DURATION-OF-STAY STORAGE ASSIGNMENT UNDER UNCERTAINTY

Anonymous authors
Paper under double-blind review

ABSTRACT

Storage assignment, the act of choosing what goods are placed in what locations in a warehouse, is a central problem of supply chain logistics. Past literature has shown that the optimal method to assign pallets is to arrange them in increasing duration of stay in the warehouse (the Duration-of-Stay, or DoS, method), but the methodology requires perfect prior knowledge of DoS for each pallet, which is unknown and uncertain under realistic conditions. Attempts to predict DoS have largely been unfruitful due to the multi-valuedness nature (every shipment contains multiple identical pallets with different DoS) and data sparsity induced by lack of matching historical conditions. In this paper, we introduce a new framework for storage assignment that provides a solution to the DoS prediction problem through a distributional reformulation and a novel neural network, ParallelNet. Through collaboration with a world-leading cold storage company, we show that the system is able to predict DoS with a MAPE of 29%, a decrease of $\sim$30% compared to a CNN-LSTM model, and suffers less performance decay into the future. The framework is then integrated into a first-of-its-kind Storage Assignment system, which is being deployed in warehouses across United States, with initial results showing up to 21% in labor savings. We also release the first publicly available set of warehousing records to facilitate research into this central problem.

1 INTRODUCTION

The rise of the modern era has been accompanied by ever-shortening product life cycles, straining the entire supply chain and demanding efficiency at every node. One integral part of any supply chain is warehousing (storage); warehouse operations often have major impacts downstream on the capability to deliver product on time.

One of the largest cold storage companies in the world is looking to improve the efficiency of their warehouses by optimizing the scheduling of storage systems. According to Hausman et al. (1976), the scheduling of labor in warehouses can be divided into three main components:

- *Pallet Assignment*: The assignment of multiple items to the same pallet.
- *Storage Assignment*: The assignment of pallets to a storage location.
- *Interleaving*: The overarching rules for dealing with concurrent inbound and outbound requests.

For this particular paper, we focus on the problem of storage assignment. Various papers such as Goetschalckx & Ratliff (1990) show labor efficiency to be a bottleneck. In a modern warehouse, the process of storage assignment usually involves forklift drivers moving inbound pallets from the staging area of the warehouse to the storage location, so a sub-optimal assignment system causes unnecessary long travel times to store the pallet. Unfortunately, the inefficiency is quadrupled when the return of the forklift and the retrieval of the pallet are considered.

To increase the efficiency of the warehouse, we would thus like to minimize the total travel time needed to store a set of shipments from the staging area. Many different theoretical frameworks exist, and the details of such frameworks are contained in Appendix 9.1. The ones of chief interest are turnover-based, class-based, and Duration-of-Stay (DoS) based strategies.

Turnover-based strategies (e.g. Hausman et al. (1976), Yu & De Koster (2009)) assign locations so that the resultant travel distance is inversely proportional to the turnover of the product. Class-based strategies (e.g. Hausman et al. (1976), Rosenblatt & Eynan (1989), Schwarz et al. (1978) and Petersen et al. (2004)) separate products into k classes, with each class assigned a dedicated area of storage. DoS-based strategies (e.g. Goetschalckx & Ratliff (1990), Chen et al. (2016)) assign pallets to locations with travel distance proportional to the duration-of-stay.

Simulation experiments in Goetschalckx & Ratliff (1990) and Kulturel et al. (1999) demonstrated that under complex stochastic environments, DoS-based strategies outperform other methodologies significantly. However, among all three categories, the most commonly used strategy is class-based, as pointed out by Yu et al. (2015) and Yu & De Koster (2013). The authors and industry evidence suggest that this is due to the fact that class-based systems are relatively easy to implement, but also because DoS is not known in advance. To utilize a DoS system realistically would therefore require an accurate prediction model using the features available at shipment entry to the warehouse.

However, even with the availability of modern high-powered predictive methods including Gradient Boosted Trees and Neural Networks, there has been no documented progress in employing DoS-based methods. This reflects the following significant challenges in a dynamic, real warehouse:

- **Multi-valuedness**: Identical pallets arriving at the same time can leave the warehouse at different times due to differing demand. It is common in the warehouse for $10+$ pallets of the same product to arrive in a single shipment, and then leave the warehouse at different times depending on the consumption of the end consumer. This causes the ground truth for the DoS of a product entering at a given time to be ill-defined by a single number.
- **Data Sparsity**: A large warehouse would have significant available historical DoS data, but such data is scattered across thousands of products/SKUs, and different operating conditions (e.g. time of the year, day of the week, shipment size). Given strong variation of DoS in a warehouse, it is very unlikely that all environment-product combinations would exist in data for the historical average to be valid for future DoS. Furthermore, new SKUs are created relatively frequently, and the predictive algorithm needs to be robust against that as well.

To solve such difficulties, we reformulate the DoS as a distribution and develop a new framework based on nonparametric estimation. Then we combine it with recent advances in machine learning to provide a practical toolkit to realize the efficiency gains of DoS with a parallel architecture of Residual Deep Convolutional Networks (He et al. (2015)) and Gated Recurrent Unit (GRU) networks (Cho et al. (2014)) that DoS can be well estimated. As far as the authors know, this is the first documented attempt to predict DoS in warehousing systems. We further release the first public dataset of warehousing records to enable future research into this problem.

This neural network is then integrated into the larger framework, which is being implemented in live warehouses. We illustrate how initial results from the ground show appreciable labor savings.

Specifically, our contributions in this paper include:

- We develop an novel end-to-end framework for optimizing warehouse storage assignment using the distribution of DoS.
- We release a curated version of a large historical dataset of warehousing records that can be used to build and test models that predicts DoS.
- We introduce a type of neural network architecture, ParallelNet, that achieves state-of-the-art performance in estimating the DoS distribution.
- Most importantly, we present real-life results of implementing the framework with ParallelNet in live warehouses, and show labor savings by up to 21%.

The structure of the paper is as followed. In Section 2, we would review possible methods to tackle the storage assignment problem, and why distribution estimation was selected. In Section 3, we develop the storage assignment framework. We would introduce the dataset in Section 4 and Section 5 contains the implementation with ParallelNet, and its results compared to strong baselines. Section 6 shows the computational results, while real-life evidence is provided in Section 7.

2 Solving the Storage Assignment Problem

The general storage assignment problem asks for an algorithm that outputs a storage location given a pallet's features, warehouse state, and entry timestamp so the total travel time for storage is minimized.

We first note that the dynamic nature of the problem seems to naturally call for Markov decision processes or reinforcement learning. However, a typical warehouse has a decision space of over $50,000$ available storage locations, each with varying characteristics. It is not realistic to train such an algorithm in a live warehouse, and simulating a warehouse using a computational model is infeasible due to its complexity. Therefore, we would not consider these approaches and turn to classical storage assignment theory.

From Goetschalckx & Ratliff (1990), we know that the DoS strategy is theoretically optimal; therefore to utilize this result we would like to predict the DoS of a pallet at its arrival. Since historical data is sparse in environment-product combinations, we cannot use prior data directly to predict future DoS and would rather need to train a continuous predictive function f for DoS . However, in most real-life warehouses, a product P_i enters the warehouse with multiple $z_i > 1$ identical pallets, which leave at different times $t_{i1}, \cdots t_{iz_i}$. Thus, assuming the product came in at time 0, the DoS of incoming pallets of P_i could be any of the numbers $t_{i1}, \cdots t_{iz_i}$, which makes the quantity ill-defined. The uncertainty can further be very large - our collaborating company had an average variance of DoS within a shipment of 10 days when the median DoS was just over 12 days.

To alleviate this, we can either choose to impose an order on the identical shipments or accept the uncertainty and treat DoS as a distribution. For the former case, a natural order is labelling each pallet with $(1, 2, \cdots, z_i)$ so that pallet 1 leaves the warehouse before pallet 2, etc. However, such methodology is ill-equipped to deal with the randomness of a modern warehouse. The first pallet of a product P_i entering at t could leave in 3 days while the first pallet of the next P_i shipment at $t + \Delta t$ might stay for over 10 days. The only difference in features between these two pallets is the time of entry, and for a dynamic warehouse Δt is small. This presents a significant challenge to fitting a continuous function f.

Therefore, to account for the uncertainty that identical pallets map to different DoS values, we would thus assume that for every shipment S (which contains multiple pallets), the DoS of a random pallet is uncertain and follows a cumulative distribution $F_S(t)$. Furthermore, we make an assumption between the characteristics X_S of the shipment known at arrival time with F_S so that F_S is identifiable:

Assumption 1 *We assume that F_S is uniquely determined by X_S. As in, there exists a function f mapping from the space of all characteristics to the space of all cumulative distribution functions (CDFs) such that:*

$$F_S(t) = g(X_S)(t)$$

For all possible shipments S.

This assumption is not as strong as it may seem - the most important variables that affect DoS usually are the time and the good that is being transported, both of which are known at arrival. As a simple example, if the product is ice cream and it is the summer, then we expect the DoS distribution to be right skewed, as ice cream is in high demand during the summer. Moreover, the experimental results in Section 6 are indicative that g exists and is highly estimable.

If the above assumption holds true, then we can estimate g using modern machine learning techniques. In the next section, we would outline our storage assignment framework based on such assumption.

3 Overview of Storage Assignment Framework

Now let us assume we have an optimal estimate $\tilde{g}$ of g, measured relative to a loss function $l(F_S, \tilde{g}(X_S))$ denoting the loss between the actual and predicted distribution. Since by our assumption F_S is uniquely determined by X_S, we cannot obtain any further information about DoS relative to this loss function. Thus, for each shipment with characteristics X_S, we take a random sample T_S from the distribution $\tilde{F_S} = \tilde{g}(X_S)$ and let that be the estimate of the DoS of the shipment. Then we construct the storage location assignment function $A : \mathbb{R} \to \mathcal{W}$ as followed:

$$A(T_S) = \arg\min_{w \in \tilde{\mathcal{W}}} d(M(W(T_S)), w) + c(w)$$

Where $W(t)$ is the historical cumulative distribution of DoS in the warehouse, $M(k)$ is the storage location at $100k\%$ percentile distance away from the staging area, $d(v, w)$ is the distance function between location v and w in the warehouse. $c(w)$ are other costs associated with storing at this position, including anti-FIFO orders, item mixing, height mismatch between the pallet and the slot, and others. $\tilde{\mathcal{W}}$ is the set of positions that are available when the pallet enters the warehouse.

The optimal position of storing the pallet according to the DoS is the $M(W(T_S))$ position as the kth percentile in DoS should correspond to the kth percentile location in the warehouse. However, it is probable that such location is not ideal, either because it is already taken ($M(W(T_S)) \notin \tilde{W}$), or there are other factors that affect its optimality. For the collaborating company, one important factor is item mixing due to potential of cross-contamination of food allergens in close pallets. These terms are highly dependent on the specific storage company, and thus we include them as a general cost term $c(w)$ to add to the cost $d(M(W(T_S)), w)$ of not storing the pallet at the DoS ideal position. The resulting optimal location based on the combination of the two costs is then chosen for the pallet.

In summary, our framework consists of four steps:

1. Use a machine learning model to provide an estimate $\tilde{g}$ of g through training on historical data for X_S and F_S.
2. For a shipment S, calculate its approximate distribution $\tilde{F}_S = \tilde{g}(X_S)$.
3. Generate a random sample T_S from $\tilde{F}_S$.
4. Apply the optimal assignment function A under the DoS policy to determine a storage location $A(T_S)$, as defined above.

4 Warehousing Dataset Overview and Construction

In this section, we introduce the historical dataset from the cold storage company to test out the framework and model introduced in Section 3.

4.1 Overview of the Data

The data consists of all warehouse storage records from 2016.1 to 2018.1, with a total of 8,443,930 records from 37 different facilities. Each record represents a single pallet and one shipment of goods usually contain multiple identical pallets (which have different DoS). On average there are 10.6 pallets per shipment in the dataset. The following covariates are present:

- **Non-sequential Information**: Date of Arrival, Warehouse Location, Customer Type, Product Group, Pallet Weight, Inbound Location, Outbound Location
- **Sequential Information**: Textual description of product in pallets.

Inbound and Outbound location refers to where the shipment was coming from, and where it would be going (both are known at arrival). The records are mainly food products, with the most common categories being (in decreasing order): chicken, beef, pork, potato, and dairy. However, non-food items such as cigarettes are also present.

The item descriptions describe the contents of the pallet, but most of them are not written in a human readable form, such as ”NY TX TST CLUB PACK”. Acronyms are used liberally due to the length restriction of item descriptions in the computer system. Furthermore, the acronyms do not necessarily represent the same words: ”CKN WG” means ”chicken wing” while ”WG CKN” means ”WG brand chicken”. Therefore, even though the descriptions are short, the order of the words is important.

To enable efficient use of these item descriptions, we decoded common acronyms used by hand (such as tst $\rightarrow$ toast). We would stress that the resulting dataset is not perfectly clean (intentionally so to mimic item descriptions encountered in real life) and contains many broken phrases, misspelled words, unidentified acronyms, and other symbols.

4.2 PUBLIC RELEASE VERSION [1]

We release the above dataset, which as far as the authors know, is the first publicly available dataset of warehousing records.

The collaborating company transports an appreciable amount ($> 30\%$) of the entire US refrigerated food supply, so US law prohibits the release of the full detail of the transported shipments. Furthermore, NDA agreements ban any mentioning of the brand names. Thus, for the public version, we removed all brands and detailed identifying information in the item descriptions. The testing in the section below is done on the private version to reflect the full realistic scenario in the warehouse, but the results on the public version are similar and the conclusions carry over to the public version.

5 IMPLEMENTING THE FRAMEWORK

The Framework described in Section 3 requires the knowledge of four parameters: X_S and F_S for pallets in the training data, the loss function $l(F_S, \tilde{F_S})$, and the machine learning model estimate $\tilde{g}$.

X_S is immediately available in the form of the non-sequential and sequential information. For the textual description, we encode the words using GloVe embeddings. (Pennington et al. (2014)) We limit the description to the first five words with zero padding.

For F_S, we first exploit the fact that each shipment arriving most often contains $p \gg 1$ units of the good, and we could treat these p units as p copies of a shipment of one unit, denoted $S_1, \cdots S_p$. Then by using the known DoS for each of these p units $(T_1, \cdots T_p)$, we could create an empirical distribution function $\hat{F}_S(t) = \frac{1}{p}\sum_{i=1}^{n} \mathbf{1}_{T_i \leq t}$ for the DoS of the shipment. This is treated as the ground truth for the training data.

To obtain a loss function for the distribution, we selected the $5\%, \cdots 95\%$ percentile points of the CDF, which forms a 19-dimensional output. This definition provides a more expressive range of CDFs than estimating the coefficients for a parametric distribution. Then we chose the mean squared logarithmic error (MSLE) as our loss function to compare each percentile point with those predicted. This error function is chosen as the error in estimating DoS affects the positioning of the unit roughly logarithmically in the warehouse under the DoS policy. For example, estimating a shipment to stay 10 days rather than the true value of 1 day makes about the same difference in storage position compared to estimating a shipment to stay 100 days rather than a truth of 10. This is due to a pseudo-lognormal distribution of the DoS in the entire warehouse as seen in the historical data.

Thus, our empirical loss function is defined as:

$$L(\hat{F}_S, \tilde{F}_S) = \frac{1}{19}\sum_{i=1}^{19}\Big(\log(\hat{F}_S^{-1}(0.05i) + 1) - \log(\tilde{F}_S^{-1}(0.05i) + 1)\Big)^2$$

Now, we would introduce the machine learning algorithm $\tilde{g}$ to approximate F_S.

5.1 INTRODUCTION OF PARALLELNET

For the dataset introduced in Section 4, the textual description carries some of the most important information relating to the DoS distribution. The nature of a product usually determines its seasonality and largely its average value of DoS, and therefore it would be desirable to extract as much information as possible through text. In this area, there are three popular classes of architectures: convolutional neural networks (CNN), recurrent neural networks (RNN), and transformers. In particular recently transformers have gained popularity due to their performance in machine translation and generation tasks (e.g. Devlin et al. (2018), Radford et al.). However, we argue that transformers are not the appropriate model for the textual descriptions here. The words often do not form a coherent phrase so there is no need for attention, and there is a lack of long-range dependency due to the short length of the descriptions.

We then proceed to use both CNN and RNN to model the textual description. As illustrated in Section 3, word order is important in this context, and RNNs are well equipped to capture such ordering. As

[1] Academic users can currently obtain the dataset by inquiring at an email address available in the ArXiv version. It would be hosted online in the near future.

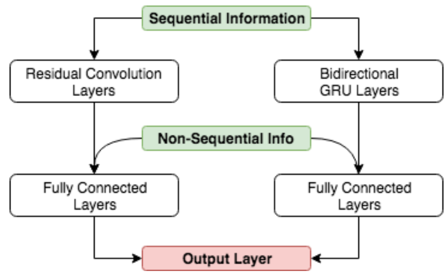

Figure 1: ParallelNet Simplified Architecture. Green boxes are inputs and the red box is the output. We separate inputs into sequential data and non-sequential data to maximally exploit different types of data.

the textual information is critical to the DoS prediction, we would supplement the RNN prediction with a CNN architecture in a parallel manner, as presented in Figure 1.

We designed the output layer as below in Figure 2 to directly predict the 19 percentile points ($\tilde{F}_S^{-1}(0.05)$, $\tilde{F}_S^{-1}(0.1)$,$\cdots$,$\tilde{F}_S^{-1}(0.95)$):

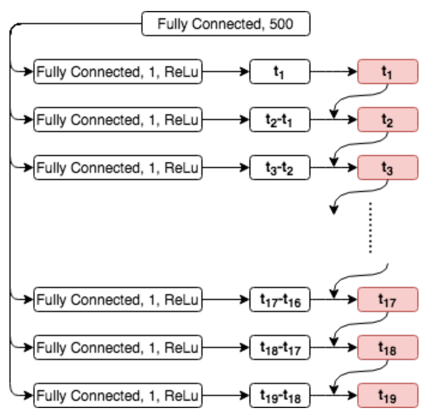

Figure 2: Output Layer. Here $t_1, \cdots t_{19}$ corresponds to the $5\%, \cdots 95\%$ percentile points.

Note that the 19 percentile points are always increasing. Thus the output is subjected to 19 separate 1-neuron dense layers with ReLu, and the output of the previous dense layer is added to the next one, creating a *residual output layer* in which each (non-negative) output from the 1-neuron dense layer is only predicting the residual increase $t_{i+1} - t_i$.

This architecture is similar to ensembling, which is well known for reducing bias in predictions (see Opitz & Maclin (1999)). However, it has the additional advantage of a final co-training output layer that allows a complex combination of the output of two models, compared to the averaging done for ensembling. This is particularly useful for our purpose of predicting 19 quantile points of a distribution, as it is likely the CNN and RNN would be better at predicting different points, and thus a simple average would not fully exploit the information contained in both of the networks. We would see in Section 6 that this allows the model to further improve its ability to predict the DoS distribution. We also further note that this is similar to the LSTM-CNN framework proposed by Donahue et al. (2014), except that the LSTM-CNN architecture stakcs the CNN and RNN in a sequential manner. We would compare with such framework in our evaluation in Section 6.

In interest of brevity, we omit the detailed architecture choice in RNN and CNN respectively and include it in Appendix 9.2. Hyperparameters are contained in Appendix 9.3.

6 Computation Results

In this section, we test the capability of ParallelNet to estimate the DoS distributions F_S on the dataset introduced in Section 4. We separate the dataset introduced in Section 4 into the following:

- Training Set: All shipments that exited the warehouse before 2017/06/30, consisting about 60% of the entire dataset.
- Testing Set: All shipments that arrived at the warehouse after 2017/06/30 and left the warehouse before 2017/07/30, consisting about 7% of the entire dataset.
- Extended Testing Set: All shipments that arrived at the warehouse after 2017/09/30 and left the warehouse before 2017/12/31, consisting about 14% of the entire dataset.

We then trained five separate neural networks to evaluate the effectiveness of ParallelNet. Specifically, we evaluated the parallel combination of CNN and RNNs against a vertical combination (introduced in Donahue et al. (2014)), a pure ensembling model, and the individual network components.

- **CNN-LSTM** This implements the neural network model introduced in Donahue et al. (2014). To ensure the best comparability, we use a ResNet convolutional neural network and a 2-layer GRU, same as ParallelNet.
- **CNN+LSTM** This implements an ensembling model of the two architectures used in ParallelNet, where the two network's final output is averaged.
- **ResNet (CNN)** We implement the ResNet arm of ParallelNet.
- **GRU** We implement the GRU arm of ParallelNet.
- **Fully-connected Network (FNN)** We implement a 3-layer fully-connected network.

All neural networks are trained on Tensorflow 1.9.0 with Adam optimizer Kingma & Ba (2014). The learning rate, decay, and the number of training epochs are 10-fold cross-validated. We used a 6-core i7-5820K, GTX1080 GPU, and 16GB RAM. The results on the Testing Set are as followed:

Architecture	**Testing Set**		**Extended Testing Set**	
	MSLE	MAPE	MSLE	MAPE
ParallelNet	0.4419	29%	0.7945	51%
CNN-LSTM	0.4812	41%	0.9021	80%
CNN+LSTM	0.5024	47%	0.9581	91%
CNN	0.6123	70%	1.0213	124%
GRU	0.5305	47%	1.1104	122%
FNN	0.8531	120%	1.0786	130%

Table 1: Table of Prediction Results for Different Machine Learning Architectures

We can see that ParallelNet comfortably outperforms other architectures. Its loss is lower than the vertical stacking CNN-LSTM, by 8%. The result of 0.4419 shows that on average, our prediction in the 19 percentiles is 44% away from the true value. We also note that its loss is about 15% less than the pure ensembling architecture, indicating that there is a large gain from the final co-training layer.

We then look at a different statistic: the Median Absolute Percentage Error (MAPE). For every percentile in every sample, the Absolute Percentage Error (APE) of the predicted number of days $\hat{T}$ and the actual number of days T is defined as:

$$\text{APE}(\hat{T}, T) = \frac{|\hat{T} - T|}{T}$$

Then MAPE is defined as the median value of the APE across all 19 percentiles and all samples in the testing set. This statistic is more robust to outliers in the data.

As seen in Table 1, ParallelNet has a MAPE of 29%. This is highly respectable given the massive innate fluctuations of a dynamic warehouse, as this means the model could predict 50% of all percentiles with an error less than 29%. The result also compares well with the other methods, as ParallelNet reduces the MAPE by 29.3% when compared to the best baseline of CNN-LSTM.

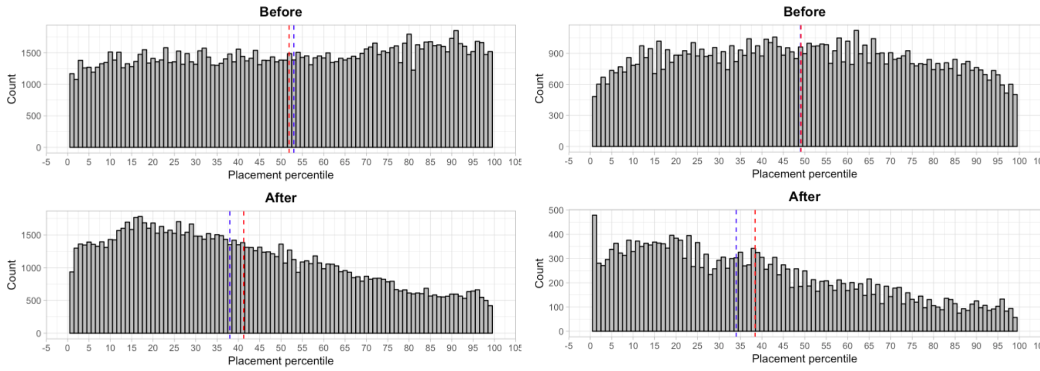

Figure 3: Placement percentile of putaway pallets before and after using a DoS system for 5 months in two selected facilities. Red line denotes the mean and blue line denotes the median.

If we look further out into the future in the Extended Testing Set, the performance of all algorithms suffer. This is expected as our information in the training set is outdated. Under this scenario, we see that ParallelNet still outperforms the other comparison algorithms by a significant amount. In fact, the difference between the pairs (CNN-LSTM, ParallelNet) and (CNN+LSTM, ParallelNet) both increase under the MSLE and MAPE metrics. This provides evidence that a parallel co-training framework like that of ParallelNet is able to generalize better. We hypothesize that this is due to the reduction in bias due to ensembling-like qualities leading to more robust answers.

7 Real-life Implementation Results

With the favorable computational results, the collaborating company is implementing the framework with ParallelNet across their warehouses, and in this section we would analyze the initial results.

The graphs in Figure 3 records the empirical distribution of the placement percentiles before and after the DoS framework was online. The placement percentile is the percentile of the distance from the staging area to the placement location. Thus, 40% means a pallet is put at the 40th percentile of the distance from staging. The distance distribution of locations is relatively flat, so this is a close proxy of driving distance between staging and storage locations, and thus time spent storing the pallets.

Ideally, according to the DoS strategy, the histogram of placement locations should roughly resemble $y = \frac{1}{x}$, as items are stored at the location proportional its length of stay (and thus inversely proportional to number of pallets sent in during a period of time). We see that such trend is observed in the 2 facilities shown in Figure 3. The slight drop in the lower percentiles is due to some locations close to the staging area reserved for packing purposes and thus not available for storage.

Specifically, Facility A had an average placement percentile of 51% before, and 41% after, while Facility B had an average placement percentile of 50% before, and 39% after. On average, we record a 10.5% drop in absolute terms or 21% in relative terms. This means that the labor time spent on storing pallets has roughly declined by 21%. An unpaired t-test on the average putaway percentile shows the change is statistically significant on the 1% level for both facilities. This provides real-life evidence that the system is able to generate real labor savings in the warehouses.

8 Conclusion and Remarks

In conclusion, we have introduced a comprehensive framework for storage assignment under an uncertain DoS. We produced an implementation of this framework using a parallel formulation of two effective neural network architectures. We showed how the parallel formulation has favorable generalization behavior and out-of-sample testing results compared with sequential stacking and ensembling. This has allowed this framework to be now implemented in live warehouses all around the country, and results show appreciable labor savings on the ground. We also release the first dataset of warehousing records to stimulate research in this central problem for storage assignment.

References

Lu Chen, Andr Langevin, and Diane Riopel. The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage. *International Journal of Production Research*, 48(4):991–1011, 2010. doi: 10.1080/00207540802506218. URL https://doi.org/10.1080/00207540802506218.

Zhuxi Chen, Xiaoping Li, and Jatinder N.D. Gupta. Sequencing the storages and retrievals for flow-rack automated storage and retrieval systems with duration-of-stay storage policy. *International Journal of Production Research*, 54(4):984–998, 2016. doi: 10.1080/00207543.2015.1035816. URL https://doi.org/10.1080/00207543.2015.1035816.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014. URL http://arxiv.org/abs/1411.4389.

Marc Goetschalckx and H. Donald Ratliff. Shared storage policies based on the duration stay of unit loads. *Management Science*, 36(9):1120–1132, 1990. doi: 10.1287/mnsc.36.9.1120. URL https://doi.org/10.1287/mnsc.36.9.1120.

Warren H. Hausman, Leroy B. Schwarz, and Stephen C. Graves. Optimal storage assignment in automatic warehousing systems. *Management Science*, 22(6):629–638, 1976. doi: 10.1287/mnsc.22.6.629. URL https://doi.org/10.1287/mnsc.22.6.629.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Sadan Kulturel, Nur E Ozdemirel, Canan Sepil, and Zafer Bozkurt. Experimental investigation of shared storage assignment policies in automated storage/retrieval systems. *IIE transactions*, 31(8): 739–749, 1999.

Charles J Malmborg. Interleaving models for the analysis of twin shuttle automated storage and retrieval systems. *International Journal of Production Research*, 38(18):4599–4610, 2000.

David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Charles G Petersen, Gerald R Aase, and Daniel R Heiser. Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, 34(7):534–544, 2004.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Meir J Rosenblatt and Amit Eynan. Notederiving the optimal boundaries for class-based automatic storage/retrieval systems. *Management Science*, 35(12):1519–1524, 1989.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Leroy B. Schwarz, Stephen C. Graves, and Warren H. Hausman. Scheduling policies for automatic warehousing systems: Simulation results. *A I I E Transactions*, 10(3):260–270, 1978. doi: 10.1080/05695557808975213. URL `https://doi.org/10.1080/05695557808975213`.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Yugang Yu and MBM De Koster. Designing an optimal turnover-based storage rack for a 3d compact automated storage and retrieval system. *International journal of production research*, 47(6):1551–1571, 2009.

Yugang Yu and René BM De Koster. On the suboptimality of full turnover-based storage. *International Journal of Production Research*, 51(6):1635–1647, 2013.

Yugang Yu, René BM de Koster, and Xiaolong Guo. Class-based storage with a finite number of items: Using more classes is not always better. *Production and operations management*, 24(8):1235–1247, 2015.

9 APPENDIX

9.1 LITERATURE REVIEW ON STORAGE ASSIGNMENT

Since Hausman et al. (1976), many different theoretical frameworks have been introduced, which can roughly be separated into two classes: dedicated storage systems and shared storage systems.

9.1.1 DEDICATED STORAGE SYSTEMS

For this class of storage systems, each product gets assigned fixed locations in the warehouse. When the product comes in, it is always assigned to one of the pre-determined locations. Under this constraint, it is optimal to dedicate positions with travel distance inversely proportional to the turnover of the product, as shown in Goetschalckx & Ratliff (1990). Turnover of a product is defined as the inverse of Cube per Order Index (COI), which is the ratio of the size of the location it needs to the frequency of retrieval needed. Heuristically, those products with the smallest surface footprint and the highest frequency should be put closest to the warehouse entry, so that those locations are maximally utilized.

9.1.2 SHARED STORAGE SYSTEMS

This class of storage systems allows multiple pallets to occupy the same position in the warehouse (at different times). It is widely considered to be superior than dedicated storage systems due to its savings on travel time and smaller need for storage space, as shown by Yu et al. (2015), Malmborg (2000). Within this category, there are mainly three strategies:

- *Turnover (Cube-per-Order, COI) Based*: Products coming into the warehouse are assigned locations so that the resultant travel distance is inversely proportional to the turnover of the product. Examples of such work includes Hausman et al. (1976), Yu & De Koster (2009), and Yu & De Koster (2013).

- *Class Based*: Products are first separated into k classes, with each class assigned a dedicated area of storage. The most popular type of class assignment is called ABC assignment, which divides products into three classes based on their turnover within the warehouse. Then within each class, a separate system is used to sort the pallets (usually random or turnover-based). It was introduced by Hausman et al. (1976), and he showed that a simple framework saves on average $20 - 25\%$ of time compared to the dedicated storage policy in simulation. Implementation and further work in this area include Rosenblatt & Eynan (1989), Schwarz et al. (1978) and Petersen et al. (2004).

- *Duration-of-Stay (DoS) Based*: Individual products are assigned locations with travel distance proportional to the duration of stay. Goetschalckx & Ratliff (1990) proved that if DoS is known in advance and the warehouse is completely balanced in input/output, then the DoS policy is theoretically optimal. The main work in this area was pioneered by Goetschalckx & Ratliff (1990). Recently, Chen et al. (2016) and Chen et al. (2010) reformulated the DoS-based assignment problem as a mixed-integer optimization problem in an automated warehouse under different configurations. Both papers assume that the DoS is known exactly ahead of time.

9.2 ARCHITECTURE CHOICE

9.2.1 BIDIRECTIONAL GRU LAYERS

Gated Recurrent Units, introduced by Cho et al. (2014), are a particular implementation of RNN intended to capture long-range pattern information. In the proposed system, we further integrate bi-directionality, as detailed in Schuster & Paliwal (1997), to improve feature extraction by training the sequence both from the start to end and in reverse.

The use of GRU rather than LSTM is intentional. Empirically GRU showed better convergence properties, which has also been observed by Chung et al. (2014), and better stability when combined with the convolutional neural network.

9.2.2 RESNET CONVOLUTIONAL LAYERS

In a convolutional layer, many independent filters are used to find favorable combinations of features that leads to higher predictive power. Further to that, they are passed to random dropout layers introduced in Srivastava et al. (2014) to reduce over-fitting and improve generalization. Dropout layers randomly change some outputs in $c_0, \cdots c_i$ to zero to ignore the effect of the network at some nodes, reducing the effect of over-fitting.

Repeated blocks of convolution layers and random dropout layers are used to formulate a deep convolution network to increase generalization capabilities of the network.

However, a common problem with deep convolutional neural networks is the degradation of training accuracy with increasing number of layers, even though theoretically a deeper network should perform at least as well as a shallow one. To prevent such issues, we introduce skip connections in the convolution layers, introduced by Residual Networks in He et al. (2015). The residual network introduces identity connections between far-away layers. This effectively allows the neural network to map a residual mapping into the layers in between, which is empirically shown to be easier to train.

9.3 Detailed Settings of Implemented Architecture

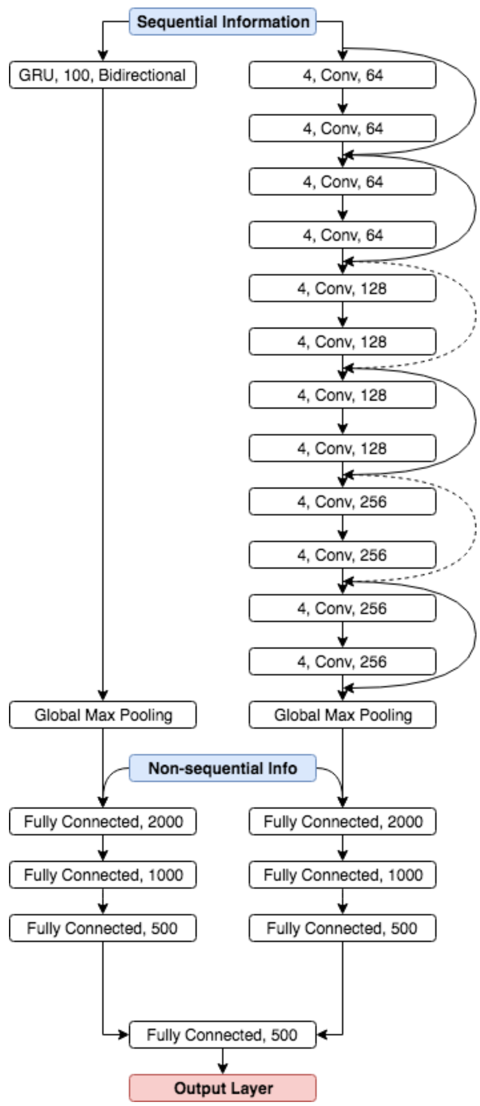

Figure 4: ParallelNet Architecture