GCF: Estimating Unobserved Demand Using Graph Causal Forecasting

Sayantan Basu, Mohit Kumar, Sivaramakrishnan Kaveri

Amazon
sayantba@amazon.com , mohitkrg@amazon.com, kavers@amazon.com

Abstract

Time series data, prevalent in fields like medical, e-commerce, finance, etc., is used for forecasting, such as predicting next quarter's product demand based on past trends. However, some problems necessitate causal models to answer questions like "What the product demand would have been without a specific intervention (e.g., products with slow delivery time suppressed from search results)?" Such questions require causal models to estimate unobserved counterfactual outcome. In this paper, we propose a novel Graph Causal Forecasting (GCF) model, that predicts the unobserved demand leveraging the relationship of a product with other similar products in the marketplace (spatial aspect), along with change in demand over time for each product (temporal aspect). The core idea is to estimate the counterfactual outcome using a synthetic control unaffected by the treatment. Our approach uses RGCN-dilated CNN based network, which leverages domain knowledge to automatically design a synthetic control during training. Using GCF for our demand forecasting problem, we achieve 75.3% lower MAPE compared to baseline. We use the forecasted values to recommend high demand products, in terms of our business metric (discussed later) which tracks the quality of these recommendations, we achieve a significant jump of 61.2%. Moreover, it adds 67.8% more high demand products to the marketplace, compared to existing model in production. Deployment of GCF in 2023, led to $+1399$ bps improvement in number of products with a view from customers, and $+310$ bps improvement in number of products with a sale. We also compare GCF with state of the art forecasting methods on a semi-synthetic data, created by simulating a treatment on open source traffic data METR-LA. We achieve 30% lower MSE against TGCN, a time series forecasting approach and 30% lower MSE against CRN and 25% lower MSE against Google Causal Impact model, both of which are causal forecasting approaches.

Introduction

Reducing product delivery time is crucial for improving customer and seller experience while lowering carbon footprint in e-commerce. A potential solution is to prioritize local product availability in search results, which can be considered a treatment. However, promoting local products while suppressing distant ones leads to reduced demand for non-local products (blue line, Fig. 1), necessitating demand forecasting to guide inventory stocking decisions for sellers. However, using a standard time-series model on observed reduced demand to identify high demand products results in a significant drop in recommendation precision (10%), recall (6%), and 58% increase in MAPE. Thus missing out on a lot of products which will have high demand if made available locally. It also negatively impacts the customer engagement metrics: Product with a glance view (PWAGV) and Product with a sale (PWAS), which measure customer views and sales for the products in the recommendation once they are brought by the sellers. We saw a significant drop of 21% in PWAGV and 20% in PWAS. Ideally we want to predict sales of products as if the treatment was not given, which is called counterfactual demand (green line in Fig 1). In our example, this means estimating sales as if far away products were not suppressed in search. Such counterfactual estimation allows us to correctly identify all products which will have high demand if made available locally and thus should be recommended to local sellers.

Identifying high-demand products typically involves building demand forecasting models. These models use historical sales data to predict future sales trends. Demand forecasting using time series data is a well-established research area, with statistical methods like autoregressive integrated moving average (ARIMA) and exponential smoothing being commonly used in the past ((Lee and Tong 2011); (Gardner Jr 1985)). However, these techniques are limited in their ability to capture long-term dependencies, as they mostly focus on recent history. In recent times, deep learning models have gained popularity in time series forecasting due to the increasing availability of data. Convolution-based models that exploit spatio-temporal information, such as those proposed by (Wang, Yan, and Oates 2017), (Ismail Fawaz et al. 2019), and (Kashiparekh et al. 2019), have shown promise in various time series tasks. Dilated CNNs(DCNNs) ((Yu and Koltun 2015)) are an extension of CNNs that can capture long-range dependencies without increasing the kernel parameters. Sequence models like Recurrent Neural Networks (RNNs) and their variants, including Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), have become popular due to their ability to effectively capture long sequential information ((Hochreiter and Schmidhuber 1997); (Cho et al. 2014); (Lim and Zohren

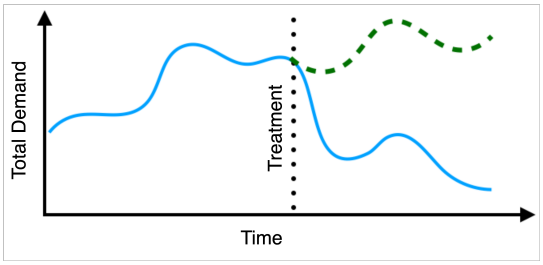

Figure 1: We show demand of a product over time. The blue line indicates the observed demand, which sees a sudden drop after treatment. The green line shows the unobserved counterfactual demand, which represents the demand if the treatment was not given. Our aim is to estimate the green time-series and not the observed blue time-series.

2021)). Attention-based models like Autoformers ((Xu et al. 2021)) and ETFormers ((Woo et al. 2022)) have been proposed to extract trend information from time series data using modified attention blocks. However, these models typically assume no or low sparsity in the data, which may not be true in real-world scenarios. Graph based approaches like Temporal graph convolutional network (TGCN) (Zhao et al. 2020) models seek to capture both spatial information through graph convolutional networks (GCNs) (Kipf and Welling 2017) and temporal dependence via gated recurrent units (GRUs) (Chung et al. 2014). While TGCN can encode these dimensions, it does not account for potential multiple relationships that may exist within the multi-relational graphs characterizing such problems.

Over the past few years, there has been an effort to apply causal models to time series data for the purpose of identifying treatment effects. An important work by (Athey and Imbens 2017), introduced the concept of difference-in-differences (DiD), where the change of the outcome of a control group is used as a counterfactual for the treatment group in the absence of the treatment. However, this method is built on the parallel trends assumption given by (Kausel 2015), which requires that in the absence of treatment, the difference between the treatment and control group is constant over time. As expected, this assumption does not hold for many real world problems. To overcome this problem, (Brodersen et al. 2015a) proposed an approach called Causal Impact, which learns the relationship between the treatment and control before any intervention and predicts the counterfactual series after the treatment. Certain deep learning models, such as counterfactual recurrent networks(CRN) (Bica et al. 2020), aim to adversarially learn balanced representations before estimating counterfactuals. CRN seeks to learn representations that are unaffected by treatment in order to perform predictions utilizing these representations. However, this approach assumes availability of historically intervened data for training, which does not apply to our scenario. In this paper we introduce GCF, a graph causal forecasting approach, which can use both the temporal and static features to first create the synthetic control, and then perform the counterfactual estimation without the presence of historically intervened data. Our key contributions are:

- We propose a dilated CNN based demand forecasting model that uses our novel Boundary-Alpha Positive Penalization(B-APP) loss to handle imbalance in the time series data used in training.
- We use domain knowledge represented as graphs, to create synthetic control, and empirically show it's superiority over other ways of creating control.
- We propose a multi-relational spatio-temporal RGCN-dilated CNN based graph causal forecasting technique, that estimates the unobserved demand from far away warehouses (out-of-region) given the local demand (in-region) forecasts.

Application Description

In this section, we start by introducing the notations and formally defining the problem, following which we discuss our approach based on causal model.

Notation

A univariate time series at a time step t is represented as $x(t)$. While $X(t) \in \mathbb{R}^n$ represent a multi-variate time series, where $\mathbb{R}$ represents the set of real numbers and n is the number of variables in the multivariate series. We represent a knowledge graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \in \mathbb{R}^{v \times p}$ is the set of nodes and $\mathcal{E} \in \mathbb{R}^{k \times d}$ is the edge set. Here $v = |\mathcal{V}|$ is the number of nodes in the graph, each having p dimensional embedding, and $k = |\mathcal{E}|$ is the number of different types of edges represented using d dimensional embedding. Let w be the look back window of the past and h be the horizon of the future. Then, an input time series $X(t-w:t)$ and future time series $X(t:t+h)$ is defined as:

$$x(t-w:t) = (x(t-w), x(t-w+1), \ldots, x(t-1))$$
$$x(t:t+h) = (x(t), x(t+1), \ldots, x(t+h-1))$$

For multivariate time-series we will use $X(t_1 : t_2)$. Let A be a binary treatment that can take values a = 0 (untreated) or a = 1 (treated). We will use $x_a(t)$ to represent the potential outcome for the treatment group (when a = 1) or control group (when a = 0). The counterfactual outcome is represented as $x^c_{a=0}(t > t_a)$, where the treatment is given at $t = t_a$. This tells us what would have happened in the treatment group had we not given the treatment.

Problem Statement

Let $y(t)$ represent the variable of interest that we want to estimate, for example demand of a product, and let $X(t)$ be a multi-variate time series representing different features of the product, for example the price, number of views on the product page, etc.

We will use A to represent the treatment that suppresses distant products from search results, with the goal of reducing logistic costs. However, this suppression of search results also leads to lower observed demand for these products. Using this reduced demand in forecasting models will result in underfitting, which means the models will not accurately predict the true high demand that would be seen once these

products are made more widely available. More formally, after receiving the treatment, we observe the impacted value, i.e., $y_{a=1}(t_a : t_a + h)$, using these values our aim is to estimate the true unobserved future demand had the product not received the treatment, i.e., $y^c_{a=0}(t_a + h : t_a + h + h')$, which is a counterfactual outcome. For simplicity, we will write $h + h'$ as h' going forward. Since, counterfactual outcomes are never observed, we estimate it by extending the ideas of causal forecasting. To be more specific, for each product i, we identify a set of products (synthetic control group) P_i, which did not receive the treatment and can help in predicting demand for i, and use it's observed demand $(y_p(t_a - w : t_a + h))$ and other features $(X_p(t_a - w : t_a + h))$ along with the observed demand $(y_i(t_a - w : t_a + h))$ and other features $(X_i(t_a - w : t_a + h))$ of product i to estimate the unobserved counterfactual demand of product i. Thus, the aim is to learn a function f such that:

$$y^c_{a=0}(t_a + h : t_a + h') = f(y(t_a - w : t_a + h), X(t_a - w : t_a + h))$$

Here $X = (X_p, X_i)$ represents the features and $y = (y_p, y_i)$ is the observed demand for products in $\{P, i\}$. In the next section we explain our demand forecasting approach with A applied to the marketplace. The ideas discussed are not restricted to this specific use-case. Later we show results on benchmark METR-LA dataset.

Use of AI Technology

For any product, the demand can be split in two categories: first, the demand coming from nearby locations, which do not get impacted when treatment A is applied and second, demand coming from far away locations, which will get suppressed due to A. Keeping this in mind, to correctly estimate counterfactual demand, $y^c_{a=0}(t_a + h : t_a + h')$, for a given product, we break the prediction problem into two parts – 1) forecasting the unaffected local demand (Factual Forecasting), and 2) forecasting the affected out-of-region demand (Counterfactual Forecasting). We then sum both the values to estimate the total demand. In detail. the approach works as follows, upon receiving past 8 quarters of data:

1. Forecasting local demand is done using CNN which is a standard time-series model, it is represented as the Factual Forecasting (FF) Model in Figure 2. For each product, this module leverages it's local demand and other features that are not affected by A to learn a function f_{FF} that forecasts local demand. Notably, the FF model can be retrained with the latest data even after applying treatment A, since the features used in training are not impacted by A.

$$y^{FF}_i(t_a + h : t_a + h') = f_{FF}(X_i(t_a - w : t_a + h))$$

2. Forecasting unobserved, out-of-region demand is complex, as this information is not directly available and cannot be easily predicted using traditional time-series models. A key challenge is to identify the appropriate set of products P that can effectively capture the unobserved counterfactual demand. This requires careful selection and modeling to ensure the synthetic control group P is a good representation of the true underlying demand dynamics. In order to learn an optimal P, we create a heterogenous knowledge graph of products. Using a heterogeneous graph to represent synthetic control has multiple benefits: first, it allows to represent domain knowledge in a concise manner, where various types of edges can be defined to represent different relations. For instance, in this case, we can use it to represent products viewed together during a session. Second, it also provides the flexibility to encode it using Relational Graph Convolution Networks (RGCN) layer, as shown in Fig 2, this allows for an end to end model that can choose a part of the knowledge graph depending on the problem we are trying to solve. Note, in causal literature, for the synthetic control group P, the selection rules should consider all the hidden confounders. Feeding all the important confounders (or atleast the relevant ones) while modelling, is a challenging task in causal, and missing out on vital confounders leads to incorrect estimates. Thus, we use RGCN which learns the selected group of local co-viewed products while predicting counterfactual demand for a faraway product. Formally, P can be stated as follows:

$$P = \arg\min_{\hat{\mathcal{E}} \subset \mathcal{E}} \Big(\sum_{i \in \mathcal{V}, k \in \mathcal{N}_i} ||y^{CF}_i - \hat{y}^{CF}_i(\hat{\mathcal{E}}_k)|| \Big) \quad (1)$$

where $\mathcal{N}_i$ denotes the neighbors of product i, $\hat{\mathcal{E}}_k$ denotes the k neighboring edges picked, and $\|.\|$ is the squared error. To form edges in the graph, we choose different strategies of similarity notions (from domain knowledge) like – similar price, normalised similar demand (considering only the movement), similar product titles, and co-viewed products. We experiment each edge creation strategy using our downstream forecasting task, and empirically show in Table 2 that the graph constructed using co-viewed products outperform other notions of similarity for our forecasting task. The final graph consists of two types of edges - co-viewed products, and same product (belonging to different region) edges. Thus, the unobserved demand for a product is based on the local demand of similar products (picked from the graph) in that time-window (eg:quarter), and how these products have evolved over time (quarter-on-quarter). To capture this spatio-temporal aspect in the data, we leverage a RGCN-dilated CNN based architecture. As shown in Figure 2, the RGCN model leverages the spatial demand information from similar products via edges in the graph, for all past quarters. The dilated CNN convolves over these RGCN generated product embeddings quarter-on-quarter to capture the temporal essence. Finally, we use fully connected layers to obtain the estimates. Mathematically, the RGCN-dilated CNN model is expressed as:

$$H^{(l)} = f_{RGCN}(H^{(l-1)}) = ReLU(Conv1D_{d_l, k_l, d_{l-1}}(H^{(l-1)}) + b_l)$$
$$h^{(l)}_t = h^{(l-1)}_t + H^{(l)}_t \quad (2)$$

where X is the input time series, $H^{(l)}$ is the output of the l-th layer, d_l is the dilation factor of the l-th layer, k_l

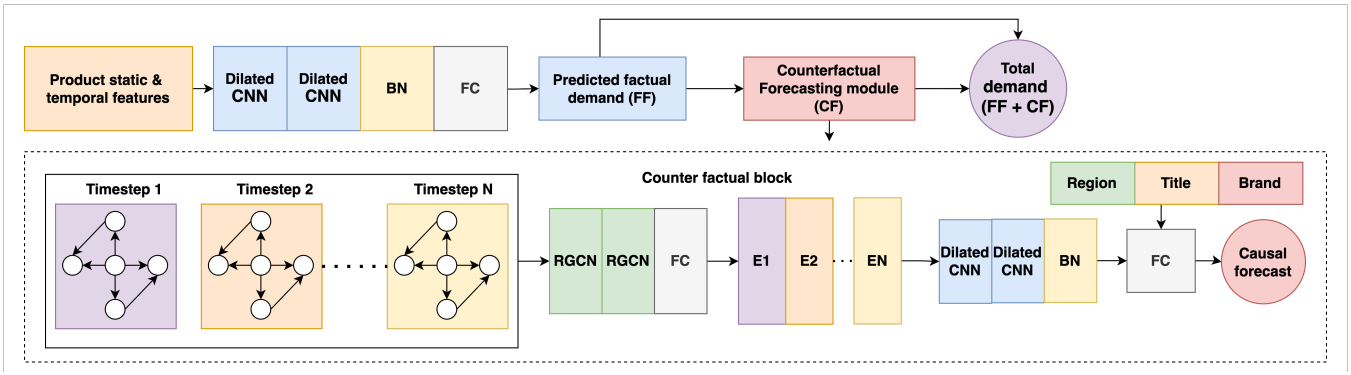

Figure 2: Graph Causal Forecasting end-to-end architecture. We use two models – Factual Forecasting (FF) for predicting the non-impacted time series, and Counterfactual Forecasting model (CF) for estimating the affected time-series.

is the kernel size of the l-th layer, b_l is the bias term of the l-th layer, and $h_t^{(l)}$ is the output of the l-th layer at time step t. f_{RGCN} is the function that applies a stack of dilated CNNs to the output of the RGCN. Finally, the local forecasts from the previous step form the base for our counterfactual forecasts. We leverage the local forecasts, knowledge graph, historical past 8 quarters data, and other static features to generate our counterfactual forecasts.

$$y_i^{CF}(t_a + h : t_a + h') = f_{CF}(X(t_a - w : t_a + h), \mathcal{G}, y_i^{FF}(t_a + h : t_a + h'))$$

3. Finally, we add the local forecasts with the counterfactual forecasts to generate our final recommendations. The products having a predicted demand higher than the business defined threshold are surfaced to the sellers as recommendations.

$$y^{TOT} = y^{FF} + y^{CF}$$

Tackling data imbalance using B-APP loss: We notice imbalance in demand data which is common in e-commerce marketplaces. The high demand products constitutes only top 5% of the total products. This imbalance in the data drives the model to perform better on the majority low demand products, which is unwanted. To handle this, we designed a novel Boundary-Alpha Positive Penalization (B-APP) loss. B-APP gives heavy penalization on samples which are predicted greater (or lesser) than our defined threshold($\hat{T}$), and originally are lesser (or greater) than $\hat{T}$. We define such samples as incorrect boundary samples. During training B-APP separates out the correctly predicted samples from the incorrect ones using a direction flag α. Formally, let y be the original values, $\hat{y}$ be the predicted values, then α is defined as:

$$\alpha = \frac{1}{2}\left|\frac{y - \hat{T} + \epsilon}{|y - \hat{T} + \epsilon|} + \frac{\hat{y} - \hat{T} + \epsilon}{|\hat{y} - \hat{T} + \epsilon|}\right| \quad (3)$$

where, $|\cdot|$ is the absolute function and ϵ is a small constant to avoid division by zero error. Note that $\alpha = 1$, when both the true and predicted forecasts are greater (or lesser) than $\hat{T}$, and 0 otherwise, thus, separating the incorrect boundary samples from rest. The final formulation of B-APP loss is as follows, where $D \geq 1$ is a hyper parameter that penalizes the samples where $|\alpha| = 0$,

$$L = (D \times (1 - |\alpha|) + |\alpha|) \times MSE(y, \hat{y}) \quad (4)$$

In eq 4, $\alpha = 1$ implies $L = MSE$, but $\alpha = 0$ implies $L = D \times MSE$, which means higher penalization for incorrect samples. The degree of penalisation can be controlled using hyperparameter D.

Model	Precision	Recall	F-score	MAPE
Catboost	-1.3%	+10.6%	+5.6%	+31.9%
LSTM	+5.2%	+16.6%	+9.8%	-11.7%
GRU	+2.6%	+21.2%	+11.2%	-25.9%
CNN	+6.5%	+15.1%	+9.1%	-21.1%
DC(MSE)	+5.2%	+16.3%	+9.8%	**-30.3%**
DC(MAE)	+6.5%	+21.4%	+14.1%	-9.7%
DC(BAPP)	**+10.5%**	**+25.7%**	**+16.9%**	-23.8%

Table 1: Factual Forecasting model results (w.r.t baseline-1). DC represents Dilated-CNN.

Application Development and Deployment

Dataset Description

Internal data: Our dataset contains features such as views, demand and price for each product over the previous eight quarters. Additionally, the dataset includes static features like product title and brand. The product graph constructed from this data has a few million nodes and the number of edges (based on co-view) is approximately three times the number of nodes.

Open source data: We also experiment with open source data METR-LA (Li et al. 2017) which poses a traffic forecasting problem. Here daytime traffic speed is used to predict speed at night in Los Angeles. We introduce a treatment where speed data at night is not available after first 2 months, due to technical issues in the sensors. Simulating this treatment results in a behaviour similar to our use

case. We experimented with three state of the art models – Counterfactual-recurrent-network(CRN) (Bica et al. 2020), TGCN (Zhao et al. 2020) and Google causal impact (Brodersen et al. 2015b).

Experimental Setup

On internal data, we implement multiple approaches to compare with our algorithm. We report the performance numbers compared to two baselines for both FF and CF model. **Baselines:** In FF model baseline-1, we use the previous quarter's local demand as the next quarter's local demand. In CF model baseline-2, we use the average internal products to estimate out-of-region (OOR) demand. This approach considers the mean demand of co-viewed and unimpacted local products as the prediction. The results in the next section are based on these baselines.

Comparing CF with other approaches: Apart from the baseline, we compare with various approaches for counterfactual estimation: 1) The Average Growth Factor approach tries to capture the average growth (G_{OD}) in OOR demand by modelling $G_{OD} = TD/ID$, where TD is the total demand and ID is local demand. We calculate the mean G_{OD} for past 4 quarters for all products and multiply it to the predicted local demand, thus, obtaining the predicted total demand. 2) For DiD approach, we create a synthetic control group by heuristically pairing non-internal products with internal products belonging to the same category. Taking inspiration from (Athey and Imbens 2017), we assume the average difference (here faraway demand) in the time series of the control group and treated group remains stable over time. We calculate the average difference between these two groups and add the difference to our predicted local demand (in the next quarter) to estimate total demand. 3)For the propensity approach we create a propensity model for all products which are likely to receive the treatment A. For every faraway product we form a control group consisting of local products with similar propensity scores. Finally, we take the mean of the control group to get the predicted demand for the faraway product. 4)To understand the importance of the local demand model, we perform an ablation study by providing zero local demand for the current quarter in our CF model. We observe a drop in performance when local demand forecasts are not used.

Edge creation: To evaluate the significance of co-viewed edges in our product graph for the CF model, we compare the performance of co-viewed edges against other popular similarity measures. We add an edge between two products if their similarity, as computed using cosine similarity, exceeds a certain threshold for a specific similarity notion, such as product titles. We consider various other similarity notions as well, such as price profile (product price over the past 8 quarters) and demand profile, as outlined in Table 2. Our results show that the co-view-based edges substantially outperform the other edge similarity measures in our case.

Evaluation metrics: For our demand forecasting setup, we assign class 1 to a sample if its predicted value exceeds the $\hat{T}$, otherwise, we assign class 0. We tag the ground truth using the same method. Using these modified predictions, we track the precision to classify high demand ones, and track MAPE, to ensure the predictions are closer to the ground truth. For traffic forecasting, we use MSE and MAPE as our metrics.

Model	Precision	Recall	F-score	MAPE
Average growth factor	+9.4%	-8.1%	-1.5%	-33.9%
DiD	-19.4%	+27.4%	+11.3%	-74.4%
Propensity	-59.7%	+32.3%	-27.4%	+15.4%
Catboost	+2.5%	+19.3%	+10.1%	+44.6%
LSTM	+0.5%	+32.2%	+14.7%	-56.4%
GRU	+3.8%	+27.4%	+14.3%	-61.4%
CNN	+2.7%	+32.2%	+15.9%	-66.4%
DCNN	+5.1%	+29.1%	+16.1%	-63.2%
CRN	+4.7%	+27.2%	+15.2%	-41.3%
No internal-demand	+5.7%	+27.4%	+15.4	-70.1%
Price profile	+6.4%	+29.8%	+17.3%	-73.9%
Demand profile	+7.7%	+29.6	+17.8%	-74.3%
Product title	+6.4%	+20.9%	+13.0%	-66.4%
Internal co-view	-0.4%	**+33.8%**	15.8%	-72.6%
Co-view	**+10.3%**	+32.2%	**+20.2%**	**-75.3%**

Table 2: Counterfactual Forecasting model results (w.r.t baseline-2). Last six rows represent RGCN-DCNN with different edge creation strategies.

Application Use and Payoff

In Table 1, we compare performance of factual model predictions with SOTA techniques and other baselines, and see dilated CNN based approach outperforms all methods. We also conduct an ablation study to empirically show dominance of B-APP loss over other popular loss functions used in forecasting tasks (MSE, MAE). In Table 2, for counterfactual model, we show how RGCNs and CNNs combined together can extract spatio-temporal relations in the underlying data, giving best performance. We conduct an ablation study and empirically show co-view data outperforming other notions of similarity, for learning our synthetic control group when combined with downstream forecasting task.

Offline results: In terms of customer impact, we use two metrics to measure performance of new offers. PWAGV (Products With A Glance View) is defined as average number of glance views per offer in given quarter. Similarly, PWAS (Products With A Sale) is defined as average sales per offer in given quarter. We compare GCF with the existing model in production on treated data. We observe a jump of 61.2% in PWAGV, 51.3% in PWAS, and a jump of 67.8% in selections added on the marketplace.

Deployed results: Post GCF deployment in 2023, we ob-

Model	MSE	MAPE
Temporal GCN	134.66	37.21
Google Causal Impact	127.07	27.87
CRN	111.71	26.14
GCF	**94.72**	**18.94**

Table 3: Results on semi synthetic METR-LA dataset

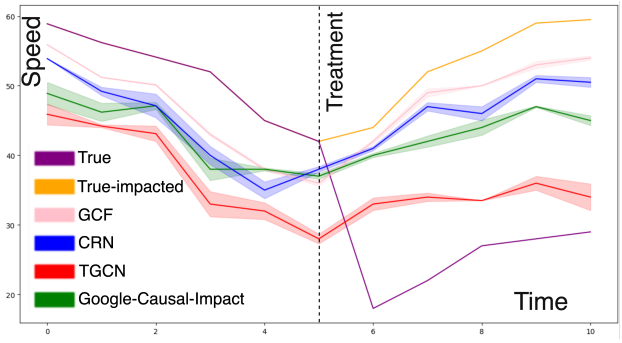

Figure 3: After the treatment (dashed line), the original time-series (in purple) drops significantly. However, GCF (in pink) is able to predict the true unobserved time-series (in yellow) effectively as compared to other SOTA techniques.

served a +1399 bps improvement in PWAGV from the customers, and +310 bps improvement in PWAS.

Open-source data results: For METR-LA traffic forecasting, in Table 3, we prove GCF's dominance by comparing against three SOTA paradigms. TGCN, which is popularly used for general forecasting tasks, Google Causal Impact model (taking all unaffected sensors as control) which serves as a popular causal intervened forecasting model choice, and Counterfactual-recurrent-network (CRN) which learns a balanced representation adversarially before estimating the counterfactual outcomes. We observe a bias towards unobserved data in all the models leading to underfitting with respect to unobserved data. GCF, on the other hand, maintains the trend, and outperforms both the models significantly as shown in figure 3.

Path to Deployment

The recommendations go once every quarter. We break the deployment process into six sub processes.

1) **Raw data ingestion:** The raw data from the platform like product demand, views and co-viewed products are dumped in hadoop clusters, consolidated quarter.

2) **Pre-processing for model consumption**: Once we have raw data at quarter level, we pre-process the data to create training-validation-test dataset, and the prediction dataset. Over a period of past 10 quarters, the first 8 quarters form the training set and 9th quarter is used for validation, and 10th(current quarter) for test. For prediction, we use quarters from 3rd to 10th quarter. Both contain past 8 quarters of sequential historical information which forms our features. We pick static features like product title, brand names from product catalogue table(maintained by catalogue team). We construct the knowledge graph of co-viewed products in this step. We use pyspark with m5.24xlarge(29 instances) for this pre-processing steps. This has been automated using AWS AI Workbench, runs every quarter on first Saturday. Data is dumped in s3 bucket.

3) **Input data monitoring checks:** We perform data checks on the pre-processed data at the category level before prediction, including metrics like demand mean/deviation, average views, total products, and total new products added. These must fall within defined thresholds to pass initial monitoring. This ensures the new data has no major distribution shift from training. If metrics fail, we raise an alarm highlighting the deviations. We leverage the data monitoring stack in the AI Workbench to implement this functionality.

4) **Model training:** Pre-processed data is read from s3. Model trains using m5.24xlarge(10 instances) cluster for 35 hours. Upon completion, evaluation script generates precision, recall and MAPE which gets monitored using AI Workbench. If metrics passes the thresholds set by business, we use the model weights(dumped to s3) for prediction. Otherwise, we raise an alarm for ML team to intervene.

5) **Prediction:** After successful training metrics, we generate the predictions and dump them to s3. This module generates the factual forecasts, and then the counterfactual forecasts, add them and generate the final demand predictions for each product. We use m5.24xlarge EMR cluster to generate predictions from both the models. The models run sequentially hence same resource is used twice. The generated recommendations are dumped in s3 bucket.

6) **Evaluation:** In this step, we measure business impact using PWAGV and PWAS. This module uses the past quarter's recommendations and the real values generated on the platform to measure PWAGV and PWAS. We use AI Workbench to monitor them, and also generate a quarterly report and share across business/tech/ML teams to keep track of quarter-on-quarter growth.

Conclusions and Future Work

We proposed a causal forecasting technique to mitigate the impact of a treatment and predict unobserved counterfactual values. We applied this for demand forecasting under localization, where product shipments from other regions are not allowed. The approach has two steps: 1) forecasting unaffected time-series using dilated CNN, 2) causal uplifting of predictions using RGCN-dilated CNN to obtain total demand. To handle imbalance, we used a novel B-APP loss and achieved 5.3% higher precision than MSE/MAE. The factual forecasting (FF) model achieved 10.5% higher precision and 23.8% lower MAPE than baselines. For the causal forecasting model, we used RGCN with co-view data to learn an optimal synthetic control, achieving 10.3% higher precision and 75.3% lower MAPE. This led to 61.2% higher PWAGV, 51.3% higher PWAS, and 67.8% more total selections. Post-deployment, we saw 1399 bps more products with views and 310 bps more with sales. On METR-LA data, GCF outperformed TGCN (30% lower MSE), CRN (15% lower MSE), and Google Causal Impact (25% lower MSE). Future work will utilize search queries and product embedding to improve synthetic control.

References

Athey, S.; and Imbens, G. W. 2017. The State of Applied Econometrics: Causality and Policy Evaluation. *Journal of Economic Perspectives*, 31(2): 3–32.

Bica, I.; Alaa, A. M.; Jordon, J.; and van der Schaar, M. 2020. Estimating Counterfactual Treatment Outcomes over Time Through Adversarially Balanced Representations. arXiv:2002.04083.

Brodersen, K. H.; Gallusser, F.; Koehler, J.; Remy, N.; and Scott, S. L. 2015a. Inferring causal impact using Bayesian structural time-series models. *The Annals of Applied Statistics*, 9(1): 247 – 274.

Brodersen, K. H.; Gallusser, F.; Koehler, J.; Remy, N.; and Scott, S. L. 2015b. Inferring causal impact using Bayesian structural time-series models. *Annals of Applied Statistics*, 9: 247–274.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv preprint arXiv:1409.1259*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:1412.3555.

Gardner Jr, E. S. 1985. Exponential Smoothing: The State of the Art. *Journal of Forecasting*, 4(1): 1–28.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep Learning for Time Series Classification: A Review. *Data Mining and Knowledge Discovery*, 33(4): 917–963.

Kashiparekh, K.; Narwariya, J.; Malhotra, P.; Vig, L.; and Shroff, G. 2019. Convtimenet: A pre-trained deep convolutional neural network for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Kausel, E. 2015. Joshua D. Angrist and Jörn-Steffen Pischke. Mastering 'Metrics: The Path from Cause to Effect. (Book Review). *Personnel Psychology*, 68: 931–933.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907.

Lee, Y.-S.; and Tong, L.-I. 2011. Forecasting time series using a methodology based on autoregressive integrated moving average and genetic programming. *Knowledge-Based Systems*, 24(1): 66–72.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. arXiv:1707.01926.

Lim, B.; and Zohren, S. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194): 20200209.

Wang, Z.; Yan, W.; and Oates, T. 2017. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 1578–1585. IEEE.

Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; and Hoi, S. 2022. Etsformer: Exponential Smoothing Transformers for Time-Series Forecasting. *arXiv preprint arXiv:2202.01381*.

Xu, J.; Wang, J.; Long, M.; et al. 2021. Autoformer: Decomposition Transformers with Auto-correlation for Long-term Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 34.

Yu, F.; and Koltun, V. 2015. Multi-scale Context Aggregation by Dilated Convolutions. *arXiv preprint arXiv:1511.07122*.

Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; and Li, H. 2020. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9): 3848–3858.