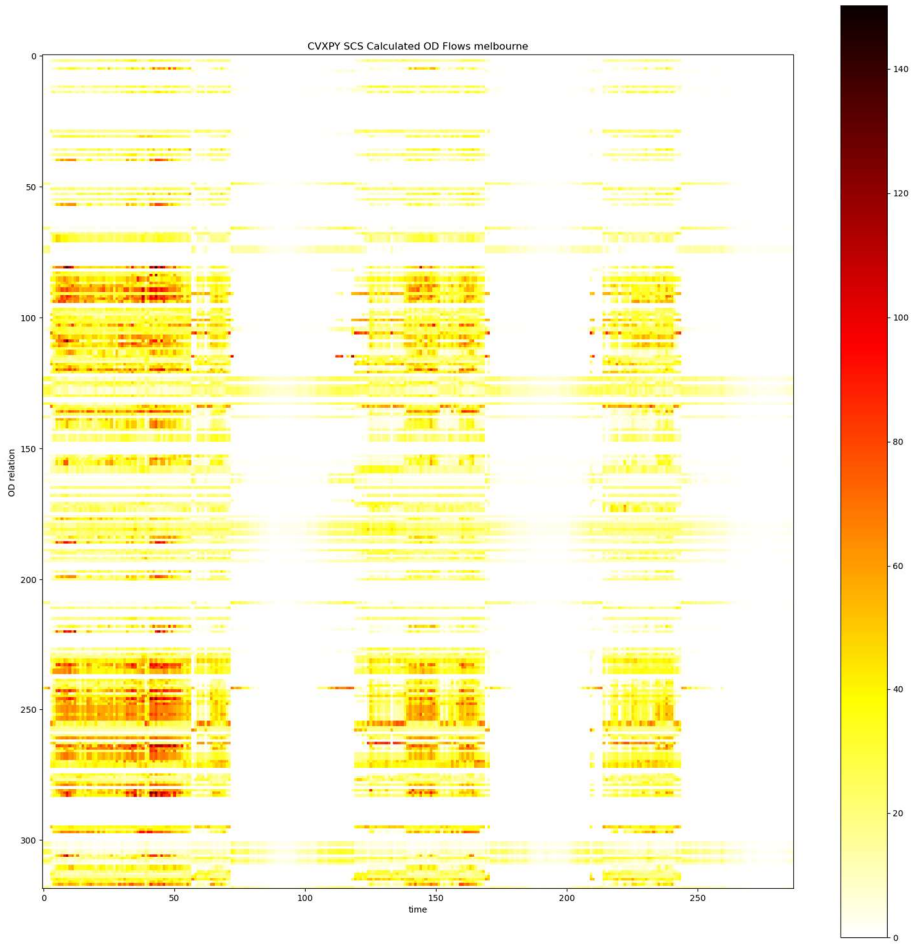
Estimated vs. actual counts on the edges for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

The origin-destination matrix is qualitatively evaluated in a heatmap of all OD pairs vs. time and by plotting time series of demand on selected OD relations. The demand vs. time reveals periods of zero traffic demand at night, but morning and evening travel peaks that are in the input counts are evident. The demand is distributed evenly across all three of these OD relations.



Estimated OD demand vs. time for all OD pairs and for selected OD pairs for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD

The heatmap of OD demand vs. time reflects the time variation of the count data, shows very low or zero values at night (vertical white bands) on many relations and also a moderate degree of peak demand concentrating on about 30% of favored OD relations(horizontal red bands).

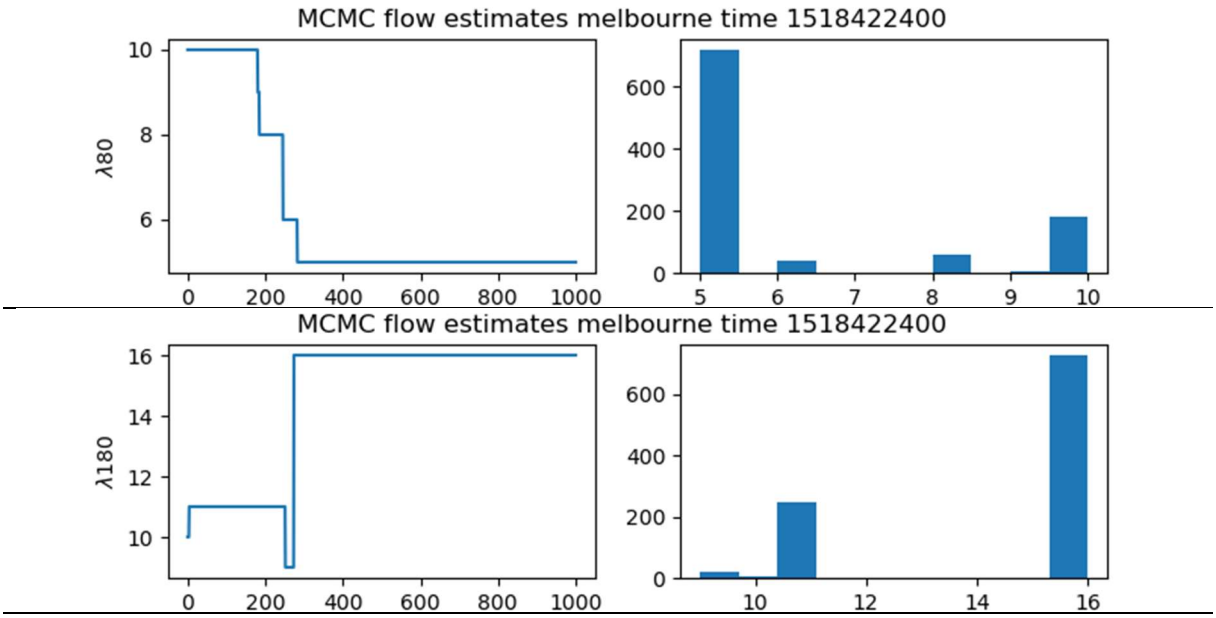


Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

MCMC

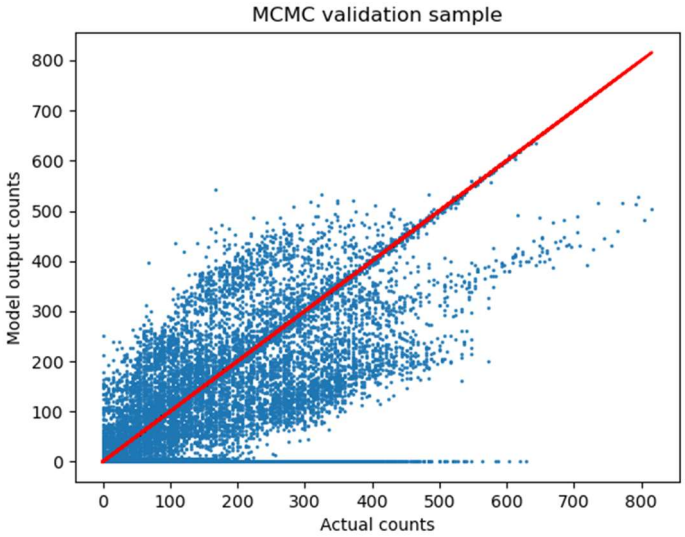
Runtime 892 minutes (ca. 15 hrs) on CPU on the validation set. The estimated-input counts have a smaller MSE compared to the CVXPY solver and there are also no zero-valued OD relations. The maximum value of the OD matrix is 607 (vehicles per 15 minute period), so there is more spreading of the traffic around the matrix in the CVXPY solution and more concentration in the MCMC solution on certain OD relations.

The tuned Metropolis-Hastings sampler provides ample opportunity for the model to improve (learn?) before tuning of the dispersion is suspended for the last 400 draws. Traces of selected route flow mean values illustrate that the model is "trying" to find different solutions but is very stable after only a few hundred draws.



Traces of the training of mean flow values in the MCMC model for routes 80 and 180. There are 627 such parameters in this model. Metropolis-Hastings tuning is performed for the first 600 iterations.

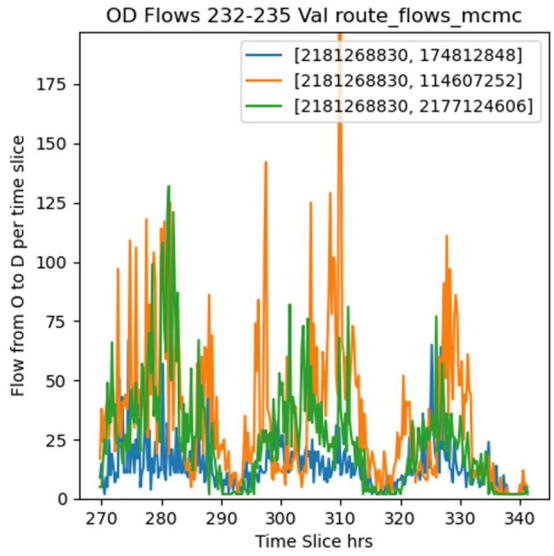
The counts generated by the MCMC are in better agreement with the input counts than in the CVXPY model. Clusters seem to exist where MCMC has overestimated the counts by double around (200,400), again in the region of the plot (300, 150) where the MCMC has underestimated by half, further along a linear pattern just below the red line, and a clear segment of nonzero real counts which the MCMC has estimated at zero. Time did not permit further examination of systematic trends in these clusters, for example if they may be linked to network topology, MCMC initialization, etc.



Estimated vs. actual counts on the edges for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

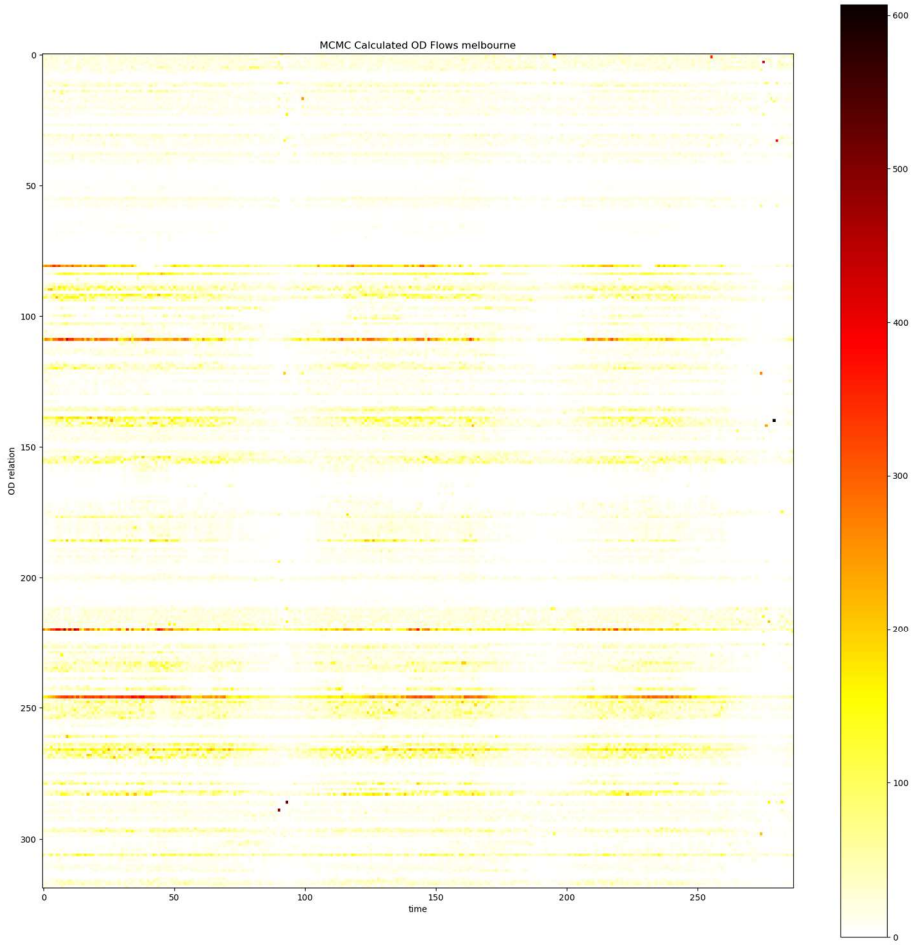
The slice of the OD matrix vs time is noisier than the CVXPY solution and does not go to zero at night, but has general daytime peaks. Due to the noise, the signature of morning and evening demand peaks is not clearly discernable. The tall spikes of double the height of the peaks on the relations

sampled from the CVXPY model contrast with the broad, rounder curves of the CVXPY. The CVXPY curves are more realistic and qualitatively follow the count curves more closely, except for the nighttime zeros. The MCMC has allocated the demand to two of the OD relations evenly, while one has about 25% as much demand. This may be realistic or not, but it differs from CVXPY. The orange trace seems not to be in time step with the other two traces. Time slice is not an explicit variable in this model (the MCMC iterates through all the route flows for a time slice, then proceeds to the next, without carrying over any information) and there is no constraint to keep the output route flows consistent with any notion of a time series, so time distributions in the final output are free of any constraints in this respect (the flow in the time slice would of course be conserved and displaced to and from other OD's in the time slice).



Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD

The matrix displayed as a heatmap reveals a pattern with some extreme peaks (very dark specks scattered throughout the pattern) with a generally evenly distributed demand in time and OD pair. Peaks and troughs occur in the expected diurnal pattern without the completely white vertical bands indicating zero night demand, but there are noisy outliers. Note the extremely high peak value in the color scale to accommodate the outliers, which makes the otherwise evenly distributed demand in very pale yellow more difficult to recognize. The MCMC solution has the OD relation with the highest maximum value of any model at 607 vehicles/15 min period

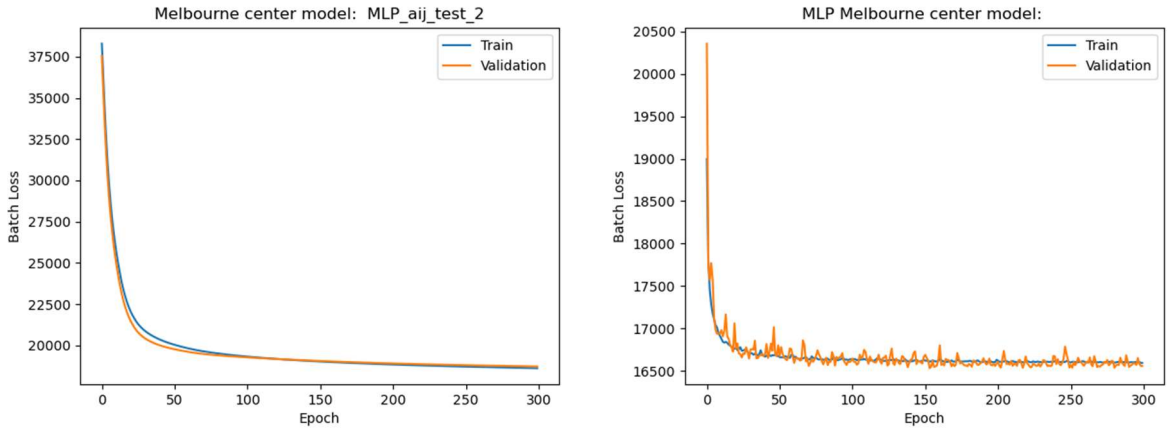


Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

MLP

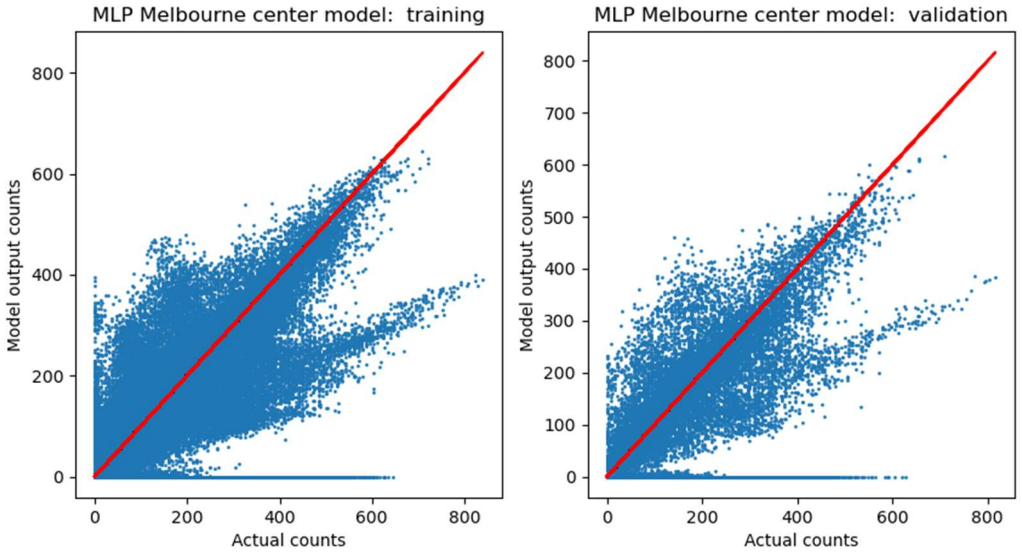
Runtime in batch is 1 minute 20 for training and validation sets for batch size 10. The disperson of the estimated counts is still lower than CVXPY or MCMC and drops even more if more routes per OD are added. The run for 25 routes per OD and batch size 32 gave the smallest MSE of estimated edge counts and took only 2 minutes 37 seconds to run to 500 epochs. The OD demand from the MLP models however is much noisier than in the other solvers and contains a significant proportion of relations with zero flow, usually one-third to one-half of the relations. Thus the flows are concentrated on a few favored OD relations. Zero-flow between origins and destinations is an unlikely occurrence in reality.

The model loss (MSE loss with a weighted *l*1 regularizer term (coefficient 0.01)) plots show consistent learning that stagnates after several hundred epochs. Varying the learning rate with an exponential cyclic schedule lowered the ultimate loss but introduced roughness to the validation predictions, so whether a particular validation run shows better or worse agreement with the inputs depends on precisely which epoch the run is stopped at, and appears to be random in a cyclic learning rate model. Runs to 3000 epochs yielded very little improvement. The agreement between validation and training loss depends on the selection of the training and validation datasets. The validation loss is higher than the training loss by a constant value for runs with more than two routes per OD.



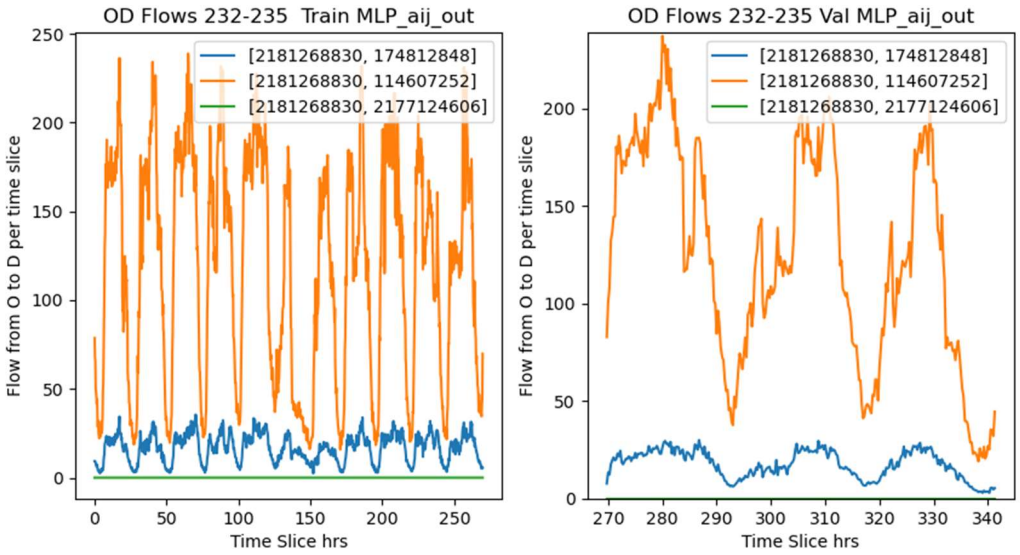
MLP training and validation loss (MSE of counts in vs. counts out plus weighted /1 regularizer term) with fixed and cyclic learning rates (exponential schedule) for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

The counts generated in the MLP are noisier than those from the previous models. The point swarm has four distinct "arms" indicating perhaps biases in the estimation of flow for particular times and/or OD relations: one tends to overestimate counts, one matches the inputs well, one underestimates in a linear dependency, and one section of the output consistently returns zero flow.



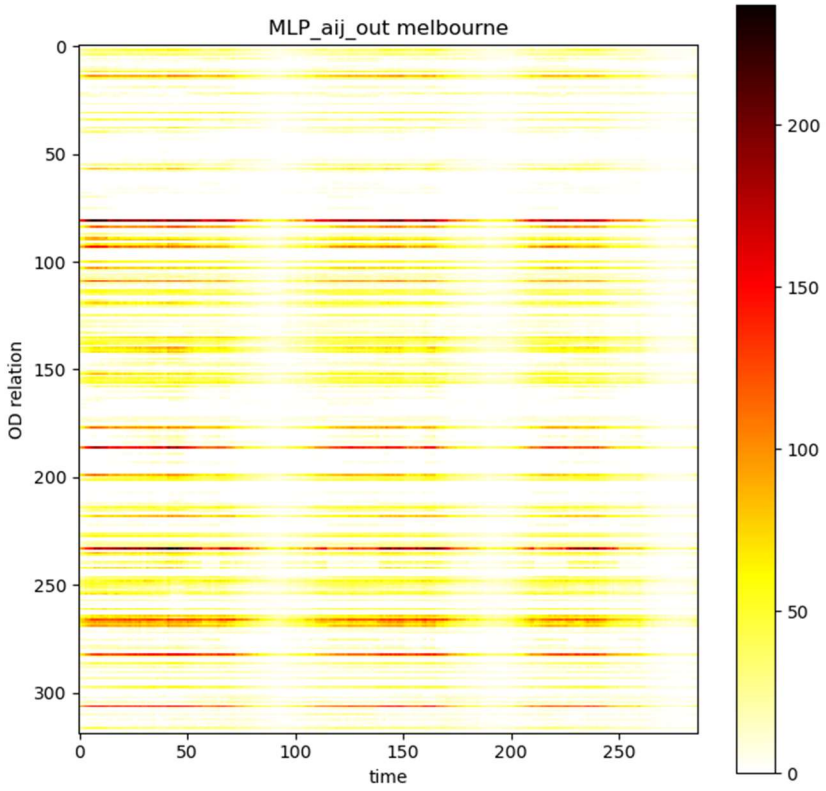
Estimated vs. actual counts on the edges for the Melbourne training and validation dataset, 2 routes per OD, fixed learning rate.

The OD flow on relations 232-235 is different from both the MCMC result and the CVXPY result, with realistic diurnal demand peaks and a suggestion of morning and evening peaks, and no zeroing-out of demand at night, but one of the demand relations clearly has zero value (no demand) for the entire time periods of both the training and validation sets.



Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD

The heatmap of OD demand vs. time shows several strongly favored (red) OD relations which also generate most of the traffic at night. Some OD relations are white all the way across the plot, indicating no or very low demand here. The dark specks indicating random high peaks that were evident in the MCMC plot are not seen here, but the demand is not distributed among the origins and destinations as evenly as in the CVXPY model. The scale of the plot colors is 100 counts higher than the CVXPY model because of systematic high demand peaks.

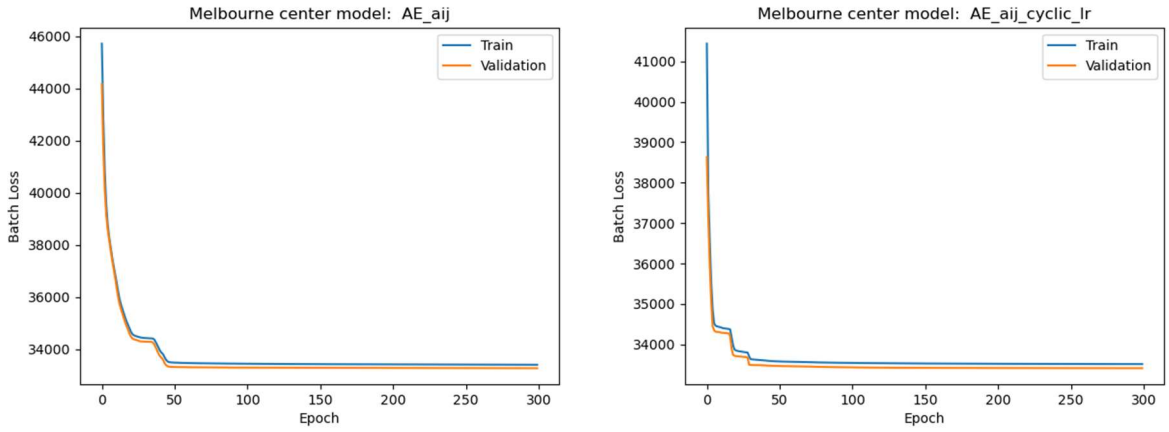


Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

Autoencoder

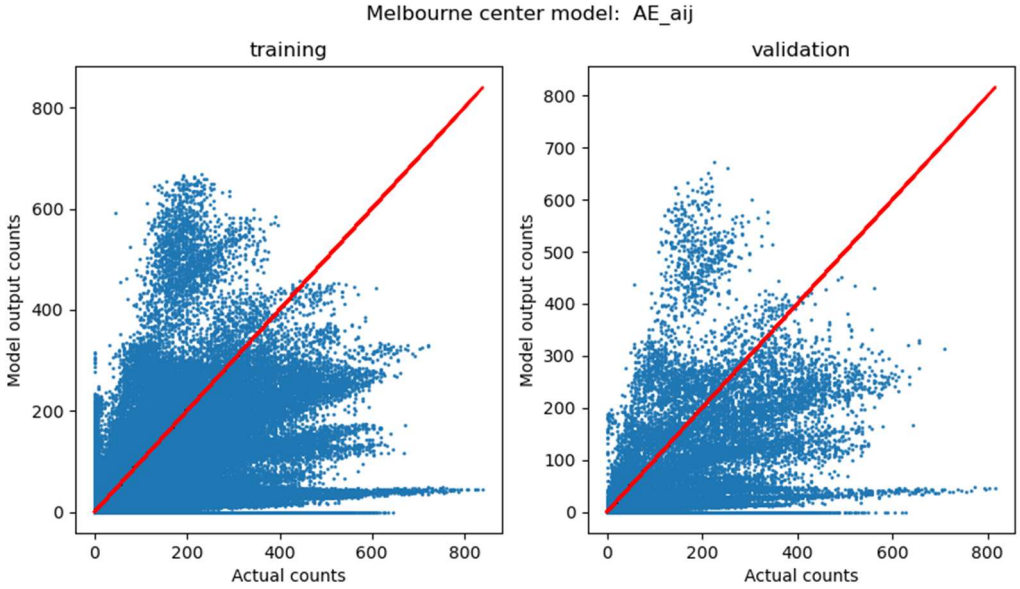
The runtimes are shorter than for the MLP. The dispersion of the counts is highest of all model types, and the distribution of the OD flows is the most-even across the relations, with the lowest-value maximium peaks and a very low incidence of zero-value flows on OD relations. Both the total number of counts and the total number of trips on the OD relations grossly underestimates the input values. At least the total counts on the network should be reproduced in a model, even if the distribution on network routes cannot be relied upon. The autoencoder is a long way from tuned or properly specified for these data.

In contrast to the MLP, a cyclic learning rate schedule did not improve the ultimate loss in the learning curve for the Autoencoder. But the validation prediction is not as noisy.



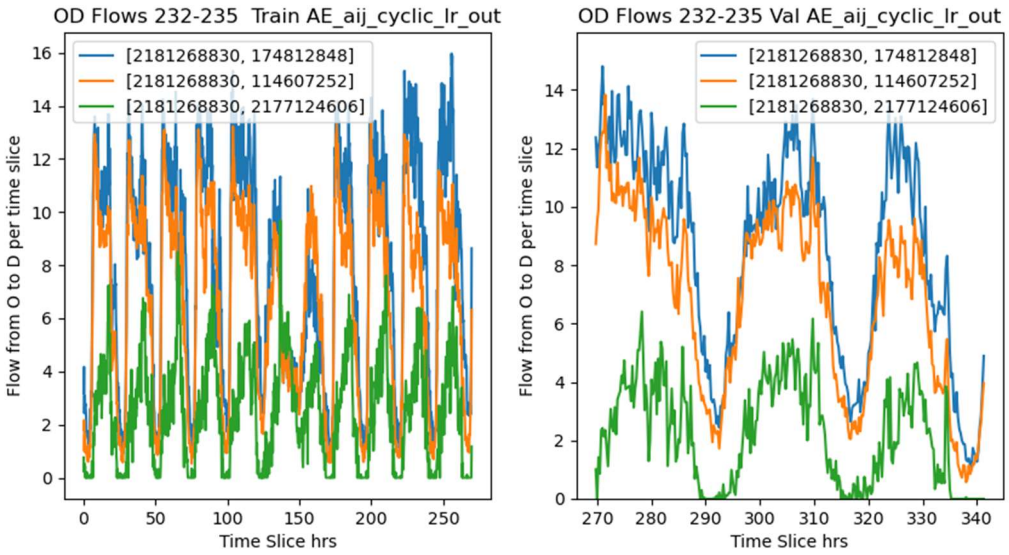
AE training and validation loss (MSE of counts in vs. counts out plus weighted /1 regularizer term) with fixed and cyclic learning rates (exponential schedule) for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

The predicted edge counts from the AE model also exhibit tendencies to over- or underestimate the real counts in a systematic fashion, with “arms” or “branches” appearing in the point swarm. The RMSE of the estimated vs. actual counts is the same for fixed or cyclic learning rates.



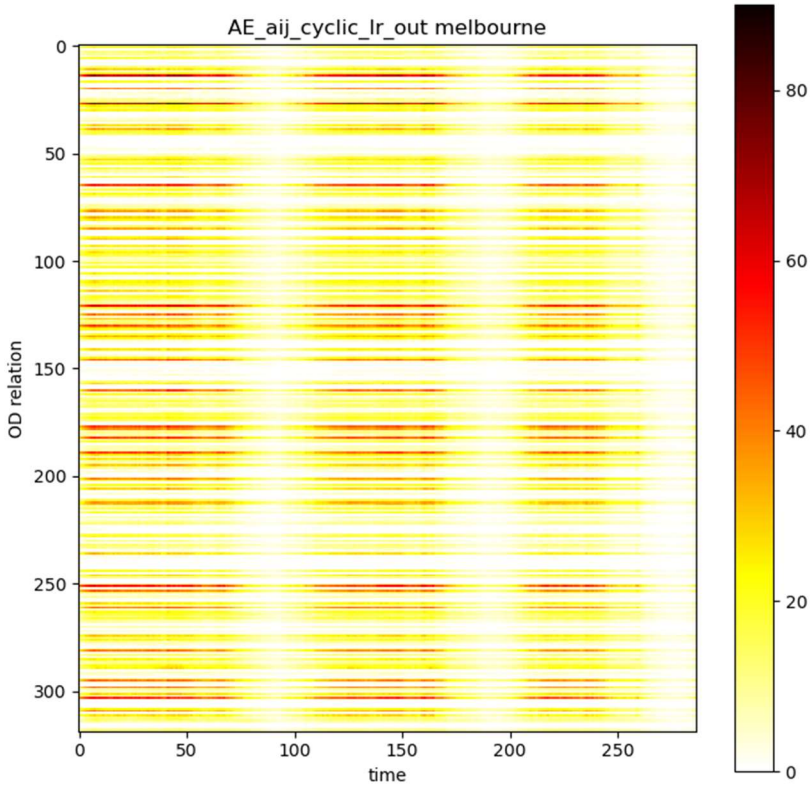
Estimated vs. actual counts on the edges for the Melbourne training and validation dataset, 2 routes per OD, fixed learning rate.

At first glance, the time profile of demand on selected OD relations appears plausible, with no zeroed-out nighttime demand and no extreme demand peaks. The noise in the signal makes it impossible to detect AM or PM peaks, but the diurnal patterns are obvious and aligned with each other temporally. However the magnitude of the demand values are highly incongruous with the other models, approximately 5 to 10 times lower.



Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD

A glance at the heatmap of demand shows that this finding of very low demand on the relations 232-235 may not be indicative of a general underestimate of total demand. Instead, the AE has distributed demand across all relations much more evenly than was seen in the other models. The moderate peaks (and color scale) of the plot illustrate a broad yellow tone of demand in the 20-30 vehicles per 15-minute time slice on about 75-80% of the OD relations, with approx. 15-20% generating consistent higher demand and a small number of OD relations contributing nearly zero.



Estimated OD demand vs. time for all OD pairs and for a selected OD pair for the Melbourne validation dataset of 3 days of measurement in 15 minute bins, 2 routes per OD.

Discussion

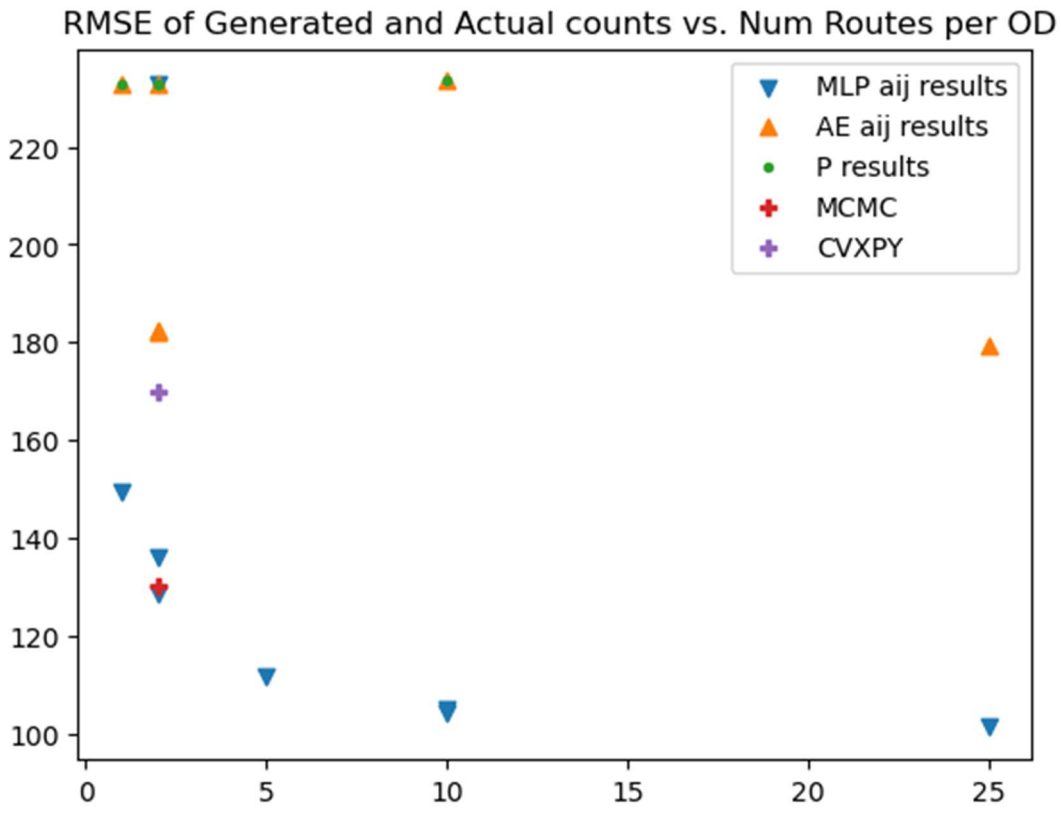
The model types tested here on the same base case yield very different results. The concern is that the model specification, regardless of solving technique, is underdetermined and the solution is subject to random fluctuation. The different results could be random. Or, the different solution techniques could themselves be driving the results to one or another pattern. It is important to know why the results have come out this way and what can be done to ensure more consistency. In the literature, authors add more and more diverse data sources to the models to validate them, above all to properly distribute the trips on the routes: floating car data (analagous to sending data packets through an internet infrastructure for compressive sensing in signal processing), calibrating on real origin and destination counts, initializing the model weights on real but out-dated OD demand, feedback of microscopic models based on the calculated OD demand, etc. Lacking these sources and keeping to the purity of the challenge to specify models of OD demand from counts only, this discussion proceeds with an investigation of and between the models, themselves.

An attempt was made to systematically analyze the base cases and the cases with varying routes per OD and a different model for the adjacency of counted edges and routes. The aim was to analyze the information in the OD demand distribution across relations and time. The table lists each model and a set of statistical indices comparing the real-value OD matrices. Some of the indices are meant for integer-value arrays and may not be appropriate, for example Mutual Information (MI).

	modelname	nroutes	rmsey	rmse	rmsn	mse	mae	mape	mi	max	mean	zero	odzero
0	route_flows_mcmc	2	130.31	27.13	3.38	736.18	13.33	1,051,292,323,578.77	2.55	607.00	13.79	0.00	0.0
1	route_flows_cvxpy	2	169.81	0.00	0.00	0.00	0.00	0.00	5.53	150.21	8.04	0.00	0.0
2	MLP_aij_out	2	136.07	23.10	2.87	533.71	12.68	855,419,201,625.68	3.33	237.08	12.34	0.45	0.2
3	MLP_aij_cyclic_LR_out	2	128.45	28.21	3.51	796.06	13.27	11,792,447,755.82	2.31	388.15	12.13	0.62	0.4
4	MLP_aij_test_25_out	25	101.32	31.01	3.86	961.91	14.83	1,460,839,466,052.02	3.64	428.46	14.30	0.39	0.1
5	MLP_aij_cyclic_lr_1_route_out	1	149.49	27.30	3.40	745.47	13.07	11,479,418,995.85	2.12	328.93	10.69	0.66	0.4
6	MLP_aij_cyclic_lr_5_route_out	5	111.75	31.07	3.87	965.37	14.57	65,069,967,812.92	2.74	502.89	13.10	0.55	0.1
7	MLP_aij_cyclic_lr_10_route_out	10	105.06	30.42	3.78	925.18	13.86	528,218,315,100.22	2.94	538.45	12.76	0.51	0.1
8	MLP_10_routes_cyclic_lr_out	10	104.12	30.71	3.82	942.96	14.05	535,532,190,970.80	2.73	482.72	12.83	0.55	0.2
9	AE_aij_out	2	182.10	16.24	2.02	263.78	10.36	390,196,202,615.14	4.36	83.68	8.40	0.27	0.2
10	AE_aij_cyclic_lr_out	2	182.47	16.76	2.09	281.02	10.64	456,671,352,410.61	4.31	90.08	8.30	0.28	0.1
11	AE_aij_test_25_out	25	179.51	14.10	1.75	198.72	9.08	274,235,255,456.17	5.52	22.76	6.34	0.01	0.0
12	MLP_P_mat_out	2	232.89	17.40	2.17	302.75	9.86	178,370,438,594.82	3.07	136.47	5.46	0.50	0.2
13	AE_P_mat_out	2	232.95	23.20	2.89	538.27	14.65	953,215,758,435.03	3.11	115.37	11.68	0.49	0.4
14	AE_P_mat_test_1_out	1	232.94	16.01	1.99	256.29	9.00	242,879,961,851.35	3.08	53.78	4.30	0.50	0.3
15	AE_P_mat_test_10_out	10	233.38	55.49	6.90	3,079.06	31.54	1,532,781,754,413.78	3.17	425.23	31.00	0.48	0.4

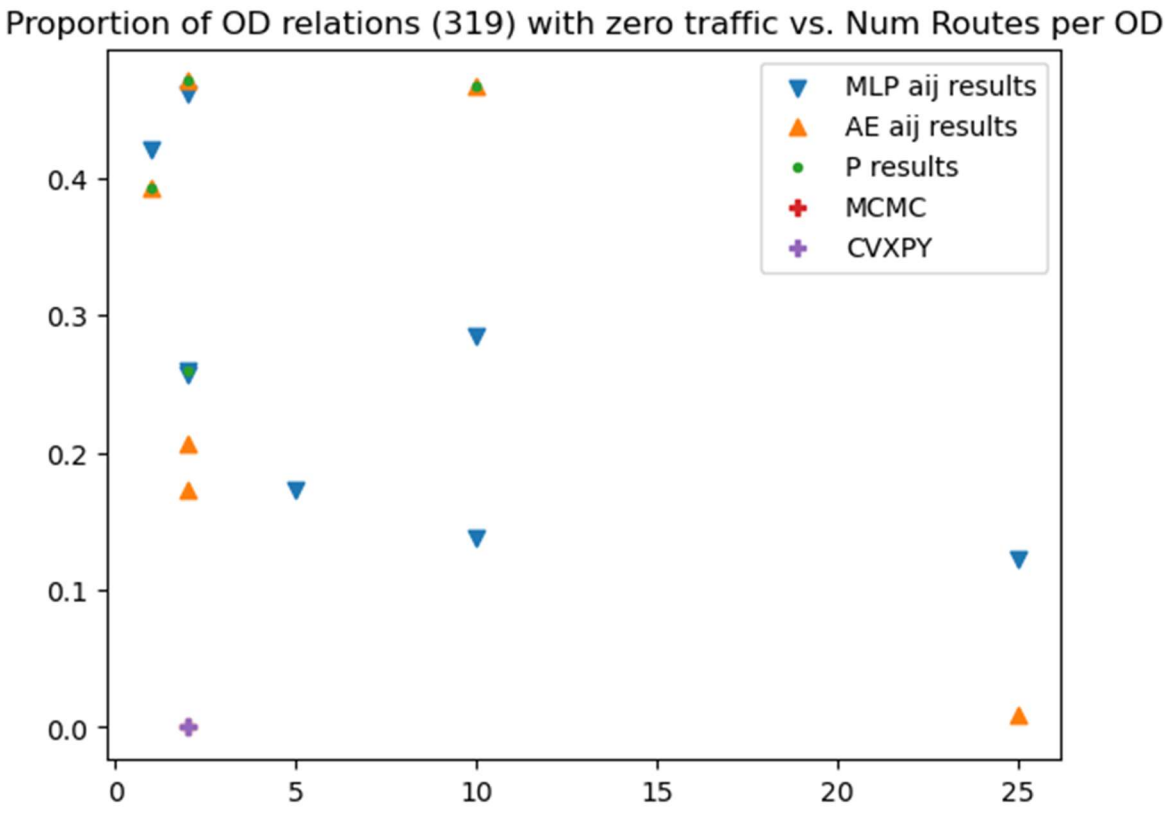
Measures of comparison of the model-generated counts compared to inputs (RMSEy) and of OD flows, relative to the CVXPY model.

A plot of the RMSE of the edge counts generated by the models with respect to the input counts of actual road measurements in the validation dataset shows that the MLP tends to generate better agreement with the inputs, followed by MCMC and then CVXPY. Using more routes per OD relation in the modelling leads to even better agreement with the inputs for MLP. There even seems to be an analytical curve delineating this improvement as though it's a mathematical function underlying the logic of the model and the contents of the *A* matrix. With more time this could be ascertained for certain. The Autoencoder produces very poor agreement between generated counts and the input counts. It does not matter if the *A* adjacency matrix or the *P* matrix is used.



The dispersion of edge counts estimated by the models relative to the input counts (measurements) in the validation dataset, plotted versus the number of routes per OD used in the model.

A plot of the proportion of OD relations with zero demand throughout the entire time period of the validation set shows less certain trends. Recall even that the MLP calculated zero demand for some OD relations for the entire training period, as well. It is realistic to expect some traffic on all the routes in the model over a time period of 287/4 =71.75 hours, especially if it is known that not nearly all the possible real routes were included in the model, i.e. that a surplus of traffic has to be assigned to fewer routes than in reality.The number of routes per OD plays a less predictable role in the result, and the same applies to the model type. However the *P* matrix solution always gives the poorest result in this regard. The Autoencoder with 25 routes per OD has almost no zero-value OD relations but the MLP has fewer than AE in models with between 1 and 10 routes per OD, so there is no pattern there. The tendency to fail to assign traffic to some routes at all, with the result that some OD relations have no traffic, is a major error in these models that must be remedied for them to be useful.



The proportion of OD relations (there are 319) with zero flow over all 287 time slices in the validation set, versus the number of routes per OD that were used for calculating the OD matrix.

The other columns in the table all compare the OD demand matrices to the CVXPY result as a benchmark. Given the nighttime zeros, this may not be a good choice. But the relative measure at least permits systematic grouping of the results to gain insights into the tendencies of the models.

The MLP models group as being different from the CVXPY and with similar statistics to the MCMC. The AE models are more similar to CVXPY, with the exception of the "AE_P_mat_out".

Clearly no conclusions can be drawn apart from the *P* matrix being unreliable without further investigation, the AE is not yet tuned, and there is an indication that supplying the models with more information (more routes per OD with probabilities for route choice) can reduce randomness in the results.

Improving the Models

It may be plausible to perform more tests with the CVXPY solver with more routes per OD, or to install a different mixed integer solver for CVXPY, or to run the solver with a (correct) *P* matrix.

It is however not plausible to re-run the MCMC for more scenarios, as this requires runs of 15 hours or more. The MCMC was run without the regularization parameter for sparse problems and may be improved with its addition.

Tuning the DNNs

Regularization is needed when the model overfits and the error on the test set is greater than that on the training set. The term "regularization" refers to measures aimed at reducing the generalization error but not the training error. Common methods include dropout (eliminating nodes of the neural network that aren't activated with high weights), early stopping conditions, constraining the weights

of the neural network layers (by using a threshold, for example), adding noise in the input or in model layers, tuning hyperparameters to improve the fit to a validation set, etc.

An underfit model fails to fit either the train or the validation sample. This seems to be more the case, here. All model types fit the toy models very well. The toy models were characterized by having a very high proportion of counted edges in the transportation network. The use of the very sparse adjacency matrix without routing preferences expressed in the matrix (apart from the hierarchical method of adding more routes to the model in the order of their shortest travel time) is the cause of the high MSE in the data-driven models. The cyclic learning rate did not succeed in bringing the models to significantly lower losses. It would seem that, indeed, the only way to improve the models, once tuned as well as possible to the data, would be to add more data to constrain the solution.

Marginal improvement in model fit could be made by automated parameter sweeps through the models and by experimenting with more model layers, particularly in the Autoencoder.

Improvements to the Model Specification

The most pressing problem is the introduce the model to hierarchical route preferences for distributing the demand realistically across the road network. The implementation of the *P* finite memory Markov transition matrix does not seem to work correctly, despite verification with the toy model of Dey et al. Other implementations for expressing route choice probabilities inside the solver should be tried, as well.

As the results attempt to find the most probable solution in a field of infinite possibilities, the modelling structure seems to be calling for an ensemble approach in which the best solution is chosen from many runs with different random seeds. This method would not introduce any more information into the modelling but would sift through the noise.

Another possible way to increase the information provided to the models would be to simulate traffic in the network. Staying at the macroscopic, abstract level, it may be most simply achieved by introducing travel time (a form of dynamic route choice probability) into the *A* adjacency matrix (*P* route probability matrix) and iteratively updating this according to the flow on each edge. The travel time is constant up to a certain volume of traffic, then it quickly increases at a critical volume. Such well-known macroscopic flow dynamics could be built into this model without much effort. Dey et al. takes this further step, for example.

Alternatively, the model could be cast as an origin flow generation problem like Dey et al. (2020)'s extension of the Method of Moments (Vardi 1996), where arrival times at destinations were explicitly calculated and the destination arrival rates did not match the origin generation rates in a time slice. This treatment would allow traffic to remain on edges from a previous time slice and be added to the traffic of the current slice, similar to a route travel time model but just adding flow to previous flow. The result is a new distribution of demand across time slices, or an implicit dependence between time slices. The solution is built into the *P* matrix that needs to be explicated further anyway.

The use of realistic spatial planning zones could help distribute trips more evenly across routes. These zones need not correspond to official zones used by planners (these can be obtained but it usually requires negotiation, credentials, well-justified reasons, etc.) but can be abstractions as a test of how the model might improve by having several entry and exit points to the network for each origin and destination. It would correlate when and where demand entered and exited the network, since they would sum to a single origin or destination, rather than independently assigning trips to all the nodes. A package called "grid2demand" is available to generate not only zones but also initial

demand matrices from OSM structural data, for example. This package relates the use category and size of building objects in the OSM database to a database of trip generation in U.S. cities.

An iterative bi-level model could tune the model parameters better. The CGAME (Li et al 2022) approach replacing an explicit traffic flow model with a graph matching DNN between the latent spaces of the OD matrix and the input counts would be an elegant start and not a long reach once an Autoencoder is developed which provides more stable results. Ou et al. (2019) structure a machine learning solution around the bi-level problem with a genetic algorithm that chooses the best OD matrix to fit the counts, which is more effort but also an exciting coding challenge.

Finally the glass wall would have to be broken to introduce other datasets to the model for calibration: an existing OD matrix for the region and zones of the study, floating car data (Uber, taxis, GPS from mobile phones) or identifiable movements from Bluetooth sensors, license plate recognition cameras, etc. Data like this is partly available for free online (some on OSM), some can be obtained free with a request or a license, and much of it is proprietary and either not available or available only for a fee, with restrictions on use and publication.

The main takeaway from this effort is that it is possible to use compressive sampling methods to reconstruct demand matrices from counts on a road traffic network and a model of the network topology. But the counts are too sparse to make a complete or very useful picture and indications of route choice preference as an intermediate step toward OD demand must be built into the model to help it along. The solution methods from signal processing, which use signals of electromagnetic waves and/or electrons, cannot be used without adaptation to queueing theory. The handful of papers published in this area that report success make use of other data, iterative techniques, or restrict themselves to toy models in order to study the statistics of the problem. There seems to be a lot yet to learn about the modelling and a lot of potential in the data.

References

Abrahamsson, Torgil, Estimation of Origin-Destination Matrices Using Traffic Counts – A Literature Survey, Interim Reports, International Institute for Applied Systems Analysis, May 1998.

Airoldi, E. M. and Blocker, A. W. (2013). Estimating latent processes on a network from indirect measurements. J. Amer. Statist. Assoc. 108 149–164. MR3174609

Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

Bauer, D. , G. Richter, J. Asamer, B. Heilmann, G. Lenz and R. Kölbl, "Quasi-Dynamic Estimation of OD Flows From Traffic Counts Without Prior OD Matrix," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 2025-2034, June 2018, doi: 10.1109/TITS.2017.2741528.

Behara Krishna Nikhil Sumanth, Origin-Destination Matrix Estimation using Big Traffic Data: A Structural Perspective. PhD Thesis, Queensland University of Technology, 2019.

Bell, M. G. H., "The estimation of origin-destination matrices by constrained generalized least squares," Transportation Research Part B: Methodological, vol. 25, no. 1, pp. 13–22, 1991.

Bera, Sharminda and K. V. Krishna Rao, "Estimation of origin-destination matrix from traffic counts: the state of the art." European Transport/ Trasporti Europei 49 (2011): 3-23

Boeing, G. 2017. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems* 65, 126-139. doi:10.1016/j.compenvurbsys.2017.05.004

Cascetta E. , "Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator," Transportation Research Part B: Methodological, vol. 18, no. 4, pp. 289–299, 1984.

Cascetta,Ennio , Andrea Papola, Vittorio Marzano, Fulvio Simonelli, Iolanda Vitiello, Quasi-dynamic estimation of o–d flows from traffic counts: Formulation, statistical validation and performance analysis on real data, Transportation Research Part B: Methodological, Volume 55, 2013, Pages 171-187, ISSN 0191-2615, https://doi.org/10.1016/j.trb.2013.06.007. (https://www.sciencedirect.com/science/article/pii/S0191261513001069)

Dey, Subrasankha., Stephan Winter and Martin Tomko, "Origin-Destination Flow Estimation from Link Count Data Only", MDPI sensors, 2020. Sensors **2020**, 20, 5226; doi:10.3390/s20185226

Diamond, Steven and Stephen Boyd, CVXPY: A Python-embedded modeling language for convex optimization, Journal of Machine Learning Research, 2016, V17 Nr83, pages 1-5.

Guo, Shengan, Youfang Lin, Ning Feng, Chao Song, Huaiyu Wan, "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting." 2019 ASTGCM Association for the Advancement of Artificial Intelligence (www.aaai.org).

https://github.com/xiaoxiong74/TrafficFlowForecasting/blob/master/papers/2019%20ASTGCN%20AAAI_Attention%20Based%20Spatial-Temporal%20Graph%20Convolutional%20Networks%20for%20Traffic%20Flow%20Forecasting.pdf

The code has been released at: https://github.com/wanhuaiyu/ASTGCN.

And for pytorch: https://github.com/wdzhong/ASTGCN-PyTorch

Kamyab, Shima, Zohreh Azimifar, Rasool Sabzi and Paul Fieguth, Deep learning methods for inverse problems, 2022 PeerJ Computer Science, May 2022, DOI 10.7717/peerj-cs.951.

Kurzhanskiy, Alex A., A Methodology for Estimating Vehicle Route Choice from Sparse Flow Measurements in a Traffic Network, MDPI mathematics,2022, 10, 527.

Li, Guanzhou, Yujing He, Jinaping Wu, Cyclic Graph Attentive Match Encoder (CGAME): A Novel Neural Network for OD Estimation. arXiv:2111.14625v4 [cs.LG] 23 August 2022

Loder, A., Ambühl, L., Menendez, M. and Axhausen, K.W. (2020) UTD19: Urban traffic data from 40 cities, , , Institute for Transport Planning and Systems (IVT), ETH Zurich, Zurich.

Mussone Lorenzo,GRANT-MULLER Susan,Chen Hai-bo. A Neural Network Approach to Motorway OD MatrixEstimation from Loop Counts[J].Journal of Transportation Systems Engineering and Information Technology,2010,10(1):88-98.

Ou, Jishun, Jiawei Lu, Jingxin Xia, Chengchuan An, Zhenbo Lu, "Learn, Assign, and Search: Real-Time Estimation of Dynamic Origin-Destination Flows Using Machine Learning Algorithms." IEEE Access Volume 7, 2019. DOI: 10.1109/ACCESS.2019.2901289 from researchgate.net

Sanandaji, Borhan M. and Pravin Varaiya, Compressive Origin-Destination Estimation, Transportation Letters, The International Journal of Transportation Research, April 2014.

Song, Chao, Youfang Lin, Shengnan Guo, Huaiyu Wan, Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting. Association for the Advancement of Artificial Intelligence (www.aaai.org). 2020. https://github.com/Davidham3/STSGCN/tree/master/paper

Syffen, Georgette, Ahmed Khalil, Ibrahim Ramadan, Origin Destination Matrix Estimation Based on Traffic Counts Using FMINCON Function in MatLab. Journal of Al-Azhar University Engineering Sector, 18(66), Jan 2023, pages 57-73.

Tebaldi, C. andWest, M. (1998). Bayesian inference on network traffic using link count data. J. Amer. Statist. Assoc. 93 557–576. MR1631325

Vahidi, Milad , Yousef Shafahi, "Time-dependent estimation of origin-destination matrices using partial path data and link counts," https://doi.org/10.21203/rs.3.rs-2225000/v1 Creative Commons November 30th, 2022. Least squares.

Van Zuylen, H. J. and L. G. Willumsen, "The most likely trip matrix estimated from traffic counts," Transportation Research Part B: Methodological, vol. 14, no. 3, pp. 281–293, 1980.

Other papers whose summaries were promising but which I could not get due to not having a university IP:

Mussone, Lorenzo, Matteo Matteucci, "OD Matrices Network Estimation from Link Counts by Neural Networks", Journal of Transporation Systems Engineering and Information Te chnology, V13#4 2013 p84-92.

Nasab, Mojtaba Rostami, Yousef Shafahi, "Path flow reconstruction problem", Transportation 47, 2923-2950 (2020) → Spiess' Model

Pamula, Teresa ,Renata Zochowska, Estimation and prediction of the OD matrix in uncongested urban road network based on traffic flows using deep learning, Engineering Applications of Artificial Intelligence, V117, Part A January 2023, 105550. https://doi.org/10.1016/j.engappai.2022.105550

Renn Qianqian, Yang Li, Yong Liu, Transformer-enhanced periodic temporal convolution network for long short-term traffic flow forecasting,https://doi.org/10.1016/j.eswa.2023.120203 https://www.sciencedirect.com/science/article/abs/pii/S0957417423007054

Appendix

Models:

- MCMC runtime 3 mins. per time slice (892 minutes for validation set)
- MLP aij 12:30 on train batch = 1, validation with cyclic LR
- MLP aij 10 routes 12:09 batch=1,
- MLP P_mat 12:23 2 routes batch =1
- AE runtime 11:20 on train, validation with cyclic LR batch =1
- CVXPY – least squares runtime 6:40 on validation set
- EM
- Moments
- P_mat likely not useful as turn probability due to using only counted edges. Results correspondingly very small counts, many zero-value OD relations. Estimated counts seems to be 4x too small.

Datasets:

- Hazelton 1 "T" intersection 5 time slices real data 5/6 edges one route OD
    - Dey method in CS_08 (is it implemented right) compare with CS_06 which apparently has mistakes
    - Cvxpy and all results from Hazelton's paper CS_02
    - Cvxpy and Lasso fit analysis CS_01
- Hazelton 2 cross one time slice on 9 of 10 edges one route/OD (30) six nodes
    - Expectation Maximization CS_04
    - Cvxpy and Hazelton's result CS_03
- Dey 1 grid varying routes/OD one-way counts made up –> moments
    - Solved with Dey method and CVXPY in CS_07 with fake counts
    - Solved with CVXPY and DNN in DNN_2
- Birmingham
    - Demonstrate pretty graphs, multiple detectors per edge and cropping
    - Demonstrate data preparation ReadUTD19_02
    - ReadUTD19_03 has prep for MLP
    -
- Melbourne
    - Demonstrate grid with high concentration of counters
    - ReadUTD19_04 first try with huge melbourne graph
    - …05 not learning
    - …06 cvxpy verbose
    - MCMC_Poisson_04 has MCMC, MLP (aij and P_mat)

Notebooks:

- CompressedSensing1
    - Model 1 Hazelton no figure
    - CVXPY
    - With optimization of lambda (between 10^-1 and 10^0)
    - Infinity norm
- CompressedSensing3
    - Model 2 Hazelton no figure of network
    - CVXPY does not agree with MCMC of Hazelton
    - Neither mixed integer nor float
    - MCMC here compared to CVXPY
- CompressedSensing4
    - Model 2 Hazelton (9 edges one time slice)
    - Same CVXPY result as 3
    - Attempt at EM/Poisson, factorial blows up
- CompressedSensing7
    - Model 1 Dey as networkx (figure)
    - CVXPY
    - Dey method (check if P_mat is correct)
- CompressedSensing8
    - Model 1 Hazelton "T" as networkx (figure)
    - CVXPY
    - Dey method (check if P_mat is correct)
- MCMC_Poisson_03
    - Model 1 Dey, et al. with 5 days' simulated counts

    - MCMC counts, route flows, od flows
    - Compare with CVXPY
- DNN_2
    - Model 1 Dey, et al. with 5 days' simulated counts
    - CVXPY comparison
    - MLP with l1 norm for sparsity
- Download_OSM
    - PLACE_NAME
    - Save as graphml
- MCMC_Poisson_04
    - Read OSM from disk and clean
    - Read detectors, counts, links and clean
    - Make data model
    - CVXPY
    - MCMC
    - MLP
- AE_03
    - Shortened data model prep
    - AE or MLP
- Analysis2
    - Re-make any necessary charts
- Analysis3
    - Calculate group statistics of model output