End-to-End Prediction of Parcel Delivery Time with Deep Learning for Smart-City Applications

Arthur Cruz de Araujo and Ali Etemad

Abstract—The acquisition of massive data on parcel delivery motivates postal operators to foster the development of predictive systems to improve customer service. Predicting delivery times successive to being shipped out of the final depot, referred to as last-mile prediction, deals with complicating factors such as traffic, drivers’ behaviors, and weather. This work studies the use of deep learning for solving a real-world case of last-mile parcel delivery time prediction. We present our solution under the IoT paradigm and discuss its feasibility on a cloud-based architecture as a smart city application. We focus on a large-scale parcel dataset provided by Canada Post, covering the Greater Toronto Area (GTA). We utilize an origin-destination (OD) formulation, in which routes are not available, but only the start and end delivery points. We investigate three categories of convolutional-based neural networks and assess their performances on the task. We further demonstrate how our modeling outperforms several baselines, from classical machine learning models to referenced OD solutions. Specifically, we show that a ResNet architecture with 8 residual blocks displays the best trade-off between performance and complexity. We perform a thorough error analysis across the data and visualize the deep features learned to better understand the model behavior, making interesting remarks on data predictability. Our work provides an end-to-end neural pipeline that leverages parcel OD data as well as weather to accurately predict delivery durations. We believe that our system has the potential not only to improve user experience by better modeling their anticipation but also to aid last-mile postal logistics as a whole.

Index Terms—last-mile, parcel delivery, origin-destination, predictive modeling, deep learning.

I. INTRODUCTION

The massive accumulation of data is rooted in the recent advances in the ubiquity of instrumentation, especially in urban environments and everyday products. Sensor systems such as Global Positioning System (GPS) now pervade day-to-day life and are at the core of the technology that makes smart cities possible, an evolving concept that gravitates towards the use of technology to enhance the quality of life of citizens, while attaining a sustainable use of natural resources [1].

Among the various aspects of smart cities, this paper focuses on smart transportation. Tracking time is of most interest in private and public transport services, for vehicles like buses, taxis, and trucks. Estimating time has become even more urgent with traffic continuously worsening and urban mobility getting increasingly complex. For instance, congestion rose by 36% in Los Angeles and 30% in New York from 2010 to 2016, aggravating health issues related to accidents and air pollution [2]. With rapid progress on smart vehicles, GPS data can improve travel time estimates, aiding traffic management, urban planning, and fuel consumption optimization.

The huge growth in retailing and ever-increasing draw of customers to e-commerce point to the need for a reformulation of delivery logistics. For retail companies, providing accurate delivery time estimates is of major importance, for both customer satisfaction and internal organization, since it allows, for example, for better routing and drivers’ scheduling. Also, the existence of large competitors with fast solutions such as next-day deliveries pushes retailers to direct resources to improve their logistics to preserve their market share. Moreover, an accurate estimate of delivery time has shown to be of considerable importance to the customer.

We tackle the problem of last-mile parcel delivery time estimation. A parcel is scanned at many depots before reaching its final destination. The qualifier last-mile denotes the last segment of that journey, from the moment the parcel is sent out for delivery to the moment it gets delivered. Formally, last-mile logistics refers to the last stretch of a business-to-consumer parcel delivery service, from the order penetration point to the consignees preferred destination point [3]. The final segments of the delivery can account for 13 – 75% of the overall supply chain cost [4]. When it comes to the impact of poor estimates on customer satisfaction, the last-mile is the most important segment to be modeled given the closeness to the delivery time and the anticipation by the customer. Furthermore, by providing accurate delivery times at that stage, the retailer can properly manage the expectations of the customer.

Most solutions on travel time estimation require information on the traveled route [5] [6] [7] [8] [9]. Although applicable in the real world, they introduce uncertainty when requiring a pre-planned route for prediction, which in practice may be impossible. This results in high demand for solutions that only depend on the start and end points of the trip, referred to as Origin-Destination Travel Time Estimation (OD-TTE). OD-based methods generally make use of large datasets for modeling spatio-temporal relationships [10] [11] [12] [13] [14] [15] [16] [17]. Some of them tackle delivery time prediction based on the geographical coordinates of the origin and destination, and the week, day, and time when the delivery started [12] [13] [17].

Despite the availability of a range of solutions and learning techniques for predicting delivery times, this problem remains challenging due to a number of open problems:

- Delivery times are strongly reliant on human driving, which is a highly unpredictable phenomenon, despite the existence of standards and protocols defined by delivery companies in general that might not fully hold in practice.

A. C. Araujo and A. Etemad are with the Department of Electrical and Computer Engineering, Queen’s University, Kingston, ON, Canada, e-mail: {18acda, ali.etemad}@queensu.ca.
• Hard-to-predict factors such as weather conditions and traffic variations can have a high impact on delivery time.
• Company-specific logistics involving drivers’ scheduling and route planning will also affect travel time estimation.

In this paper, we investigate a series of deep learning models for tackling the problem of parcel delivery time estimation, formulated as an OD-TTE problem. We explore instances from three different categories of convolutional-based neural networks and evaluate their performances on the task. Namely, we experiment with VGG architectures [13] as a very popular type of convolutional neural network (CNN) used in a variety of domains; residual neural networks (ResNet) [19], given their strong ability to learn effective representations while avoiding the vanishing gradient problem as a result of introducing the skip connections; and squeeze-and-excitation (SE) operators [20], which attempts to better capture channel interdependencies in the data through feature recalibration. We thoroughly analyze these categories of neural networks, and a number of other benchmarks, to identify the best approach capable of accurately predicting last-mile parcel delivery times. While our approach, in general, makes no attempt at directly predicting traffic or human behavior patterns, it provides a direct end-to-end neural pipeline tailored to leverage parcel data information and return accurate predictions that might be further utilized to improve last-mile logistics. We validate the system on a real-world last-mile parcel delivery dataset, collected and provided by Canada Post [21], the main postal operator in Canada. The dataset comprises last-mile information from parcels delivered in the Greater Toronto Area (GTA) within the period from January 2017 to June of the same year, including spatial and temporal data, as well as weather measurements. We compare all of our deep models against a number of benchmarks, including classical machine learning approaches and existing OD solutions from the literature, and we demonstrate the effectiveness of our system in comparison to the baselines.

Our contributions in this paper are summarized as follows: (1) we propose an end-to-end system capable of parcel delivery time prediction under the IoT paradigm for smart city applications. To this end, we study the application of a series of deep CNNs towards solving this problem, relying solely on start and end points, with no knowledge of the route taken. Our solution leverages spatio-temporal information, while also taking weather into account, using daily temperature, rain and snow precipitation, and the amount of snow in the ground; (2) we validate our solution on a very large real-world dataset on parcel delivery in Canada’s largest metropolitan area (GTA), provided by Canada’s main postal operator Canada Post; (3) our end-to-end solutions achieve competitive results with accurate performance, outperforming the benchmark techniques as well as a number of methods in the related work.

II. RELATED WORK
A. Last-Mile Parcel Delivery

According to [22], last-mile logistics can be divided into five components: logistics, distribution, fulfillment, transport, and delivery. Much of the last-mile research proposes new formulations for logistics or attempts to improve specific aspects of it. For example, the work in [23] compares the usage of delivery boxes to reception boxes for deliveries, concluding that the unattended reception of parcels can reduce up to 60% of delivery costs. In [24], the potential of drones for last-mile delivery and the accessibility to such service is investigated. In [25], a conceptual last-mile framework is proposed integrating technologies such as GPS and IoT to enhance the delivery operator visibility through a smartphone. Furthermore, many works have proposed solutions to improve last-mile deliveries by using car trip sharing [26, 27, 28, 29, 30, 31].

Although a lot can be found on last-mile logistics and delivery, sources become scarce when it comes to the specific application of travel time estimation. A recent work [32] proposes DeepETA, a spatial-temporal sequential neural network model for estimating travel (arrival) time for parcel delivery, formulated with multiple destinations and taking into account the latest route sequence by knowing the order in which the parcels were delivered. The model overcomes state-of-the-art methods on their real logistics dataset. However, this approach relies on the availability of the delivery routes, which renders it incompatible with our work’s OD-based formulation.

B. Travel Time Estimation

Travel time estimation is an area concerned with estimating travel durations from data. While it encompasses last-mile delivery, most works in the area have focused on other applications such as taxi trip durations [8, 9, 10]. The literature on travel time estimation is often subdivided into two categories: solutions that depend on knowing the traveled routes, hence called route-based methods, and those that only require the start and end points of the trip, being referred to as Origin-Destination Travel Time Estimation (OD-TTE) [16]. Recent effort has been made towards OD research, encouraged by the fact that in many cases, due to either privacy concerns, tracking costs, or the lack of ability to fully pre-plan the route, only limited information is available [15].

1) Route-based Travel Time Estimation: In [5], a neural network was used to predict bus travel times. Given hour-of-day, day-of-week, and weather conditions (precipitation), the network would estimate the travel time for each route segment, further adjusting the prediction using Kalman filtering and up-to-the-minute bus locations. Similarly, the work in [6] estimates the duration of a bus ride based on a scheduled route and a source-destination pair. In [7], travel time was predicted based on floating-car data, by combining linear models, deep, and recurrent neural networks to explore road segment sequences. In [8], DeepTTE was proposed, an end-to-end deep learning framework that predicts travel time for an entire path as a whole. Associating convolutional and recurrent stages, the approach experimented on two trajectory datasets and outperformed state-of-the-art methods. Similarly, DEEPTRAVEL [9] estimates the travel time of a whole trip directly, given a query path. The paper proposed a feature extraction structure to effectively capture spatial, temporal, and traffic dynamics, allowing for accurate estimations.

2) OD Travel Time Estimation: An approach was presented in [10] to discover spatio-temporal patterns in OD displacements. Spatial clustering of coordinates was used to identify
meaningful places, and flow measures were extracted from the clusters to understand the spatial and temporal trends of movements. In [11], taxi OD data was used for estimating the hourly average of travel times for urban links. For each trip, possible paths were inferred and the link travel times were estimated. The work in [12] estimated daily OD trips from triangulated mobile phone records, converting the phone records into clustered locations where users engaged in activities for an observed duration. Trips were then constructed for each user between two consecutive observations in a day and scaled up using census data. Rather than using routes, OD-TTE was directly tackled in [13], where a Simple Baseline to Travel Time Estimation (SB-TTE) was proposed. The work used large-scale trip data to estimate travel time between two points based on origin, destination, and actual distance traveled. It should be noted, however, that actual traveled distance is not available in real-world settings prior to the travel taking place and without a pre-planned route. In [14], insights were borrowed from network optimization to demonstrate that it is possible to extract accurate travel time estimations from a large OD dataset to reconstruct the traffic patterns in a whole city. In [15], a model called Spatio-Temporal Neural Network (ST-NN) is proposed for OD travel time estimation, which jointly predicts trip durations and traveled distances. In [16], a Multi-task Representation learning model for Arrival Time estimation (MURAT) was proposed, a solution claimed to produce meaningful data representations leveraging the underlying road network and spatio-temporal priors. Although an OD-based approach, MURAT requires route information for training purposes. Finally, in [17], we explored an NYC taxi trip dataset in the context of OD-TTE. In that work, deep neural networks were used for estimation travel times, outperforming a number of baseline techniques.

Table I summarizes the related work, organized according to problem type, that is, between route-based and route-free methods. The table also specifies what type of datasets were used, for instance, bus records, taxis, etc. It also lists what information is used as input data. Lastly, Table I outlines the prediction method of each work so as to better understand the variety of tools that have been used in this problem domain.

III. SYSTEM ARCHITECTURE

A. Smart City Contextualization

Smart transportation is one of the key services that define a smart city, being indeed an integral part of any smart city initiative. Smart transport systems are meant to improve safety, security, efficiency, and environmental friendliness of transport systems, by combining Intelligent Transportation Systems (ITS) with other vehicular technologies such as autonomous vehicles [33]. The deployment of smart transportation is enabled by technologies such as GPS and the Internet of Things (IoT), a communication paradigm according to which objects of everyday life will be electronically equipped to communicate with one another and with the users, becoming an integral part of the Internet [34].

Even though smart transportation refers to using smart technologies to improve citizens’ mobility experience, its benefits certainly go beyond that. For instance, smart transportation also provides the infrastructure to enable other transport-related initiatives, such as predictive systems for collision avoidance, traffic monitoring, and real-time traveler information. In this context, a system for travel time estimation, such as the one herein proposed, may be regarded as a beneficiary and a contributor to the realization of smart transportation.

Figure 1 illustrates the typical scenario for last-mile delivery, in which a parcel is scanned as out-for-delivery in a local depot, collected by an agent, who then delivers it at the designated address. During the process, relevant data is recorded, such as depot geographical coordinates (GPS) and destination postal codes. Weather information is also available, including records on daily temperature, precipitation of rain and snow, and the amount of snow in the ground.

In the same figure, we illustrate the deployment of such a system in an IoT-enabled scenario. A diagram presents an end-user (receiver) perspective, who should be able to request the status of their parcel at any time or place, especially during the expected day of delivery. A cloud-based server, which works in conjunction with a database, would then process the user request, feed it into an appropriate machine learning inference algorithm, and return the last-mile estimates to the user.

B. IoT Architecture

Although a number of flexible layered architectures have been proposed for IoT standardization, a reference model has not yet been established [35]. Still, in its most basic modeling, the IoT architecture generally comprises three layers of abstraction. This three-layer architecture consists of the Perception Layer, the Network Layer, and the Application Layer. In the following, we present them according to [36] while placing them in the context of our problem. Finally, Figure 2 summarizes the IoT layered architecture.

![Image of IoT architecture](image-url)
1) Perception Layer: This layer represents the information source of the IoT, that is, sensors and data collection devices. For example, the terminal board in delivery trucks enables the tracking of relevant data such as speed and gas consumption. Also, Automatic Vehicle Location (AVL) enables reliable transmission of current vehicle location. This layer could also include cameras or barcode labels used by delivery agents, besides any additional devices required for communication with the depot or the addressee, such as smartphones. Our solution will mainly exploit this layer by providing delivery agents with portable scanners associated with smartphones. With that, the agents would scan the parcels at the moment of leaving the depot, as well as at their final destinations, therefore generating the timestamps needed as inputs to our models. Each scan would also provide the respective GPS coordinates collected on their smartphones. Moreover, the agent would still be able to provide user notification in case of unforeseen events relative to the delivery.

2) Network Layer: This layer essentially transmits information from the Perception to the Application Layer. It is established based on the current mobile telecommunication and internet infrastructure. In terms of wireless communication, examples include Global System for Mobile Communication (GSM), General Packet Radio Service (GPRS), and the 4th Generation communication (4G) or 5th Generation (5G), which can allow for long-distance transmission of vehicular data. Besides, private wireless communication such as WiFi and Bluetooth can be used for communicating sensors within the trucks, for example. For our proposed last-mile delivery solution, this layer would simply make use of the mobile network in the delivery agents’ smartphones in order to transmit the required coordinates and timestamps back to their respective depots.

3) Application Layer: This layer receives information from the Perception Layer and processes it into useful applications for staff and customers. It can be subdivided into platform and business sub-layers. The platform sub-layer comprises all algorithms used, for example, for data mining and inference, with cloud computing support. The business sub-layer holds the applications per se. Regarding the platform sub-layer for our last-mile delivery solution, our deep learning models for travel time estimation will operate as the predictive analytics engine. Our models and data would be stored on the cloud and used for inference by customer demand as well as for training the models. On the other hand, the business sub-layer comprises not only the user device but also the driver’s smartphone. Based on its definition, we place the driver smartphone in both Perception in Application layers. The driver device would automatically report truck location, speed, and driving behavior, while also notifying emergencies regarding the safety of drivers. The user device in turn would allow for collecting real-time delivery status and notifications to customers, who would be able to query updates at any time using any device connected to the Internet.

### IV. METHOD AND EXPERIMENTS

#### A. Problem Formulation

We tackle the problem of last-mile parcel delivery time estimation as an *Origin-Destination Travel Time Estimation* (OD-TTE) problem, that is, we aim to predict travel time based on start and end locations, not relying on the actual route taken. Given dataset $\mathcal{D}$ containing $N$ samples $d$ defined as:

$$\mathcal{D} = \{ d_i \mid i \in [1, N] \}$$

$$d = (x, y, t),$$

TABLE I

| Reference            | Year | Data Type | Location       | Input Data                                                                 | Prediction Method                                      |
|----------------------|------|-----------|----------------|----------------------------------------------------------------------------|--------------------------------------------------------|
| Chen et al. [15]     | 2015 | Bus       | New Jersey     | GPS, travel distance, departure timestamp                                  | Multinomial logistic function                          |
| Gal et al. [16]      | 2017 | Bus       | Dublin         | GPS, time period in a year, month and a day, day-of-week, hour-of-day     | Arthur’s travel time estimation                        |
| Wang et al. [17]     | 2018 | Taxi      | Chengdu/Beijing| GPS, distance of path, day-of-week, time-slot of travel start, weather     | LSTM                                                   |
| Wu and Wu [18]       | 2019 | Parcel    | Beijing        | GPS, date, GPS, timestamp, driving state                                  | BiLSTM                                                 |
| Guo et al. [19]      | 2012 | Taxi      | Shenzhen       | GPS, timestamp                                                             | Clustering; statistical summaries                      |
| Zhan et al. [20]     | 2013 | Taxi      | Small area     | GPS, road network, hourly intervals                                       | Multinomial logistic function                          |
| Alexander et al. [21]| 2015 | Mobile    | Boston         | Call detail records (CDRs) with time-stamped GPS coordinates              | Rule-based user activity inference                      |
| Wang et al. [22]     | 2016 | Taxi      | NYC/Chengdu/Beijing| GPS, travel distance, departure time (weekly-basis and absolute)       | Neighbor-based estimation                              |
| Jindal et al. [23]   | 2017 | Taxi      | NYC            | GPS, travel distance, departure timestamp                                | Multi-Layer Perceptron (MLP)                          |
| Li et al. [24]       | 2018 | Taxi/Didi | Beijing/NYC    | GPS, travel distance, departure time (hourly and day-of-week)             | ResNet-based neural network                            |
| de Araujo and Etemad [25]| 2019 | Taxi      | NYC            | GPS, day-of-week and hour-of-day taken at departure                       | Multi-Layer Perceptron (MLP)                          |
| Bertsimas et al. [26]| 2019 | Taxi      | NYC            | GPS, date and time                                                         | Shortest-path convex optimization                     |
Fig. 2. An IoT 3-layered architecture is shown: the layer descriptions are adapted from [36] to the context of the last-mile parcel delivery problem, highlighting some major points for each layer. We highlight in bold the terms relevant to the context of our last-mile delivery solution.

where $d$, the last-mile parcel delivery record, consists of:

- the longitude/latitude coordinates for the depot $(o_x, o_y)$;
- the coordinates for the postal code destination $(d_x, d_y)$;
- the timestamp $o_t$ relative to the out-for-delivery scan;
- a vector $w$ encoding weather conditions;
- the registered delivery time target $t$, the time elapsed from $o_t$ up to the moment when the last scan happened.

Problem (OD Travel Time Estimation). Assuming the availability of a dataset of past deliveries $D$ (according to Equation 1), the end goal of OD-TTE is to leverage $D$ so that, given a parcel query $q = (o_x, o_y, o_t, d_x, d_y, w)$, its correspondent travel (delivery) time $t$ can be predicted with minimal error.

B. VGG-based Architectures

The first class of convolutional networks explored is based on VGG modules [18]. The VGG block generally comprises a number of convolutional layers followed by a pooling layer. Figure 3 shows a 1D version of the VGG block, formed by two convolutional ReLU-activated layers with a kernel of 3, and a max-pooling layer with a stride of 2.

By combining such blocks, we build a series of VGG-based neural networks of varying depths. Figure 4 presents a general diagram of the VGG architecture utilized. On the input end, we concatenate the encoded features, which feeds a series of VGG blocks of increasing depth, allowing the model to learn complex representations in the data. Then, this deep representation is flattened into a one-dimensional tensor and fed to a fully-connected two-layered network, which outputs the estimated prediction of delivery time.

The first and shallowest reported VGG-based model has 3 blocks of increasing depth, a total of 6 convolutional layers and 3 pooling stages. In principle, the number of pooling layers, and therefore VGG blocks, is limited by the dimensionality of the input. In our case, for an input dimension of 12, that limit would be 3 (for a stride of 2). To account for that when building deeper VGG models, we distribute the pooling layers evenly, while prioritizing earlier layers if needed. We further expand the experiments to deeper models containing up to 10 VGG blocks, therefore with 6 to 20 convolutional layers.

C. ResNet-based Architectures

Residual learning explores the building of computational units that learn residual functions of the input, in contrast to learning unreferenced mappings [19]. This helps to mitigate the problem of vanishing gradients that affects deeper networks. Practically, a shortcut (or skip) connection is associated with a two-layer feed-forward ReLU-activated neural network, resulting in the ResNet block, the residual learning unit used in ResNet architectures [19], as shown in Figure 5.

The second class of convolutional neural networks investigated is based on this concept, i.e. residual modules (ResNet). Their innovation relates to the presence of skip connections serving as auxiliary paths for gradient flow. It is worth noting that ResNet blocks by definition do not presume pooling. Figure 6 shows the counterpart architectural diagram for the ResNet models built. Similarly, we experiment with models of varying depth, ranging from 3 to 10 residual blocks, that is, models with 6 to 20 convolutional layers.

D. Squeeze-and-Excitation Architectures

The Squeeze-and-Excitation (SE) block [20] is an approach for modeling interdependencies between channels in convolutional networks. While prior architectures usually learn spatial and channel-wise feature maps jointly, SE blocks attempt to
Generally, global average pooling is used, so each channel the squeeze operator outputs a channel-wise vector that squeezes the input along its spatial dimension, collapsing the transformations that take place.

The SE block can be used to re-calibrate the feature map from any transformation $F : \mathbf{X} \rightarrow \mathbf{U}$, $\mathbf{X} \in \mathbb{R}^{D' \times C'}$, $\mathbf{U} \in \mathbb{R}^{D \times C}$, so $F$ could represent a regular convolution, a residual module, an inception module, etc. In the following, the main components of an SE block adapted to 1D inputs for our problem space are presented. Figure 7 summarizes the SE block pipeline: a diagram at the top illustrates how the input tensor dimensions change through each stage of the block; at the bottom, a complementary schema highlights the transformations that take place.

**Squeeze operator.** The first operator $F_{sq}$ in the SE block squeezes the input along its spatial dimension, collapsing the tensor so that only the channel dimension remains. Therefore, the squeeze operator outputs a channel-wise vector $s \in \mathbb{R}^{C}$. Generally, global average pooling is used, so each channel component $u_c$ is mapped to a scalar $s_c$ according to:

$$s_c = F_{sq}(u_c) = \frac{1}{D} \sum_{i=1}^{D} u_c(i). \quad (3)$$

**Excitation operator.** The second operator $F_{ex}$ in the SE block is intended to exploit the “squeezed” representation $s$ in order to capture the importance of each channel regardless of the spatial distribution. That is achieved by a two-layer fully connected (FC) neural network given by:

$$e = F_{ex}(s) = \sigma(W_2 \text{ReLU}(W_1 s)), \quad (4)$$

where $e \in \mathbb{R}^C$, $W_1 \in \mathbb{R}^{C/r \times C'}$, $W_2 \in \mathbb{R}^{C \times C/r}$. The first layer reduces the depth of $s$ by a factor of $r$, while the second restores it back to $C$. A sigmoid activation function at the last layer assures that channels can be independently excited.

**Feature scaling.** In the final step, each channel of $U$ is re-scaled by the correspondent factor in the vector $e$, an independent re-calibration of features that allows the model to select most relevant ones. The following equation denotes a channel-wise multiplication:

$$U = e \cdot U \quad (5)$$

In this study, we incorporate the SE block into our best VGG and ResNet models as an attempt to better capture more intricate relationships between the learned feature maps. For the VGG models, we insert the SE block after the non-linearity, right before the pooling operator, while in the ResNet architectures, the SE block is best placed before the aggregation with the skip connection. We illustrate the augmented versions of each block in Figure 8.

### E. Training and Implementation Details

For training our model we utilized the Adam optimizer [50], and the Mean Squared Error (MSE) as loss function. Considering a given set of $N$ samples $y_i$ of delivery times and $N$ associated predictions $f_i$, the MSE is defined by:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} |y_i - f_i|^2. \quad (6)$$

We monitor the loss over the validation set and apply early stopping if no improvement happens after 25 consecutive epochs. We set an initial learning rate of $10^{-4}$ and reduce it by half every 40 epochs. We utilize a batch size of 64 samples. We implement our models using the Keras API with TensorFlow [51] backend, using an Nvidia RTX 2080 Ti GPU.

### F. Dataset

We utilize a real-world dataset on last-mile parcel delivery provided by Canada Post, the main postal operator in Canada. The dataset holds 3,253,252 deliveries within the Greater Toronto Area (GTA), from January to June of 2017. There are 72 different depot locations in the data, covering 83,730

---

**Fig. 6.** A general diagram is presented describing the architecture of our deep residual models. Likewise, the number of blocks is not specified since we experiment with different instances of varying depth.

**Fig. 7.** A diagram on the squeeze-and-excitation block to 1D data is shown.

**Fig. 8.** Diagrams for the squeeze-and-excitation versions of the VGG and ResNet blocks. The insertion point for SE-VGG and SE-ResNet are respectively before pooling and before aggregating the skip connection.
delivery points. Geographical coordinates (longitudes and latitudes) are given for the depots, but, due to privacy, delivery addresses are limited to postal codes, from which coordinates are obtained. Timestamps are also available, indicating the date and time when the *out-for-delivery* scan and the *delivered* scan happened, from which delivery time is computed.

The last-mile parcel delivery problem shows a clear imbalance between the number of origins and destinations, unlike other OD-based problems such as predicting taxi trip durations, for example. Since there are considerably fewer depots than delivery addresses, each depot usually covering a defined region, it is reasonable to expect spatio-temporal patterns to vary across different depots as they relate to different traffic and road conditions. Hence, it is intuitive to interpret the data spatially as a set of graphs, each one centered in one depot, as shown in Figure 9, which illustrates the distribution of the data within GTA. For visualization purposes, the 5 busiest locations are shown, which are jointly responsible for 35.19% of the deliveries in the data. Figure 10 shows the distribution of delivery time for each of the 10 busiest depots.

Weather information is also used, as it configures a critically relevant factor for delivery times. Besides, the period spanned covers diametrically opposite climate conditions (e.g., January vs. June). The features used are daily temperature, rain and snow precipitation, and amount of snow on the ground. Figure 11 shows how these features are distributed in the data. The plot on the top-left shows the daily temperature, which ranges from \(-10^\circ\) to \(30^\circ\). This figure shows wide distributions of different weather conditions, indicating that deliveries have occurred under many different situations, contributing to the hardness of this dataset. We observe that rain precipitation has been as high as 40 millimeters, and snow precipitation has reached up to 16 centimeters. While local measurements might have been different, the reported values are daily averages, computed across multiple weather stations. Finally, Table II summarizes some high-level information on the GTA dataset.

### G. Input Features

In terms of coordinates, especially for final destinations, we quantize the latitudes and longitudes to a coarser resolution to alleviate GPS noise and to cluster neighboring locations together. Figure 12 illustrates the quantization, where the red dots represent the binned coordinates. Next, we normalize them based on a geographical bounding box enclosing the GTA. The Haversine distance, which measures the distance between two points on Earth based on their coordinates, is computed for each delivery and further normalized as well. From the out-for-delivery timestamp, we extract hour-of-day, day-of-week, and the week itself. The daily temperature, precipitation of both rain and snow, and the amount of snow on the ground are also used. All temporal and weather features are normalized within the range of 0 to 1. Finally, we concatenate all aforementioned features into a 12-dimensional input vector.
We report the absolute growth and watching for overfitting.

We define a grid search over their main hyperparameters, constraining exaggerated growth. We report two instances of 2 and 5 ReLU-activated layers, each with 50 units, trained with early stopping based on validation loss.

\[ MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - f_i| \]

\[ MAPE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - f_i}{y_i} \right| \]

\[ MARE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - f_i}{|y_i|} \right| \]

Additionally, we measure the window of error (±EW) that includes a certain percentage of the delivery predictions. For a desired percentage of \( p \), EW can be found by:

\[ \frac{1}{N} \sum_{i=1}^{N} u(EW - |y_i - f_i|) = p \]

where \( u(\cdot) \) is the step function, counts the number of samples whose absolute errors are within ±EW hours. In particular, we report EW_{90\%}. This metric was indicated by Canada Post as a common evaluation metric in that organization.

I. Benchmarks

1) Random Forest: we explored random forests as a representative of classical ensemble methods. We define a grid search over their main hyperparameters, constraining exaggerated growth and watching for overfitting.

2) Multi-Layer Perceptrons: we also explored neural networks based on Multi-Layer Perceptrons (MLPs). We report two instances of 2 and 5 ReLU-activated layers, each with 50 units, trained with early stopping based on validation loss.

A remark on referenced baselines. Adaptations were necessary when re-implementing the solutions found in the literature, mostly because the closest approaches, although OD-based, would still indirectly rely on the availability of the actual distance traveled by the driver [13, 15, 17], information usually available in public datasets.

V. Results and Analysis

A. Evaluating VGG Architectures

We now present the performance results on the VGG architectures over the GTA dataset. We varied the number of VGG blocks to identify the best trade-off between performance and complexity. Table III enlists all metrics computed over the validation set for the VGG models with the depth varying from 3 to 10 blocks. Regarding performance, even though we observe an improvement as we explore additional blocks, it deteriorates on the deeper instances, particularly after VGG-7 (14 layers). This could be due to overfitting or the vanishing gradient problem. We display the validation losses during training in Figure 13 in which a similar trend can be observed. Accordingly, VGG-6 displays the best performance: although it does not have the lowest loss, it shows the best overall results when considering all the metrics.

| VGG | MSE | #MSE | MAE | EW_{90\%} | MAE | MARE |
|-----|-----|------|-----|-----------|-----|-------|
| VGG-3 | 1.8273 | 1.3518 | 0.9789 | 2.0117 | 44.04 | 30.71 |
| VGG-4 | 1.7542 | 1.3245 | 0.9417 | 1.9527 | 41.83 | 29.55 |
| VGG-5 | 1.7028 | 1.3049 | 0.9861 | 1.8749 | 39.45 | 28.12 |
| VGG-6 | 1.6979 | 1.3030 | 0.9867 | 1.9160 | 39.36 | 27.82 |
| VGG-7 | 1.6743 | 1.2939 | 0.9719 | 1.8864 | 41.46 | 28.09 |
| VGG-8 | 1.6955 | 1.3021 | 0.9003 | 1.9623 | 40.35 | 28.25 |
| VGG-9 | 1.7411 | 1.3195 | 0.9363 | 1.8931 | 42.17 | 29.38 |
| VGG-10 | 1.7494 | 1.3227 | 0.9408 | 1.9258 | 41.94 | 29.52 |

B. Evaluating ResNet Architectures

Next, we evaluate the different number of layers in our ResNet architectures. Table IV shows the results for instances with depth varying from 3 up to 10 ResNet blocks. Even though we can observe a generally monotonic decrease across all metrics as the models go deeper, the improvement diminishes. A similar trend is observed in the validation loss in Figure 14. Small circles denote the epochs at which the models
have reached their best performance, illustrated in the zoomed-in plot. Even though ResNet-8 reaches the lowest MSE, the best MAPE value, for example, was shown by ResNet-10. Metrics penalize errors differently and it is normal for them not to be always consistent, reason why we report six of them.

The ResNets show a quite different evolution in comparison to the VGGs: the skip connections significantly help the learning process by providing direct paths for gradient flow during training. Accordingly, we infer that the decline in the performance of deeper VGGs may have been gradient-related.

TABLE IV
EVALUATING THE EFFECT OF DEPTH IN ResNet ARCHITECTURES

| Architectures | MSE  | RMSE | MAE  | EWmae | MARE |
|---------------|------|------|------|-------|------|
| ResNet-3      | 1.3401 | 1.3107 | 0.9503 | 1.9569 | 42.88 | 29.81 |
| ResNet-4      | 1.6320 | 1.2775 | 0.8911 | 1.8524 | 39.37 | 27.96 |
| ResNet-5      | 1.5752 | 1.2551 | 0.8510 | 1.8419 | 37.28 | 26.70 |
| ResNet-6      | 1.5614 | 1.2495 | 0.8516 | 1.8350 | 37.23 | 26.72 |
| ResNet-7      | 1.5496 | 1.2448 | 0.8461 | 1.7942 | 36.61 | 26.55 |
| ResNet-8      | 1.5492 | 1.2447 | 0.8404 | 1.7680 | 36.55 | 26.37 |
| ResNet-9      | 1.5617 | 1.2497 | 0.8475 | 1.8248 | 36.95 | 26.59 |
| ResNet-10     | 1.5540 | 1.2486 | 0.8411 | 1.7938 | 36.21 | 26.39 |

C. Assessing the Squeeze-and-Excitation Augmentation

We now report the results on the SE-based models. As discussed, the SE block is an architectural unit designed to enhance the representational power of a model through feature re-calibration [20]. Accordingly, we augmented VGG-6 and ResNet-8, our best VGG and ResNet instances, with SE blocks to assess their effect on the performance.

Table V shows the metrics for all four models. We observe that the insertion of SE blocks does not seem to improve performance. In fact, the evolution of losses in Figure 15 for each pair is quite similar, especially for ResNet, and the SE blocks could actually be slightly worsening the performance.

D. Analysis of Model Complexity

Here we discuss the differences in model complexity in our results and associate them to their relative performance to assess which instances display the best trade-off.

Starting with ResNet-3, we append a new block twice as deep as the previous one, resulting in ResNet-4 with 2.05 million trainable parameters, and we repeat that for ResNet-5, reaching 7.60 million. If we were to proceed with this pattern, the next model would reach 29.19 million parameters, which is unnecessarily large. For reference, ResNet-50 [19] itself has 25 million. The same reasoning applies to the VGG architectures. In order to avoid a brute force procedure, the approach for expanding the models was to replicate the middle layer first and then alternate between a prior and posterior layer, moving towards both ends of the convolutional pipeline. That process is followed from ResNet/VGG-5 up to ResNet/VGG-10.

Table VI summarizes all architectures in terms of the number of blocks and their respective depth values, enlisting the number of trainable parameters for each case. VGG-6 has the lowest error among the VGGs, displaying the best complexity trade-off since it only has 6.73 million parameters. As for the ResNets, even though we observed ResNet-10 as the model with the best metrics, the improvement obtained is incremental for the complexity added, which doesn’t justify using it. Also, the results for ResNet-9 are not consistently better than its predecessor. Such remarks favor ResNet-8 as the most appropriate model depth. Finally, Table VII compares VGG-6 and ResNet-8 with their SE-augmented versions. Although the complexity added by the SE blocks is indeed minimal, mainly due to the low number of parameters from their two FC layers, the metrics still favored the original models.

| Architectures | MSE  | RMSE | MAE  | EWmae | MARE |
|---------------|------|------|------|-------|------|
| VGG-6         | 1.6979 | 1.3030 | 0.8867 | 1.9160 | 39.36 | 27.82 |
| SE-VGG-6      | 1.5492 | 1.2447 | 0.8404 | 1.7680 | 36.55 | 26.37 |
| ResNet-8      | 1.5516 | 1.2456 | 0.8512 | 1.8292 | 37.34 | 26.71 |

Table VII compares VGG-6 and ResNet-8 with their SE-augmented versions. Although the complexity added by the SE blocks is indeed minimal, mainly due to the low number of parameters from their two FC layers, the metrics still favored the original models.
TABLE VI
MODEL COMPLEXITY COMPARISON: NUMBER OF TRAINABLE PARAMETERS IN VGGs AND ResNETs

| X | Depth Summary | # Trainable Parameters |
|---|---|---|
| VGG-X | ResNet-X |
| 3 | [64, 128, 256] | 397k | 580k |
| 4 | [64, 128, 256, 512] | 1.59M | 2.05M |
| 5 | [64, 128, 256, 512, 1024] | 6.34M | 7.60M |
| 6 | [64, 128, 256, 512, 1024] | 6.73M | 7.99M |
| 7 | [64, 128, 256, 512, 1024] | 8.30M | 9.57M |
| 8 | [64, 128, 256, 512, 1024] | 8.40M | 9.69M |
| 9 | [64, 128, 256, 512, 1024] | 8.43M | 9.69M |
| 10 | [64, 128, 256, 512, 1024] | 14.72M | 15.98M |
| | | | |

TABLE VII
MODEL COMPLEXITY COMPARISON: NUMBER OF TRAINABLE PARAMETERS IN SE-AUGMENTED MODELS

| # Trainable Parameters |
|---|
| VGG-6 | 6,730,907 |
| ResNet-8 | 9,664,923 |
| SE-VGG-6 | 6,916,071 |
| SE-ResNet-8 | 9,885,583 |

E. Performance Comparison

In the following, we compare the results obtained by the benchmarks against our best deep learning models. Table VIII shows the performance metrics over the validation set. Among the machine learning benchmarks, the MLP-2 neural network works best, followed by the random forest. As for the related work benchmarks, ST-NN-TimeDNN [15] has relatively low performance and it is not a good fit for our dataset complexity. DNN [17], on the other hand, is deep enough to outperform the classical benchmarks. Finally, SB-TTE [13] works remarkably well despite the simplicity of the model. Lastly, ResNet-8 has the best results, followed by VGG-6 and SB-TTE. Although we elect one model as the best solution for our problem, we still provide a series of alternatives for solving it.

TABLE VIII
RESULTS COMPARISON

| Model | MSE | RMSE | MAE | $R^2_{\text{val}}$ | MARE | MAPE |
|---|---|---|---|---|---|---|
| SB-TTE [13] | 2.0918 | 1.4463 | 0.9728 | 2.0698 | 39.34 | 30.51 |
| ST-NN-TimeDNN [15] | 2.3629 | 1.5372 | 1.1891 | 2.2197 | 56.87 | 37.31 |
| DNN [17] | 2.2649 | 1.5050 | 1.1377 | 2.2032 | 50.34 | 35.69 |
| Random Forest | 2.4412 | 1.5831 | 1.2081 | 2.2573 | 52.76 | 37.90 |
| MLP-1 | 2.4506 | 1.5655 | 1.2147 | 2.2748 | 58.07 | 38.11 |
| MLP-2 | 2.2940 | 1.5146 | 1.1587 | 2.2017 | 53.88 | 36.36 |
| VGG-6 | 1.6979 | 1.3030 | 0.8867 | 1.9160 | 39.36 | 27.82 |
| SE-VGG-6 | 1.7048 | 1.3057 | 0.9003 | 1.8814 | 39.86 | 28.25 |
| ResNet-8 | 1.5492 | 1.2447 | 0.8404 | 1.7680 | 36.55 | 26.37 |
| SE-ResNet-8 | 1.5516 | 1.2456 | 0.8512 | 1.8292 | 37.34 | 26.71 |

F. Error Analysis

We discuss the error distribution across the data and how that relates to the behavior of ResNet-8. For visualization purposes, we concentrate on the MAPE and MARE metrics.

1) Variations across depots: We have discussed the imbalance between the number of origins and destinations in a parcel delivery problem, and the assumption of different spatio-temporal patterns for each depot. Figure 16 illustrates the spatial distribution of the errors and depots, overlaid onto a map of the GTA. Circles of equal radius are drawn at each location and a color map is defined according to the MAPE. As shown, bluer circles with lower MAPE tend to be found on the outskirts of GTA, possibly due to easier traffic or to a greater portion of routes being covered on highways. Conversely, depots located near Downtown Toronto (pink circles) display much higher MAPE.

2) Variations across OD distances: We now analyze the error variations against the direct distance between origin and destination. Figure 17 displays the error distribution versus the euclidean distance binned into integer kilometers. The performance is nearly the same up to 7 km, where the data concentrates, indicating that, within that interval, this feature has no direct relation to the prediction ability. Interestingly, the metrics do improve for longer distances, suggesting they are more predictable despite their low sample density (smaller training set). This may be due to the fact that longer distances often entail traveling through highways which are less susceptible to traffic and unforeseen driving circumstances.

3) Variations across temporal features: We analyze the distribution of error across temporal features, since traffic, and therefore predictability, are intrinsically time-variant phenomena. Figure 18 shows the distribution of the MAPE and MARE, calculated over the validation set, versus the hour-of-day when the parcel was sent out for delivery. A green curve
showing the distribution of samples tells that the majority of parcels are sent out in the morning, peaking around 9 a.m. The number of samples drastically decreases before 7 a.m. and after noon, which could relate to the protocols and policies of Canada Post for last-mile delivery. Finally, the metrics show smaller errors for earlier hours, indicating that deliveries sent out earlier are likely more predictable, while the model performs approximately the same from 9 a.m. to noon.

Moreover, Figure 19 presents error distributions against entire weeks, for a total of 25 weeks covered in the dataset (6 months), and against the days of the week. Both metrics have shown no relevant variation across different weeks, so the model performs roughly the same every week, which is unexpected as it would be reasonable to anticipate higher errors in winter and lower errors in spring. As for days of the week, the error shows a slight decrease for Saturdays and a big improvement for Sundays, which could be due to the fact that weekends generally have better flowing traffic.

4) Variations across targets: finally, we observe the performance for different delivery durations in Figure 20. Since this is the model target, the loss (MSE) is displayed in addition to the metrics. We show in green the number of samples per hour. Considerably low MSE is obtained for deliveries taking up to 7 hours, while for longer ones we observe that the loss increases with the duration. That could hint on the predictability of short versus long deliveries, but the MSE definition also plays a role. Based on squared absolute durations, it is directly affected by the target: while a 20% error over a 12-hour delivery gives an MSE of 5.76, the same 20% error over a 1-hour delivery returns an MSE of 0.04. As for the metrics, we observe an increase with duration, except in the first few bins, which we associate to MAPE/MARE being sensitive to errors for small targets, but also to the fact that deliveries that short might be abnormal and eventually need to be reviewed in the data. Interestingly, by grouping samples according to delivery time, the MAPE and MARE are roughly the same at each bin, which can be easily verified at the limit of making \( y_i \) constant in Equations 8 and 9.

The analysis of error with respect to both spatial and temporal dimensions helps in better understanding the model behavior, providing insightful interpretations on the predictability of parcel deliveries. Moreover, it confirms prior assumptions on the relevance of the selected features for prediction purposes.

G. Visualizing Learned Representations

As an attempt to better understand the model behavior, we present visualizations on the model representation of data across some of its features. Essentially, we project the learned representations onto a 2D space and observe how the model deals with some of the features in the data.

Since our model maps the data onto a very non-linear high-dimensional space, our approach to reducing dimensionality is to utilize principal component analysis (PCA) [52]. We move away from the direct use of PCA since solving the eigenvalue decomposition for our dataset becomes unfeasible. In order to access the data principal components, we utilize an autoencoder, a neural network that is essentially trained by an autoencoder, a neural network that is essentially trained to copy its input to its output [53]. The network encodes the input data onto a latent representation and then decodes it back to the original space. An undercomplete autoencoder, more specifically, has its latent layer with a smaller dimension than the input, being forced to discover the most informative dimensions in the data. Finally, an undercomplete autoencoder with linear activations optimized for MSE spans the same subspace as PCA [53]. Therefore, we freeze our model, so their weights are no longer updated, and replace the FC layers by an autoencoder, which reduces the dimension down to 2. The training is unsupervised, optimizing for the reconstruction loss (MSE). The autoencoder is summarized in Figure 21.

Once trained, the encoded 2D representations of the data are collected. Figure 22 shows the distribution of the 2 units at encoder output. Given their equivalence to PCA, we refer to them as principal components for clarity. The representations are shown by hour-of-day, from the 6 a.m. to 2 p.m period when most deliveries take place (see Figure 18). The distribution shifts and deforms as time goes by, showing a similar representation for 6 – 7 a.m., as well as for 10 – 12 a.m. Also, the distribution centroid is tracked on the right plot,
We present our solution as a smart city application under the IoT paradigm and discuss how a cloud-based architecture could render such a system feasible for real-world usage. With an OD-TTE formulation, we only rely on the out-for-delivery and delivered parcel scans, taking into account spatio-temporal information as well as weather data. We explore several deep CNNs, and compare them against multiple benchmarks, from classical machine learning models to OD-based works from the literature. We demonstrate that a ResNet with 8 residual convolutional blocks achieves the best results, outperforming the other methods while providing a good performance-complexity balance. Further, we seek a better understanding of the model by thoroughly analyzing the errors across different features, and by visualizing the learned deep features through dimensionality reduction, which has led to interesting remarks on data predictability. Our work delivers an end-to-end neural pipeline tailored to leverage parcel OD information as well as weather data to estimate accurate predictions.

A possible future direction would be expanding the data and pivot the approach to the drivers perspective, knowing, for example, which driver made the delivery and their delivery history. Should the order of deliveries be available, Recurrent Neural Networks (RNN) can be used for time-series modeling of parcel sequences, allowing for more frequent updates to the user about the remaining parcels in the truck. Moreover, auxiliary inputs like traffic measurements or a map of potential routes could help to enhance performance.

ACKNOWLEDGMENT

The authors would like to thank Innovapost Inc., Natural Sciences and Engineering Research Council of Canada (NSERC), and Ontario Centres of Excellence (OCE) for funding this work.

REFERENCES

[1] E. Ismagilova, L. Hughes, Y. K. Dwivedi, and K. R. Raman, “Smart cities: Advances in research An information systems perspective,” International Journal of Information Management, vol. 47, pp. 88 – 100, 2019.
[2] E. Hannon, S. Knupfer, S. Stern, and J. T. Nijssen, “The road to seamless urban mobility,” https://www.mckinsey.com/business-functions/sustainability/our-insights/the-road-to-seamless-urban-mobility January 2019, accessed on 09/10/2020.
[3] L. S. F. W. T., J. Xin, and S. J. Singh, “Consumer-driven e-commerce: A literature review, design framework, and research agenda on last-mile logistics models,” International Journal of Physical Distribution & Logistics Management, vol. 48, no. 3, p. 308332, Jan 2018.
[4] R. Gevaers, E. Van de Voorde, T. Vanelander et al., “Characteristics of innovations in last-mile logistics - using best practices, case studies and making the link with green and sustainable logistics,” Association for European Transport and contributors, 2009.
[5] M. Chen, X. Liu, J. Xia, and S. I. Chien, “A dynamic bus-arrival time prediction model based on APC data,” Computer-Aided Civil and Infrastructure Engineering, vol. 19, no. 5, pp. 364–376, 2004.
[6] A. Gal, A. Mandelbaum, F. Schnitzler, A. Senderovich, and M. Weidlich, “Traveling time prediction in scheduled transportation with journey segments,” Information Systems, vol. 64, pp. 266 – 280, 2017.
[7] Z. Wang, K. Fu, and J. Ye, “Learning to estimate the travel time,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ser. KDD 18. New York, NY, USA: Association for Computing Machinery, 2018, p. 858866.
[8] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, “When will you arrive? estimating travel time based on deep neural networks,” in AAAI 2018, January 2018.
