Foresight Plus: Scalable Serverless Real-Time Spatio-Temporal Traffic Forecasting

Joe Oakley (✉️ J.Oakley@warwick.ac.uk)
The University of Warwick

Chris Conlan
The University of Warwick

Gunduz Vehbi Demirci
Imagination Technologies

Alexandros Sfyridis
Imperial College London

Hakan Ferhatosmanoglu
The University of Warwick

Research Article

Keywords: Real-Time Traffic Forecasting, Graph Neural Networks, Cloud Computing, Serverless Inference, Dynamic Urban Events, Vehicle-Level Flow Data

Posted Date: September 19th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3353187/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.
Abstract

Building a real-time, cost-effective, spatio-temporal forecasting system is a challenging problem with many practical applications such as traffic and road network management. Most forecasting research focuses on average prediction quality, with less attention paid to building practical pipelines and achieving timely and accurate forecasts when the network is under heavy load. Additionally, transport authorities need to leverage dynamic data sources (e.g., scheduled roadworks) and vehicle-level flow data, while also supporting ad-hoc inference workloads at low cost. The cloud-based system Foresight, developed in collaboration with Transport for the West Midlands (TfWM), is able to ingest, aggregate and process streamed traffic data, as well as dynamic urban events/flow data to produce regularly scheduled forecasts with high accuracy. In this work, we extend our system with several novel enhancements. First, we present an efficient method for extending the forecasting scale, enabling transport managers to predict traffic patterns further into the future than existing methods. In addition, we augment the existing inference architecture with a new, fully serverless design. This offers a more cost-effective inference solution, which seamlessly handles sporadic inference workloads over multiple forecasting models. We observe that Graph Neural Network (GNN) forecasting models are robust to extensions of the forecasting scale, achieving consistent (and sometimes even improved) performance up to
24 hours ahead. This is in contrast to the 1 hour forecasting horizons popularly considered in the literature. Further, our serverless inference solution is shown to be significantly more cost-effective than provisioned alternatives in appropriate use-cases. We identify the optimal memory configuration of serverless resources to achieve an attractive cost-to-performance ratio.

**Keywords:** Real-Time Traffic Forecasting, Graph Neural Networks, Cloud Computing, Serverless Inference, Dynamic Urban Events, Vehicle-Level Flow Data

1 Introduction

Traffic data collected at roadside sensors can offer significant value to transport managers. The raw data is typically transformed into a time series format, capturing metrics such as the vehicle count or average speed over the road network. This information can be used to make forecasts about the state of the road network in the near future, which can enable proactive responses when heavy or unusual load on the network is predicted.

A wide range of forecasting approaches have been applied to the traffic prediction task, from statistical methods such as ARIMA [28], to Deep Learning (DL) models such as LSTM [18]. In recent years, Graph Neural Network (GNN) approaches have achieved state-of-the-art results, due to their ability to capture spatial dependencies between sensors [17, 24, 26, 29, 31, 44, 52, 57]. GNNs typically model the road sensor network as a graph structure, whose weighted adjacency matrix reflects the strength of inter-sensor relationships.

Despite an extensive body of work on traffic forecasting, there are still several major challenges in building a practical traffic forecasting system. First, there are operational requirements around the scalable handling and pre-processing of the streaming traffic data, as well as in its use for real-time forecasting. Typically, most forecasting models are developed offline, without considering the challenges of producing forecasts on streaming data. The real-time forecasting problem requires that the prediction/scoring process takes place continuously within a given time lag of each real-world traffic event occurring. This is an important problem to resolve for practical data-driven systems, as transport managers need to be able to take action based on responsive short-term forecasts. It has also been identified as an open research issue, and entails significant data management challenges, particularly when DL models are employed [8]. Furthermore, existing research has largely considered forecasting 1 hour ahead, on static benchmark data, with only a few works seeking to predict further ahead [5, 56]. It would be beneficial to transport managers if accurate forecasts could be made further into the future, allowing more time for responsive action to be taken. Next, in addition to data captured at roadside sensors, dynamic urban events (DUE) and vehicle-level flow data should also be incorporated dynamically into forecasting models to improve predictive performance. Finally, many inference workloads are sporadic in nature, with queries arriving at irregular intervals and being distributed over multiple forecasting models/scales. An inference platform built on provisioned VM resources may not be
a cost-effective solution for such workloads, but there are challenges associated with achieving efficient inference performance on lightweight alternatives such as serverless computing.

The Foresight cloud-based forecasting system [11] achieves real-time forecasting over streaming traffic data, and effectively leverages DUE and vehicle-level flow data to improve predictive performance. In this work, we extend Foresight with several novel enhancements; we term the improved system **Foresight Plus**.

First, we present an approach for extending the forecasting scale beyond the ‘1 hour ahead’ horizons typically seen in forecasting literature. Our efficient method is able to satisfy real-time forecasting requirements while enabling transport managers to forecast many hours into the future, with little to no degradation in the quality of predictions. Further, we design a fully serverless inference solution for traffic forecasting. Foresight Plus is significantly more cost-effective than a provisioned inference solution for many workloads, and seamlessly handles requests over multiple predictive models with an impressive cost-to-performance ratio.

The contributions of this work are as follows:

- We present Foresight Plus, which enhances the Foresight cloud-based real-time traffic forecasting system.
- We present a novel method for extending the forecasting scale, which enables predictions further into the future than previous approaches. This lightweight scheme enables real-time forecasting requirements to be satisfied, even when forecasting up to 24 hours ahead.
- We design a fully serverless inference solution, which effectively handles sporadic inference workloads. We also present a cost model for serverless forecasting, and consider the implications of several design choices in this context.
- We observe that GNN forecasting models are robust to extensions of the forecasting scale up to 24 hours ahead, and can actually achieve improved performance as the scale grows.
- We identify fully serverless inference as a cost-effective and efficient solution for sporadic inference workloads. We study the scalability, cost and performance characteristics of serverless offerings to optimize resource configurations.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 formalizes the traffic forecasting problem, and describes its real-time extensions. Section 4 illustrates Dynamic Urban Events. Section 5 describes the Flow-based GNN Adjacency Matrix. Section 6 illustrates the Foresight cloud-based real-time forecasting system. Section 7 presents the novel enhancements we make in Foresight Plus. Section 8 covers our experimental analysis, both of Foresight and Foresight Plus. Finally, Section 9 concludes the paper.

---

1 This work extends the workshop paper ‘Real-Time Spatio-Temporal Forecasting with Dynamic Urban Event and Vehicle-Level Flow Information’, presented at the 5th International Workshop on ‘Big Mobility Data Analytics’ BMDA @ EDBT 2023, which described work at the University of Warwick, prior to GV Demirci joining Imagination Technologies, A Sfyridis joining Imperial College London, and H Ferhatosmanoglu joining Amazon Web Services. Work in this publication is not affiliated with Imagination Technologies, Imperial College London or Amazon Web Services.
2 Related Work

Identifying the future state of a system via forecasting has been applied in a wide range of disciplines such as economics [27], energy and environmental studies [3, 16, 22, 36], epidemiology [21, 46] and transport [13, 32, 41, 49, 55], among others. Forecasting is typically undertaken with the use of statistical and machine learning models, which may be embedded in an end-to-end forecasting system.

Numerous forecasting methods have been applied in the traffic prediction domain. In traffic flow forecasting studies, various models and systems have been applied to improve current modelling outcomes, such as achieving accurate traffic predictions on the road network. The Autoregressive Integrated Moving Average (ARIMA) and its variations have been consistently popular time-series models [2, 14, 28]. In addition, many Machine Learning (ML) approaches have been applied, with the Support Vector Machine (SVM) [53, 60] and the Random Forest [4, 38, 53] being the most common. Deep Learning (DL) solutions based on Artificial Neural Networks have been increasingly utilized due to their improved forecasting accuracy and the ability to account for non-linear dependencies [30]. Long Short-Term Memory (LSTM) and Feed Forward Neural Networks (FFNN) are among the popular models applied to forecast traffic flows [18, 33, 34], with several hybrid approaches also investigated [47, 59]. Finally, Graph Neural Networks (GNNs), which can capture the spatial dependencies between the traffic monitoring sensors by representing the road network as a graph structure, have further improved prediction accuracy. Hence, multiple GNN applications for traffic flow forecasting have been presented in recent years [8, 17, 24–26, 29, 31, 44, 52, 57].

2.1 Dynamic Urban Events

Urban events such as roadworks have been shown to significantly impact traffic flow [7, 42]. Hence, the incorporation of auxiliary information about such events can further improve traffic forecasting performance. For example, roadwork and accident information has been utilized in traffic simulation systems, ML models and GNNs [1, 6, 31]. A combination of roadworks and weather conditions have also been added to a bi-directional LSTM Autoencoder for short-term traffic prediction [15].

2.2 GNN Adjacency Matrix

In GNN models, the underlying graph structures are usually represented with an adjacency matrix which represents the spatial relationships between the nodes of a graph [17]. Although GNN adjacency matrices are typically binary [20], multiple variations have been proposed [25]. For example, a real-valued distance-based adjacency matrix is a common alternative for representing the spatial dependencies between nodes, and has been applied in numerous traffic forecasting studies with GNNs [9, 40, 45, 50, 58]. The travel time between nodes has also been considered as an alternative to distance-based metrics [51]. More recently, dynamic matrices which capture changes in the spatial dependencies of the graph have been introduced [12]. Coarse origin-destination (OD) data has been applied as a substitute for a distance-based adjacency matrix [54].
2.3 Forecasting Systems

In addition to statistical and ML/DL modelling approaches, forecasting systems have also been developed as general tools for time series prediction and road management optimization. For example, the AutoAI for Time Series Forecasting (AUTOAI-TS) [43] automates forecasting techniques and addresses specific requirements for time series data. DeepTRANS [48] combines the DeepTTE system [35] with DCRNN [29] for bus travel time estimation. The system uses archive information about bus and traffic flow from sensor data, and DCRNN is used to estimate traffic speed at buses’ locations. The TrafficStream forecasting system leverages GNNs and Continual Learning (CL) [10]. It constructs a sub-graph to capture network expansion, and constraints are applied on the current training model to integrate information from historical data.

3 Real-Time Spatio-Temporal Forecasting

In this section, we first describe the traffic forecasting problem, before introducing its real-time variant. A key requirement of this procedure is that the aggregation, pre-processing and inference of the traffic data must take place within a certain time lag of the real traffic events occurring. These practical aspects of forecasting have attracted relatively little attention in the large body of research on the topic.

3.1 Traffic Forecasting Problem

We first present the definition of the traffic forecasting problem, where the goal is to predict the future state of the road network, given a sequence of previously observed time series readings. Traffic information is typically obtained from roadside sensors, which can capture features such as traffic flow or average speed, to form a (multivariate) time series. Given a set of sensors $S$, we denote the traffic information observed across all sensors as $X \in \mathbb{R}^{H \times |S| \times P}$, where $H$ is the total number of historical traffic readings, and $P$ is the number of predictive features used. Let $X^{(t)} \in \mathbb{R}^{|S| \times P}$ denote the traffic signal observed at time $t$, and $Y^{(t')} \in \mathbb{R}^{|S| \times Q}$ denote the traffic signal to be predicted at time $t'$. Note that the number of target features $Q$ may be different to $P$. We aim to learn a function $f(\cdot)$ which maps from $T'$ historical traffic signals to $T$ future traffic signals:

$$[X^{(t-T'\!\!+1)}, \ldots, X^{(t)}] \xrightarrow{f(\cdot)} [Y^{(t+1)}, \ldots, Y^{(t+T)}]$$ (1)

3.2 Real-Time Forecasting

The real-time variant of the traffic forecasting problem adds the constraint that all processing takes place within a specified duration following the end of each time bin. Performing real-time forecasting, particularly with DL models, has been identified as a significant challenge [8]. In Foresight/Foresight Plus, anonymized streaming data is collected at road cameras and ingested into the platform via an API endpoint. Further details of this procedure are illustrated in Section 6.1.

The real-time forecasting routine begins at the end of each time bin, which are each $B$ minutes long. First, the raw vehicle-level data (held in cloud storage) is aggregated
for the most recent time bin (i.e., the \(B\) minutes from \(X^{(t-1)}\) to \(X^{(t)}\)). We denote the time taken for this aggregation as \(T_{Agg}\). Next, the aggregated data is pre-processed so that it is in the correct format for model inference. This includes fetching and processing the aggregated traffic count information for the last \(T'\) time bins, as well as retrieving any additional model-specific data used for inference (e.g., roadwork time series, adjacency matrix). The time taken for this phase is referred to as \(T_{PreProc}\). Once the required data have been produced, the inference API endpoint is invoked to perform the forecast. The time taken for inference processing to occur, as in Equation 1, is denoted by \(T_{Inf}\).

We require the following expression to be satisfied for a system to be capable of real-time forecasting:

\[
T_{Total} = T_{Agg} + T_{PreProc} + T_{Inf} \leq B \tag{2}
\]

A value of \(T_{Total} \leq B\) ensures that the shortest forecasting horizon still pertains to information that is yet to be aggregated in the system, and is therefore relevant to network managers.

4 Dynamic Urban Events

Foresight is able to leverage DUE data dynamically to improve the accuracy of its forecasts. We use roadworks data as an illustrative example, but other such information (e.g., social event data) could readily be applied in a similar fashion. In the context of traffic forecasting, planned and unplanned roadworks frequently influence the volume and nature of traffic propagation through the road network \([7, 42]\), and so incorporating information about them into predictive models is important for accurate predictions. Foresight automatically ingests DUE data and processes it into a format which forecasting models can easily exploit.

Roadworks data is ingested into Foresight via the Street Manager API\(^2\), which is invoked to receive a feed of planned roadwork events. We denote the set of all roadworks listed by a given API call as \(R\). For each roadwork \(r \in R\), we obtain its latitude/longitude, as well as its start and end dates \(T_s\) and \(T_e\). In order to associate the live roadworks on a given day \(T\) with the road sensor network \(S\), we first select only those roadworks where \(T_s \leq T \leq T_e\). Next, we calculate the road network distance (using an indicative driving speed over a shortest path calculation on the road network) between each \(r \in R\) and each \(s \in S\). These distances populate an \(|R| \times |S|\) matrix \(W\), with each entry \((i, j)\) denoting the road network distance from live roadwork \(i\) to traffic sensor \(j\) in the network.

To incorporate this roadwork-to-camera influence information into the forecasting models, we convert \(W\) into a time series format at the same temporal granularity as the observed traffic data. This has been shown to be an effective method for adding roadwork data to forecasting models \([31]\). We define this as a new feature set \(\hat{X} \in \mathbb{R}^{H \times |S|}\). Each entry \(\hat{x}_{i} \in \hat{X}^{(t)}\) has a value between 0 and 1 which denotes the strength of the influence of the nearest active roadwork to sensor \(i\) at time \(t\). We consider two approaches to approximate this influence. The first is a binary thresholding approach,

\(^2\)https://www.gov.uk/guidance/find-and-use-roadworks-data
where entries are activated if there is a roadwork within threshold distance \( d \) metres of the sensor. The second method involves first calculating the distance from each sensor to its nearest live roadwork, before normalizing these distances into \([0, 1]\). We perform this normalization using a thresholded Gaussian kernel, with threshold \( k \).

Combining \( X \) and \( \hat{X} \), a new matrix \( \tilde{X} = \begin{bmatrix} X & \hat{X} \end{bmatrix} \) is constructed, which is the new feature vector passed to the forecasting models. We evaluate these approaches within the context of a GNN model in Section 8.

## 5 Flow Aggregated Adjacency Matrix

Graph Neural Networks (GNNs) are popularly used in state-of-the-art forecasting models [9, 12, 24, 26, 29, 40, 44, 45, 50, 52, 58]. These methods typically represent the traffic sensor network as a graph structure, whose adjacency matrix aims to capture spatial relationships between the sensors. The principle of GNN message passing and node aggregation approaches in the context of traffic forecasting, such as diffusion convolution [29], is to simulate traffic propagation in the network. This method of extracting features is typically embedded into a wider learning structure so that temporal features can be learnt along with spatial features in an integrated fashion.

The graph structure which models the traffic sensor network is described by a \(|S| \times |S|\) (weighted) adjacency matrix. The value at position \((i, j)\) approximates the strength of the relationship between sensor \( s_i \) and sensor \( s_j \). A popular method to assign weights in the adjacency matrix is to calculate pairwise sensor distances measured in the road network [29, 50, 52].

The aim of our approach is to more realistically reflect the actual flow of traffic in the network, compared to coarse sensor separation measures such as Euclidean distance. Simple distance-based measures alone are insufficient, as sensor separation per-se does not necessarily indicate traffic flow levels. Even though two sensors are spatially co-located, traffic might rarely pass between them consecutively, or may flow in one direction significantly more than the other; these properties cannot be easily captured by this approach.

We therefore develop a method for computing the adjacency matrix weights which uses vehicle-level flow data to more accurately determine the relationships between sensors. By leveraging the properties of granular ANPR (Automatic Number Plate Recognition) data, our method can anonymously capture (in order) the sequence of sensors which vehicles pass as they traverse the road network. By aggregating this information at the network level, we are able to determine actual flows within the network. The new adjacency matrix is designed to retain the same dimensions used in most GNN methods for spatio-temporal forecasting, so it can be directly applicable within these methods.

The Flow Aggregated Adjacency Matrix (FAAM), denoted as \( F \in \mathbb{R}^{|S|\times|S|} \), is constructed by aggregating observed flow between cameras within a given time frame. 1 unit of flow is recorded between cameras \( i \) and \( j \) when a car is observed at camera \( s_i \in S \) at time \( t \), and is then next observed \( s_j \in S \) no later than \( t + \tau \), where \( \tau \) is a parameter given in seconds which denotes the acceptable transition period. To
construct $F$, each entry $F_{i,j}$ is incremented by 1 for each observed unit of flow. $F_{i,j}$ is then averaged over all the time periods during which flow was observed, before being normalized into $[0, 1]$. Each entry $F_{i,j}$ thus approximates the likelihood of a vehicle transitioning directly from $s_i$ to $s_j$ within transition period $\tau$. This can be periodically updated to reflect changes in the network over time, such as seasonality. We note that a more granular time scale would be possible in this formulation, e.g., to capture shifting traffic patterns throughout the day, but we leave this to be explored in future work.

6 Foresight System Architecture

![Figure 1: High-level architecture of the Foresight cloud-based forecasting system](image)

In this section we present an overview of the Foresight system, as illustrated in Figure 1. The overall goal of Foresight is to provide continuous forecasts for transport managers by leveraging streaming traffic data as well as dynamic urban event and flow information. We will first describe how streaming traffic data is ingested and aggregated, before presenting the MLOps pipeline and inference procedure. Details of how DUE data and flow information are processed are given in Sections 4 and 5 respectively.
6.1 Streaming Data Ingestion, Aggregation and Storage

Foresight’s primary data source is anonymized ANPR vehicle capture information in the West Midlands road network managed by TfWM. This data flows into Foresight using a POST request to an API endpoint, before being forwarded to a streaming ETL service (Kinesis Data Firehose\(^3\)). Individual vehicle captures (including a timestamp, salted hash of vehicle registration, camera/lane of observation and vehicle type) are buffered using this service, and are periodically flushed to object storage (once the buffer fills, or a short time period elapses). The buffered file is also converted to a columnar format (Apache Parquet\(^4\)) for improved query performance.

We next use a serverless data integration offering (AWS Glue\(^5\)), to crawl the object storage buckets containing these intermediate files periodically. This enables the use of serverless SQL queries (via AWS Athena\(^6\)) over the columnar Parquet data. These queries generate aggregated traffic count data, illustrating the total number of vehicles of each type (e.g., petrol car, HGV) that have passed each roadside camera within the current time bin, i.e., the last \(B\) minutes. We use scheduling functionality in a cloud monitoring service (AWS CloudWatch\(^7\)) to trigger the SQL processing (via lightweight serverless functions) for the current time bin. This procedure writes a single file to object storage (AWS S3\(^8\)) per the current time bin, which can later be used as an input to ML workflows.

6.2 MLOps Pipeline and Training

We leverage an AWS SageMaker\(^9\) MLOps pipeline to create and deploy forecasting models. Data scientists can run experiments (e.g., in SageMaker Studio Notebooks) over data held in object storage, using standard libraries such as NumPy, PyTorch, TensorFlow, etc. Once a model has successfully been designed, its source code can be pushed to one of two Git repositories (test, production) hosted in AWS CodeCommit\(^10\).

Once a code update is performed in either repository, the MLOps pipeline provisions a compute instance to perform the necessary pre-processing and training of the model. The trained model is then deployed to a SageMaker inference endpoint (in Foresight, this is on a VM instance; see Foresight Plus enhancements below in Section 7). The MLOps pipeline can be configured to re-train the model periodically, e.g., once per week, to continually incorporate the latest traffic data.

7 Foresight Plus Enhancements

7.1 Extending the Forecasting Scale

Many DL-based forecasting models (and systems such as Foresight) are usually evaluated on benchmark datasets such as METR-LA\(^{23}\), and seek to forecast 1 hour ahead

\(^3\)https://aws.amazon.com/kinesis/data-firehose/
\(^4\)https://parquet.apache.org
\(^5\)https://aws.amazon.com/glue/
\(^6\)https://aws.amazon.com/athena/
\(^7\)https://aws.amazon.com/cloudwatch/
\(^8\)https://aws.amazon.com/s3/
\(^9\)https://aws.amazon.com/sagemaker/
\(^10\)https://aws.amazon.com/codecommit/
While this is useful in short-term forecasting applications, it would be beneficial for transport managers (and members of the public) to be able to extend the forecasting horizons further into the future. Having the ability to forecast traffic patterns multiple hours ahead offers more time to perform interventions, such as re-routing the traffic or travelling via an alternative path.

In light of this, we extend the architecture of the existing Foresight system to offer multiple forecasting scales, henceforth denoted $K$. In particular, we cater for traffic forecasts multiple hours into the future (up to 24 hours ahead at present), with the user able to dynamically select from the supported scales at inference time with no additional provisioning time/costs. Foresight Plus forecasts further into the future than several previous approaches [5, 56], which have considered forecasts up to 4 and 10 hours ahead, respectively.

This is a challenging task for several reasons. Firstly, previous work has stated that GNN models, such as DCRNN [29] and ASTGCN [19], can have extremely high computational cost when attempting to forecast multiple hours ahead [61]. Second, while a high level of forecast accuracy is important regardless of the scale, predictive errors are naturally likely to increase as forecasts extend further into the future. We aim to address these concerns, and present an efficient method to enable DL-based forecasting models to forecast at multiple scales.

For the original 1 hour ahead forecasts generated by Foresight, the underlying data was aggregated into $B$-minute bins, as discussed in Section 3. In Foresight Plus, we introduce an efficient aggregation and upscaling pipeline. This procedure is abstracted away from the forecasting models, which still execute the same prediction function $f(\cdot)$ illustrated in Section 3.1; $T'$ historical traffic signals are used to forecast $T$ future signals.

Recall that for a set of sensors $S$, the entire historical traffic data is denoted as $X \in \mathbb{R}^{H \times |S| \times P}$, where $H$ is the total number of historical $B$-minute bins. To upscale the traffic data to a new scale $k \in K$, we sum each group of $k$ rows of $X$. This results in a new historical time series $X_k \in \mathbb{R}^{\frac{H}{k} \times |S| \times P}$. We then train bespoke forecasting models on each of the upscaled datasets (requiring only minimal adjustments to the models themselves). To process an inference query at scale $k$, data from the appropriate $(k \times T')$-minute bins are first collected. These can be efficiently upscaled as described above, to produce inference input data $X'_k \in \mathbb{R}^{T' \times |S| \times P}$. $X'_k$ is then sent to the appropriate inference endpoint as input; further details of the Foresight Plus inference workflow are given in Section 7.2. We evaluate the effectiveness of our approach in Section 8.3.

### 7.2 Cost-Effective Serverless Inference for Sporadic Workloads

The original Foresight architecture, which was initially designed with short-term forecasting in mind, provides a pipelined MLOps solution for regularly scheduled predictions (to satisfy TIWM operational requirements and integration with traffic reporting systems). However, spatio-temporal forecasting requests that spread over multiple scales may be more ad-hoc in nature (for example, in response to unusual traffic/cultural events). In scenarios such as this, inference requests are often triggered manually (e.g., via mobile apps) and arrive in a sporadic fashion. While Foresight is
well-suited to handling regularly scheduled predictions, its provisioned inference solution may not be cost-effective for sporadic workloads due to low utilization. Further, several endpoint instances may be required to handle bursty traffic and/or multiple models (for different forecasting scales).

With this in mind, we extend Foresight’s existing MLOps pipeline in Foresight Plus by adding a fully serverless inference solution across multiple models/scales. Upon the receipt of an inference request at a given scale (as described in Section 7.1), a lightweight serverless instance performs the necessary data extraction, upscaling and pre-processing, before invoking a further serverless inference endpoint. We maintain a unique serverless inference endpoint for each forecasting scale. These incur no cost when not in use (as is also the case with the aforementioned serverless pre-processing instance), and can rapidly scale to accommodate parallel requests. A serverless inference solution can be significantly more cost-effective than a provisioned alternative for many workloads [39].

7.2.1 Serverless Inference Cost Model

We now formalize the cost model for fully serverless ML inference. This procedure consists of a pre-processing phase, followed by the invocation of a serverless inference endpoint:

\[
C_{\text{Total}} = C_{\text{PreProc}} + C_{\text{Inf}} \tag{3}
\]

Both stages run on lightweight AWS Lambda Function-as-a-Service (FaaS) instances.

We first consider the detailed cost model for the pre-processing phase. \( C_{\text{PreProc}} \) consists of the expenses incurred by running the FaaS instance, as well as those of the corresponding requests to object storage to fetch the necessary data. It is defined as follows:

\[
C_{\text{PreProc}} = C_{\lambda(\text{PreProc})} + C_{S3(\text{PreProc})} \tag{4}
\]

\[
C_{\lambda(\text{PreProc})} = C_{\lambda(\text{Inv})} + T_{\text{PreProc}} M_{\text{PreProc}} C_{\lambda(\text{Run})} \tag{5}
\]

\[
C_{S3(\text{PreProc})} = L C_{S3(\text{List})} + G C_{S3(\text{Get})} \tag{6}
\]

Where \( C_{\lambda(\text{Inv})} \) is the cost of invoking a single FaaS instance, \( T_{\text{PreProc}} \) is the duration of pre-processing (i.e., the runtime of the FaaS invocation), \( M_{\text{PreProc}} \) is the memory assigned to the instance (in MB), and \( C_{\lambda(\text{Run})} \) is the cost per MB-second of FaaS runtime. Note that increasing the amount AWS Lambda memory leads to a larger vCPU allocation, introducing a cost-to-performance trade-off which we examine in Sections 8.3.2 and 8.3.3. \( L \) and \( G \) correspond to the number of required LIST and GET operations, respectively, with \( C_{S3(\text{List})} \) and \( C_{S3(\text{Get})} \) representing their costs. In Foresight Plus, we leverage the metadata provided in object storage solutions to filter out redundant files, hence minimizing the number of LIST and GET requests.

Next, we define the cost model for inference \( C_{\text{Inf}} \) as follows:

\[
C_{\text{Inf}} = T_{\text{Inf}} M_{\text{Inf}} C_{\text{ServInf}} + Y C_{\text{Byte}(\text{In})} + Z C_{\text{Byte}(\text{Out})} \tag{7}
\]

Similarly to above, \( T_{\text{Inf}} \) and \( M_{\text{Inf}} \) reflect the serverless inference instance runtime and memory allocation, respectively. The cost of the AWS Lambda worker used for
inference, $C_{ServInf}$, is $\sim 19\%$ more expensive per MB-second than a regular FaaS instance at present\textsuperscript{11,12}. The final two terms represent the costs of data transfer in/out of the serverless inference instance. As described in Section 3, this has dimensionality $T \times |S|$.

The standard alternative to a serverless inference solution is to permanently provision one or more inference endpoints (hosted on VM instances). Such a solution can incur high passive running costs over time (as instances must always be provisioned to accommodate peak traffic), while achieving poor resource utilization under sporadic inference workloads. We evaluate the cost savings of our fully serverless inference solution in Section 8.3.

8 Experimental Analysis

In this section we present the results of our experiments to test the effectiveness of popular traffic forecasting methods in a new setting. We then evaluate the impact of incorporating DUE data as an additional dimension to the input feature vector. We also consider the performance impact of using the FAAM, in place of a distance-based adjacency matrix, in a GNN forecasting model. After exploring the error profiles of our models, and their efficiency within Foresight, we evaluate the enhancements made in Foresight Plus. In particular, we assess forecasting performance at longer scales, as well as the cost-effectiveness and performance of our fully serverless solution (compared against non-serverless alternatives).

8.0.1 Road Camera Dataset

The anonymized and aggregated data used for the experiments is from a set of ANPR cameras in the West Midlands region of the UK, covering several large conurbations including Birmingham and Coventry. The precise locations of cameras remain private. The set of cameras are spread over a variety of different road types, including many roads from inner city locations and smaller connecting roads. This is different to many prior datasets, such as METR-LA\textsuperscript{23}, where road sensors are typically located on freeways where one can expect a high volume of free-flowing traffic. The quality of this data is high; the rate of missingness is only 2.3%, compared with 8.1% for METR-LA. We use linear interpolation to impute these missing values.

8.0.2 Experimental Setup

Unless stated otherwise, the vehicle count data used in the following experiments was collected between August 5th and December 5th 2021 (inclusive), and was aggregated at 15 minute intervals. DUE data was collected for the same period. The flow was measured between August and November 2021 in order to compute the FAAM. Experiments on Foresight Plus are described in Section 8.3; the data used for these experiments was collected over a different date range.

The data was split into training, validation and test sets in a 70/10/20 ratio. We evaluate performance using mean absolute error (MAE) and mean absolute percentage error.
error (MAPE). We also calculate the error distribution’s coefficient of variation, which we refer to as the error coefficient of variation (ECV). We refer to the set of absolute errors across all test samples as \( E \), and hence \( ECV = \frac{\sigma(E)}{\mu(E)} \). The ECV allows us to compare the dispersion of the error terms across different distributions (i.e., the sets of errors made by different models), as it normalizes by the mean error. A high ECV indicates that predictions are inconsistent.

We evaluate the results firstly over all time periods in the test data, which we refer to as ‘Any Time’ (AT) experiments. We also perform evaluation focusing only on ‘Peak Times’ (PT). We identify peak times as those that have historically shown high average traffic counts, but also high levels of variability. High average traffic counts indicate heavy load on the network, which we assume are periods of interest for transport managers. High levels of variability are a sign of challenging forecasting conditions, and may denote periods of unusual traffic conditions on the network. We identify these periods of interest by first dividing the dataset into weekends and weekdays, and then further splitting each of these into hourly subsets. The mean and coefficient of variation of each subset is then calculated. Any of these subsets with both mean and coefficient of variation in the upper two quartiles is classified as peak

### Table 1
Table of results for all time periods (AT). Evaluation metrics include Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Error distribution Coefficient of Variation (ECV).

<table>
<thead>
<tr>
<th>AT Results</th>
<th>15m</th>
<th>30m</th>
<th>45m</th>
<th>60m</th>
<th>15m</th>
<th>30m</th>
<th>45m</th>
<th>60m</th>
<th>15m</th>
<th>30m</th>
<th>45m</th>
<th>60m</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA</td>
<td>17.212</td>
<td>20.897</td>
<td>24.839</td>
<td>27.326</td>
<td>0.292</td>
<td>0.388</td>
<td>0.485</td>
<td>0.580</td>
<td>1.176</td>
<td>1.182</td>
<td>1.165</td>
<td>1.147</td>
</tr>
<tr>
<td>FFNN</td>
<td>22.452</td>
<td>26.005</td>
<td>28.034</td>
<td>30.386</td>
<td>0.395</td>
<td>0.476</td>
<td>0.532</td>
<td>0.597</td>
<td>1.075</td>
<td>1.075</td>
<td>1.094</td>
<td>1.097</td>
</tr>
<tr>
<td>LSTM</td>
<td>16.477</td>
<td>18.326</td>
<td>21.046</td>
<td>24.389</td>
<td>0.264</td>
<td>0.333</td>
<td>0.403</td>
<td>0.482</td>
<td>1.260</td>
<td>1.263</td>
<td>1.248</td>
<td>1.231</td>
</tr>
<tr>
<td>DCRNN</td>
<td>13.041</td>
<td>14.543</td>
<td>15.716</td>
<td>16.744</td>
<td>0.322</td>
<td>0.419</td>
<td>0.537</td>
<td>0.642</td>
<td>1.260</td>
<td>1.263</td>
<td>1.248</td>
<td>1.231</td>
</tr>
<tr>
<td>DCRNN-RW-T</td>
<td>13.041</td>
<td>14.543</td>
<td>15.716</td>
<td>16.744</td>
<td>0.322</td>
<td>0.419</td>
<td>0.537</td>
<td>0.642</td>
<td>1.260</td>
<td>1.263</td>
<td>1.248</td>
<td>1.231</td>
</tr>
<tr>
<td>DCRNN-RW-G</td>
<td>16.009</td>
<td>18.684</td>
<td>21.260</td>
<td>24.333</td>
<td>0.317</td>
<td>0.354</td>
<td>0.393</td>
<td>0.426</td>
<td>1.173</td>
<td>1.251</td>
<td>1.288</td>
<td>1.116</td>
</tr>
</tbody>
</table>

### Table 2
Table of results for peak time (PT) periods only. Evaluation metrics include Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Error distribution Coefficient of Variation (ECV).

<table>
<thead>
<tr>
<th>AT Results</th>
<th>15m</th>
<th>30m</th>
<th>45m</th>
<th>60m</th>
<th>15m</th>
<th>30m</th>
<th>45m</th>
<th>60m</th>
<th>15m</th>
<th>30m</th>
<th>45m</th>
<th>60m</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA</td>
<td>24.002</td>
<td>31.965</td>
<td>40.559</td>
<td>49.468</td>
<td>0.292</td>
<td>0.378</td>
<td>0.458</td>
<td>0.552</td>
<td>1.011</td>
<td>1.096</td>
<td>1.096</td>
<td>1.058</td>
</tr>
<tr>
<td>FFNN</td>
<td>30.153</td>
<td>35.277</td>
<td>37.862</td>
<td>41.332</td>
<td>0.386</td>
<td>0.460</td>
<td>0.503</td>
<td>0.575</td>
<td>1.063</td>
<td>1.056</td>
<td>1.056</td>
<td>1.025</td>
</tr>
<tr>
<td>LSTM</td>
<td>19.823</td>
<td>21.697</td>
<td>23.508</td>
<td>23.813</td>
<td>0.301</td>
<td>0.373</td>
<td>0.415</td>
<td>0.533</td>
<td>1.193</td>
<td>1.207</td>
<td>1.211</td>
<td>1.186</td>
</tr>
<tr>
<td>DCRNN</td>
<td>16.157</td>
<td>18.431</td>
<td>20.171</td>
<td>21.848</td>
<td>0.415</td>
<td>0.510</td>
<td>0.585</td>
<td>0.651</td>
<td>1.066</td>
<td>1.087</td>
<td>1.103</td>
<td>1.119</td>
</tr>
<tr>
<td>DCRNN-RW-T</td>
<td>16.157</td>
<td>18.431</td>
<td>20.171</td>
<td>21.848</td>
<td>0.415</td>
<td>0.510</td>
<td>0.585</td>
<td>0.651</td>
<td>1.066</td>
<td>1.087</td>
<td>1.103</td>
<td>1.119</td>
</tr>
<tr>
<td>DCRNN-RW-G</td>
<td>16.009</td>
<td>18.684</td>
<td>21.260</td>
<td>24.333</td>
<td>0.317</td>
<td>0.354</td>
<td>0.393</td>
<td>0.426</td>
<td>1.173</td>
<td>1.251</td>
<td>1.288</td>
<td>1.116</td>
</tr>
<tr>
<td>DCRNN-RW-F</td>
<td>15.282</td>
<td>16.983</td>
<td>18.730</td>
<td>19.918</td>
<td>0.359</td>
<td>0.430</td>
<td>0.485</td>
<td>0.527</td>
<td>1.102</td>
<td>1.128</td>
<td>1.145</td>
<td>1.169</td>
</tr>
<tr>
<td>DCRNN-RW-F</td>
<td>15.282</td>
<td>16.983</td>
<td>18.730</td>
<td>19.918</td>
<td>0.359</td>
<td>0.430</td>
<td>0.485</td>
<td>0.527</td>
<td>1.102</td>
<td>1.128</td>
<td>1.145</td>
<td>1.169</td>
</tr>
</tbody>
</table>
time. The only time periods which satisfy this are 7am-8am, and 8am-9am on weekdays, hence we select these as our peak times. This selection also conforms closely to the domain knowledge of our partners at TfWM.

8.1 Forecasting Models

The following forecasting models have been evaluated on the road camera dataset.

- **Historical Average (HA):** We produce a historical average matrix based on the training set. The average reading over the training set is calculated at each sensor in \( S \) for each of the 672 (4x24x7) weekly time steps. To perform inference, we give the historical average value of the target time period as our prediction (the notion of \( T' \) historical traffic signals is not applicable to this method).

- **ARIMA:** We iterate over all sensors and all test examples. In each iteration, we train an ARIMA model\(^{13}\), using the previous \( T' = 100 \) values as the training input.

- **Feed Forward Neural Network (FFNN):** We implement an FFNN, where the input consists of the previous \( T' \) readings across all sensors \( s \in S \). The model produces predictions for the next \( T \) forecasting horizons. The network is constructed with two hidden linear layers, with ReLU activation functions. Model parameters are learned using backpropagation, with an L1 loss function.

- **Long Short Term Memory (LSTM):** This is implemented similarly to FFNN, except using LSTM layers in place of linear layers. Within the LSTM layers, input data is treated as a sequence and temporal patterns are learnt using an additional hidden layer to capture the cell state, which passes information along the sequence.

- **Diffusion Convolutional Recurrent Neural Network (DCRNN):** We select DCRNN [29] as an illustrative example of an effective GNN method. This method has been previously identified as one of the best-performing (GNN) approaches for traffic forecasting on benchmark datasets [8]. The model utilizes a distance-based adjacency matrix to model the spatial relationships between road sensors, and employs diffusion convolution and bidirectional random walks to simulate traffic propagation in the network. We utilize the PyTorch implementation of DCRNN [29].

- **DCRNN-RW-T / DCRNN-RW-G:** DCRNN with DUE adaption to include roadwork data. DCRNN-RW-T associates live roadworks to all sensors within a 1000m distance threshold. DCRNN-RW-G uses thresholded Gaussian kernel normalization (threshold \( k = 0.1 \)).

- **DCRNN-F:** DCRNN with the FAAM representing the underlying graph structure. An acceptable transition period \( \tau \) between sensors is given as 3600 seconds and thresholded Gaussian kernel normalization \( (k = 0.1) \) is applied on the matrix.

- **DCRNN-RW-F:** DCRNN with roadworks (using Gaussian kernel method) and FAAM.

All models are implemented in AWS SageMaker Studio using Python 3.6, on a ml.g4dn.xlarge instance. We use PyTorch 1.8 to implement FFNN, LSTM, and DCRNN (including all variants). Unless stated otherwise, \( T' = T = 4 \), and \( B = 15 \) minutes. In practice, we make predictions over horizons of 15, 30, 45 and 60 minutes (henceforth referred to as 15m, 30m, 45m, 60m). Note that we make predictions over all

\(^{13}\)\( p = 1, d = 0, q = 1 \)
horizons simultaneously, i.e. the model does not gain information about the observed time series at \( t+1 \) when predicting for \( t+2 \). The more distant forecasting horizons (i.e., 45m, 60m) offer transport managers more time to implement pre-emptive interventions on the road network. Hence, performance gains here are particularly valuable.

8.2 Experimental Results: Foresight

We describe the key findings from our experimental results, which are presented in Tables 1 (AT) and 2 (PT). First, we compare the performance of several existing forecasting approaches in our new data setting. We then consider the impact of incorporating roadworks as an exogenous input feature, as well as using flow to determine edge weights in the adjacency matrix. Next, we discuss our findings pertaining to prediction reliability using ECV. Finally, we analyze the efficiency of Foresight.

8.2.1 Analysis of Existing Approaches

We first consider the performance of several existing spatio-temporal forecasting approaches in our new data setting. During the AT experiments it can be observed that DCRNN makes more accurate predictions across all four time horizons (MAE improvements - 15m: 20.9%, 30m: 20.6%, 45m: 25.3%, 60m: 22.7%) compared to the next closest model (LSTM), with the largest improvements seen at the longest forecasting horizons. Similarly, during the PT experiments DCRNN remains the most accurate model. However, it is interesting to note that the improvements compared to LSTM are now much smaller (MAE improvements - 15m: 18.5%, 30m: 15.1%, 45m: 14.2%, 60m: 8.3%), and the trend at longer horizons is reversed where we see the smallest MAE improvements. ARIMA tends to be a competitive model for shorter horizons, during both PT and AT experiments, however the performance deteriorates quickly at longer forecasting horizons, which indicates that this model requires fresh data to support accurate predictions. HA and FFNN make the least accurate predictions across all forecasting horizons.

Different trends emerge when MAPE performance is considered. DCRNN now exhibits poorer performance than LSTM across all forecasting horizons during the AT experiments (MAPE degradation - 15m: 22%, 30m: 25.8%, 45m: 21.3%, 60m: 11.3%); these discrepancies are further exacerbated in PT experiments (MAPE degradation - 15m: 37.7%, 30m: 36.6%, 45m: 41.2%, 60m: 18.3%). This is an interesting result as it suggests that while LSTM makes poorer predictions on average (i.e., MAE), it also makes fewer mistakes of a significant margin, leading to a lower MAPE (this metric is highly sensitive to outliers in the error term). It may therefore be inferred that LSTM is better than DCRNN at predicting unusual traffic patterns, especially at peak times.

In terms of MAPE, ARIMA was shown to be a highly competitive model across all forecasting horizons, outperforming DCRNN in all cases, with more pronounced gains in PT experiments. As ARIMA is retrained on the most recent data when evaluating each test sample (see Section 8.1), it will naturally be more responsive to unusual traffic patterns than models trained using a conventional train/test split. LSTM still largely outperforms ARIMA in regards to MAPE. HA performs particularly poorly on this metric, due to its inability to dynamically respond to current network conditions.
8.2.2 DUE Analysis

We also evaluate the impact of adding dynamic urban events to GNN models. Mixed results are achieved when MAE is considered. DCRNN-RW-G, which associates roadworks using a thresholded Gaussian kernel, generally yields higher MAE than DCRNN across both AT and PT experiments. These discrepancies in MAE are particularly pronounced for long forecasting horizons during peak times (MAE degradation - 45m: 5.4%, 60m: 11.4%). On the other hand, DCRNN-RW-T (binary thresholding) achieves lower MAE compared to DCRNN over all AT experiments. However, it still yields inferior performance at more distant forecasting horizons at peak times (MAE degradation - 45m: 1.4%, 60m: 2.1%).

The results for MAPE present a contrasting picture, where DCRNN-RW-G outperforms DCRNN-RW-T across all experiments. During AT experiments (especially at longer horizons), DCRNN-RW-G achieves significant improvements compared to DCRNN (MAE improvements - 45m: 29.8%, 60m: 29.1%). We observe a similar pattern during peak times. These findings indicate that using a thresholded Gaussian kernel during the construction of a FAAM yields a reduction in large outlier errors (likely resulting in improved performance under unusual road network conditions).

8.2.3 FAAM Analysis

Our experimental results indicate that using vehicle-level flow data to model inter-sensor relationships is an effective strategy. For AT experiments, DCRNN-F achieves lower MAE than DCRNN across all time horizons, and is particularly effective at long forecasting horizons (MAE improvement - 60m: 3.2%). Further, we observe even larger MAE gains for DCRNN-F compared to DCRNN at peak times (MAE improvement - 60m: 8.8%). These findings support the inclusion of vehicle-level flow data into GNN models for improved predictive performance.

We note that leveraging the FAAM in place of a distance-based adjacency matrix (i.e., DCRNN) yields MAPE improvements in all cases. However, the most significant MAPE gains are still experienced by DCRNN-RW-G, indicating that incorporating roadworks is a more effective strategy for minimizing outlier errors. It should be noted that DCRNN-F mitigates much of the degradation in MAPE performance at peak times that DCRNN suffers in comparison to LSTM, while also offering leading MAE results.

8.2.4 Error Coefficient of Variation

As illustrated in Tables 1 and 2, DL models, particularly those which have been enhanced by DUE data or the FAAM, experience the highest ECV (especially at longer forecasting horizons). As shown above, it is at these more distant horizons that the biggest performance improvements (MAE/MAPE) are observed for our augmented models. This suggests that while these solutions produce the best forecasts on average, their errors are the least consistent. This finding is noteworthy, and we would recommend further investigation to better understand its implications.
Table 3 Table illustrating the time periods encapsulated by each horizon, for all five forecasting scales ($k = 1, 3, 6, 12, 24$).

<table>
<thead>
<tr>
<th>Scale</th>
<th>Horizon 1</th>
<th>Horizon 2</th>
<th>Horizon 3</th>
<th>Horizon 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1hr ahead</td>
<td>15mins</td>
<td>30mins</td>
<td>45mins</td>
<td>60mins</td>
</tr>
<tr>
<td>3hrs ahead</td>
<td>45mins</td>
<td>1hr 30mins</td>
<td>2hrs 15mins</td>
<td>3hrs</td>
</tr>
<tr>
<td>6hrs ahead</td>
<td>1hr 30mins</td>
<td>3hrs</td>
<td>4hrs 30mins</td>
<td>6hrs</td>
</tr>
<tr>
<td>12hrs ahead</td>
<td>3hrs</td>
<td>6hrs</td>
<td>9hrs</td>
<td>12hrs</td>
</tr>
<tr>
<td>24hrs ahead</td>
<td>6hrs</td>
<td>12hrs</td>
<td>18hrs</td>
<td>24hrs</td>
</tr>
</tbody>
</table>

8.2.5 Efficiency Analysis

As discussed in Section 3.2, the real-time forecasting task requires that $T_{Total} = T_{Agg} + T_{PreProc} + T_{Inf} \leq B$. In the current version of Foresight, we allow for $T_{Agg} \leq 40$ seconds. For all of the implemented forecasting models, $T_{PreProc} \leq 6$ seconds. Each model except ARIMA achieves $T_{Inf} \leq 2$ seconds. As discussed above, at inference time we train an ARIMA model over the previous $T' = 100$ values for each $s \in S$. For ARIMA, $T_{Inf} \leq 16$ seconds. Hence, all of the presented models achieve $T_{Total} \approx 1$ minute, satisfying Equation 2 with significant headroom for $B = 15$ minutes. Further, these results conform to alternative notions of real-time forecasting [37], where predictions were produced in a single-digit order of minutes.

8.3 Experimental Results: Foresight Plus

In this section, we focus on the new components in Foresight Plus, namely the extended forecasting scales, as well as the fully serverless real-time inference solution. As DCRNN was identified above as the best-performing forecasting model on the ANPR dataset, we select it as for all experiments in this section. To evaluate Foresight Plus, we now utilize traffic data collected between 1st January 2022 and 30th June 2023 (inclusive), which is still aggregated at 15 minute intervals. DUE data was not available for the entire date range in question, hence we exclude it from this analysis. We still leverage the Flow Aggregated Adjacency Matrix (FAAM) to improve the performance of DCRNN.

8.3.1 Extending the Forecasting Scale

We first evaluate the resilience of DL-based forecasting models to the extension of the forecasting scale further into the future, as well as the impact of a larger historical traffic time series. As discussed in Section 7, Foresight Plus offers 5 different forecasting scales; namely $K = \{1, 3, 6, 12, 24\}$ hours ahead. Each still forecasts 4 horizons into the future, with the time period ($B$ minutes) covered by each horizon expanding (uniformly) with the scale. For instance, the ‘1hr ahead’ scale includes forecasting horizons 15mins, 30mins, 45mins and 60mins into the future, while the horizons for ‘12hrs ahead’ reflect 3hrs, 6hrs, 9hrs and 12hrs ahead. The forecasting scales, together with their corresponding horizons, are presented in Table 3.
Table 4 Table of results for all forecasting scales ($k = 1, 3, 6, 12, 24$). Each scale has four unique horizons (see Table 3). Evaluation metrics include Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).

<table>
<thead>
<tr>
<th>Scale</th>
<th>Horizon 1</th>
<th>Horizon 2</th>
<th>Horizon 3</th>
<th>Horizon 4</th>
<th>Horizon 1</th>
<th>Horizon 2</th>
<th>Horizon 3</th>
<th>Horizon 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1hr ahead</td>
<td>12.221</td>
<td>13.265</td>
<td>14.142</td>
<td>14.949</td>
<td>0.170</td>
<td>0.196</td>
<td>0.216</td>
<td>0.235</td>
</tr>
<tr>
<td>3hrs ahead</td>
<td>33.040</td>
<td>39.494</td>
<td>44.492</td>
<td>49.292</td>
<td>0.151</td>
<td>0.195</td>
<td>0.232</td>
<td>0.263</td>
</tr>
<tr>
<td>6hrs ahead</td>
<td>67.128</td>
<td>85.976</td>
<td>97.188</td>
<td>105.632</td>
<td>0.155</td>
<td>0.215</td>
<td>0.247</td>
<td>0.272</td>
</tr>
<tr>
<td>12hrs ahead</td>
<td>152.379</td>
<td>180.859</td>
<td>182.935</td>
<td>191.493</td>
<td>0.179</td>
<td>0.212</td>
<td>0.219</td>
<td>0.227</td>
</tr>
<tr>
<td>24hrs ahead</td>
<td>230.714</td>
<td>255.402</td>
<td>291.103</td>
<td>316.201</td>
<td>0.110</td>
<td>0.124</td>
<td>0.137</td>
<td>0.148</td>
</tr>
</tbody>
</table>

Fig. 2 Figure showing the daily query cost against the daily query volume, for each serverless inference solution as well as a provisioned endpoint alternative. Assumes a uniform distribution over forecasting scales and cold/warm requests.

Our experimental findings for the ‘1hr ahead’, presented in Table 4, indicate that DL/GNN-based forecasting models such as DCRNN benefit significantly from a larger historical dataset of traffic signals. Expanding from 4 to 18 months of ANPR readings leads to a reduction in MAPE from $\sim 26-38\%$ to $\sim 15-37\%$. Beyond the obvious benefit of learning from a greater number of training samples, the fact that the model has observed data from each month in the calendar year allows it to better capture the seasonality of traffic patterns.

Furthermore, we see that prediction errors remain stable as the forecasting scale expands significantly further into the future, and can even improve. We observe minimal degradation between $k = 1, 3, 6, 12$, outside of minor lapses in performance at $k = 12$ (horizon 1). Forecasting errors then significantly reduce for $k = 24$. While perhaps a reflection of the relative stationarity of the ANPR dataset, it is nonetheless
8.3.2 Fully Serverless Real-Time Inference: Cost

We now evaluate Foresight Plus' fully serverless solution for real-time forecasting inference. In this section, we consider three variants of Foresight Plus, each with a different memory allocation. Namely, we configure serverless inference pipelines with 1GB, 3GB and 6GB memory (note that this applies to both $M_{\text{PreProc}}$ and $M_{\text{Inf}}$). These configurations were selected as 1GB/6GB are the current min/max memory allocation for AWS SageMaker Serverless Inference. Note that increasing the memory allocation to AWS Lambda instances (which both the pre-processing and inference stages are executed on) entails an increase in vCPU allocation and network bandwidth.

We first compare cost-effectiveness of Foresight Plus to that of a standard inference solution; that is, a provisioned endpoint, running on VM instances (analogous to the original Foresight solution). In this experiment, we model the non-serverless solution as running with two AWS t2.medium instances (2 vCPU, 4GB memory). We allocate two instances to handle overlapping queries (which could exceed the memory of one instance), and to offer redundancy. To simulate the sporadic inference workloads which Foresight Plus targets, we model queries being received over a 24-hour period, which are uniformly distributed over the 5 forecasting scales ($k = 1, 3, 6, 12, 24$).

We also consider both cold and warm inference requests. Each AWS Lambda request runs in a specific AWS Firecracker container. Upon the receipt of a request, the AWS Lambda service attempts to allocate it to a container. If no containers are immediately available, one must be started; this incurs a delay of several seconds, and

---

Fig. 3 Figure showing the per-request runtime and cost for each serverless inference implementation, across all forecasting scales. Left: cold inference requests. Right: warm inference requests.
is known as a ‘cold start’. Once a request has been completed, its container remains warm for $\sim 15$ minutes. If another request is received during this period, it can be rapidly assigned to the warm container (this effect is termed ‘container re-use’), hence avoiding the cold start delay. It should be noted that container re-use only applies to instances of the same Lambda function, so consecutive requests relating to different forecasting scales will not benefit from this effect. However, all inference requests initially invoke a shared pre-processing function, which will benefit from container re-use for all queries. In this experiment, we assume a 50/50 split of cold/warm inference requests.

On the other hand, the provisioned endpoint solution will have a fixed cost regardless of utilisation, as VM instances are simply billed at an hourly rate. In Figure 2, we observe that each serverless variant is significantly more cost-effective than the provisioned alternative, until the daily query volume becomes very high (1GB: 5851 queries/day, 3GB: 3441 queries/day, 6GB: 1908 queries/day). This highlights the significant cost savings which a lightweight serverless inference solution can provide.

8.3.3 Fully Serverless Real-Time Inference: Performance

We next consider the cost-to-performance ratio of each serverless solution. As illustrated in Figure 3, while 1GB is clearly the cheapest solution, it is also the least performant. We identify the 3GB configuration as a strong compromise of cost and performance, for both cold and warm inference requests. Its end-to-end inference runtimes are significantly faster than 1GB, and are on par with 6GB for all tests. It also costs significantly less than 3x as much as 1GB; this is because Lambda functions are billed by the second of active runtime, so improving the computational efficiency can also reduce costs. While the 6GB solution is usually the most performant, it typically only achieves small improvements over 3GB while incurring significantly higher costs.

However, it should be noted that all three solutions comfortably satisfy the real-time inference requirements described in Sections 3.2 and 8.2.5. Therefore, even the 1GB solution is a viable option for deployment, particularly if the inference workload will predominantly focus on shorter forecasting scales (1-3hrs ahead). However, if responsiveness is of especially important (e.g., if queries are performed by members of the public via a mobile app), and/or many queries will target the longer-to-execute scales, then 3GB is likely a more suitable configuration.

9 Conclusion

In this work, we develop Foresight Plus (in collaboration with Transport for the West Midlands), which builds upon the existing Foresight system to enhance its real-time spatio-temporal forecasting capabilities. We present a novel method for extending the forecasting scale, enabling predictions to be made 1, 3, 6, 12 or 24 hours ahead. This is a significant improvement when compared to the 1 hour ahead scale which is commonly seen in forecasting literature (and was studied in Foresight), and greatly improves the utility of the system for transport managers/members of the public who require accurate forecasts multiple hours in advance. Our experimental analysis shows that GNN forecasting models such as DCRNN can achieve impressive performance
under extended forecasting horizons, with MAPE frequently reducing as the forecasting scale increases. These results highlight the effectiveness of Foresight Plus as both a short and longer-term forecasting system. Further, we consider the fact that many inference workloads will be sporadic in nature, with queries arriving at irregular intervals and being spread over multiple different predictive models/scales. We identify that a fully serverless, pay-as-you-use inference solution is a far more cost-effective approach for such workloads than a provisioned approach (as was utilised in Foresight). We develop an optimized serverless inference procedure in Foresight Plus, and evaluate its scalability, cost and performance across multiple resource configurations. Further work could evaluate the robustness of additional forecasting model types to extensions in the forecasting scale. Also, additional optimisations could be made to accelerate queries made in quick succession (e.g., caching of previous results for a short period of time).

Acknowledgments. We thank Transport for the West Midlands and our collaborators there, particularly Tim Katheru, Andrew Burns and Stuart Lester in the Data Insights team. This research is also supported in part by the Feuer International Scholarship in Artificial Intelligence, WM5G, and the UK Engineering and Physical Sciences Research Council under Grant No. EP/L016400/1.

Declarations

- **Conflicts of Interest:** The authors have no competing interests to declare that are relevant to the content of this article.
- **Data Availability:** The data that support the findings of this study are managed by Transport for the West Midlands, and restrictions apply to the availability of these data, which were used under licence for the current study, and so are not publicly available.

References


Spatiotemporal_traffic_flow_prediction_with_KNN_and_LSTM

[34] Ma D, Song X, Li P (2021) Daily traffic flow forecasting through a contextual
convolutional recurrent neural network modeling inter- and intra-day traffic pat-
https://doi.org/10.1109/TITS.2020.2973279

time traffic information. Transportation Research Part C: Emerging Technologies
105:536–549. https://doi.org/10.1016/J.TRC.2019.06.008

system over europe: The macc-ii daily ensemble production. Geoscientific Model

tion real-time dynamic traffic assignment system. Proceedings - 2nd International
doi.org/10.1109/SIMUL.2010.28

[38] Mohammed O, Kianfar J (2019) A machine learning approach to short-term traffi-
c flow prediction: A case study of interstate 64 in missouri. 2018 IEEE Interna-
tional Smart Cities Conference, ISC2 2018 https://doi.org/10.1109/ISC2.2018.8656924

on cold data using serverless cloud infrastructure. Association for Computing
Machinery, pp 115–130, https://doi.org/10.1145/3318464.3389758, URL https:
//doi.org/10.1145/3318464.3389758

temporal data using deep meta learning. Proceedings of the ACM SIGKDD
International Conference on Knowledge Discovery and Data Mining pp 1720–
1145/3292500.3330884

[41] Sakr M, Ray C, Renso C (2022) Big mobility data analytics: recent
1007/S10707-022-00483-0/METRICS, URL https://link.springer.com/article/
10.1007/s10707-022-00483-0

[42] Schwietering C, Feldges M (2016) Improving traffic flow at long-term road-
TRPRO.2016.06.023

25


