Demand Forecasting in the Presence of Privileged Information

Mozhdeh Ariannezhad[1], Sebastian Schelter[2,3], and Maarten de Rijke[2,3]

[1] AIRLab, University of Amsterdam
[2] University of Amsterdam
[3] Ahold Delhaize
{m.ariannezhad, s.schelter, m.derijke}@uva.nl

Abstract. Predicting the amount of sales in the future is a fundamental problem in the replenishment process of retail companies. Models for forecasting the demand of an item typically rely on influential features and historical sales of the item. However, the values of some influential features (to which we refer as *non-plannable features*) are only known during model training (for the past), and not for the future at prediction time. Examples of such features include sales in other channels, such as other stores in chain supermarkets. Existing forecasting methods ignore such non-plannable features or wrongly assume that they are also known at prediction time. We identify non-plannable features as privileged information, i.e., information that is available at training time but not at prediction time, and design a neural network to leverage this source of data accordingly. We present a dual branch neural network architecture that incorporates non-plannable features at training time, with a first branch to embed the historical information, and a second branch, the *privileged information* (PI) *branch*, to predict demand based on privileged information. Next, we leverage a single branch network at prediction time, which applies a simulation component to mimic the behavior of the PI branch, whose inputs are not available at prediction time. We evaluate our approach on two real-world forecasting datasets, and find that it outperforms state-of-the-art competitors in terms of mean absolute error and symmetric mean absolute percentage error metrics. We further provide visualizations and conduct experiments to validate the contribution of different components in our proposed architecture.

1 Introduction

Demand forecasting aims to predict future sales and has the potential to significantly improve supply chain management. An accurate forecast prevents overstocking and reduces costs, waste, and losses. At the same time, it also avoids understocking and thereby helps to prevent unfulfilled orders and unsatisfied customers [7, 29]. In practice, demand forecasting is modeled as a time series forecasting (TSF) problem, where the goal is to predict future sales volumes based on historical sales and influential features [5]. Following the success of deep neural networks in sequence-to-sequence tasks such as machine translation [28], recent work has studied the effectiveness of neural networks for TSF in general [16, 21, 23], and demand forecasting in particular [8, 11, 20].

We divide influential features into two categories, based on their availability at the time of forecast. *Plannable features* are known for the past and the future; examples are time-dependent features such as calendar events, or static features such as product characteristics. *Non-plannable features* are time-dependent and only known for the past, e.g., sales in other stores. Many neural-based approaches use an encoder-decoder structure to map the history of a time series to its future. A common strategy when treating influential features is to feed their historical values to the encoder, either alongside the historical sales data [11], or to a different layer that is responsible for encoding them [8]. The decoder then either uses their future values to produce the forecast [11, 31], or assumes that they are unknown [8]. Neither of these schemes is ideal for non-plannable features. Using future values of influential features in the decoding stage makes sense for plannable features, but the approach is not applicable for non-plannable features. A model trained in that way is not applicable in a real world setting as the non-plannable features are not known at prediction time. Simply ignoring non-plannable features – both at training and prediction time – is not optimal either as they carry important information that should be leveraged at the time of training the model.

In this paper, we therefore propose an indirect approach to model the effects of non-plannable features, and treat them as *privileged information* [18], i.e., information that is available at the time of training the model but not at prediction time (Section 3.1). We introduce a neural network architecture to capture the effect of these features at training time, and use this effect to produce a forecast at prediction time (Section 3.2). To this end, we propose *two different network architectures for training and prediction*. At the time of training, the network has two different branches. The first branch is responsible for modeling the effect of historical sales and plannable features, while the second branch uses non-plannable features as input to produce a forecast based on them. At prediction time, the second branch is not available, and is *replaced by a simulation network*, which is trained to mimic its behavior (Section 3.3).

Our experimental evaluation demonstrate that the proposed model outperforms state-of-the-art baselines on several datasets in terms of mean absolute error and symmetric mean absolute percentage error. Our experiments show that the proposed network is not only able to learn from non-plannable features at training time, but can also embed this type of information for use at prediction time.

We summarize the contributions of our research as follows:

- We categorize the influential features for demand forecasting into two categories, based on their availability for the time of forecast. We propose to treat the non-plannable features as privileged information (Section 3.1).
- We design a novel neural network architecture that is able to leverage privileged information. To this end, we propose to use two different networks for training and prediction time, where the network at prediction time simulates the effect of the unknown privileged information (Section 3.2).

– We conduct extensive experiments on two publicly available datasets. Our experimental results show that our approach outperforms state-of-the-art baselines for demand forecasting in the majority of cases in terms of mean absolute error and symmetric mean absolute percentage error metrics (Section 5).

2 Related Work

Time series forecasting. Prior work on time series forecasting (TSF) mostly focuses on linear approaches [22], such as Auto-Regressive Integrated Moving Average (ARIMA) model [6], with a solid underlying theory and relatively few parameters. However, linear methods cannot model non-linear temporal dependencies and complex relationships between different dimensions of a time series. Recently, neural network-based approaches have found their way into TSF. The most dominant type of network used are recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) [2, 4, 21, 23, 25]. Convolutional neural networks (CNNs) are also considered in the literature [14, 24], and some work studies networks with both recurrent and convolutional components [16, 32].

For the task of demand forecasting, state-of-the-art approaches mostly employ neural-based models. TADA [8] uses different LSTM layers to model different kinds of influential features, and the multimodal-attention model proposed in [11] uses a bidirectional LSTM with an attention mechanism to better capture latent patterns in historical data. To incorporate the impact of substitutable products with respect to the target product, DSF [20] uses a sequence-to-sequence structure with gated recurrent units.

Our focus is on modeling non-plannable features. In previous work, non-plannable features are either treated as plannable, i.e., with the unrealistic assumption that their future values are known at prediction time, or are only used as historical data. None of these approaches is able to incorporate non-plannable features in a realistic manner. In contrast, we propose a neural architecture, that (1) is capable of modeling non-plannable features, and (2) has different components for historical sales and influential features, which are usually treated in the same way in previous work.

Learning under privileged information. The learning under privileged information (LUPI) framework was originally proposed for support vector machines [30]. The idea was again popularized in [18], where it was unified with knowledge distillation for neural networks. Most of the work done in this area is in the field of computer vision [9, 12, 17], with teacher-student networks as the dominant approach. Teacher-student networks are mostly based on distilling knowledge from a 'teacher' network to a 'student' network at training time through the loss function, which makes sense for classification problems [18], where class probabilities produced by the teacher network are treated as 'soft targets' for training the student network.

To the best of our knowledge, LUPI has never been used in time series forecasting and utilizing existing LUPI frameworks is not straight-forward in a forecasting scenario. In this work, we propose a network architecture to leverage non-plannable features. We achieve this with two different networks at training

and prediction time. In contrast to common teacher-student networks, our second network is not guided by a loss component, but learns a simulation component that mimics the output of the missing branch.

3 A Privileged Information-Aware Neural Network

We present a dual branch neural network architecture for demand forecasting. It incorporates non-plannable features as privileged information (PI) at training time, with a first branch (the *historical branch*) that embeds historical information and plannable features via dilated causal convolutional layers, and a second branch, the *PIBranch*, which leverages fully-connected feed-forward layers to predict sales based on privileged information. At prediction time, we apply a slightly different single branch network with a simulation component to mimic the behavior of the PIBranch, whose inputs are not available at prediction time.

3.1 Problem Statement

The goal of a demand forecasting model is to predict the amount of sales in the future. Many different settings exist for building a forecasting model. Without loss of generality, we consider the case of multiple stores and multiple items, and forecast the demand of each item per store. Formally, for an arbitrary target product in a target store, the goal of the forecasting model is to predict

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = \{\hat{y}_{T+1}, \dots, \hat{y}_{T+\Delta}\},$$

where T is the length of history being considered, Δ is the forecast horizon, and $\hat{y}_t \in \mathbb{R}$ denotes the predicted sales of the target item in the target store at time t. Future sales are affected by both the history of sales in the past, and other influential features. Influential features can be divided into two categories. *Plannable features* are known both for the past and the future; they can be static, such as product-dependent features like category and brand, or time-dependent, such as promotional campaigns and calendar events. Future values of *non-plannable features* are not known at the time of forecast; examples include behavior data from users on an online shopping website, sales of similar items in the same store, or sales of the same item in other stores.

We design a forecasting model that incorporates both types of feature alongside the historical sales. Formally:

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = F\big(\{y_t\}_{t=1}^{T}, \{\mathbf{x}_t^p\}_{t=1}^{T+\Delta}, \{\mathbf{x}_t^{np}\}_{t=1}^{T}\big), \tag{1}$$

where $F(\cdot)$ is a non-linear mapping function that we learn, and

$$\{y_t\}_{t=1}^{T} = \{y_1, y_2, \dots, y_T\} \tag{2}$$

$$\{\mathbf{x}_t^p\}_{t=1}^{T+\Delta} = \{\mathbf{x}_1^p, \mathbf{x}_2^p, \dots, \mathbf{x}_{T+\Delta}^p\} \tag{3}$$

$$\{\mathbf{x}_t^{np}\}_{t=1}^{T} = \{\mathbf{x}_1^{np}, \mathbf{x}_2^{np}, \dots, \mathbf{x}_T^{np}\} \tag{4}$$

denote the historical sales of the target item, and the corresponding plannable features $\mathbf{x}^p \in \mathbb{R}^n$ and non-plannable features $\mathbf{x}^{np} \in \mathbb{R}^m$ for the item, respectively, with the corresponding feature dimensions n and m.

For training our model, we adopt the 'walk-forward' training schema that is a common choice for time series data [8, 15, 27]. In this approach, which we illustrate in Fig. 1, we apply a sliding window to divide the data into history

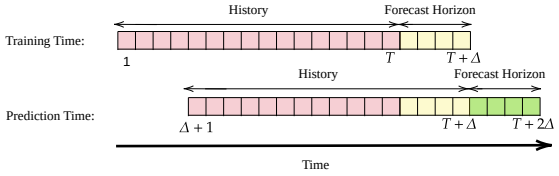


Fig. 1: The 'walk-forward' training scheme. A history of length T is used for training, and validation is performed on $[T+1, T+\Delta]$. For predication time, the history is shifted Δ steps, and $[T+\Delta+1, T+2\Delta]$ is used as forecast horizon.

and forecast horizon, and shift this sliding window Δ steps from training time to prediction time. In other words, assuming the length of the whole dataset is $T+2\Delta$ and $T \gg \Delta$ is the length of the sliding window, the $[1, T]$ interval is used as the history training time, the $[T+1, T+\Delta]$ interval is used as the forecast horizon at training time (i.e., the validation window), the $[\Delta+1, T+\Delta]$ interval is used as the history at prediction time and finally, the $[T+\Delta+1, T+2\Delta]$ interval is used as the forecast horizon at prediction time.

We evaluate our model on the forecast horizon at prediction time, for which the non-plannable features are not available. For training and hyperparameter selection, we leverage the history and the validation window, for which non-plannable features are known. This assumption is inline with real world cases, where a forecasting model is trained on the past data, for which the values of all influential features are already known. We rephrase Eq. 1 into Eq. 5 at training time and into Eq. 6 at prediction time in order to account for this setup:

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = F_1\left(\{y_t\}_{t=1}^{T}, \{\mathbf{x}_t^p\}_{t=1}^{T+\Delta}, \{\mathbf{x}_t^{np}\}_{t=1}^{T+\Delta}\right) \tag{5}$$

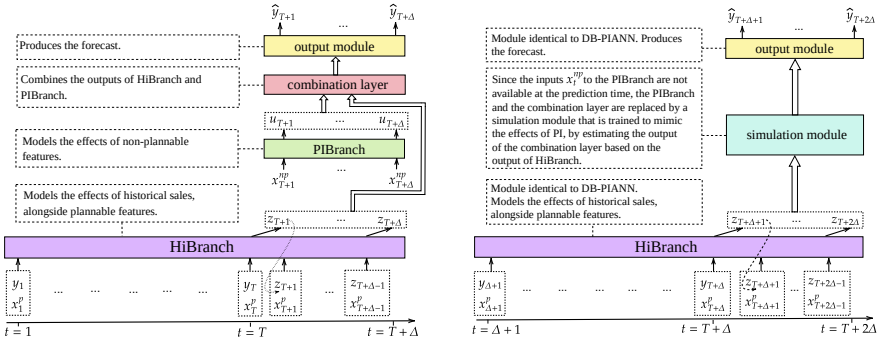
$$\{\hat{y}_t\}_{t=T+\Delta+1}^{T+2\Delta} = F_2\left(\{y_t\}_{t=\Delta+1}^{T+\Delta}, \{\mathbf{x}_t^p\}_{t=\Delta+1}^{T+2\Delta}\right), \tag{6}$$

where $F_1(\cdot)$ is the function that we learn at training time and $F_2(\cdot)$ is the function that we apply at prediction time to produce the forecast.

3.2 Architecture Overview

We model non-plannable features as privileged information (PI), i.e., information that is available at the training time but not available at prediction time. This approach requires different forecasting models for training and prediction time. We therefore propose: a *Dual Branch PI Aware Neural Network* (DB-PIANN) to embed both the historical sales and PI at training time, and a *Single Branch PI Aware Neural Network* (SB-PIANN) to produce the forecast at prediction time. Fig. 2a illustrates the architecture of DB-PIANN. Historical sales and plannable features are fed into one branch of the network, while privileged information is fed into a different branch. The outputs of these two branches are then merged with a combination layer, and fed into the output module to produce the forecast. The unavailablity of the PI for the future implies that we cannot produce a forecast with DB-PIANN at prediction time. However, only the input to the privileged information branch (PIBranch) is missing at prediction time, and we can still utilize the historical branch (HiBranch) of DB-PIANN.

We tackle this challenge by *training an additional simulation network to mimic the behavior of the missing PI Branch*. This network takes the output of



(a) DB-PIANN (dual branch) (b) SB-PIANN (single branch)

Fig. 2: Neural network architectures for PI-aware demand forecasting. Fig. 2a illustrates the architecture of DB-PIANN applied at training time and Fig. 2b details the architecture of the SB-PIANN, which we leverage at prediction time

.

the historical branch as input, and learns to reproduce the output of the combination layer. Fig. 2b details the architecture of the SB-PIANN. The difference to DB-PIANN is that the PIBranch and the combination layer are replaced with the simulation network; the other branch is identical. DB-PIANN and SB-PIANN model $F_1(\cdot)$ and $F_2(\cdot)$ in Eq. 5 and Eq. 6, respectively.

3.3 Architecture Details

Next, we introduce the details of DB-PIANN and SB-PIANN. As illustrated in Fig. 2a, DB-PIANN consists of four modules: (1) a *historical branch* (HiBranch), which is responsible for modeling the effects of historical sales data, along with plannable features, (2) a *privileged information branch* (PIBranch), which takes care of non-plannable features, (3) a *combination layer*, which combines the outputs of the two previous branches, and (4) an *output module*, which produces the final forecasts.

Based on these definitions, we break up Eq. 5 as follows:

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = F_1\big(\{c_t\}_{t=T+1}^{T+\Delta}\big), \tag{7}$$

where F_1 is the output module, and $\{c_t\}_{t=T+1}^{T+\Delta}$ is the output of the combination layer, defined as:

$$\{c_t\}_{t=T+1}^{T+\Delta} = \{z_t + u_t\}_{t=T+1}^{T+\Delta}, \tag{8}$$

where $\{z_t\}_{t=T+1}^{T+\Delta}$ and $\{u_t\}_{t=T+1}^{T+\Delta}$ are the outputs of the HiBranch and PIBranch, respectively (Fig. 2a).

Historical branch. The input of this branch, namely N_1, consists of the historical sales of an item y_t with of length T, and the plannable features $\mathbf{x_t^P}$ of length $T+\Delta$ corresponding to that item. The output of this branch, z_t, has the length of Δ, and will be fed to the combination layer. Formally:

$$\{z_t\}_{t=T+1}^{T+\Delta} = N_1\big(\{y_t \oplus \mathbf{x_t^P}\}_{t=1}^{T}, \{\mathbf{x_t^P}\}_{t=T+1}^{T+\Delta}\big), \tag{9}$$

where $\oplus$ denotes the concatenation operation. With this definition, HiBranch can be applied in SB-PIANN at prediction time, where we shift the time Δ

steps. However, one cannot include non-plannable features as well at prediction time, since their values are unknown for the forecast horizon.

We choose a stack of dilated causal 1D convolution layers [19] for the historical branch, followed by two fully-connected feed-forward layers to produce the final output of the branch. Formally:

$$N_1\big(\{y_t \oplus \mathbf{x_t^P}\}_{t=1}^{T}, \{\mathbf{x_t^P}\}_{t=T+1}^{T+\Delta}\big) = W_2^h \mathrm{ReLU}(W_1^h c_{L_1} + b_1^h) + b_2^h, \tag{10}$$

where c_{L1} is the output of the last convolutional layer, and W_1^h, b_1^h, W_2^h, and b_2^h are the weights and biases of the first and second feed-forward layer.

Causal convolutions are typically faster to train compared to RNNs and LSTMs (the common choice for sequence to sequence problems), since they do not have any recurrent connections [3]. Dilated convolutions allow us to process a long sequence without exponentially increasing the number of layers, by skipping input values with a certain step size. This is especially useful for the case of demand forecasting, where the history of the data is relatively long. More formally, for a sequence input x and a filter f with size k, a causal dilated convolution operation g on element s of the sequence is defined as:

$$g(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i}, \tag{11}$$

where d is the dilation factor. Following [19], we increase d exponentially with the depth of the network, i.e., $d = O(2^j)$ at level j of the network, where $j \in \{0, \ldots, L_1 - 1\}$ and L_1 denotes the number of layers. All parameters are shared across the whole forecast horizon.

The historical branch N_1 operates on a rolling basis, meaning that the output $z_t \in \mathbb{R}$ is produced one step at a time and is concatenated to the history in the input, which is is then shifted one step forward, and fed back into the first convolutional layer, until the end of the forecast horizon is reached (see Fig. 2a).

Privileged information branch. The input of this branch, namely N_2, are the non-plannable features with a length of Δ, and the output of this branch, u_t, (also of length Δ), will be fed to the combination layer. To make use of all the available training data, we apply a sliding window of length Δ and move it forward one step at a time, until we reach the end of the training interval, i.e., $T + \Delta$. For the PIBranch N_2, we leverage fully-connected feed-forward layers with dropout [26] applied after each hidden layer. We use a ReLU activation function for the hidden layers, and a linear fully-connected layer for the output. Formally:

$$\{u_t\}_{t=i}^{i+\Delta} = N_2(\{\mathbf{x^{np}}_t\}_{t=i}^{i+\Delta}) = W_{L_2}^{np} \mathrm{ReLU}(W_l^{np} x_l + b_l^{np}) + b_{L_2}^{np}, \tag{12}$$

where $1 \leqslant i \leqslant T$ is the start of the sliding window, L_2 denotes the number of layers, and W_l^{np} and b_l^{np} are the weights and biases of the l-th layer for $l \in \{1, \ldots, L_2\}$; x_l is the input of the l-th layer, which is the output of the previous layer for $l \in \{2, \ldots, L_2\}$, and equal to $\{\mathbf{x^{np}}_t\}_{t=T+1}^{T+\Delta}$ for $l = 1$.

Note that the PIBranch cannot be used in SB-PIANN at prediction time, since its input values are unknown for the forecast horizon.

Output module. In SB-PIANN, this module consumes the output of the combination layer, and produces the predicted sale values for the validation period. We again apply fully-connected feed-forward layers:

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = F_1(\{u_t + z_t\}_{t=T+1}^{T+\Delta}) = W_{L_3}^o \text{ReLU}(W_l^o s_l + b_l^o) + b_{L_3}^o, \quad (13)$$

where W_l^o and b_l^o are the weights and biases of the l-th layer for $l \in \{1, \ldots, L_3\}$, s_l is the input of l-th layer, which is the output of the previous layer for $l \in \{2, \ldots, L_3\}$, and equal to $\{u_t + z_t\}_{t=T+1}^{T+\Delta}$ for $l = 1$.

Together, the definitions of the aforementioned modules complete the architecture of DB-PIANN, according to Eq. 7.

SB-PIANN. Next, we define our second architecture SB-PIANN, which is going to produce the forecast at prediction time. We break down Eq. 6 as follows:

$$\begin{aligned} \{\hat{y}_t\}_{t=T+\Delta+1}^{T+2\Delta} &= F_2\left(\{y_t\}_{t=\Delta+1}^{T+\Delta}, \{\mathbf{x^P}_t\}_{t=\Delta+1}^{T+2\Delta}\right) \\ &= F_1\left(S(\{z_t\}_{t=T+\Delta+1}^{T+2\Delta})\right) \\ &= F_1\left(S(N_1(\{y_t \oplus \mathbf{x_t^P}\}_{t=\Delta+1}^{T+\Delta}, \{\mathbf{x_t^P}\}_{t=T+\Delta+1}^{T+2\Delta}))\right), \end{aligned} \quad (14)$$

where S is the simulation module, which we define in Eq. 15 below, N_1 refers to the historical branch defined in Eq. 10, and F_1 depicts the output module, defined in Eq. 13.

Simulation module. The purpose of the *simulation module* is to replace the missing PIBranch of DB-PIANN. As we cannot use the PIBranch at prediction time, one of the inputs of the combination layer is not available at prediction timeas well (see Fig. 2). Therefore, we train a feed-forward neural network to estimate the output of the combination layer based on the output of HiBranch. Formally:

$$S(\{z_t\}_{t=T+\Delta+1}^{T+2\Delta}) = W_{L_4}^s \text{ReLU}(W_l^s p_l + b_l^s) + b_{L_4}^s, \quad (15)$$

where W_l^s and b_l^s are the weights and biases of the l-th layer for $l \in \{1, \ldots, L_4\}$. p_l is the input of l-th layer, which is the output of the previous layer for $l \in \{2, \ldots, L_4\}$, and equal to $\{z_t\}_{t=T+\Delta+1}^{T+2\Delta}$ for $l = 1$.

3.4 Learning Process

Many options exist for training a forecasting model for a collection of time series, such as the sales per item per store in our case. Following previous work [8, 23], we train a single model for all of the items; each training sample contains the sales of a single item in a single store, along with its corresponding influential features.

We train the PIBranch N_2 and the historical branch N_1 separately, and then train DB-PIANN using the learned models, as outlined in the previous section. We subsequently train the simulation network S in isolation. With this scheme, SB-PIANN does not actually need training and can be composed from the previously learned modules, i.e., HiBranch, the simulation module and the output module. We leverage the mean absolute error between the predicted values and the actual values as objective function to train all the components of our architecture. All of our proposed modules are smooth and differentiable, which allows us to learn their parameters by standard back propagation. With

Table 1: Dataset statistics.

Dataset	Items	Stores	#time series	Time series' length
Favorita	1,656	54	11,614	365
Dunnhumby	1,101	26	8,825	117

mean absolute error as the objective function, we define the loss for each module as follows:

$$\mathcal{L} = \frac{1}{N}\Big(\sum_{n=1}^{N}\sum_{t=T+1}^{T+\Delta} \mid y_{nt}^{a} - y_{nt}^{p} \mid \Big), \tag{16}$$

where N is the number of training samples, n is an index for the samples, y_{nt}^{a} is the label for sample n at time t, and y_{nt}^{p} is the output of the corresponding module. Specifically, when training the HiBranch, PIBranch, and output module, y_{nt}^{a} is equal to the actual sales of the training sample n at time t, and y_{nt}^{p} is equal to the corresponding value of z_t, u_t, and $\hat{y}_t$ for sample n. For the simulation module, y_{nt}^{a} translates to the corresponding values of $z_t + u_t$ of sample n.

4 Experimental Setup

Datasets. Unfortunately, most of the sales datasets from existing work are not publicly available. For example, we could neither obtain the 'One Stop Warehouse' dataset from [8], the Amazon Demand Forecasting dataset from [31], nor the JD50K Online Sales Forecasting dataset from [11].

We therefore evaluate the performance of SB-PIANN on two publicly available datasets to ensure reproducibility. Both datasets contain sales numbers at the product and store level; we therefore set the goal of predicting the sales per product per store for a certain forecast horizon. Here, for a target item in a target store, we treat its sales in other stores as the privileged information. The *Favorita* dataset contains daily sales of thousands of items across 54 stores located in Ecuador.[4] We use the data from the 15th of August 2016 to the 15th of August 2017, and only consider items that have less than five days of sales data missing.[5] We also use the *The Complete Journey* dataset published by *Dunnhumby*.[6] This dataset contains around 300 million transactions for ∼5,000 items across ∼760 distinct stores, spanning more than two years of history. We randomly select a subset of items and stores for our experiments, and aggregate sales on a weekly basis to reduce the sparsity. The statistics of the datasets are shown in Table 1. Not all of the items are sold in all of the stores, so the total number of time series, i.e., training samples, is different from the number of items multiplied by the number of stores.

Influential features. Our design of the network architecture does not restrict the types of influential features that our approach can incorporate. However,

[4] https://www.kaggle.com/c/favorita-grocery-sales-forecasting/data

[5] The same dataset is used in [8], however the authors do not mention how they created a subset of the original dataset in their paper. Nevertheless, we achieve a comparable performance with their method on our version of the data.

[6] https://www.dunnhumby.com/careers/engineering/sourcefiles

we work with non-plannable features only in the experiments, as their effect on the forecast is the focus of this paper. For such non-plannable features, we rely on the sales of an item in other stores. In many demand forecasting datasets, including both of the datasets that we use, the retail company has more than one store, and the sales of a target item in other stores can help to forecast the demand in the target store. Formally:

$$\mathbf{x^{np}}_t = \{(x_t^{s_0}, x_t^{s_1}, \dots, x_t^{s_n})\}, \tag{17}$$

where n is the number of stores and $x_t^{s_i}$ denotes the sales for the target item in store s_i at time t.

Training and testing. Analogous to previous work, we use the walk-forward strategy [27, 8] for the train-test split, as detailed in Section 3.3. Following this strategy, we split the data according to the time dimension, and not based on stores and items. Given a time series with a total size of T for the history of sales data and Δ steps to predict, we use $[1, T]$ as the training data, $[T+1, T+\Delta]$ for validation and $[T+\Delta+1, T+2\Delta]$ for testing (as illustrated in Fig. 1). We experiment with $\Delta \in \{2, 8, 16\}$.

Implementation details. Our models are implemented using the Keras framework [10] with TensorFlow as backend [1]. We use mini-batch stochastic gradient descent (SGD) together with the Adam optimizer [13] to train the models. We set the batch size to 32, and stop training when the loss on the validation set converges. All the methods run on a Linux server with an Intel Xeon CPU, and a GeForce GTX 980 (Maxwell GM204) GPU. The GPU code is implemented using CUDA 9.

Metrics. We consider two metrics for evaluation: mean absolute error (MAE) $= \frac{1}{N \times \Delta} \sum_{n=1}^{N} \sum_{t=T+\Delta+1}^{T+2\Delta} |y_t - \hat{y}_t|$ and symmetric mean absolute percentage error (SMAPE) $= \frac{100}{N \times \Delta} \sum_{n=1}^{N} \sum_{t=T+\Delta+1}^{T+2\Delta} \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2}$, where y_t and $\hat{y}_t$ denote real and predicted sales, respectively.

Parameter settings. For all of the parameters of the networks, we conduct a grid search and leverage the parameters with the best performance on the validation set. For the feed-forward modules, we conduct a grid search over $\{1, 2, 3, 4\}$ for the number of hidden layers, {16, 32, 64, 128, 256, 512} for the number of nodes in each hidden layer, and $\{0.1, 0.2, 0.3, 0.4\}$ for the dropout rate. For the convolutional layers, we search over $\{16, 32, 64\}$ for the number of filters, $\{2, 4, 8, 16\}$ for the filter size, and $\{4, 5, 6, 7, 8\}$ for the number of layers.

5 Experimental Results

5.1 Capturing the Effects of Privileged Information

Our first set of experiments addresses the following research question: *To what extent is the proposed model, SB-PIANN, able to capture the effect of privileged information?* We conduct experiments with different components of our proposed model to answer our first research question. The purpose of these experiments is to reveal the importance of privileged information and to quantify its impact on the final performance of SB-PIANN. We compare its performance

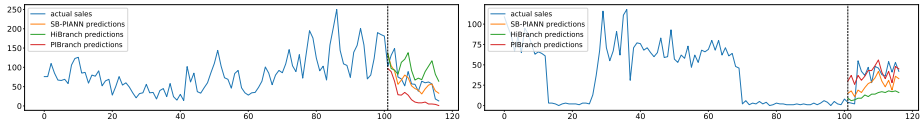
Table 2: Performance of SB-PIANN and DB-PIANN and their components on the Favorita and Dunnhumby datasets. * indicates that a component is significantly outperformed by SB-PIANN; $^-$ indicaties that there is no significant difference between DB-PIANN and SB-PIANN (student t-test, $\alpha = 0.05$).

Method	$\Delta = 2$		$\Delta = 8$		$\Delta = 16$	
	MAE	SMAPE	MAE	SMAPE	MAE	SMAPE
Favorita dataset						
HiBranch	6.126*	38.840*	6.494*	38.563*	7.278*	38.794*
PIBranch	6.415*	38.235*	6.664*	39.692*	7.080*	38.431*
SB-PIANN	6.069	37.139	6.277	38.102	6.868	38.241
DB-PIANN	5.985$^-$	36.956$^-$	6.052$^-$	37.304$^-$	6.752$^-$	37.314$^-$
Dunnhumby dataset						
HiBranch	3.593	43.042*	3.661*	44.348*	4.079*	50.289*
PIBranch	3.669*	44.108*	3.640	44.006	3.691	45.059
SB-PIANN	3.558	42.567	3.612	43.506	4.059	49.540
DB-PIANN	3.555$^-$	42.531$^-$	3.601$^-$	43.384$^-$	3.899$^-$	47.817$^-$

to that of the HiBranch and PIBranch in isolation, as well to the "oracle" performance of DB-PIANN. DB-PIANN's performance is only reported for the sake of comparison; the model cannot be used for prediction in real world cases due to the unavailability of the values of non-plannable features at prediction time.

Results and discussion. Table 2 shows the performance of HiBranch, PIBranch, SB-PIANN and DB-PIANN in terms of MAE and SMAPE, on testing time intervals, i.e., $[T + \Delta + 1, T + 2\Delta]$. The results are reported for all Δ settings on the Favorita and Dunnhumby datasets. We compare the performance of two different branches for different forecast horizons. We observe that the performance of the PIBranch is comparable to that of the HiBranch for shorter forecast horizons, i.e., 2 and 8, and outperforms HiBranch for the longest forecast horizon. This shows that the PIBranch is capable of predicting the sales at least as well as HiBranch, and is more robust with respect to forecast horizon. We attribute the ability of PIBranch to predict future sales to two aspects: First, we note that the historical sales data are not taken into account in the PIBranch, and the forecast is produced solely based on the privileged information. This indicates the usefulness of this information for demand forecasting. Second, the obtained performance points out the effectiveness of the proposed network to model the effects of privileged information. In other words, a deep feed-forward neural network is capable of producing a forecast based on the sales of products in other stores, and this forecast is comparable to and in some cases superior to the one based on the historical sales.

We also report the performance of DB-PIANN, while noting (again) that it cannot be used in a real world setting, as it requires the privileged information, which is not available at prediction time. DB-PIANN outperforms individual branches on both datasets and for all of the forecast horizons, which shows that it is able to leverage both the historical sales and the privileged information



(a) (b)

Fig. 3: Forecasts produced by HiBranch, PIBranch and SB-PIANN, along with the actual sales, for sample products. 3a is taken from Favorita and 3b is selected from Dunnhumby. y axis shows the sales and x axis is the time. 3a and 3b show the improved forecasts using PI.

to produce the final forecast. The performance gain also indicates that both branches contribute positively to the final forecast; their contribution is not overlapping, and they are both essential to achieve the highest performance.

Finally, we observe that the performance of SB-PIANN is better than that of HiBranch and is relatively close to DB-PIANN, on both datasets and for all forecast horizons. Recall that SB-PIANN is not using the inputs of the PIBranch, and indirectly models the effects of privileged information via the simulation component. The close performance of SB-PIANN to that of DB-PIANN shows that the proposed approach is able to embed the PI from train to test time effectively. In other words, the simulation component is capable of transferring the effects of privileged information absorbed in the train time to the prediction time. The superiority of SB-PIANN compared to the HiBranch supports the hypothesis that utilizing the privileged information leads to a better performance than a forecast that only relies on the historical data.

To gain more insights into forecasts by our proposed method, we visualize a few examples in Fig. 3. Each figure shows the demand for a product in a store, taken from both Favorita and Dunnhumby. We show 101 days of history for both datasets. The history of sales as well as the actual sales in the future are plotted, along with forecasts made by PIBranch, HiBranch, and SB-PIANN. We observe that the forecast based on PI has a different trend from the one based on the history of sales. This might be a result of an event that affects the sales of the product in the stores, and leads to a better prediction for the future sales, compared to the forecast of HiBranch, which is based on the history of sales in the target store. We can also recognize that SB-PIANN picks-up on this difference, and its forecast is superior to that of HiBranch.

5.2 Comparison to Existing Approaches for Demand Forecasting

Next, we focus on the following question: *How effective and accurate is SB-PIANN for the task of demand forecasting?* We compare SB-PIANN to the following state-of-the-art neural network-based forecasting models:

DA-RNN [**21**]. This dual-stage attention-based RNN method is proposed for time series prediction. It uses an encoder-decoder structure with two different attention mechanisms. In this model, the PI are available also for the future. However, this is an unrealistic assumption; we use this model as the baseline to evaluate the ability of our model in making use of PI.

LSTNet [16]. The long- and short-term time-series network uses both CNNs and RNNs to capture both short-term and long-term trending patterns of the time series. It also has an auto-regressive component. The architecture is proposed for multi-variate TSF, therefore we build a model for each item and forecast for the target stores simultaneously. For LSTNet, we only make use of the historical values of PI for both training and test time.

TADA [8]. An encoder-decoder based architecture that uses two LSTM branches for encoding different types of feature. Since the method was tested on the Favorita dataset, we also report their results based on their original division of features for this dataset. For the Dunnhumby dataset, we randomly divide the data from other stores into two categories. We again only make use of the historical values of PI available for both training and test time.

All baselines report superior performance compared to auto-regressive models, decision tree models, and simpler neural network-based approaches. We therefore omit these methods from our evaluation. Aside from their state-of-the-art performance, we chose these methods as baseline to cover a wide range of possible approaches. Specifically, they apply different training schemes; DA-RNN trains a model per time-series, LSTNet trains a model per store, and TADA uses an approach similar to ours, training a single model for all of the data. They also differ in the ways they treat the PI. For DA-RNN, the assumption is that these features are available for both history and the future (which is not true in real world settings), while the other two only use the historical values of the PI.

Implementation details. For the baseline implementation, we asked the authors of TADA and DA-RNN for the code and they kindly provided us with their own implementation. For LSTNet, we leverage the code which is made publicly available by the authors.[7] We use the same testing environment as SB-PIANN for the baselines, as outlined in Section 4.

Results and discussion. Table 3 shows the performance of SB-PIANN compared to the baselines for different forecast horizons on Favorita and Dunnhumby, respectively. The best performance is highlighted with bold face. In almost all cases, SB-PIANN outperforms the baselines in terms of MAE and SMAPE. We observe that the performance of all methods starts to drop with the increase of the forecast horizon, but SB-PIANN is more robust with respect to the length of the forecast horizon. In all cases, the performance is better on Favorita than on Dunnhumby in terms of SMAPE. This might be due to the fact that the sales data in Dunnhumby is more scarce.

According to the results, SB-PIANN even outperforms DA-RNN, which makes the unrealistic assumption of having the PI available at prediction time. The fact that SB-PIANN outperforms even DA-RNN shows the effectiveness of our proposed approach to leverage privileged information. The lower performance of DA-RNN might also be a result of its training scheme, i.e., building a forecast model per time series. In this scenario, the model cannot learn from the similarity and differences between different products and stores.

[7] https://github.com/laiguokun/LSTNet

Table 3: Comparison of neural network-based forecasting methods on the Favorita and Dunnhumby datasets. * indicates that a method is significantly outperformed by SB-PIANN (student t-test, $\alpha = 0.05$).

Method	Uses PI		$\Delta = 2$		$\Delta = 8$		$\Delta = 16$	
	Train	Test	MAE	SMAPE	MAE	SMAPE	MAE	SMAPE
Favorita dataset								
DA-RNN	+	+	9.343*	48.864*	8.583*	46.174*	8.709*	43.132*
LSTNet	−	−	6.619*	40.509*	7.481*	43.920*	8.332*	45.016*
TADA	−	−	6.428*	**36.744**	7.431*	41.389*	8.463*	43.165*
SB-PIANN	+	−	**6.069**	37.139	**6.277**	**38.102**	**6.868**	**38.241**
Dunnhumby dataset								
DA-RNN	+	+	4.535*	52.779*	3.995*	47.154*	4.168*	**48.942**
LSTNet	−	−	4.134*	48.484*	4.818*	55.310*	5.673*	59.864*
TADA	−	−	4.367*	49.018*	5.777*	63.834*	8.251*	78.055*
SB-PIANN	+	−	**3.558**	**42.567**	**3.612**	**43.506**	**4.059**	49.540

While the special type of PI that we experiment with, i.e., sales in other stores, suggests the use of a multi-variate TSF model, our experiments show that compared to LSTNet, a state-of-the-art model proposed for the multi-variate TSF, SB-PIANN, performs better in all cases. On the Favorita dataset, we compare the performance of SB-PIANN with the original version of TADA, i.e., using the definition that the authors propose for internal and external features. In this version, a set of 13 attributes are used as features, such as the location of the stores and the oil price. In our approach, we only rely on the sales of the items in other stores as features, and we observe that our model performs significantly better in terms of both error metrics, while using no manual categorization of features. The gain in performance is more significant on the Dunnhumby dataset, which indicates the limitation of TADA when the division of features into internal and external as required by TADA is not straightforward. In other words, the performance of TADA degrades when influential features cannot be characterized as internal and external, and accordingly, cannot be fed into the corresponding LSTM layers.

6 Conclusions and Future Work

Demand forecasting is a fundamental problem in the replenishment process of retail companies. Models for forecasting the demand for a product usually consider patterns in the history of sales data and influential features. Such influential features can be divided into two categories: plannable features that are known for the past and the future, and non-plannable features, for which the future values are unknown. Neural forecasting models usually ignore non-plannable features when predicting the amount of sales in the future.

In this paper, we identify non-plannable features as privileged information and design a novel neural network to utilize them. We propose two different network architectures for training and prediction time. At the time of training, the network has two different branches to model the effect of historical sales and

non-plannable features. At prediction time, the second branch is not available, and is replaced by a simulation network that is trained to mimic its behavior. We extensively evaluate our proposed approach on two real-world forecasting datasets, and find that it outperforms state-of-the-art baselines in terms of mean absolute error and symmetric mean absolute percentage error metrics.

While our proposed architecture is capable of plannable features, our focus in this paper is on modeling non-plannable features. In future work, it will be appealing to study different approaches to incorporate plannable features in the model; aside from treating them as an extra dimension of the historical sales, they can be fed into the PIBranch, or an extra dedicated branch. Moreover, although we make no specific assumptions about the type of non-plannable features in our model, we rely on a single type of non-plannable features in our experiments. We also aim to investigate the impact of more sources of privileged information.

Code and data. To facilitate the reproducibility of the reported results, this work only made use of publicly available data and our experimental implementation is publicly available at https://github.com/mzhariann/PIANN.

Acknowledgments. This research was supported by Ahold Delhaize. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI. pp. 265–283 (2016)
2. Adhikari, B., Xu, X., Ramakrishnan, N., Prakash, B.A.: Epideep: Exploiting embeddings for epidemic forecasting. In: KDD. pp. 577–586 (2019)
3. Bai, S., Kolter, J.Z., Koltun, V.: Convolutional sequence modeling revisited. In: ICLR (2018)
4. Bao, W., Yue, J., Rao, Y.: A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS one **12**(7), e0180944 (2017)
5. Boese, J., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M.W., Wang, B.: Probabilistic demand forecasting at scale. PVLDB **10**(12), 1694–1705 (2017)
6. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)
7. Carbonneau, R., Laframboise, K., Vahidov, R.M.: Application of machine learning techniques for supply chain demand forecasting. European Journal of Operational Research **184**(3), 1140–1154 (2008)
8. Chen, T., Yin, H., Chen, H., Wu, L., Wang, H., Zhou, X., Li, X.: TADA: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction. In: ICDM. pp. 49–58 (2018)
9. Chen, Y., Jin, X., Feng, J., Yan, S.: Training group orthogonal neural networks with privileged information. In: IJCAI. pp. 1532–1538 (2017)
10. Chollet, F., et al.: Keras. https://keras.io (2015)

11. Fan, C., Zhang, Y., Pan, Y., Li, X., Zhang, C., Yuan, R., Wu, D., Wang, W., Pei, J., Huang, H.: Multi-horizon time series forecasting with temporal attention learning. In: KDD. pp. 2527–2535 (2019)
12. Hoffman, J., Gupta, S., Darrell, T.: Learning with side information through modality hallucination. In: CVPR. pp. 826–834 (2016)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
14. Koprinska, I., Wu, D., Wang, Z.: Convolutional neural networks for energy time series forecasting. In: IJCNN. pp. 1–8 (2018)
15. Ladyzynski, P., Zbikowski, K., Grzegorzewski, P.: Stock trading with random forests, trend detection tests and force index volume indicators. In: ICAISC. pp. 441–452 (2013)
16. Lai, G., Chang, W., Yang, Y., Liu, H.: Modeling long- and short-term temporal patterns with deep neural networks. In: SIGIR. pp. 95–104 (2018)
17. Lambert, J., Sener, O., Savarese, S.: Deep learning under privileged information using heteroscedastic dropout. In: CVPR. pp. 8886–8895 (2018)
18. Lopez-Paz, D., Bottou, L., Schölkopf, B., Vapnik, V.: Unifying distillation and privileged information. In: ICLR (2016)
19. Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)
20. Qi, Y., Li, C., Deng, H., Cai, M., Qi, Y., Deng, Y.: A deep neural framework for sales forecasting in e-commerce. In: CIKM. pp. 299–308 (2019)
21. Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., Cottrell, G.W.: A dual-stage attention-based recurrent neural network for time series prediction. In: IJCAI. pp. 2627–2633 (2017)
22. Ristanoski, G., Liu, W., Bailey, J.: Time series forecasting using distribution enhanced linear regression. In: PAKDD. pp. 484–495 (2013)
23. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2019)
24. Shen, Z., Zhang, Y., Lu, J., Xu, J., Xiao, G.: Seriesnet: A generative time series forecasting model. In: IJCNN. pp. 1–8 (2018)
25. Shih, S., Sun, F., Lee, H.: Temporal pattern attention for multivariate time series forecasting. Machine Learning **108**(8-9), 1421–1441 (2019)
26. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
27. Stein, R.M.: Benchmarking default prediction models: Pitfalls and remedies in model validation. Moody's KMV, New York **20305** (2002)
28. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NeurIPS. pp. 3104–3112 (2014)
29. Vairagade, N., Logofatu, D., Leon, F., Muharemi, F.: Demand forecasting using random forest and artificial neural network for supply chain management. In: ICCCI. pp. 328–339 (2019)
30. Vapnik, V., Vashist, A.: A new learning paradigm: Learning using privileged information. Neural Networks **22**(5-6), 544–557 (2009)
31. Wen, R., Torkkola, K., Narayanaswamy, B., Madeka, D.: A multi-horizon quantile recurrent forecaster. arXiv preprint arXiv:1711.11053 (2017)
32. Wu, X., Shi, B., Dong, Y., Huang, C., Faust, L., Chawla, N.V.: Restful: Resolution-aware forecasting of behavioral time series data. In: CIKM. pp. 1073–1082 (2018)