SPEEDY: Framework for Sharpening Promise Time Estimates in Sub-Same Day Delivery

Arindam Sarkar
Amazon
India
arindsar@amazon.com

Prakash Mandayam Comar
Amazon
India
prakasc@amazon.com

Abstract

In today's fast-paced world, customers increasingly value quick and reliable delivery services, with many prioritizing speed as a decisive factor in their purchasing decisions. E-commerce stores serve customers through specialized programs ensuring delivery within same day. Facilitated by strategically placed delivery networks, this provides an ultra-fast delivery experience to the end customers enabling them to receive their orders within the same day in a chosen fixed size window. While intra-day deliveries conclusively improve the customer experience, in the age of quick commerce, customers want the orders faster, and many scenarios (for e.g. periods of the day when customers would be traveling to school or work) make long *static* windows inconvenient for customers. However, faster deliveries increase the cost of shipping. In this work, we leverage the observation that in many instances orders reach ahead of time because of the geographical proximity of the shipping address and the order density in the neighborhood. This presents an opportunity to improve the delivery experience of customers without incurring any additional costs for customer or the seller. We present a framework to recommend *dynamic*, faster delivery time slots to customers. We create multiple heterogeneous views of order-to-delivery data capturing the spatial and spatio-temporal aspects of the data, and leverage a *novel* deep view interaction network which computes the higher order interactions among the views. The proposed model outperforms multiple representative baselines and allows us to predict narrower slots for **60%+** eligible orders for the locale under experimentation. During a 21 day online A/B test, the treatment recorded a significant gain of +17 bps in units, +21 bps for views and +19 bps increase in search interactions, establishing the efficacy of the framework.

CCS Concepts

• **Computing methodologies** → **Multi-task learning**; **Neural networks**; • **Applied computing** → **Forecasting**.

Keywords

Location business intelligence, Spatio-temporal analysis, E-commerce, Delivery Time Estimation, Forecasting, Deep Neural Networks

SIGSPATIAL '25, Minneapolis, MN, USA

ACM ISBN 979-8-4007-2086-4/2025/11
https://doi.org/10.1145/3748636.3764604

ACM Reference Format:
Arindam Sarkar and Prakash Mandayam Comar. 2025. SPEEDY: Framework for Sharpening Promise Time Estimates in Sub-Same Day Delivery. In *The 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '25), November 3–6, 2025, Minneapolis, MN, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3748636.3764604

1 Introduction

Online stores today offer ultra fast delivery experience to the customers allowing them to receive their orders by a certain time within the same day if the orders are placed by a specific time window. Faster delivery is known to improve customer satisfaction resulting in better engagement and retention [3, 23, 27]. Customers who are located in the respective servicable zones amenable to fast delivery services are provided the option to choose between the preferred delivery slot among evenly distributed slots starting at 6AM, and ending at 11PM. In this work we refer to these as 'promise' slots. The quickest slot presented to the customer is contingent on the on-ground delivery capacity for the remaining slots of the day. A common practice in these delivery systems is to pool orders that arrive within a predefined pooling window, and optimize shipments for timely dispatch and efficient route planning. Due to the inherent uncertainty in predicting which other addresses will place orders in the same window as a given customer, fixed-size promise windows are typically configured to standardize fulfillment planning and ensure operational reliability. However, in this age of quick commerce, customers want faster speeds. This is achievable by supplementing the delivery capacities to enable shorter delivery promise windows. But this reduces the delivery density (delivery/hour), and thus is detrimental to the existing cost structures associated with the shipping logistics. Another crucial problem with static slots is that these introduce an ambiguity for customers ordering in the peak slots. For e.g., a morning slot of 6-10AM, or 8AM-12PM or an evening slot of 4PM-9PM may overlap with travel to/from office/school and hence might not appeal to these set of customers. This is especially important for high value purchases/exchanges/setup, where the customer might want to be present in person. Even in the same fixed slot, some customers receive orders before others due to the way delivery executives batch orders for delivery, resulting in shorter effective delivery times for some of the customers. This motivates us to explore data driven approaches to sharpen the promise time windows presented to the customers by predicting shorter *dynamic* windows on the fly, while incurring no additional investments towards the logistics.

Predicting delivery estimates at (address, slot) level is a challenging problem as (1) demand distribution at individual address level is often sparse, (2) last mile delivery time is contingent on multiple factors like nearby addresses placing order for the same

slot, on-ground capacity deployed, traffic conditions and so on. The high sparsity of order events (in terms of the possible day, slot combinations) makes the task of predictive modelling on the basis of locality identifier alone difficult. This is complicated by the fact that the sub-same day windows are made available to customers for windows spread over upto 2 days (observed across multiple online stores) to allow for flexibility, and hence, the order data has an latency of roughly 48 hours, which implies that when the customer is searching for a product, or placing the order, there is little to no information available on the other orders which will placed in the same time window or the available delivery capacity of the station (the nodal points which are established in the vicinity of fulfillment-centers for direct delivery to the end customer). As such, our problem is different from origin-destination based shipping time estimation, since in a sub-same day delivery setting, customers usually receive shipments from a local hub within the same city; in our case local factors like neighborhood orders and station-address delivery time spread are more important, and usually there is little control over the route a delivery executive is likely to take. Due to the pooling of orders, our setting is also different from real time delivery prediction/tracking wherein real time location and capacity of delivery executives are present. Rather our goal is to opportunistically create high precision *dynamic* windows offering faster delivery to the customer.

To tackle the unique challenges for our problem, we leverage a few key observations - (1) delivery executives tend to visit nearby addresses in same sub-interval of time window, and are more likely to visit clusters which are closer to the delivery station first (delivery time is negatively correlated with the distance of the address from the station), and (2) while the temporal information at smaller locality levels is sparse, the coarser view of the data (combining multiple localities) is more amenable to predictive modelling. Based on these observations, we propose **SPEEDY**, a deep learning based framework for sharpening promise time estimates for sub-same day delivery, which leverages hierarchical multi-view features across multiple geo-spatial resolutions, and efficiently computes higher order interactions of the hierarchical views of data via deep late stage fusion. Our key contributions in this work are as follows:

(1) We create multiple views from delivery data to capture the distribution of delivery times. For instance, to capture delivery distribution in the neighborhood of an address, we create efficient multi-resolution, hierarchical grids summarizing demand and delivery time information at multiple granularities, while, to efficiently leverage the spatio-temporal nature of the order delivery data, we leverage a *deep* Convolutional LSTM [21] based neural network.
(2) To model the heterogenous views of data, we present a deep multi-task (MTL) model [19, 33] which fuses multi-view geo-spatial features via *explicit* higher order interactions [20, 29].
(3) Finally, we demonstrate the effectiveness of our proposed approach by carefully chosen quantitative/ablation studies, as well as an online A/B test spanning across 3 weeks.

2 Related Work

Our problem is closely related to that of delivery time prediction in the last mile delivery networks. However a key distinction from recent works like [6], where the authors deal with the problem of route planning and delivery time estimation for delivering multiple orders to different addresses in a quick delivery, we have multiple orders (per executive) to be delivered in a multi-hour window, and estimates need to be generated upto 48 hours in advance. In [4], authors solve the problem of parcel delivery time prediction and experiment with multiple deep Convolutional Neural Network (CNN) [14] based approaches, and find that CNNs can effectively model spatio-temporal dependencies. Compared to the broader problem of origin-destination delivery time estimation [4, 17, 34], in our case, station to customer addresses distances already fall within a serviceable area, and we rather need to model the how the delivery dynamics within a locality will affect the delivery time to a particular address in a given slot. Somewhat related is the task of traffic congestion prediction in urban setup, in which given the routes of a region, task is to predict traffic at different traffic intersections, and this heavily influence our model design wherein we leverage the spatio-temporal modelling capabilties demonstrated by CNN + LSTM based models in [30–32]. In [32] authors present a meta-learning based approach and use CNN + LSTM based architecture to model spatio-temporal patterns, in [31], authors propose a deep spatio temporal ResNet based architecture to forecast inflow/outflow of crowds, and in [30], authors propose the use of Convolutional LSTM network for predictive modelling of crashes for accident prevention.

3 Proposed Approach

In this section, we introduce the preliminaries, describe the problem formulation, and present our solution approach for predicting dynamic promise slots for sub-same day delivery.

3.1 Problem Formulation

In a typical e-commerce store, when a customer arrives at the product listing on search page or is checking out the product details on the detail page (DP), multiple services come into play to fetch information on customer's active address, and possible delivery options (promise times) to be surfaced to the customer. If the product is eligible for fast delivery at the customer's shipping address, the earliest delivery option is shown to the customer on search and DP, and all the available slots are shown when the customer is on checkout page. For a given shipment, we define the early delivery delta as the difference between promise time and the delivery time: $t_\Delta = t_{promise\_end} - t_{delivery}$. As such, our objective to estimate for the available promise time slots for a given order, how much can the slot be narrowed down to by tapering the end time of the slot. Thus the central problem here is to predict if the shipment will be delivered to the customer M minutes ahead of the *static* promise end time for multiple values of M:

$$\underset{M}{\operatorname{argmax}}\, P(t_\Delta = M | addr_c, \mathbf{C}),\ \ M \in \mathbb{Z}^+ \tag{1}$$

where $\mathbf{C}$ is the additional context like date/time of order, historical shipping data etc. For fixed slots like 6-10AM, this in effect allows offering narrower *dynamic* slots like 6-8AM, 6-9AM and so on. In this work, we pose this as a classification problem, in which for given $(addr_c, \mathbf{C})$, we estimate the probability in eq. 1 for multiple

values of M, where the target values of M are chosen in accordance to the product/UX considerations.

3.2 Preliminary Data Analysis

We performed extensive data analysis to understand the nature of the problem, and the data characteristics. We used 12 months of anonymized order data ($\mathbf{X}_{i=1}^{N}$), where ground-truth was derived based on the promise window selected at the checkout, and the delivery time was as registered by the delivery executive. One of the key findings was that at small locality levels, there is lack of repeatability of outcomes in terms of same localities receiving orders early or late. For instance, in the sample proprietary data we observed less than 10% overlap of localities with early delivery between consecutive days. This can be attributed to the fact that every day, different set of customers from different neighborhoods (small localities) place orders in an online store. As expected, repeatability improves as we increase the locality size, but the precision inevitably goes down as all addresses in a larger neighborhood will not necessarily receive the orders in the same time sub-window. Moreover we found that distance of an address from the delivery has a weak negative monotonic relationship with t_Δ. While intuitively, one may expect a strong correlation, usually delivery hubs are strategically located to guarantee timely deliveries for all addresses in the covered regions. This makes it important to effectively encode the spatio-temporal dynamics for our task.

3.3 Constructing Multiple Views of Data

Hierarchical locality based spatial views ($\mathbf{X_h}$): As we go from smaller to larger localities, the historical order data becomes less sparse (making it easier to train a predictive model), however, this also causes loss of precision, as not all addresses which fall in the same locality will get the orders delivered *equally* early or late since they will share a small subset of delivery executives. Faced with this trade-off, we propose creating hierarchical views of data to aggregate features at multiple resolutions. Specifically, we hierarchically partition the address space $\mathbf{R} = \{\text{addr}_c\}$, where shipping address c is composed of the (latitude, longitude) pairs, into uniform (square) grid based segments [32], $r_{G_k}^{m,n}$ where G_k represents a grid with fixed dimensions ($w_{G_k} \times w_{G_k}$) at level k. The conversion logic from coordinates to grid cell is outlined in Algorithm 1. We create grids of size w_{G_k} = [50m, 100m, 200m, 500m]. Intuitively, smaller grids capture the more specific properties of the address/locality, whereas larger grids enable efficient neighborhood aggregation for each address. Note that these grids have higher resolution compared to the existing pre-defined hierarchy defined by postal-code, station, fulfilment-center, city level identifiers in the addressing system. Finally, both the pre-defined and the newly introduced grid-based hierarchy is used to aggregate historical distributions of order volume, fraction of early deliveries, etc. Following the observation that delivery executives tend to traverse nearby addresses in certain precedence, for the grids, we additionally compute the average rank in which the address was served among other addresses receiving order in the same time window.

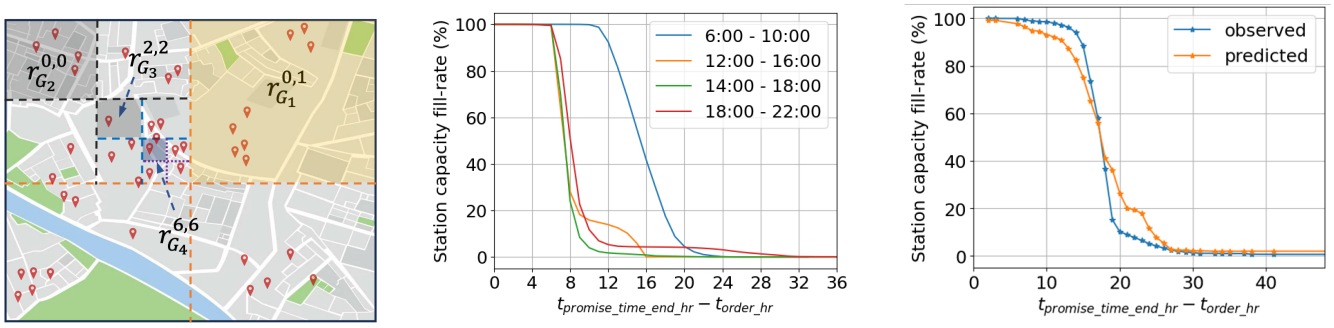
Spatio-temporal views ($\mathbf{X_{st}}$): While the spatial view captures aggregate level features at multiple resolutions, it does not capture how the distribution of t_Δ for a given locality changes over time.

Algorithm 1 Coordinate-to-Grid Mapping for Hierarchical Region Partitioning

Input: Set of delivery coordinates $\mathcal{D}_{\text{coord}} = \{(lat_i, lng_i)\}_{i=1}^{N}$
Output: Assigned grid cell $r_{G_k}^{m,n}$ for each address $c \in \mathbf{R}$
$M_{\text{lat}} \leftarrow 111{,}000$ ▷ Approximate meters per degree latitude
$lat_{\min} \leftarrow \min_{(lat,\cdot)\in\mathcal{D}_{\text{coord}}} lat$
$lng_{\min} \leftarrow \min_{(\cdot,lng)\in\mathcal{D}_{\text{coord}}} lng$
function CoordToGridCell($(lat, lng), G_k$)
  $\Delta_{\text{lat}} \leftarrow \frac{w_{G_k}}{M_{\text{lat}}}$
  $\Delta_{\text{lng}} \leftarrow \frac{w_{G_k}}{M_{\text{lat}} \cdot \cos(\text{radians}(lat_{\min}))}$
  $m \leftarrow \left\lfloor \frac{lat - lat_{\min}}{\Delta_{\text{lat}}} \right\rfloor$
  $n \leftarrow \left\lfloor \frac{lng - lng_{\min}}{\Delta_{\text{lng}}} \right\rfloor$
  return $r_{G_k}^{m,n}$
end function
for $G_k \in \mathcal{G}$ **do** ▷ $\mathcal{G}$ is the set of grid configurations at different levels
  for $(lat, lng) \in \mathcal{D}_{\text{coord}}$ **do**
    Assign $r_{G_k}^{m,n} \leftarrow$ CoordToGridCell(lat, lng, G_k)
  end for
end for

We adapt the Convolutional LSTM architecture [21] to model the spatio-temporal view of the localities, and create distinct views for each classification task corresponding to prediction of early delivery by M minutes. For brevity, we drop the task indicator in this discussion. We therefore adapt the Convolutional LSTM (ConvLSTM) architecture to obtain a sequence of latent maps $\{\mathbf{H}_t\}_{t=1}^{T}$ that capture local, short-range interactions in both space and time. Although newer visual representation backbones such as pre-trained Vision Transformers (ViT) [5] could be plugged in as well, two considerations motivate our deliberately conservative choice: (1) local inductive bias: convolutional kernels are well suited for dense fields in which neighbouring cells exhibit the strongest dependence (2) data efficiency: ViTs (and transformer models in general) are known to be data-hungry [15], and as such ConvLSTM remains a practical baseline for usecases like spatio-temporal forecasting [22]. Given that our inputs are intensity maps and not complex visual objects, the ConvLSTM's local spatial bias and proven robustness make it a pragmatic choice for learning the dynamics of spatio-temporal relationships for our task.

For a given region (e.g., grid/city) $\mathbf{r} \subseteq \mathbf{R}$, we denote the temporal snapshot at time t as $A_r^{(t)} \in \mathbb{R}^{W_r \times H_r \times 3}$. To form the snapshot, we start from the smallest sized grids ($50{\times}50m^2$ in this work), where W_r and H_r are sides of the rectangular box which bounds all the grids in the region r. We assign grid at the bottom left corner of the box as the origin for the region, and shift the other grid coordinates relative to this, thereby mapping the bottom left corner to (0, 0). For rest of the discussion, we use $\Phi_r(addr_c) = (m, n)$ to denote the operator which maps an address to a position in $\mathbf{A_r}$. For a grid cell



(a) Vis. of spatial hierarchy of localities (b) Observed capacity-fill rate over time. (c) Predicted fill-rate at different query times.

Figure 1: (a)–(c): Visualizations from SPEEDY for hierarchical locality-based view of delivery address segments, and estimated fill-rates as a function of time before promise slot end.

(m, n) in the shifted space, let the fraction of orders which were delivered before M minutes to end of promise time be $\rho^{(t)}(m, n)$, then a 3-dimensional vector is assigned to each grid forming the individual entries:

$$A^{(t)}_{r(m,n)} = \begin{cases} [0,0,0], & \text{if no order was placed} \\ [0,1,0], & \rho_r(m,n) \geq \rho^{thresh}_r \\ [0,0,1], & \rho_r(m,n) < \rho^{thresh}_r \end{cases} \quad (2)$$

Given a sequence of D_{hist} temporal snapshots $[\mathbf{A}^{(t)}_r, .., \mathbf{A}^{(t+D_{hist})}_r]$ for a region r, predictive saptio-temporal representation at a future time horizon $+D_{hz}$ is generated, forming spatio-temporal views of data. In Here we have used day-level snapshots aggregating delivery events throughout the day.

Static view ($\mathbf{X_s}$): Apart from these, multiple static features are created to represent the shipment data: date/time features, distance of the location from the station, event indicators (e.g., High Velocity Events like Prime Day), promise-slot etc. A vital factor which determines how crowded a promised delivery slot will be is the number of orders which will be placed in the slot. However this data is not available during the inference. So, we approximate the percentage capacity that is already filled at the serving station at the time of ordering (fig. 1). This is modelled by a lightweight GBDT model [2, 11], predicting the remaining-capacity as a function of (station, promise-time, request-time), where the request is triggered when the user lands on search/detail/checkout page. This is modelled as a function of (station, promise-time, request-time). Figure 1c shows the fitted curve for a station on sample data.

$$\hat{\rho}_{stn,\tau} = \mathcal{F}^{\text{GBDT}}_{\theta}(stn,\ \tau,\ \Delta t), \qquad \Delta t = \tau - t_{\text{ord}}$$

where stn denotes the station, $\tau = t_{\text{promise\_end}}$ is the end time of the promised slot, t_{ord} is the order timestamp, and $\hat{\rho}_{s,\tau}$ represents the predicted fill-rate for station s and slot τ. The mapping $\mathcal{F}^{\text{GBDT}}_{\theta}(\cdot)$ is a gradient-boosted decision-tree model parameterised by θ.

3.4 Model Architecture

Modelling Spatio-temporal View: To model the complex intra-region relationships with respect to the probability of early delivery across the temporal dimension, we employ a deep sequential neural network architecture which leverages Convolutional LSTM [21] which determines the future state of a certain point in a region by the current inputs and past states of the local neighbors. As discussed in the last section, an address $addr_c$ has multiple spatial views, however the smaller localities usually have sparse order events, and mapping of address to station/FC can also change over time leading to noisy transitions. In this work, we model the spatio-temporal dynamics at the city-level which have sufficient order events, and are relatively free of mapping changes. Different cities have varying sized snapshots (W_r, W_h). For processing the snapshots via a neural network model, we need the shapes of tensors to be identical. Resizing is a popular operation for this purpose in computer vision, but in this case, the change in aspect ratio could lead to mis-representation of distance between two addresses. Instead, first the tensors are padded to $(W_{fimg} \times W_{fimg} \times 3)$, s.t., $W_{fimg} \leq max_r(W_r, H_r)$. For our data $W_{fimg} = 1024$ was selected. After padding, the tensors are downsampled to $(512 \times 512 \times 3)$ using nearest-neighbor based interpolation for the efficiency of computation. Starting with the downsampled historical snapshots $\tilde{\mathbf{A}}_\mathbf{r} = [\tilde{\mathbf{A}}^{(t)}_\mathbf{r}, .., \tilde{\mathbf{A}}^{(t+D_{hist})}_\mathbf{r}]$, we compute the future spatio-temporal representation as:

$$\begin{aligned} \mathbf{H_r}^{(t+D_{hz})} &= \text{Conv2d}(\text{Conv2dBlock}(\ldots \\ &\qquad \text{ConvLSTM2dBlock}(\ldots(\tilde{\mathbf{A}}_\mathbf{r})))), \\ \text{Conv2dBlock}(\mathbf{Z}) &= \text{LeakyRelu}(\text{Drop}(\text{BNorm}(\text{Conv2d}(\mathbf{Z})))), \\ \text{ConvLSTM2dBlock}(\mathbf{Z}) &= \text{Drop}(\text{BNorm}(\text{ConvLSTM2d}(\mathbf{Z}))) \end{aligned} \quad (3)$$

Here Drop and BNorm represent the Dropout [26] and BatchNorm [10] operations respectively. We use a variant of the original ConvLSTM architecture as implemented in Keras [12], with the final hidden state $\mathcal{H}^{(t)}$ as output (see details in A.1). Here $*$ represents the convolution operator, $\odot$ represents the Hadamard product, and we set $D_{hist} = 7$, and $D_{hz} = 2$ to mimic the lag in data.

Higher Order View Fusion: Different views of the data capture various important aspects of the dataset. The static view captures intrinsic properties of a shipping address/order, the spatial view captures the neighborhood characteristics at multiple resolutions, and finally the spatio-temporal view captures how the *spatial dynamics* change over time. Apart from their individual importance, the views also interact with each other to affect the delivery speed. For instance, occurrence of an sales event like Prime Day (static

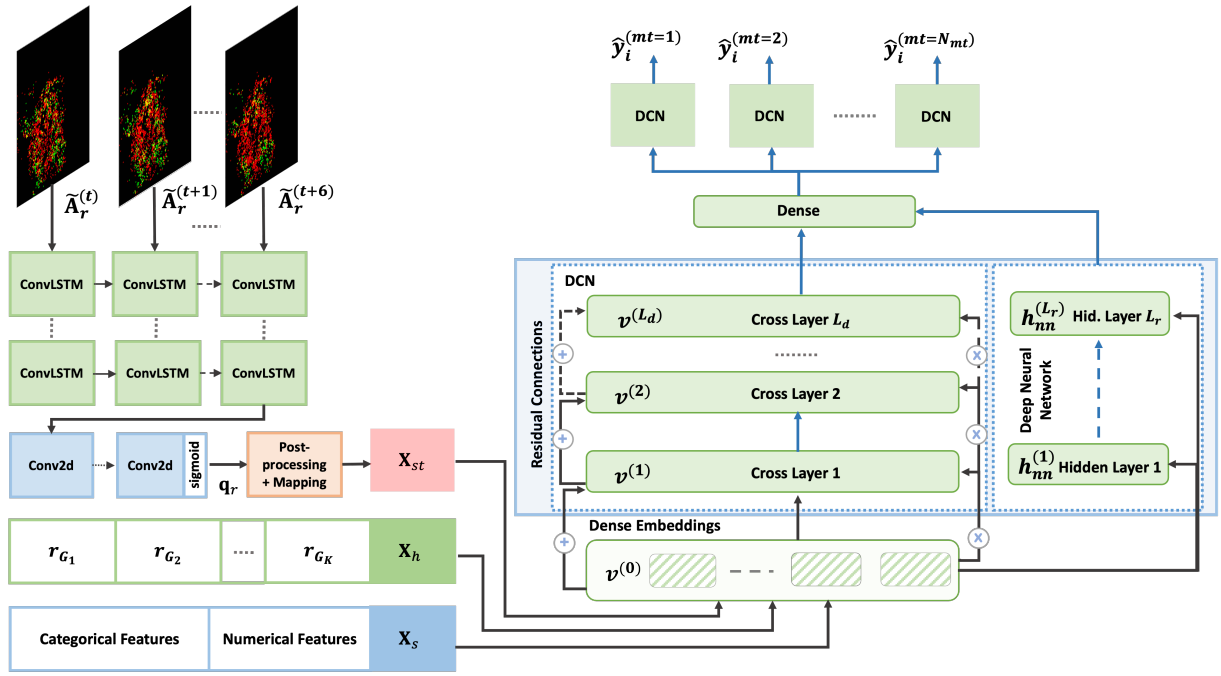


Figure 2: Architecture of the proposed framework, illustrating the spatio-temporal view modelling and the deep view fusion module which computes higher order interactions among the views.

view), Valentines Day, etc., should in principle affect the spatio-temporal view as the order volumes and correspondingly delivery capacities usually increase during these times. Similarly, if the distribution of orders which are delivered early in a smaller region changes, it will affect the corresponding distribution in the regions higher up in the hierarchy as well. To effectively combine these heterogeneous views, we leverage the idea of computing the *higher order interactions* via deep crossing networks [29]. Given the views of the data for a given address, we propose modelling the early arrival probability for a particular value of M as:

$$P(t_\Delta = M \mid addr_c, \mathbf{C}) = \text{sigmoid}\Big(f_{DVIN}(\mathbf{X_h}(addr_c)); \mathbf{X_{st}}^{(t)}(addr_c \mid t); \mathbf{X_s}(addr_c)\Big) \tag{4}$$

Here, $\mathbf{X_h}(.) \in \mathbb{R}^{N \times h_h}$, $\mathbf{X_s}(.) \in \mathbb{R}^{N \times h_s}$, and $\mathbf{X_{st}}(.) = \left[\mathcal{I}(\mathbf{q_r})_{m,n} \;\middle|\; (m,n) = \Phi_r(addr_c) \,\forall\, addr_c\right] \in \mathbb{R}^{N \times 1}$, where $\mathbf{q_r}^{(t)} = \text{sigmoid}\left(\text{Conv2d}\left(\mathbf{H_r}^{(t)}\right)\right)$. $\mathcal{I}(.)$ is the interpolation operator which is used to resize $\mathbf{q_r}^{(t)} \in \mathbb{R}^{512 \times 512 \times 3}$ to original dimension of the temporal snapshot ($W_r \times H_r \times 3$). Note that the static view consists of categorical features like the event indicator, day of week etc., which are projected to fixed size continuous embeddings via an embedding layer. f_{DVIN} represents the deep view interaction network which performs explicit higher order interactions among the various views in latent space. This is implemented via an adaptation of deep cross networks [29] and is composed of multiple cross-interaction layers. For input $\mathbf{u^{(0)}}$, representation at an intermediate layer l is given by:

$$\mathbf{u}^{(l)}(\mathbf{u}^{(0)}) = \mathbf{u}^{(0)} \odot (\mathbf{W}^{(l)}\mathbf{u}^{(l-1)} + \mathbf{b}^{(l)}) + \mathbf{u}^{(l-1)} \tag{5}$$

Each of these layers explicitly increments the order of feature-feature interaction relative to previous layer representation. We will hereby use the shorthand **DCN**(.) to represent this construct. With $\mathbf{v}^{(0)} = [\mathbf{X_h}(addr_c)); \mathbf{X_{st}}(addr_c|t); \mathbf{X_s}(addr_c)] \in \mathcal{D}$,

$$\begin{aligned} f_{DVIN}(.) &= \mathbf{DCN}([\mathbf{v}^{(L_d)}; \mathbf{h}_{nn}^{(L_r)}]), \\ \mathbf{v}^{(l)} &= \mathbf{DCN}(\mathbf{v}^{(0)}) \quad (l = 1, \ldots, L_d), \quad \text{and,} \\ \mathbf{h}_{nn}^{(l_r)} &= \text{Drop}\Big(\sigma\big(\text{BNorm}\big([\mathbf{v}^{(0)}; \mathbf{W_r}^{(l_r)}\mathbf{v}^{(l_r-1)} + \mathbf{b}^{(l_r)}]\big)\big)\Big) \end{aligned} \tag{6}$$

where L_d is the number of interaction layers, L_r is the number of layers in the resnet-style [8] neural network component with output $\mathbf{h}_{nn}^{(L_r)}$. For the choice of non-linearity σ, we found the Swish/SiLu [18] activation function led to more stable training for our model compared to the ReLU/Sigmoid.

As discussed before, for each value of M, probability of early delivery (eq. 4) needs to be modelled. For our problem, probabilities corresponding to the different values of M are related to each other, and $P(t_\Delta = M + h|.) \leq P(t_\Delta = M|.)\,\forall h \geq 0$. While we do not enforce this constraint, we employ multi-task learning for better information sharing across the tasks. All the layers except the final deep crossing layer are shared between the tasks. For each task the logits are given as:

$$f_{DVIN}^{(M)}(.) = \mathbf{DCN^{(M)}}([\mathbf{v}^{(L_d)}; \mathbf{h}_{nn}^{(L_r)}]) \tag{7}$$

Optimization: For modelling the spatio-temporal view, final layer representation $\mathbf{H_r}^{(t+D_{hz})}$ is used to predict the future temporal snapshot $\tilde{\mathbf{A}}_{\mathbf{r}}^{(t+D_{hz})}$. Network parameters are learned by minimizing the reconstruction-style loss:

$$\mathcal{L}_{st} = -\frac{1}{N}\sum_{N}\frac{1}{\kappa W_{fimg}^2}\sum_{(\tilde{m},\tilde{n},\kappa)} w_{(\tilde{m},\tilde{n},\kappa)} \cdot \Big(\tilde{\mathbf{A}}_{\mathbf{r}_{\tilde{m},\tilde{n},\kappa}} \log(\mathbf{q}_{\tilde{m},\tilde{n},\kappa}) + \left(1 - \tilde{\mathbf{A}}_{\mathbf{r}_{\tilde{m},\tilde{n},\kappa}}\right) \log(1 - \mathbf{q}_{\tilde{m},\tilde{n},\kappa}) \Big) \tag{8}$$

Here $w_{(\tilde{m},\tilde{n},\kappa)}$ is used for down-weighting loss on snapshot patches with no non-zero channel and κ represents the channel. We use the Adam optimizer [13] with weight decay based regularization. The final representations are frozen and used in the downstream deep crossing network.

The parameters of the higher order fusion model are learned by simultaneously predicting the early delivery probability for all M, and minimizing the joint loss:

$$\mathcal{L}_{mtl} = -\frac{1}{N_{mt}}\sum\frac{1}{N}\sum_{\mathbf{v}_i^{(0)} \in \mathcal{D}_{tr}} y_i^{(mt)} \log(\hat{y}_i^{(mt)}) + (1 - y_i^{(mt)}) \log(1 - \hat{y}_i^{(mt)}) \tag{9}$$

where $\hat{y}_i^{(mt)} = f_{DVIN}^{(mt)}(\mathbf{v}_i^{(0)})$ (refer Eq. 7), mt is the task identifier, N_{mt} represents the number of tasks, and $\mathcal{D}_{tr}$ represents the training dataset. For the deep view crossing network, we found that AdamW optimizer [16] with weight decay worked slightly better than the Adam optimizer and was used in our experiments. The model architecture is illustrated in figure 2.

4 Evaluation

In this section we present the empirical results of evaluation of the proposed model, and ablation studies to justify the design choices. For the task of predicting if a shipment will reach early to the customer, we compare the proposed model **SPEEDY** to the following representative baselines:

- **Gradient Boosted Decision Trees**: We use LightGBM [11], a fast and efficient GBDT implementation, and is a very powerful baseline for tabular data.
- **Deep Neural Network (DNN)**: A multilayer Feedforward Neural Network [1], with Batch-Normalization [10] and Dropout [26] is used.
- **TabTransformer**: Transformer based architecture proposed in [9] for tabular data, in which the categorical features are first transformed to embeddings, and are passed through a stack of multiple Transformer [28] encoder layers. Finally the categorical feature representation and numerical features are concatenated and passed through a shallow neural network.
- **FTTransformer**: Proposed in [7], it is a transformer based model for tabular data. The tokenizer component transforms all features to tokens and passes the tokens via a stack of transformer layers. A CLS token is prepended to the token sequence, and the embedding of the CLS token is used for prediction.

4.1 Experimental Setup

Dataset: The proprietary dataset is comprised of 19 months of anonymized checkout/delivery data of sub-same day delivery orders which consists of the promise slot chosen by the customer at checkout, the actual delivery time when the order was delivered to the customer, and various date/time related features. We discard orders which were not delivered in the first attempt, as these are late delivered due to reasons outside our control. We split the data into 2 out-of-time splits (1) Train data consisting of 18 months, (2) Test data comprises of data for following 1 month period. Validation split is created by further partitioning the train set in 80:20 split. The resulting train set has ~30M, validation set ~7.7M, and the test set has ~2.2M samples.

Metrics For our problem, we want to predict if an order can be delivered M minutes before the end of a *static* N-hour promise time window. For this binary classification problem (for each M), we report two metrics (1) AUC-ROC, which summarizes the model performance at varying thresholds, (2) Recall at 90% Precision (Recall@90P), since having a low precision w.r.t promised delivery time erodes customer trust. All the results are reported in relative terms to one of the baselines to preserve the anonymity of the internal dataset.

Model Selection We perform careful parameter tuning by varying tree depth/number of leaves/number of trees for LightGBM model. For the neural network based models, the no. of layers, hidden dimension size and number of heads (self-attention) are varied, and the model with minimum validation loss is retained. These are trained with binary cross entropy loss, with ADAM optimizer.

4.2 Discussion

Comparison with baselines: From table 1, it can be observed that the the proposed model outperforms all the baselines models. The top performing multi-task variant of **SPEEDY** gains upto **1.47%** on AUC, and **27.3%** on the recall % metric (relative) compared to the next best baseline across various tasks with access to two of the views, and including the spatio-temporal view improves the gain in performance to **2.34%** on AUC, and **60.9%** on recall. Note that even the non-multitask base SPEEDY model (with two views) gains 0.16% to 1.05% on AUC, and 17.3%-43.8% on the recall metric compared to the next best baseline, improving over both the GBDT and the specialized neural network based models for tabular data, demonstrating the effectiveness of the presented architecture. This can be attributed to the fact that by design our model *explicitly* captures both weighted sum based non-linear interactions as well as product style interactions (similar to conjunctive rules in decision tree based models), and by utilizing multiple layers of such interactions, can readily models complex relationships among the views. Apart from this, it can be noted that as hypothesized, multi-task variants of SPEEDY improve upon the single task variants as the label information results in constructive information sharing in the multi-task setup [19, 33]. Among the baselines, GBDT outperforms the vanilla neural network based models on most of the metrics. Despite the advent of complex transformer based architectures for tabular data, GBDT based models tend to outperform them or are still on par in general tabular data settings [24]. One

Table 1: Performance improvement (relative) of SPEEDY versus baselines, expressed as ± % change over the DNN baseline

		M = 60		M = 120		M = 180	
Views	**Model**	**AUC**	**Recall@90**	**AUC**	**Recall@90**	**AUC**	**Recall@90**
$[X_s; X_h]$	LightGBM	+1.767%	+6.736%	+1.102%	−1.085%	−0.351%	−1.579%
	TabTransformer	−0.483%	+0.053%	+0.318%	−9.483%	−1.8522%	+24.967%
	FTTransformer	+0.106%	+1.698%	−0.126%	–	−0.7%	–
	SPEEDY	+1.928%	+7.045%	+1.831%	+17.375%	+0.696%	+49.775%
	SPEEDY (multi-task)	**+2.377%**	**+8.409%**	**+2.242%**	**+27.364%**	**+1.116%**	**+84.743%**
$[X_s; X_h; X_{st}]$	SPEEDY	+2.925%	+10.211%	+3.091%	+43.889%	+1.3650%	+80.480%
	SPEEDY (multi-task)	**+3.780%**	**+12.730%**	**+3.370%**	**+60.993%**	**+1.985%**	**+102.692%**

Table 2: Ablation study showing the impact of various feature groups in terms of Recall @90% Precision. We use the multi-task variant of SPEEDY as the base model. Results are reported relative to X_s (w/o est. capacity).

SPEEDY Variant	M = 60	M = 120	M = 180
X_s	0.022	8.938	3.899
+ $X_{h(G_k)}$ (grid IDs)	0.270	24.411	3.914
+ $X_{h(edb)}$ (early del.)	0.415	37.523	6.904
+ $X_{h(vol)}$ (vol distr.)	0.424	39.111	5.238
+ $X_{h(del\_rank)}$	0.443	40.769	7.563
+ X_{st} (spatio-temporal)	**0.480**	**49.211**	**7.734**

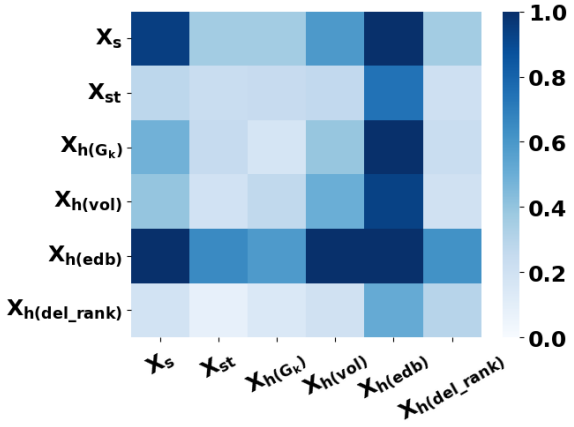


Figure 3: Visualization of the learned feature-group interaction matrix.

trend we noticed was that neither TabTransformer or FTTransformer conclusively outperform the 5-layer DNN model. In fact, for FTTransformer, for M = [120, 180], recall @90% precision was close to 0 (and hence indicated with '-' in the table). One reason could be that these models are over-parameterized w.r.t our dataset, hence increasing the generalization gap. In SPEEDY, we leverage various task specific structures and inductive biases - for instance the related-ness of the views, tasks and by explicitly modelling the higher order interactions between the views, improving upon all the baselines.

Analysis of feature groups: To understand the feature interaction induced by the view crossing layers, we plot the weights of cross-layer (eq. 5). Since the input to the cross layer is embedding of actual input features, we first group the input embeddings corresponding to 6 feature groups: static features, spatial features (grid IDs, aggregated early delivery distribution, agg. volume distribution, and avg. delivery rank), and the spatio-temporal features. Block matrices are constructed between the feature groups, and Frobenius norm of each block is computed to get the interaction strength between each pair (fig. 3). As expected the view crossing layers effectively learn strong interactions between the feature groups. Further we perform an ablation study to understand the incremental contribution of each view/feature group to the final model (table 2). It is evident that incrementally adding multiple views of the data improves the efficacy of the base model.

Customer Impact: In an offline back-testing, we observe that using the best performing variant of SPEEDY, we are able to significantly reduce the promise time by **75%** for ~1.01% of orders, **50%** for ~6.51%, and by **25%** for ~60.44% of the orders, operating at 90% precision. To test the effect of displaying narrower promise times on customer engagement, we performed an online A/B test where the Control was shown the original fixed promise time windows, while the Treatment was exposed to dynamic promises. In order to meet a strict sub 5 ms online latency budget, we pre-materialised predictions at grid cell level and persisted them in key-value datastore (e.g., DynamoDB) [25]. At runtime, each incoming (latitude, longitude) coordinate associated with customer address is mapped to its grid index, enabling a constant-time key lookup that delivers the pre-computed score to downstream services. Freshness of this caching layer is maintained by re-computing the scores every 24 hours. Over the 21-day online A/B experiment, the treatment variant delivered significant uplift: total units increased by 17 bps, page views by 21 bps, and search interactions by 19 bps, demonstrating the effectiveness of the framework.

Training/Inference Infrastructure: Data processing jobs were run on a distributed cluster of 15-30 nodes with 32 CPUs in each node. All the models were trained/tested on a single NVIDIA V100 GPU node with 512 GB RAM.

5 Conclusion & Future Work

In this work, we present SPEEDY, a framework for recommending *dynamic* promise time estimates for sub-same day eligible orders, enabling us to replace the current *static* promise windows on multiple interaction points like the search, detail and the checkout page. We construct multiple views of data, capturing static, spatial and spatio-temporal characteristics of order delivery data, and propose a *novel* deep view interaction network which models the probability of early delivery by M minutes by computing higher order interactions among the views. By exploiting the problem specific structure, our model is able to outperform representative baselines like - Gradient Boosted Decision Trees, and Neural Network based models, and reduces the promise time estimate by atleast 25% for 60%+ of same day delivery eligible orders and results in significant lift in various customer engagement metrics as well. As part of future improvements, we plan to experiment with more specialized architectures to improve the spatio-temporal modelling aspects of the data by leveraging models like dynamic spatio-temporal graph neural networks which can model the joint dependencies of the spatial grids over time more naturally.

References

[1] G. Bebis and M. Georgiopoulos. 1994. Feed-forward neural networks. *IEEE Potentials* 13, 4 (1994), 27–31. doi:10.1109/45.329294

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. doi:10.1145/2939672.2939785

[3] Ruomeng Cui, Zhikun Lu, Tianshu Sun, and Joseph Golden. 2020. Sooner or Later? Promising Delivery Speed in Online Retail. *SSRN Electronic Journal* (March 2020). doi:10.2139/ssrn.3563404 Accessed: 2025-06-05.

[4] Arthur Cruz de Araujo and Ali Etemad. 2021. End-to-End Prediction of Parcel Delivery Time With Deep Learning for Smart-City Applications. *IEEE Internet of Things Journal* 8, 23 (2021), 17043–17056. doi:10.1109/JIOT.2021.3077007

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

[6] Chengliang Gao, Fan Zhang, Guanqun Wu, Qiwan Hu, Qiang Ru, Jinghua Hao, Renqing He, and Zhizhao Sun. 2021. A Deep Learning Method for Route and Time Prediction in Food Delivery Service. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2879–2889. doi:10.1145/3447548.3467068

[7] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems* 34 (2021), 18932–18943.

[8] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 770–778. https://api.semanticscholar.org/CorpusID:206594692

[9] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678* (2020).

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37* (Lille, France) *(ICML'15)*. JMLR.org, 448–456.

[11] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 3149–3157.

[12] Keras. 2024. ConvLSTM2D layer. https://keras.io/api/layers/recurrent_layers/conv_lstm2d/. Accessed: 2024-04-29.

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60 (2012), 84 – 90. https://api.semanticscholar.org/CorpusID:195908774

[15] Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco De Nadai. 2021. Efficient Training of Visual Transformers with Small Datasets. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[16] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bkg6RiCqY7

[17] Xiaowei Mao, Huaiyu Wan, Haomin Wen, Fan Wu, Jianbin Zheng, Yuting Qiang, Shengnan Guo, Lixia Wu, Haoyuan Hu, and Youfang Lin. 2024. GMDNet: A Graph-Based Mixture Density Network for Estimating Packages' Multimodal Travel Time Distribution. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 4 (Sep. 2024), 4561–4568. doi:10.1609/aaai.v37i4.25578

[18] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2018. Searching for Activation Functions. *ArXiv* abs/1710.05941 (2018). https://api.semanticscholar.org/CorpusID:10919244

[19] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *ArXiv* abs/1706.05098 (2017). https://api.semanticscholar.org/CorpusID:10175374

[20] Arindam Sarkar, Dipankar Das, Vivek Sembium, and Prakash Mandayam Comar. 2022. Dual Attentional Higher Order Factorization Machines. *Proceedings of the 16th ACM Conference on Recommender Systems* (2022). https://api.semanticscholar.org/CorpusID:252216566

[21] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf

[22] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM Network: a machine learning approach for precipitation nowcasting. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 802–810.

[23] Allison Shirreffs. 2020. Optimizing the Delivery Speed Promise Can Boost Sales. *Emory Business* (October 2020). https://www.emorybusiness.com/2020/10/09/optimizing-the-delivery-speed-promise-can-boost-sales/ Accessed: 2025-06-05.

[24] Ravid Shwartz-Ziv and Amitai Armon. 2021. Tabular Data: Deep Learning is Not All You Need. In *8th ICML Workshop on Automated Machine Learning (AutoML)*. https://openreview.net/forum?id=vdgtepS1pV

[25] Swaminathan Sivasubramanian. 2012. Amazon dynamoDB: a seamlessly scalable non-relational database service. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, Arizona, USA) *(SIGMOD '12)*. Association for Computing Machinery, New York, NY, USA, 729–730. doi:10.1145/2213836.2213945

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[27] SupplyChainBrain. 2024. Pricey, But Worth It – Same-Day Delivery. *SupplyChainBrain* (2024). https://www.supplychainbrain.com/articles/39311-pricey-but-worth-it-same-day-delivery Accessed: 2025-06-05.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[29] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1785–1797. doi:10.1145/3442381.3450078

[30] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-ConvLSTM: A Deep Learning Approach to Traffic Accident Prediction on Heterogeneous Spatio-Temporal Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 984–992. doi:10.1145/3219819.3219922

[31] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). doi:10.1609/aaai.v31i1.10735

[32] Yingxue Zhang, Yanhua Li, Xun Zhou, Jun Luo, and Zhi-Li Zhang. 2022. Urban Traffic Dynamics Prediction—A Continuous Spatial-temporal Meta-learning Approach. 13, 2, Article 23 (jan 2022), 19 pages. doi:10.1145/3474837

[33] Yu Zhang and Qiang Yang. 2017. A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering* 34 (2017), 5586–5609. https://api.semanticscholar.org/CorpusID:11311635

[34] Xin Zhou, Jinglong Wang, Yong Liu, Xingyu Wu, Zhiqi Shen, and Cyril Leung. 2023. Inductive Graph Transformer for Delivery Time Estimation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining* (Singapore, Singapore) *(WSDM '23)*. Association for Computing Machinery, New York, NY, USA, 679–687. doi:10.1145/3539597.3570409

Appendix

A.1 ConvLSTM Implementation Details:

ConvLSTM architecture as implemented in Keras [12], with the final hidden state $\mathcal{H}^{(t)}$ as the output:

$$
\begin{aligned}
\mathcal{H}^{(t)} &= o^{(t)} \odot \tanh(C^{(t)}), \\
\text{where} \quad o^{(t)} &= \sigma(W_{zo} * Z^{(t)} + W_{ho} * \mathcal{H}^{(t-1)} + b_o), \\
C^{(t)} &= f^{(t)} \odot C^{(t-1)} \\
&\quad + i^{(t)} \odot \tanh(W_{zc} * Z^{(t)} + W_{hc} * \mathcal{H}^{(t-1)} + b_c), \\
i^{(t)} &= \sigma(W_{zi} * Z^{(t)} + W_{hi} * \mathcal{H}^{(t-1)} + b_i), \\
f^{(t)} &= \sigma(W_{zf} * Z^{(t)} + W_{hf} * \mathcal{H}^{(t-1)} + b_f)
\end{aligned} \tag{10}
$$