# ETA Prediction with Graph Neural Networks in Google Maps

Austin Derrow-Pinion[1], Jennifer She[1], David Wong[2*], Oliver Lange[3], Todd Hester[4*], Luis Perez[5*], Marc Nunkesser[3], Seongjae Lee[3], Xueying Guo[3], Brett Wiltshire[1], Peter W. Battaglia[1], Vishal Gupta[1], Ang Li[1], Zhongwen Xu[6*], Alvaro Sanchez-Gonzalez[1], Yujia Li[1] and Petar Veličković[1]
[1]DeepMind [2]Waymo [3]Google [4]Amazon [5]Facebook AI [6]Sea AI Lab [*]work done while at DeepMind
{derrowap,jenshe,wongda,petarv}@google.com

## ABSTRACT

Travel-time prediction constitutes a task of high importance in transportation networks, with web mapping services like Google Maps regularly serving vast quantities of travel time queries from users and enterprises alike. Further, such a task requires accounting for complex spatiotemporal interactions (modelling both the topological properties of the road network and anticipating events—such as rush hours—that may occur in the future). Hence, it is an ideal target for graph representation learning at scale. Here we present a graph neural network estimator for estimated time of arrival (ETA) which we have deployed in production at Google Maps. While our main architecture consists of standard GNN building blocks, we further detail the usage of training schedule methods such as MetaGradients in order to make our model robust and production-ready. We also provide prescriptive studies: ablating on various architectural decisions and training regimes, and qualitative analyses on real-world situations where our model provides a competitive edge. Our GNN proved powerful when deployed, significantly reducing negative ETA outcomes in several regions compared to the previous production baseline (40+% in cities like Sydney).

## CCS CONCEPTS

• **Applied computing** → **Transportation**.

## KEYWORDS

Graph neural networks, MetaGradients, Google Maps

**ACM Reference Format:**
Austin Derrow-Pinion[1], Jennifer She[1], David Wong[2*], Oliver Lange[3], Todd Hester[4*], Luis Perez[5*], Marc Nunkesser[3], Seongjae Lee[3], Xueying Guo[3], Brett Wiltshire[1], Peter W. Battaglia[1], Vishal Gupta[1], Ang Li[1], Zhongwen Xu[6*], Alvaro Sanchez-Gonzalez[1], Yujia Li[1] and Petar Veličković[1]. 2021. ETA Prediction with Graph Neural Networks in Google Maps. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3481916

## 1 INTRODUCTION

Web mapping services such as Google Maps are invaluable tools for many users for interactively navigating areas of the Earth—especially so in metropolitan areas. On a daily level, such products

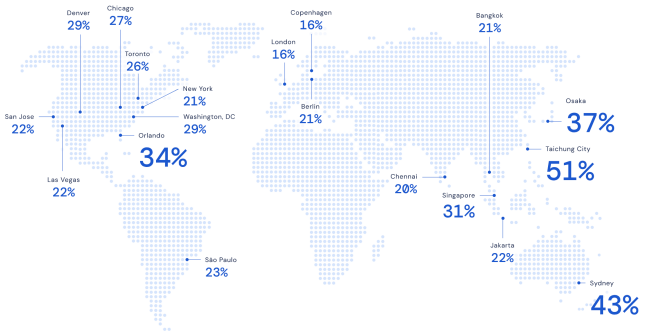


**Figure 1: Google Maps estimated time-of-arrival (ETA) prediction improvements for several world regions, when using our deployed graph neural network-based estimator. Numbers represent relative reduction in negative ETA outcomes compared to the prior approach used in production. A negative ETA outcome occurs when the ETA error from the observed travel duration is over some threshold and acts as a measure of accuracy.**

are used not only by everyday users finding the (optimal) routes between waypoints, but also by enterprises which rely on accurate mapping features (such as food delivery services, which may seek to provide accurate and optimised delivery time estimates). One critical task in this context is *expected time of arrival* (ETA) prediction: given current conditions on the road network, and for a particular candidate route between waypoints, predict the expected time of travel along this route.

The significance of such a predictor is clear: accurate ETA predictions allow traffic participants to make more informed decisions, potentially avoiding congested areas and minimising overall time spent in traffic. It is important to note that powerful ETA predictors need to meaningfully reason about conditions which take place in the *future*, and may not be directly obvious from current road state. As a simple example, the number of traffic participants may substantially increase during daily *rush-hour* periods, when most people commute to and from the workplace—but in many cases such situations may be more subtle. Further, there is a wealth of historical travel data available which may be used to support detection and exploitation on such subtleties. As such, this problem is substantially amenable to machine learning approaches.

As the road network is naturally modelled by a *graph* of road segments and intersections, ETA prediction is amenable to graph representation learning [1, 2, 10] approaches, particularly *graph neural networks* (GNNs) [8, 15, 25]. Here we present our graph neural network model for ETA prediction, which we deployed in


*CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia*

ACM ISBN 978-1-4503-8446-9/21/11.
https://doi.org/10.1145/3459637.3481916

production at Google Maps, observing significant reductions in negative ETA outcomes across all trips worldwide—above 40% in cities like Sydney—compared to the previous production baseline. See Figure 1 for improvements in additional regions. A negative ETA outcome occurs when the ETA error from the observed travel duration is over some threshold and acts as a measure of accuracy.

The task of arrival time estimation can be described as providing an estimated travel time given a user-provided starting location and a route suggested by a prior system. This problem is difficult because it needs to take into account both spatial information (captured within the road network) as well as temporal information (the evolution of traffic conditions over time). In particular, we need to anticipate that travel times across road segments may be affected by traffic conditions that are further away, or not immediately on the user track—potentially also accounting for "background" signals such as traffic light configurations. We also need to anticipate that traffic conditions may have changed by the time a user traverses to a road segment further along their route.

*Contributions.* We demonstrate how a powerful graph neural network estimator can be constructed for the ETA prediction task, while following simple design principles and carefully designed loss functions. Such a GNN then remains stable and performant when deployed in production, providing substantial gains against prior production baselines which did not use the road network graph reprsentations. Beyond this, our key contributions to this problem are as follows:

- Innovations on how the road network data is **featurised** and presented to the GNN;
- Innovations in the GNN **model design** and **operation**. While the GNN itself is constructed of standard building blocks, we found several benefits for the training regime (such as Meta-Gradients [31] and semi-supervised training [16, 26]) as well as architectural ablations (e.g. carefully tuned combinations of aggregators [5]) that proved particularly important for maintaining stability of the GNN in this regime and making it production-ready.
- Finally, we **deploy** the trained GNN model in production within Google Maps, observing strong **real-world benefits**. As displayed in Figure 1, we noticed clear quantitative improvements in negative ETA outcomes, but besides the on-line performance, we performed several offline ablations on various architectural and setup decisions, as well as visualisations of particular traffic situations where our model has a clear advantage over the prior baseline.

On one hand, our work represents another in a series of successful *web-scale* deployed applications of graph neural networks with real-time positive user experience impact [12, 18, 32, 33]. On another, we offer ablative studies and insights into the GNN model's predictions which simultaneously (a) elucidate the kinds of real-world traffic use cases where such a model may provide a significant edge; and (b) provide prescriptive advice for other practitioners in the general area of user-facing traffic recommendation.

## 2 RELATED WORK

We outline several related works that either directly inspire the building blocks of our proposed GNN architecture and pipeline, or sit more broadly in the space of GNNs for traffic forecasting.

*Graph representation learning.* Being a graph neural network, our work directly builds upon the previous years of progress in graph representation learning. In particular, our model is fundamentally based on the Graph Network framework [1] and follows the encode-process-decode [11] paradigm. This allows us to align better with the iterative nature of traffic computations, as well as pathfinding algorithms. Improving such an alignment is known to help GNNs generalise better to a diverse distribution of graphs [27].

*Travel-time prediction.* Given its wide applicability, travel-time prediction is a problem that has historically been closely studied under various settings. Earlier work modifies convolutional neural networks to be mindful of the spatial properties of the trajectory [28] and employs graph neural networks coupled with recurrent mechanisms [13]. The concurrently developed work of Yuan *et al.* [35] makes an important observation in this space: travel time is often largely dependent on the historical travel times in the given time-of-day—an observation that we extensively use. The recent CurbGAN framework [38] may also be of interest, with its leveraging of generative adversarial networks for evaluating urban development plans in the context of estimated travel times.

*Spatiotemporal traffic forecasting.* Using GNNs to forecast traffic conditions is historically a very active direction: most influential papers in the area of *spatiotemporal graph representation learning* rely on traffic datasets. The spatiotemporal setup typically assumes a graph structure over which each node maintains a *time series*—in the case of road networks, such time series typically correspond to historical speeds measured at regular time intervals. While our GNN model detailed here processes *static* inputs, and does not directly forecast future speeds, having some estimate about future traffic flow behaviour is likely to be meaningful for ETA prediction.

One of the first influential papers in the area was the *diffusion-convolutional recurrent neural network* (DCRNN) [17]. This paper illustrated how explicitly accounting for the graph structure provides a significant reduction in forecasting error over several horizons. This generic blueprint of a spatial GNN component combined with a temporal component yielded several subsequent proposals, including STGCN [34], GaAN [37], Graph WaveNet [29], ST-GRAT [19], StemGNN [3], and GMAN [39]. We refer interested readers to [23] for a survey of interesting problems and methodologies in the area.

Lastly, we find it important to mention that the assumption of the time-series being aligned across nodes does not necessarily hold in practice—we may instead expect data to be provided in an *asynchronous* fashion. A recent line of research on *dynamic graph representation learning* [20, 30] tries to explicitly account for such situations, both on the featurisation and the topological side.

*Graph neural networks at scale.* As graph neural network methodologies became more widely available and scalable [4, 9, 21], this also enabled a wider range of *web-scale* applications of graph representation learning. Perhaps the most popularised applications

concerned recommender systems, which are very naturally representable as a graph-structured task: with Pinterest being one of the most early adopters [18, 33]. GNNs have also been deployed for product recommendation at Amazon [12], E-commerce applications at Alibaba [32], engagement forecasting and friend ranking in Snapchat [22, 24], and most relevantly, they are powering traffic predictions within Baidu Maps [6].

*Training regimes and components.* While the core components of our architecture correspond to the Graph Network paradigm, ETA prediction in production invites particularly unstable training conditions across many batches of queries, particularly over routes of different scales. We adapted the MetaGradients [31] methodology from reinforcement learning, which allowed us to dynamically tune the learning rate during training and stabilise it across many uneven query batches, enabling a production-ready GNN model.

Further, there exist many patterns inherent to the *road network topology* that may be particularly pertinent to forecasting traffic flow. Similar motifs in the road network of the same region are likely to correspond to similar traffic dynamics—as such, automatically discovering and compressing them is likely to be useful. To enable such effects, we have directly adapted established unsupervised GNN methodologies such as graph auto-encoders [16] and deep graph infomax [26]—for both of them, we demonstrate benefits to more accurate ETA predictions.

Additionally, different kinds of heuristical computations over road networks may require different ways of *aggregating* information. For example, shortest path-finding would require *optimising* the signals over a node's neighbourhood, whereas flow estimation may require *summing* such signals. Inspired by the principal neighbourhood aggregation [5] architecture, we have investigated various combinations of GNN aggregation functions, showing them to be beneficial across many modelling scenarios.

## 3 METHOD

In this section, we describe our travel time prediction problem in the context of arrival time estimation. We also provide the technical details of the GNN models that we used for this problem, including their architecture and training details. The key components of our method are models that operate on networks of route segments in order to leverage their structure, and predict travel times for multiple time horizons into the future, in order to provide end users with more accurate ETAs. Key technical details of our GNN models are the various stabilizing methods that are used to reduce training variance, which have critical impact on user experience. Further, we will demonstrate that several changes to our launched model provide offline improvements under specific conditions.

### 3.1 Problem Setup

To achieve accurate travel time estimates, we model the road network using **supersegments**, which are sequences of connected road segments that follow typical traffic routes. For a given starting time, we learn the travel time of each supersegment for different fixed time horizons into the future. At serving time, the sequence of supersegments that are in a proposed route are queried sequentially for increasing horizons into the future. This process specifically involves sequentially using the prediction time of an earlier supersegment to determine the relevant fixed horizons for the next supersegment, and interpolating between the prediction times of the fixed horizons to arrive at a travel time for the next supersegment. This process enables us to provide accurate travel time estimates in a scalable way, taking into account both spatial and temporal information.

A *supersegment*, $\mathcal{S}$, is defined as a graph $\mathcal{S} = (S, E)$ where each node $s \in S$ is a road segment, and an edge $e_{ij} \in E$ exists if two segments $s_i$ and $s_j$ are connected (see Figure 2 for an overview). The supersegment travel time prediction task is then: given a supersegment $\mathcal{S}_t$, predict the travel time across the supersegment at different horizons into the future $y_t, y_{t+h_1}, y_{t+h_2}, ..., y_{t+h_k}$, where $h_1, h_2, ...h_k$ correspond to the different horizons.

*3.1.1 Data.* The world is divided into regions that confine similar driving behaviors and capture most trips without splitting. Data is constructed for each of these regions in order to build region-specific GNN models. For training and evaluation, each example was collected from a traversal event across a supersegment and its underlying segments. Specifically, a unique supersegment may appear in multiple examples corresponding to multiple traversals over that supersegment. The actual traversal times along segments and supersegments in seconds are used as node-level and graph-level labels for prediction, and each unique supersegment can appear in the training and evaluation data multiple times, corresponding to multiple traversals. The features, describing the traffic conditions of the supersegments prior to the traversal event, were collected backwards in time by the duration of each fixed horizon. The road segments are on average 50 - 100 meters long, and the supersegments contain on average about 20 road segments. Horizons we use are 0 seconds, 600 seconds, 1200 seconds, 1800 seconds, and 3600 seconds. In Section 4, we also investigate the trade-off of expanding supersegment graphs to include neighbouring nodes, or segments that extend beyond an immediate route.

*3.1.2 Features.* For this task, we utilize both *real-time* information that captures traffic patterns at the time of prediction and *historical* data for traffic patterns specific to the time-of-day and day-of-week when the prediction is made. On a segment (node) level, we provide the "average real-time" and "historical segment travel speeds" and times, as well as "segment length" and "segment priority" (eg. road classifications such as highways) as features. On a supersegment (graph) level, we additionally provide real-time supersegment travel times as a feature. Real-time travel speeds and times are provided for 17 2-minute windows prior to horizon 0. Historical speeds and times are provided for five 8-minute windows prior to horizon 0, and seven 8-minute windows after horizon 0, with the values being an average across the past 17 weeks. We additionally provide learnable segment- and supersegment-level embedding vectors (of sizes 16 and 64, respectively). This enables sharing of information whenever the same segment appears in different supersegments, and whenever the same supersegment appears in different candidate routes.

*3.1.3 Baselines.* Possible baselines for this problem include (a) *non-parametric* methods, that compute travel times by averaging speeds across segments and supersegments. (b) *segment-level* models, that

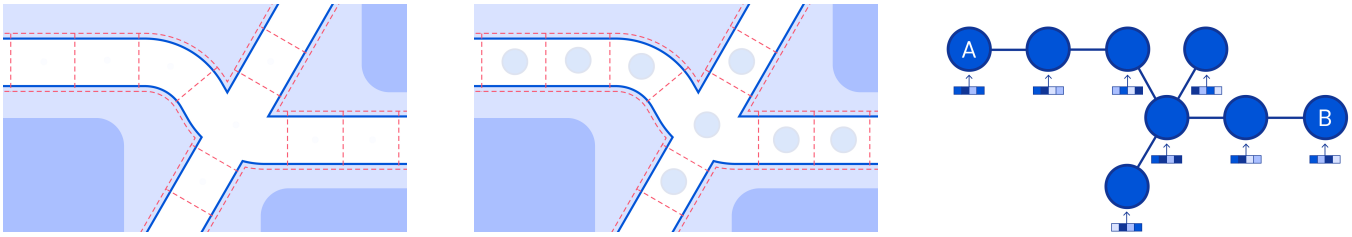


**Figure 2: An example road network with shared traffic volume, which is partitioned into segments of interest (left). Each segment is treated as a node (middle), with adjacent segments connected by edges, thus forming a *supersegment* (right). Note that for *extended supersegments* (discussed in Section 4.1.4), extra off-route nodes may be connected to the graph.**

bypass the graph structure induced by supersegments. Such models predict travel times along every segment in isolation, adding them together to predict across supersegments. Prior production models used segment-level linear regression models.

## 3.2 Model Architecture

Our GNN model leverages full *Graph Network* (GN) blocks exactly as described in [1]. Specifically, a GN block is defined with three "update" functions $\phi$ corresponding to edge, node, and global or supersegment-level updates, and three "aggregation" functions, $\rho$:

$$\begin{aligned} \mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{t_k}, \mathbf{u}\right) & \bar{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\ \mathbf{v}'_i &= \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \bar{\mathbf{e}}' &= \rho^{e\to u}\left(E'\right). \\ \mathbf{u}' &= \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) & \bar{\mathbf{v}}' &= \rho^{v\to u}\left(V'\right) \end{aligned} \tag{1}$$

For each edge $k$, $\phi^e$ takes in edge features $e_k$, source and target node features $v_{s_k}$ and $v_{t_k}$, supersegment features $u$, and outputs an edge representation $e'_k$. For each node $i$, $\phi^v$ takes in node features $v_i$, aggregate edge representations of node $i$'s edges $\bar{e}'_i$, supersegment features $u$, and outputs a node representation $v'$. Finally, $\phi^u$ takes in aggregate node representations $\bar{v}'$, aggregate edge representations $\bar{e}'$, supersegment features $u$, and outputs a supersegment representation $u'$.

We compose 3 GN blocks into an *encode-process-decode* architecture [11], and learn a separate model for each horizon $h$. These blocks are defined as $\mathrm{GN}^{(h)}_{enc}$, $\mathrm{GN}^{(h)}_{proc}$, $\mathrm{GN}^{(h)}_{dec}$.

The encoder is first applied to the supersegment $\mathcal{S}$ with the previously described raw features. It produces *latent* representations of nodes, edges and the supersegment itself. These latents are then passed to the processor, which is applied 2 times (with shared parameters) to *update* these representations. Finally, these representations are transformed into appropriate *predictions* by applying the decoder.

The model predictions are $\hat{y}^{(i)}_{t+h}$, $\hat{y}^{(i)}_{j,t+h}$, $\hat{y}^{(i)}_{c,j,t+h}$, which are supersegment, segment, and cumulative segment predictions. On the supersegment level, the models predict the estimated travel time for the entire supersegment, $\hat{y}^{(i)}_{t+h}$. In addition, for every node, the models predict the estimated travel time across the segment $\hat{y}^{(i)}_{j,t+h}$ as well as the cumulative estimated travel time up to and including that segment $\hat{y}^{(i)}_{c,j,t+h}$, which we found are helpful for representation learning. In production, we use the supersegment-level output $\hat{y}^{(i)}_{t+h}$ instead of any of the node-level outputs. Intuitively, summing the individual segment predictions, for example, can lead to an accumulation of error.

The update functions, $\phi$, are all simple multilayer perceptrons (MLPs). Aggregation functions $\rho$ are *summations*. While this is common practice, there exist benefits from using multiple aggregator functions like *summing* and *averaging* and concatenating their results together [5]. For certain regions and horizons in offline testing, this was true in our case as well. We will discuss the comparative benefits of various aggregators in Section 4.

## 3.3 Model Training

While training our GNN models, we found it very useful to use a *combination* of various loss functions. This resulted in strong serving performance when deployed in production. The primary bottleneck to deploying was the *high variability* of models during training. Variance is an issue in application because models saved at different points of training will result in different end user experiences. To address this, we use a MetaOptimizer that makes use of **MetaGradients** [31] for adapting the learning rate during training, as well as applying an exponential moving average over model parameters during evaluation and serving.

*3.3.1 Losses.* Although our main objective is to predict supersegment-level travel time, we found that using auxiliary losses for good segment-level travel time prediction helps the final performance. Our final loss for a specific horizon $h$ is

$$L = \ell_{\mathrm{ss}} + \lambda_{\mathrm{s}}\ell_{\mathrm{s}} + \lambda_{\mathrm{sc}}\ell_{\mathrm{sc}}, \tag{2}$$

with additional weight decay, where $\lambda_{\mathrm{s}} = 1$, $\lambda_{\mathrm{sc}} = 0.15$ and

$$\ell_{\mathrm{ss}} = \sum_{i=0}^{N}\left(\frac{1}{\max\{f^{(i)}, 1\}^{0.75}}\right)\cdot \mathcal{L}_{400}\left(y^{(i)}_{t+h}, \hat{y}^{(i)}_{t+h}\right) \tag{3}$$

$$\ell_{\mathrm{s}} = \sum_{i=0}^{N}\sum_{j=0}^{m^{(i)}}\left(\frac{1}{\max\{f^{(i)}_j, 1\}^{0.75}}\right)\cdot \mathcal{L}_{400}\left(y^{(i)}_{j,t+h}, \hat{y}^{(i)}_{j,t+h}\right) \tag{4}$$

$$\ell_{\mathrm{sc}} = \sum_{i=0}^{N}\sum_{j=0}^{m^{(i)}}\left(\frac{1}{\max\{\sum_{k=0}^{j} f^{(i)}_k, 1\}^{0.75}}\right)\cdot \mathcal{L}_{400}\Big(\sum_{k=0}^{j} y^{(i)}_{k,t+h}, \hat{y}^{(i)}_{c,j,t+h}\Big). \tag{5}$$

The provided labels and weights are $y^{(i)}_{t+h}$ and $y^{(i)}_{j,t+h}$, which are supersegment and segment traversal times, and $f^{(i)}$ and $f^{(i)}_j$, which

| Dataset | # (Unique) Training Supersegments | # (Unique) Testing Supersegments | Average # Segments per Supersegment | Average # Segments per Extended Supersegment | Average Segment Length (m) |
|---|---|---|---|---|---|
| LAX | 661M (16k) | 126M (16k) | 21.79 | 204.60 | 81.02 |
| NYC | 502M (12k) | 91M (12k) | 20.75 | 147.67 | 109.03 |
| SGP | 257M (5k) | 54M (5k) | 18.80 | 96.31 | 106.29 |
| TYO | 115M (4k) | 22M (4k) | 24.52 | 236.54 | 57.94 |

**Table 1: Information on datasets, including the number of total supersegments, and unique supersegments in each training and evaluation dataset, the average number of segments in a supersegment computed across a training dataset, and the average length of segments computed across a training dataset.**

are supersegment and segment free flow times, an estimate of traversal times when little to no traffic is present. We predict $\hat{y}^{(i)}_{t+h}$ in addition to segment-level travel times in order to avoid an accumulation of errors over segments. $\mathcal{L}_\delta$ is a Huber loss function with delta value $\delta$, which is important in being less sensitive to outliers. The losses are summed up across the segments and supersegments, so losses such as the supersegment-level loss is more influenced by supersegment length whereas losses like the segment-level loss is more independent from length. We exponentially scale the Huber loss down for examples with greater free flow times to better control the loss's growth.

We also experimented with purely *unsupervised* auxiliary losses, and will discuss their trade-offs in Section 4.

*3.3.2 MetaGradients.* In order to reduce the variance of the recovered GNN models across epochs, we resort to a careful treatment of the *learning rate* for each step of training. Our method of choice, MetaGradients [31], follows the online cross-validation paradigm and jointly optimizes the hyper-parameter $\eta$ with the model parameters $\theta$ at each training iteration. Although the original work targets at reinforcement learning, the approach can be generalized well to supervised learning applications. MetaGradients frames the optimization problem as $L(\tau, \theta, \eta)$, where $\tau$ is a training example, $\theta$ are the model parameters, $\eta$ is the hyper-parameter (e.g., the learning rate) and $L$ is the original loss function additionally parameterized by $\eta$. For a new training example, $\tau'$, an underlying update to the model parameters $\theta' = \theta + f(\tau, \theta, \eta)$ with some update function $f$ (eg. a gradient step) additionally parameterized by $\eta$, and a reference meta-parameter value $\eta'$, MetaGradients computes the update for $\eta$ using the following derivative:

$$\frac{\partial L(\tau', \theta', \eta')}{\partial \eta} = \frac{\partial L(\tau', \theta', \eta')}{\partial \theta'} \frac{\partial \theta'}{\partial \eta}, \tag{6}$$

where the gradient w.r.t. the model parameter is multiplied by a factor $\partial\theta'/\partial\eta$. Xu et al. [31] has shown that, in practice, the second term can be approximated using an accumulative trace, i.e.,

$$\frac{\partial \theta'}{\partial \eta} = \frac{\partial \theta}{\partial \eta} + \frac{\partial L(\tau, \theta, \eta)}{\partial \eta}. \tag{7}$$

In our application, we have successfully combined MetaGradients with the Adam optimizer [14].

*3.3.3 Exponential Moving Average (EMA) of Parameters.* During evaluation and serving, we use an EMA of model parameters

$$\theta_{\text{EMA}} = \alpha \cdot \theta_{\text{EMA}} + (1 - \alpha) \cdot \theta, \tag{8}$$

where $\alpha = 0.99$. EMAs were another key component in reducing variance in saved models within training runs.

# 4 EXPERIMENTS

## 4.1 Experimental Setup

*4.1.1 Datasets.* Although our models were launched across regions world-wide, we evaluate our model specifically over: *New York* (**NYC**), *Los Angeles* (**LAX**), *Tokyo* (**TYO**) and *Singapore* (**SGP**), finding that the results generalize to other regions (see Figure 1). For offline evaluation, we used training data collected during January 2020. For online evaluation, our data was collected during November 2020. We note that November 2020 data may be susceptible to COVID-19 traffic patterns, and hence we attempted to select regions that minimize this deviation for evaluation in order to capture performance under regular traffic conditions. The distributions of the datasets and their extended versions are described in Table 1.

*4.1.2 Baselines.* We evaluate our GNN models against nonparametric travel times computed using real-time speeds and historical speeds, as well as models that do not leverage graph structure:

- **Real-time travel times**: Summation of segment-level travel times computed using segment speeds averaged over a two minute window prior to the time of prediction. These are numbers computed from data.
- **Historical travel times**: Summation of segment-level travel times computed using segment speeds specific to hour of day and day of week at the time of prediction, averaged across last 17 weeks. These are numbers computed from data.
- **DeepSets**: Transform node representations using a segment-level feed-forward network followed by a sum aggregation over the segment-level representations before using a final feed-forward network to make graph-level predictions [36]. This effectively treats supersegments as a "bag-of-segments", *ignoring* their graph structure.

*4.1.3 Ablations of Launched Models.* We ablate over the specific elements of our launched GNNs that contribute to performance improvements and variance reduction. This includes learnable segment and supersegment embeddings.

*Embeddings.* We ablated over the embeddings defined for unique segment and supersegment IDs. We use a dimension of 16 for segment-level embeddings and a dimension of 64 for supsersegment-level embeddings across all models that use embeddings. The embedding vocabularies are region-specific, covering 99.5% of segment

| | Real-Time | | Historical | | DeepSets | | GN | |
|---|---|---|---|---|---|---|---|---|
| | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 |
| NYC | 42.31 | 69.60 | 48.03 | 51.80 | 37.33 | 46.68 | **36.84** | **46.55** |
| LAX | 47.58 | 79.31 | 53.40 | 60.22 | 40.71 | 50.06 | **40.32** | **49.94** |
| TYO | 63.56 | 81.01 | 63.76 | 67.63 | 52.15 | 60.07 | **51.64** | **59.86** |
| SGP | 45.09 | 75.83 | 58.25 | 61.39 | 37.56 | 50.81 | **36.79** | **50.67** |

**Table 2: Offline model performance against baseline methods reported in RMSE, averaged across five random seeds. The graph network (GN) significantly outperforms all reported baselines for each region and horizon ($p < 0.01$ using a $t$-test).**

IDs and supersegment IDs in the corresponding datasets, and assigning the rest to be out-of-vocabulary (OOV), spread out across 200 and 20 OOV buckets for segments and supersegments respectively.

*MetaGradients & EMA.* We directly compare our models trained with and without MetaGradients. When trained with MetaGradients, we set the meta-learning rate to 0.01 and the update frequency to every 100 iterations.

We compare our GNN models that use EMA (with a decay factor of $\alpha = 0.99$) to models that do not.

*4.1.4 Notable Extensions.* We describe and compare extensions to the launched GNN models that show offline improvements, and can be useful in specific settings. This includes extended supersegment graph data, combinations of aggregators, and unsupervised auxiliary losses.

*Extended Supersegments.* We investigate performance improvements for GNN models that operate on **extended** supersegments. Extended supersegments are larger graphs that include nodes from neighbouring segments, in addition to the original segments. These include segments from nearby, possibly connected traffic. The prediction task and labels remain the same. Specifically, the goal is still to predict travel times across the original segments and supersegments within the extended supersegments.

This enables our models to make use of additional spatial context—such as traffic congestion on a connecting route—at the expense of data storage, slower training, and inference. We note that the same GNN architecture is used for both supersegment and extended supersegment experiments, making the *model* storage cost remain constant. Further, we experiment with binary features that indicate whether segments are on the main route in an extended graph.

*Combinations of Aggregators.* We further investigated the benefits of swapping out the default *sum* aggregation function with *combinations* of different aggregators. We combine by *concatenating*; for example, the combination [Min, Max] has an aggregation function for nodes of:

$$\rho_i^{s\to u}(S) = \left[\min_{j\in N(i)} s_j || \max_{j\in N(i)} s_j\right] \tag{9}$$

where $N(i)$ are the road segments that neighbour $s_i$. We ablate over all combinations of Max, Min, SqrtN, and Sum. In the interests of brevity, only the best-performing ones are included in Table 6.

*Unsupervised Auxiliary Losses.* Lastly, we experimented with incorporating *unsupervised* losses, which are designed to produce representations that capture the *structural* information of input graphs. This, in turn, may simplify discovery of interesting motifs within the road network topology, which may implicitly drive traffic dynamics. In all experiments, these losses are combined with the supervised objective (Equation 2) to assist learning.

We combine two main kinds of unsupervised loss functions. Firstly, *Deep graph infomax* (DGI) [26] encourages the node embeddings to encode *global* structural properties, by carefully contrasting the road network against an artificially *corrupted* version. Conversely, the *graph auto-encoder* (GAE) [16] leverages a *link prediction*-style loss and is hence mindful of *local* topology.

## 4.2 Offline Evaluation

For offline evaluation, we use the root mean squared error (RMSE) between the predicted supersegment travel times and the actual traversal times across supersegments, in seconds:

$$\text{RMSE}_{\text{ss}} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}\left(y_{t+h}^{(i)} - \hat{y}_{t+h}^{(i)}\right)^2}. \tag{10}$$

Reported values are computed using the temporally held-out test datasets of the same regions. Other metrics such as mean absolute error (MAE) show similar results and are thus omitted. RMSE is used to make decisions for which architectures are used in online evaluations and launches, so we focus on that metric here.

*Baseline Comparison.* Table 2 shows the offline evaluation results of our GNN models compared to the baselines, averaged over five random seeds. Across all studied regions and horizons, our GNN models outperform the baselines. Compared to the closest baseline, DeepSets, our GNN models show an improvement of 0.12 to 0.77 RMSE. One note is that these improvements are computed over individual supersegments, and they will accumulate over entire routes—hence our improvements may quickly become more important on a whole-route level. Across all of the considered regions and horizons, the improvements of the GNN are ***statistically significant***: $p < 0.01$ using a $t$-test on the individual run results, across five runs.

*Embeddings.* Table 3 studies the effect of the various learnable input embeddings, by attempting to ablate them out. Across most regions and horizons, such embeddings are useful for improving RMSE.

*MetaGradients & EMA.* Figure 3 compares training with and without applying MetaGradients and EMA decay over several seeds. For brevity, we only show results for two settings (TYO Horizon 0, LAX Horizon 3600); the trends generalize across different regions

| | GN | | GN without Segment Embeddings | | GN without Supersegment Embeddings | | GN without Segment and Supersegment Embeddings | |
|---|---|---|---|---|---|---|---|---|
| | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 |
| NYC | **36.87** | **46.59** | 37.03 | 46.59 | 37.04 | 46.61 | 37.55 | 46.86 |
| LAX | 40.39 | **49.96** | 40.48 | 50.03 | **40.38** | 49.96 | 41.11 | 50.33 |
| TYO | 51.76 | **59.90** | 52.02 | 59.98 | **51.72** | 60.00 | 52.63 | 60.60 |
| SGP | **36.84** | 50.60 | 37.29 | 50.83 | 36.98 | **50.52** | 37.41 | 51.25 |

**Table 3: An ablation on the segment and supersegment learnable embedding representations used as input to the Graph Net model. Performance for each reported in RMSE.**

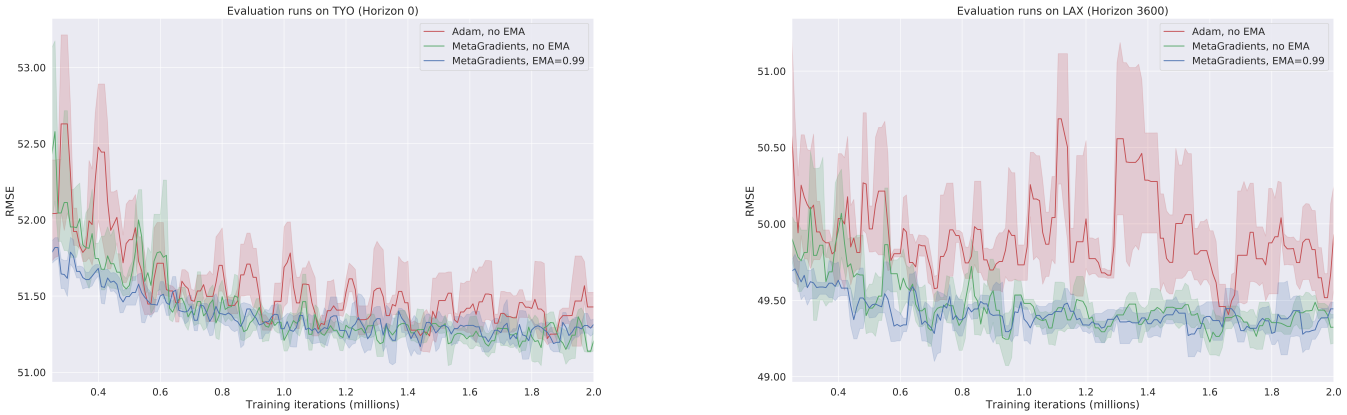


**Figure 3: Validation RMSE during training, with and without MetaGradients and EMA decay, aggregated across five seeds. Both methods contribute to variance reduction. Results shown for two metro/horizon setups – same trends hold elsewhere.**

and horizons. Both MetaGradients and EMA contribute to lowering within-run and across-run variance, which are critical for inference in production. Further, it was shown that MetaGradients consistently decays the learning rate over time, which ensures stable training.

*Extended Supersegments.* Table 4 compares GNNs that operate on supersegments to variants that operate on extended supersegments. Combining extended supersegments with additional binary features that indicate whether a segment or an edge is part of the original supersegment is capable of improving RMSE (by 0.01 to 0.29). In a scenario where data storage and inference cost increases are not core issues, the extended supersegments may hence be a useful data representation.

*Unsupervised Auxiliary Losses.* Table 5 compares the performance between different unsupervised losses in terms of RMSE. Augmenting the loss function with deep graph infomax (DGI) or graph auto-encoders (GAE) by a tuned weight parameter achieves an improvement of 0.05 to 0.23 RMSE. The optimal variant and hyperparameters were different for each region and prediction horizon. Thus, it is likely expensive to tune such losses for all existing regions and each horizon. However, GN+DGI would be a reasonable option as it showed some degree of improvement in all cases. Note that there are several important design decisions involved with deploying DGI: e.g. deciding the readout, corruption, and discriminator functions. We, however, did not explore many options in this space, as the potential performance improvements were not deemed significant enough for the additional costs incurred in a production environment.

*Combinations of Aggregators.* Table 6 compares model performance when using different aggregation functions. When compared to our production default of sum, other variants may achieve an improvement of up to 0.34 RMSE depending on the region and horizon. Since the optimal aggregation function differs between regions and horizons, it is likely difficult and expensive to tune for all regions and horizons of the world. Using all five aggregations at once is a reasonable preference, and it may be beneficial to investigate dynamic aggregators as a future line of work.

## 4.3 Online Evaluation

We compare our GNNs against baselines in an online setting in order to show the effectiveness of the models during serving. Both real-time travel times and historical travel times are used in precursor systems and fall-back systems. For online evaluation, we use:

$$\text{RMSE}_{\text{track}} = \sqrt{\frac{1}{N}\sum_{i=0}^{N}\left(u^{(i)} - \hat{u}^{(i)}\right)^2} \tag{11}$$

which is the root mean squared error between the predicted track travel times and actual travel times, computed over all tracks within a week, specifically January 15th 6:00 PM - January 22nd 2021 6:00 PM. We note that the GN and DeepSets models additionally use

| | Supersegments | | Extended Supersegments | | Extended Supersegments + Extra Features | |
|---|---|---|---|---|---|---|
| | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 |
| NYC | 35.85 | 45.91 | 36.11 | 45.71 | **35.64** | **45.65** |
| LAX | 39.51 | 49.01 | 39.67 | 48.99 | **39.22** | **48.79** |
| TYO | 52.59 | 60.72 | 53.24 | 60.85 | **52.57** | **60.64** |
| SGP | 34.89 | 49.51 | 35.29 | **49.48** | **34.73** | 49.50 |

**Table 4: A comparison between different input graph formats. Travel time predictions only include road segments in the traversed track. Same embedding vocabularies are used across variants. Performance for each reported in RMSE.**

| | NYC | | LAX | | TYO | | SGP | |
|---|---|---|---|---|---|---|---|---|
| | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 |
| GN | 36.88 | 46.56 | 40.30 | 49.95 | 51.59 | 59.82 | 36.91 | 50.77 |
| GN+DGI | 36.79 | 46.53 | **40.18** | **49.90** | 51.48 | **59.76** | **36.76** | **50.54** |
| GN+GAE | **36.76** | **46.38** | 40.23 | 49.91 | **51.43** | 59.82 | 36.85 | 50.57 |

**Table 5: Compares using different unsupervised algorithms for augmenting the Graph Net as an auxiliary loss function. Performance for each reported in RMSE.**

| | NYC | | LAX | | TYO | | SGP | |
|---|---|---|---|---|---|---|---|---|
| | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 | Horizon 0 | Horizon 3600 |
| Sum | 36.88 | 46.56 | **40.30** | 49.95 | **51.59** | **59.82** | 36.91 | 50.77 |
| SqrtN | 37.06 | 46.60 | 40.39 | 49.95 | 51.75 | 59.96 | 36.93 | 50.66 |
| Mean | 36.92 | 46.60 | 40.45 | 49.98 | 51.92 | 60.01 | **36.78** | 50.66 |
| Min | 36.98 | 46.57 | 40.49 | 49.91 | 51.83 | 59.95 | 36.98 | 50.81 |
| Max | 37.01 | 46.57 | 40.48 | 49.95 | 51.84 | 59.96 | 36.97 | 50.94 |
| All | **36.83** | **46.53** | 40.33 | **49.88** | 51.60 | 59.90 | 36.95 | **50.43** |

**Table 6: Performance in RMSE using various aggregation functions. Only the individual aggregation operations and the full combination are included for brevity due to similar or worse results. Note that the optimal aggregation combination differs between regions as well as horizons.**

| Region | Real-Time | Historical | DeepSets | GN |
|---|---|---|---|---|
| NYC | 130.05 | 154.83 | 119.76 | **118.63** |
| LAX | 154.19 | 171.25 | 139.02 | **138.10** |
| TYO | 211.85 | 250.30 | 182.73 | **181.98** |
| SGP | 127.54 | 138.59 | 109.39 | **108.78** |

**Table 7: Online model performance against baselines reported in RMSE of track travel times.**

a series of fallback models for segments that do not appear in supersegments, and segments that do not have real-time or historical travel time input features, whereas the same is not done for real-time and historical travel times.

Table 7 shows the results of our GNN models compared to the baselines in the online setting. It shows a similar pattern compared to Table 2 with GNNs outperforming the baseline methods. Compared to DeepSets, the strongest baseline approach, GNNs outperform by 0.05 to 1.35 RMSE in this online setting.

## 4.4 Analysis

In this section, we provide specific examples that provide insight into scenarios where our GNN models are able to provide more accurate estimated time of arrivals compared to baselines, by better making use of spatial and temporal information. Figure 4 (left) shows the supersegment (which is part of LAX) which we will use to illustrate these qualitative findings.

Figure 4 (above) compares the predicted travel times over the supersegment to the actual travel times, over a range of departure times. The upward slopes of the GNN predicted travel times match the upward slopes of the actual travel times better than the real-time travel times baseline. In this setting, our GNN model is able to detect traffic congestions more accurately than the baseline by making use of traffic information from a larger neighborhood.

Figure 4 (below) compares predicted travel times over the supersegment for various horizons into the future. The predictions of our GNN for $h = 3600$ better match the actual travel times 60 minutes into the future, compared to the predictions of our GNN for $h = 0$. This example illustrates the value of having multiple prediction horizons in providing accurate ETA.

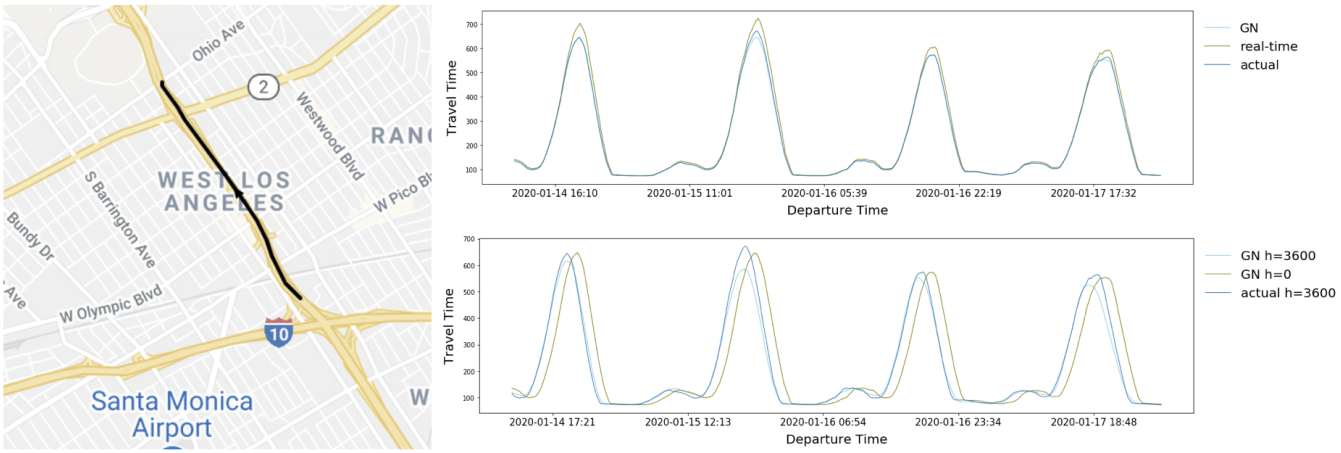


**Figure 4: Qualitative analysis over a supersegment in LAX (shown on the left). Above: Travel times and predictions over different departure times. Our GNN is able to detect traffic congestions more accurately than using real-time travel times. Below: Travel times and predictions for different horizons into the future, over different route departure times. Having multiple prediction horizons is important to providing accurate ETA.**

## 5 ENGINEERING CHALLENGES

**Caching** It is challenging to meet the latency requirements of Google Maps while keeping the cost of evaluating Graph Net models low. The path in an ETA request usually includes multiple supersegments and making predictions for these supersegments on the fly is not practical nor scalable. Approaches like CompactETA [7] learn high level representations that can be queried during inference time for use in a very simple MLP. To resolve this issue, we instead created a shared lookup table caching fresh predictions for predefined supersegments for a fixed set of horizons, which is periodically updated. The server fetches necessary predictions upon a request and interpolates the predictions of adjacent horizons to compute the estimated arrival time for each supersegment. We have tested an update frequency of 15 seconds and found no measurable regression up to a frequency of 2 minutes, indicating staleness is not an issue.

**Turn speeds** It is common to have a multimodal speed distribution for a busy road segment heading to a split of traffic, such as an intersection or a highway ramp. For such cases, we used the real time and historical speed for the specific turn which the supersegment follows.

**Coverages** Currently we have about 1 million predefined supersegments that cover the most common routes taken by our users. The effect is that most freeways, major arterials and some popular short-cuts in metropolitan areas are selected, whereas less busy surface streets will not be covered. Multiple supersegments may cover any given road segments to account for multiple common routes that traverse through that road segment. For less frequently visited road segments, we use simpler per-segment models.

## 6 CONCLUSION

We presented how a graph neural network can be engineered to accurately predict travel time estimates along candidate routes. Applying stabilising techniques such as MetaGradients and EMA was a necessary addition to make the GNNs production-ready. We deployed our GNN model for ETA prediction in Google Maps, where it now serves user queries worldwide. We presented offline metrics, online evaluations, and user studies: all showing significant quantitative improvements of using GNNs for ETA predictions. Further, through extensive ablations on various design and featurization choices, we provide prescriptive advice on deploying these kinds of models in practice. We believe our findings will be beneficial to researchers applying GNNs to any kind of transportation network analysis—which is certain to remain a very important application area for graph representation learning as a whole.

## ACKNOWLEDGMENTS

We would like to thank Paulo Estriga and Adam Cain for designing several of the figures featured in this paper.

## REFERENCES

[1] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[3] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 17766–17778. https://proceedings.neurips.cc/paper/2020/file/cdf6581cb7aca4b7e19ef136c6e601a5-Paper.pdf

[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*

(2018).
[5] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718* (2020).
[6] Xiaomin Fang, Jizhou Huang, Fan Wang, Lingke Zeng, Haijin Liang, and Haifeng Wang. 2020. ConSTGAT: Contextual Spatial-Temporal Graph Attention Network for Travel Time Estimation at Baidu Maps. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2697–2705.
[7] Kun Fu, Fanlin Meng, Jieping Ye, and Zheng Wang. 2020. *CompactETA: A Fast Inference System for Travel Time Prediction.* Association for Computing Machinery, New York, NY, USA, 3337–3345. https://doi.org/10.1145/3394486.3403386
[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212* (2017).
[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.
[10] William L Hamilton. 2020. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14, 3 (2020), 1–159.
[11] Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. 2018. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203* (2018).
[12] Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. 2020. P-Companion: A Principled Framework for Diversified Complementary Product Recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2517–2524.
[13] Jilin Hu, Bin Yang, Chenjuan Guo, Christian S Jensen, and Hui Xiong. 2020. Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1417–1428.
[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[16] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[17] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).
[18] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2311–2320.
[19] Cheonbok Park, Chunggi Lee, Hyojin Bahng, Yunwon Tae, Seungmin Jin, Kihwan Kim, Sungahn Ko, and Jaegul Choo. 2020. ST-GRAT: A Novel Spatio-temporal Graph Attention Networks for Accurately Forecasting Dynamically Changing Road Speed. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1215–1224.
[20] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).
[21] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. 2020. SIGN: Scalable Inception Graph Neural Networks. *arXiv preprint arXiv:2004.11198* (2020).
[22] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph Neural Networks for Friend Ranking in Large-scale Social Platforms. (2021).
[23] Xingjian Shi and Dit-Yan Yeung. 2018. Machine learning for spatiotemporal sequence forecasting: A survey. *arXiv preprint arXiv:1808.06865* (2018).
[24] Xianfeng Tang, Yozen Liu, Neil Shah, Xiaolin Shi, Prasenjit Mitra, and Suhang Wang. 2020. Knowing your FATE: Friendship, Action and Temporal Explanations for User Engagement Prediction on Social Apps. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2269–2279.
[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[26] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. (2019).
[27] Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. 2019. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593* (2019).
[28] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[29] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).
[30] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive Representation Learning on Temporal Graphs. *arXiv preprint arXiv:2002.07962* (2020).
[31] Zhongwen Xu, Hado P van Hasselt, and David Silver. 2018. Meta-gradient reinforcement learning. *Advances in neural information processing systems* 31 (2018), 2396–2407.
[32] Hongxia Yang. 2019. Aligraph: A comprehensive graph neural network platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3165–3166.
[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.
[34] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).
[35] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective Travel Time Estimation: When Historical Trajectories over Road Networks Matter. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2135–2149.
[36] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep Sets. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3391–3401. http://papers.nips.cc/paper/6931-deep-sets.pdf
[37] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294* (2018).
[38] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. 2020. Curb-GAN: Conditional Urban Traffic Estimation through Spatio-Temporal Generative Adversarial Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 842–852.
[39] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 1234–1241. https://aaai.org/ojs/index.php/AAAI/article/view/5477