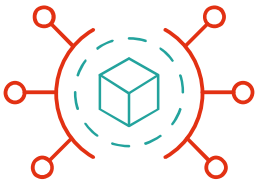


New ICT infrastructure and reference architecture to support Operations in future PI Logistics NETworks

D2.14 Intelligent Optimization of PI-Containers and PI-Means in PI-Nodes (Final)

Document Summary Information

Grant Agreement No	769119	**Acronym**	ICONET
Full Title	New **ICT** infrastructure and reference architecture to support **O**perations in future PI Logistics **NET**works		
Start Date	01/09/2018	**Duration**	30 months
Project URL	https://www.iconetproject.eu/		
Deliverable	D2.14 Intelligent Optimization of PI-Containers and PI-Means in PI-Nodes (v3)		
Work Package	WP2		
Contractual due date	30/9/2020 (as per AMD)	**Actual submission date**	30/9/2020
Nature	Other	**Dissemination Level**	Public
Lead Beneficiary	IBM		
Responsible Authors	Gabriele Ranco (IBM), Kieran Flynn (IBM), Gordon Doyle (IBM), Faisal Ghaffar (IBM)		
Contributions from	CLMS, ELU		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 769119.

Disclaimer

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.
While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the ICONET consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.
Neither the ICONET Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.
Without derogating from the generality of the foregoing neither the ICONET Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© ICONET Consortium, 2018-2021. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

Table of Contents

© ICONET, 2020

List of Figures

List of Tables

© ICONET, 2020

Glossary of terms and abbreviations used

Abbreviation / Term	Description
API	Application Programming Interface
BPP	Bin Packing Problem
SBPP	Single Bin Packing Problem
GRPP	Guillotine Rectangular Packing Problems
SPP	Strip Packing Problem
DI	Digital Internet
NFP	Network Flow Problem
NP	Non-Polynomial
GA	Grant Agreement
Intra/Inter-AS Goods	Intra/Inter-Autonomous Systems Goods
IoT	Internet of Things
OSI Model	Open Systems Interconnection Model
OLI Model	Open Logistics Interconnection Model
NOLI Model	New Open Logistics Interconnection Model
PI	Physical Internet
PoA	Port of Antwerp
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem Living Lab 1
SOM	Self-Organizing Map
SOFM	Self-Organising Feature Map

© ICONET, 2020

Executive Summary

As part of the research project of optimization in Physical Internet, IBM has incorporated the feedback provided by the project officer and partners into this final deliverable. We have identified several key areas that can support the work done in ICONET's Living Labs and other deliverables. This deliverable serves as an additional set of evaluations and validations of different Machine learning techniques, and demonstrates the following:

- Analysis of data from Living Labs.
- Application of different approaches to a number of Use Cases in the Living Labs.
- that the proposed solutions use the most up-to-date and modern technologies and approaches.
- that Machine Learning techniques are employed where possible.
- The provided data can be used to optimize the Supply Chain operations.

After the mid-term review of the project, there were new initiatives by industry partners within the ICONET project to collect the relevant datasets for the creation of machine learning models capable of providing novel insights for the optimization of operations in the supply chain. IBM has used these unique datasets to predict the evolution of product consumption as it relates to both Living Lab 3 – Sonae and Living Lab 4 - Stockbooking. Section 5 of this deliverable discusses the application of the proposed prediction models in more detail. Key objectives of our work are outlined below:

i) Helping the decentralization of order fulfilment in the ecommerce domain (LL3)
ii) Stock out prediction to increase order fulfilment efficiency (LL3)
iii) Providing relevant planning of warehouse space utilization (LL4)
iv) Increase warehouse Quality of Service by optimizing picking time of orders (LL4)
v) Inform the Routing Service to maximise centralised route and capacity planning (LL3/LL4)

In addition to the key objectives outlined above for LL3 and LL4, for LL1 which is focused in the area of loading and composition, IBM have improved upon the two-dimensional (2D) Bin Packing implementation which is discussed in the previous versions of this deliverable, D2.12 & D2.13. Section 2 of this deliverable presents a three-dimensional (3D) Bin Packing solution which will ensure all possible *bundling* scenarios are handled within the logistics supply chain.

© ICONET, 2020

1 Introduction

In this final deliverable, IBM presents a series of machine learning models, trained and validated, for the prediction of product stocks in warehouses and stores. Also, a further improvement of the Bin Packing algorithm implementation used for the optimization of operations within port yards is outlined.

A well-known goal of the Physical Internet is to create a network of shared resources and information along the entire supply chain (Fazili, 2017). This facilitates the creation of more intelligent and cognitive software solutions, capable of automatically leveraging data collected along the logistics chain, and also provides insights about the future evolution of the demand of network usage (Zhong, 2017).
A key focus of this deliverable is order fulfilment in e-commerce scenarios. Following several meetings with Living Lab partners, that operate in the field of logistics, it became evident that historical data regarding the movement of products, can be used to predict the distribution of goods in their stores.

The models presented in section 5 of this deliverable, which have been trained using Living Lab data, can lead to a de-centralisation of their activities, resulting in increased time savings and a better usage of available resources. An additional benefit, in relation to the fulfilment of food orders, is the reduction in overall food wastage which is one of the drivers of high cost, both economically and socially (Göbel, 2015).
The datasets provided by Sonae and Stockbooking represent a unique and significant amount of data for research. Access to these datasets have allowed for the application of the most up to date machine and deep learning techniques. Models like Long Short-Term Memory neural network have shown accuracy in prediction and flexibility towards the statistical properties of the dataset.

In the previous versions of this deliverable, D2.12 and D2.13, IBM focused on the optimization of PI-Movers' usage in PI-Hubs, a series of possible approaches were presented, some of which leveraged other services such as the Routing Service (D2.5 PI networking, routing, shipping and encapsulation layer algorithms and services). Subsequently IBM focused more on the implementation of the 2D Bin Packing algorithm into the PI Service stack. In this deliverable, together with the work done for the prediction models, the 2D Bin Packing implementation has been improved upon to produce a 3D Bin Packing solution which considers additional constraints effective across all bundling scenarios present in the logistics supply chain, this is discussed in more detail in section 2 of this deliverable.

© ICONET, 2020

1.1 Summary of previous versions

In deliverable D2.13 IBM identified several key focus areas:

- Acquire data from Living Labs
- Expand the number of potential Living Lab Use Cases that Optimization is applicable to
- Introduce Machine Learning
- Perform initial evaluations of machine learning models on the different datasets

After eventually acquiring datasets suitable for machine learning, IBM consulted with partners in Living Labs 1, 3 and 4 to identify existing use cases where optimisation (and specifically Machine Learning powered optimisation) could be applied. Living Lab 4, in particular, showed early promise with regards to utilizing machine learning and also the quality of the dataset provided.

Per the data available for machine learning, IBM outlined a strategy - the ensemble approach. This approach sees the use of multiple models trained from diverse datasets. The weaknesses or inaccuracies in a single model will be overcome by the strengths in another, and vice versa. By combining multiple models with a diverse set of strengths and weaknesses the overall results have a greater chance of accuracy and success.

The datasets were successfully used to evaluate machine learning algorithms that optimize warehouse and port operations by reducing the distance travelled by picking operations. A number of ML algorithms were evaluated, and it was shown how Self Organised Maps for Clustering is considered the best match for the use cases in the Living Labs.

For Living Lab 1 (Port of Antwerp), the Bin Packing algorithm was researched, as this technology is widely adopted in the logistics industry. Further research in this area was performed and a multi-dimensional Bin Packing implementation, enhanced by reinforcement learning techniques, was introduced. This allows algorithms to learn from past failures and successes without the need for pre-training or pre-processed datasets.

D2.13 also showed how reinforcement learning and neural networks can be applied in the Port of Antwerp shunting yard. To increase efficiency and clustering techniques they can be applied in the container storage yards to reduce picking operational costs.

While initial machine learning models and training performed in D2.13 showed promise, a more detailed and expansive exploration has taken place in this deliverable, D2.14.

© ICONET, 2020

2 Heuristics - PI-Node Optimization

An overarching goal of the Physical Internet is the eventual competitive collaboration between transport and logistics actors to realise more efficient supply chain operations across several transport mediums (Montreuil, 2011). As professor Franklin mentioned during his keynote speech at the ICONET project kick-off, the journey to realise the Physical Internet concept will necessitate the gradual adoption of new services, standards, protocols and concepts. One of the logical starting points to begin this journey is the Optimisation of the PI-Nodes within the PI-Networks.

In the framework of Physical Internet, the movements of goods in a PI-Node is made possible by specific units (PI-Containers) moved by specific transporters (PI-Movers), in order to ensure timely delivery of goods and to the correct destination. To achieve this, a solution is required that leverages all possible PI-Movers for all PI-Containers that have to be transported.
This scenario is very close to the Bin Packing problem, which for decades, computer scientists of all generations have tried to resolve. Due to the challenges of obtaining optimal solutions of Bin Packing problems, data scientists have proposed various approximations or heuristic algorithms. To achieve satisfactory results, heuristic algorithms have to be designed specifically for different types of problems or use cases: in particular this is true for Bin Packing of three-dimensional PI-Containers.

In the previous version of this deliverable, D2.13, IBM implemented a solution using the bidimensional Bin Packing algorithm which allows for the assignment of PI-Containers to PI-Movers using
two constraints, the *width* and *length* of the PI-Containers. The height of the PI-Container was not considered because the solution was developed in the context of LL1, where PI-Containers are containers loaded on Wagons using only the *width* and *length* constraints, the height constraint is not part of the optimisation strategy employed. However, the ultimate intention of the Optimization Service is to provide a generic solution that can be applied at scale and across all possible regulatory scenarios in the context of the logistics supply chain network. For this reason, IBM have implemented a solution that considers all three-dimensions of a PI-Containers.

The complexity of finding optimal solutions for the three-dimensional Bin Packing problem is compounded by the difficulty of providing a useful problem formulation. In order to formulate the problem, each PI-Container i in the finite set S to have three dimensions; w_i, h_i and d_i. Each identical PI-Mover $\boldsymbol{b}$ has dimensions W, H and D. The PI-Containers and PI-Movers are rectangular boxes and the three dimensions correspond to the width, height, and depth (length) values. PI-Containers are allowed to rotate orthogonally. Rotating a PI-Container simply means swapping its width (w_i), height (h_i) and depth (d_i) values around in a defined and ordered manner. Each PI-Container has 6 rectangular facets, but there are only 3 distinct facets because the opposite facets are identical. Each of the three facets can be rotated orthogonally to obtain a new configuration of the box. Each PI-Container can therefore have 6 different rotation configurations.

2.1 Problem definition

To find the solution for a PI-Mover ***b,*** assume without loss of generality that:

$$\sum_{i=1}^{b} w_i \leq W$$

$$\sum_{i=1}^{b} h_i \leq H$$

$$\sum_{i=1}^{b} d_i \leq D$$

As such, it is correct to conclude the following:

$$\sum_{i=1}^{b} w_i\, h_i\, d_i \leq WHD$$

Therefore, for each PI-Mover, intend to minimize the wasted volume given, by using the following:

$$WHD - \sum_{i=1}^{b} w_i\, h_i\, d_i$$

As indicated in previous versions of this deliverable, Bin Packing being a NP-Hard problem suggests that an exhaustive search for the optimal solution is, in general, computationally intractable, and also that there is thus no known computationally feasible optimal solution method for the problem. Other means to obtain a solution therefore need to be found. Most popular are heuristic solution methods.

2.2 Solution Methodology

In order to solve the problem definition outlined in section 2.1 above, IBM have leveraged and built upon a solution proposed by (Dube, 2006). An additional combined weight constraint of both PI-Containers and Wagons has been implemented to ensure LL1 objectives are met, which is to optimise the formation of wagons to trains.

The presented 3D Bin Packing implementation uses a heuristic approach to perform the core task of the Bin Packing problem and satisfies the following objectives:

i) Guarantees a solution to the problem
ii) Obtains a solution in a reasonable time (i.e. solution is computationally feasible to obtain)
iii) Allows for general data input
iv) Provides continuity between the solution and the problem

© ICONET, 2020

The objectives listed above provide a solid reasoning as to why a heuristic approach is appropriate. Any approach which fails to satisfy any of the above objectives will not completely meet user requirements and thus cannot be considered an optimal solution.

Failure to satisfy (i) and (ii) above would be unsatisfactory and if the solution does not satisfy (iii) then it would lose its generality and flexibility. Condition (iii) is also of importance because bins (or containers) need to be packed with items (or cargo) of *different dimensions*. Failure to satisfy (iv) would also make it difficult for users to retrieve data and test alternative solutions during the process of problem solving. Thus, failure to satisfy (iv) would mean that the system does not readily support continuity between the solution and the problem.
The two primary heuristic Bin Packing algorithms that were used in the presented solution were the *First Fit Decreasing* and the *Best Fit* (see Annex I – Pseudocode). They were chosen over other heuristic algorithms because they have a faster running time, and also produce solutions that are much closer to the optimal solution than most other heuristic algorithms. More details about the usage of the presented approach in real cases (Living Lab 1) can be found in section 6 of this deliverable – PI Web Services.

2.2.1 3D Best Fit Algorithm

The purpose of this algorithm is to optimise the volume used in a *bin*. First, a packing direction is decided. Each PI-Mover has three directions in which to pack: a width (or x) direction, a height (or y) direction, a depth (or z) direction. One PI-Mover is packed at a time. First, a ***pivot*** point is chosen. The pivot is a (x, y, z) coordinate which represents a point in a particular *3D bin* at which an attempt to pack a PI-Container will be made.

The back lower left corner of the PI-Container will be placed at the pivot. If the PI-Container cannot be packed at the pivot position, then it is rotated until it can be packed at the pivot point or until we have tried all 6 possible rotation types. If after rotating it, the PI-Container still cannot be packed at the pivot point, then we move on to packing another PI-Container and add the unpacked PI-Container to a list of PI-Containers that will be packed after an attempt to pack the remaining PI-Containers is made. The first pivot in an empty PI-Mover is always (0,0,0).

2.2.2 3D First Fit Decreasing Algorithm

To pack a PI-Container, firstly a packing direction needs to be decided. The longest side of the PI-Mover corresponds to the packing direction. Then rotate each PI-Container such that the longest side of this PI-Container is the side which is the packing direction, i.e. if we are packing by width, then we want the longest side of the PI-Container to be the container's width, so for example if the packing direction is by width and the current height of the container is longer than its width, then rotate the container. If after performing the rotation(s) the item cannot fit into the PI-Mover (i.e. one or more of the dimensions of the containers exceeds the mover's corresponding dimension), then we rotate the container until the *second* longest side of this container is the side which corresponds to the packing direction. If after performing the rotation(s) the container cannot fit into the mover, then we rotate the container until the *third* longest side of this container is the side which corresponds to the packing direction. Next sort the containers in decreasing order of width, height or depth depending on packing direction. The pseudocode for this can be seen in Annex I.

© ICONET, 2020

3 Learning based PI-Optimization

3.1 Time-Series Forecasting

IBM investigated the possibility of using historical data of activities within a PI-Node, so as to improve the efficiency of the operations in it. Considering the PI concepts presented in D1.8 – Generic PI Case Study & Associated PI-Hubs plan v2, a PI-Node continuously receives PI-Containers and manipulates them in order to send them to the final destination. In the scenarios defined in D1.8, a PI-Node only sees the demand of the nodes in the neighbourhood or local network; it has limited visibility of the demand of the terminal users, so it can use the flow of containers through the node to derive the information around the demand of the final user. This information can be used to a) forecast future demands of consumer goods and b) to cluster PI-Containers on arrival to ensure correct and optimal storage. In this deliverable, IBM introduce the concept of time series forecasting and also the usage of a neural network in order to make relevant predictions regarding stored quantities.

Timeseries forecasting is a well-known technique of econometrics and statistics: in recent years there was an increase of its application in several fields from Financial Markets (Ranco, 2016) to Urban Traffic (Guo, 2020), mostly due to a significant increase in the volume of data recorded. Timeseries forecasting provides a solid basis for planning the future evolution of systems subjected to respective studies. For this reason, a tentative step to apply timeseries forecasting techniques to the PI context has to be made, given the temporal nature of data available within the ICONET Living Labs.

The Routing Service is the first and most relevant application of time series forecasting in the PI context defined by ICONET. In deliverable D1.8, the Physical Internet concept defines a specific series of layers in the supply chain in an analogy to digital internet e.g. the OLI layer; of those layers routing is the one that can leverage insights about future distribution of goods. The routing layer, in fact, is the layer responsible for identifying the suggested route for each PI-Mover. Therefore, if a trained machine learning forecasting model is employed as part of the Optimisation Service and the gathered insights are published, the routing service will calculate the route and can take into account the most probable distribution of goods, resulting in an optimal outcome in terms of time and mover's utilization. The final result of sending the PI-Mover to the right PI-Node to collect a PI-Container is an important goal of the Physical Internet.

© ICONET, 2020

3.2 Background & State of The Art

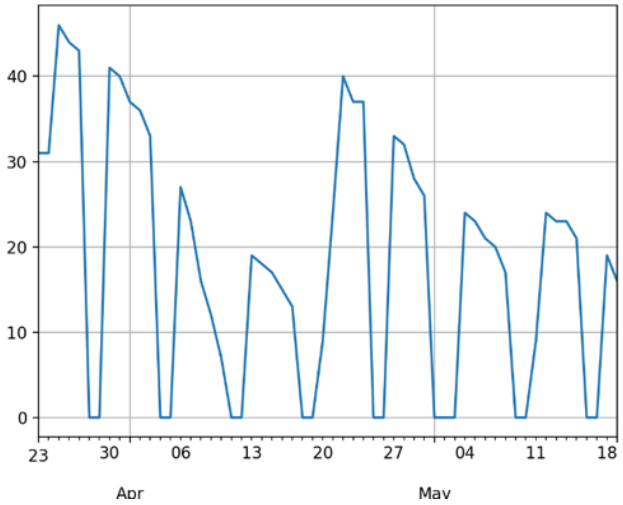


Figure 1 – Product Quantity in Storage

Figure 1 above represents the usual shape of a time series for the product quantity in storage, in a given store, at the end of each day. This information is leveraged in order to predict the future evolution of this curve. This prediction may then be used by the Routing Service to determine which store to visit to fill an order.

Per Figure 1, it is immediately evident that there are issues with some of the key features of time series forecasting in that, autocorrelation exists which violates the hypothesis under which standard econometric models can be applied and also a seasonal trend exists. As can also be seen, the mean value is changing over time. In order for a time series to be predictable, it should satisfy two conditions:

1. Low level of correlation between the numerous data points
2. The time series should be stationary

While the first condition above simply implies that the correlation between the time series and its lagged version is close to zero, the second condition is more complicated. Stationarity means that the time series should:

1. have constant mean across time
2. have constant standard deviation across time
3. not have any source of trend in it

If above conditions are met, the time series is stationary and can be forecasted. It is generally the case that these conditions are violated by real-world time series data such as the time series of daily stocks in a store. Data scientists have developed several techniques to overcome this issue (Kwon, 2007) (Halevy, 2015). Neural network models are able to make predictions without too much consideration in the stationarity of a time series.

The idea of adding time series forecasting to the Physical Internet context can be fruitful for supply chain optimization. In fact, the paper (Kantasa-Ard, 2019) proposes an approach to reduce the complexity of a Hub's connection by leveraging a predictive model of the demand. The approach is based on the concept of dynamic clustering for the retailers' demand, solving retailers to PI-hubs' clusters assignment problem,

© ICONET, 2020

and then tackling a routing problem for each cluster. The dynamic clustering is based on a forecasted demand calculated from a learning algorithm - Long Short-Term Memory (LSTM) Recurrent Neural Network. Another paper (Qiao, 2019) investigates a dynamic pricing problem for less-than-truckload (LTL) carriers during several auction periods in Physical Internet (PI), in consideration of peak demand forecasting.

3.2.1 Introduction to LSTM (Long Short-Term Memory)

Neural networks, in particular LSTM recurrent neural networks, can represent a solution to several computational problems for time series forecasting, in particular in a real deployment in which it is impossible to make all the tests necessary to understand the statistical properties of the data. See Annex II - Overview of LSTM for a brief explanation of what LSTM models are and how they are used.

3.3 Our Methodology

Our approach was to use an LSTM model to predict the next value of the stored quantity given *n* previous observations. In order to do so, we split each time series representing quantity of a product in a series of sequences. Given that there are several products, instead of training a model for each product we decided to train only one. This is possible if we normalize all the time series to be constrained between 0,1.

In order to rescale we have used the following transformation:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
where min, max = 0, 1
```

The length of the sequence was decided on according to the result of the measurement of the autocorrelation of the time series used in training. To train the model we passed several randomly sampled time series from the set of the possible products to use. For each time series, the learning process was repeated 50 times. In the next section, the training and application of the proposed models are described in more detail.

© ICONET, 2020

4 Application in LLs Perspective

In order to improve the efficiency of the Routing Service, the predictive models proposed in this deliverable are used for predicting the future quantity of stocks of a given product in a store and enable intelligent storage of goods in a warehouse. Both machine learning models are applicable in LL3 and LL4 at the Routing Service layer, as the prediction of future distribution of goods would help the routing service to identify the best route for optimal allocation of PI-Movers.

4.1 Living Lab 3 Goals

Over the course of the project, Sonae (for its food e-commerce service) have identified the following Living Lab 3 goals:

- Decentralize and optimize order initiation/preparation/fulfilment to inform the routing service
- Shorten delivery times
- Provide faster and more convenient shopping experience

The potential role of machine learning in achieving these goals is to predict future Stock Keeping Unit (SKU) Levels & Stockouts. If the routing service can consume the result of the machine learning model, it will be possible to decentralize the fulfilment of the orders received by Sonae. Knowing in advance the distribution of the products between the stores can lead to better utilization of PI-Movers, making it possible to avoid moving stocks from stores, unless an order is to be fulfilled. It can also reduce delivery time because with good planning there will be no stockouts, as the PI-Movers will be able to avoid stores that do not contain insufficient stocks to fulfil the order.

To enable training of the predictive model, Sonae has collected 56 days of storage records of around 60,000 products in 4 stores located in the city of Porto, Portugal.

4.2 Living Lab 3 Data

The records were collected only during the business day. For this reason, we re-sampled the time series to be regular only during business hours, i.e. we discarded weekends. This has reduced the length of the timeseries to 42 days. Table 1. below shows the summary statistics for each store. The values shown are computed for each time series and then the mean of them was computed.

	DAY	STOCK	STOCK	STOCK	STOCK	SKU
	count	mean	min	max	std	COUNT
STORE						
A	38.5948851	33.953754	0.68685199	56.5175478	12.9223681	48095
B	38.4471396	50.8024221	0.71729754	88.4664147	20.1113731	55079
C	38.7545982	40.4017437	0.63057038	64.389396	14.1121618	47954
D	39.3461568	31.6502373	0.27462652	51.5378535	11.7727301	26462

Table 1 - Sonae Data

4.3 Methods

As already mentioned, we decided to use Long-Short Term Memory (LSTM) neural network and compared its accuracy with respect to an Auto Regressive Integrated Moving Average model (ARIMA). This decision was made because of the high level of autocorrelation of the time series. It is possible that in the time period used to make the prediction that there is a violation of the stationarity condition as already outlined in section 3.2 above. As a result, making predictions with standard econometric model is not possible. Being stationary for a time series means in practice three things:

1. The average of the time series value cannot change during the period of analysis.
2. The standard deviation should not change during the period of analysis.
3. There should not be trends or seasonal patterns.

Autocorrelation between the various lags of the timeseries is a clear sign that the above conditions are not satisfied. In Figure 2 below, the presence of autocorrelation is evident until lag 5 for a given timeseries. After lag 10, the autocorrelation is increasing which can be a dangerous sign of seasonality. Given the nature of the data of residual products in the stores, it is not unexpected to see such seasonal fluctuations illustrated in Figure 2 below.

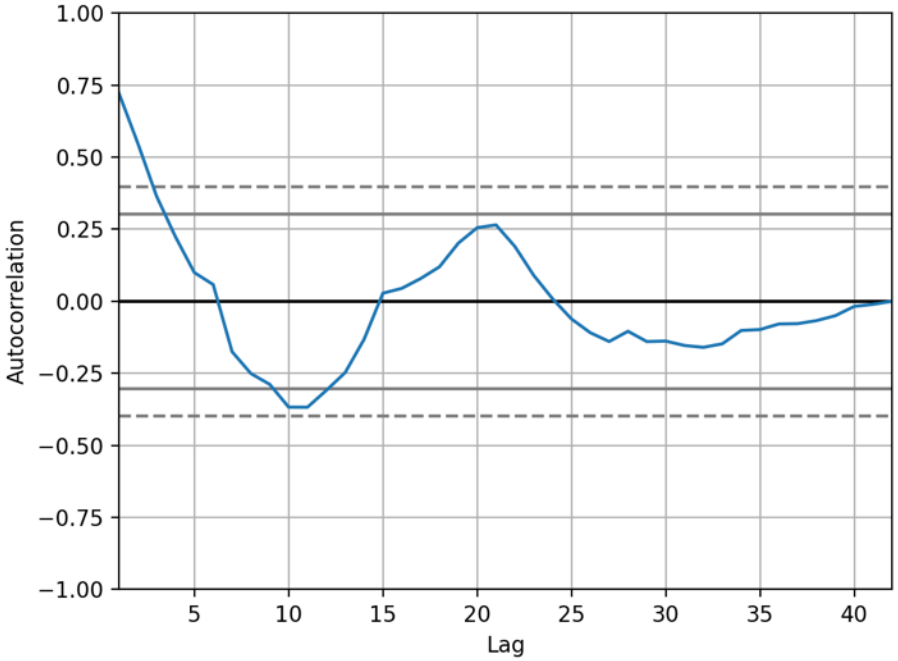


Figure 2 - Autocorrelation
Note: Lag 5 = 1 week / Lag 10 = 2 weeks and so on

Due to time constraints, stationarity for each time series was not tested however, one possible way to overcome this potential risk is to use a model, like LSTM, that is able to avoid this kind of issue. With a particular set of parameters even the ARIMA model can face the challenges of such types of time series however, due to time limitations, it is not possible to finetune an ARIMA model for each product. Understandably, creating a model for each product was almost impossible, given the number of products, but it is also not the best approach given that we will lose information provided due to the volume of data.

© ICONET, 2020

Therefore, a decision was made to train a model that will learn from each product's history. Unfortunately, the products are not all the same i.e. there are products that always have a high volume in terms of quantity. In order to address this heterogeneity, we decided to renormalize the time series according to their minimum and maximum values. Figure 3 below is a distribution of the area of the curves between (sum of all the values of each time series) in each store in Porto city.

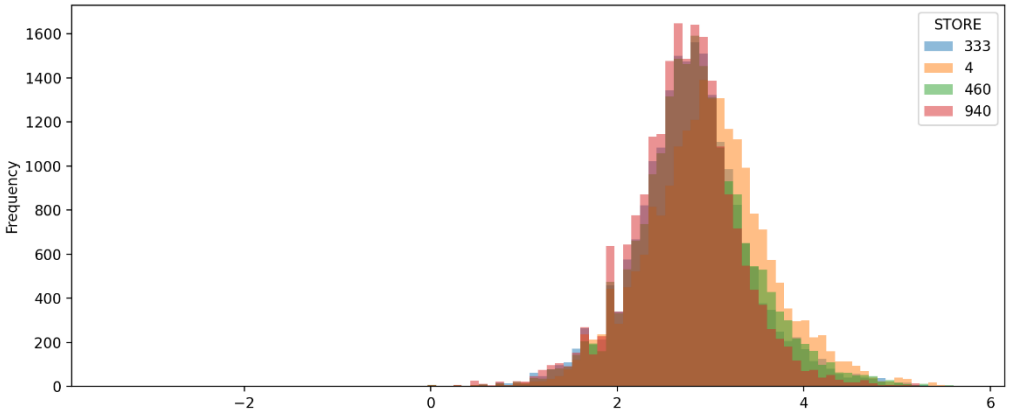


Figure 3 - Heterogeneity in the sum of the values of the timeseries for the Porto dataset

Figure 4 below provides an indicative idea of the distribution of the heterogenicity of the data because it shows a sum of the values for each product in the Living 3 dataset. It can be seen that some products are present in some stores and not in others and that more or less the distribution of the timeseries values is the same in all the stores; even if some stores seem to have more frequent residuals. These graphs enable us to understand that there are differences between products, and this heterogeneity needs to be considered.

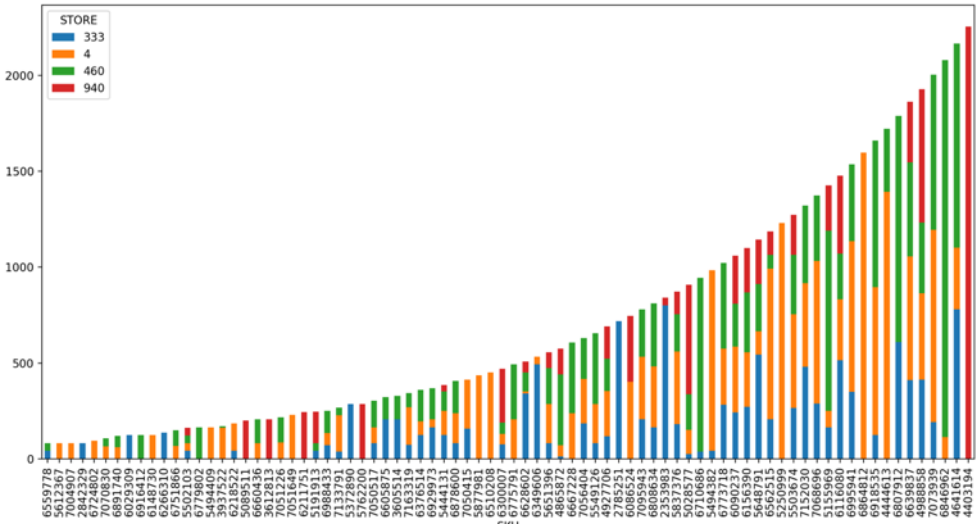


Figure 4 - Products per Store

To remove heterogeneity, and to overcome the issue of stationarity, a lag of 6 was used for our model. This decision was made after looking at the autocorrelation graph in Figure 2 above. Having taken the value of the time series in the previous 6 days to make the prediction, we decided to renormalize the value of the timeseries by the minimum and the maximum using the following approach:

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

Note: where min, max = feature_range.

This type of renormalization transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. After this renormalization was done, the model was trained as indicated in the following section 4.3.1.

4.3.1 LSTM training

As already mentioned, the neural network was trained and tested over each time series of a product. This process can be summarized as follows:

1. Randomly select a product and a store
2. Renormalize the time series
3. Take 66% of its initial length for training (this number was chosen according to existing common practices for the implementation of LSTM models)
4. Repeat the training process for 50 loops over the same time series (the number of loops was a compromise between the computational power of the GPU used and the number of time series to process)
5. Test the result over each remaining day on the sequence
6. Compute the mean square error for the time series
7. Select a new time series and the process restarts. It is important to notice that the weights of the times series were not reset, so the model will keep the information about the timeseries *j* used for training.

This process removed the necessity to keep a separate model for each product. Instead, one is kept that contains mean information about the entire dataset. Given the GPU capacity available, over 12,000 products were used to train the model (20% of the entire dataset).

4.3.2 ARIMA training

Given the unique nature of the dataset provided by Living Lab 3, it was not possible to compare our results with the examples from the existing literature. So, in order to have an estimate of the usefulness of the trained model, a decision was made to also train an ARIMA model. Given the well consolidated understanding of this econometric model, it represents a baseline for our prediction.

The training process of the ARIMA model was similar to the one described in section 4.3.1 above:

1. Randomly select a time series
2. Use 66% of its length for training and the rest for testing
3. Train the model
4. Compute the mean square errors of the predictions.

© ICONET, 2020

Given the nature of the ARIMA model, we have trained a single ARIMA for each time series. There is no concatenation here because it is not possible to keep the weights of the trained model for future training.

4.3.3 Comparison (LSTM & ARIMA models)

The standard and correct approach of a comparison of two models in data mining is to train two models over the same dataset and measure their accuracy. But given the limitations in the Statsmodels python library used for training the ARIMA model, we decided to sample some time series and compare the mean square error of the trained model of LSTM with each ARIMA model trained for each time series.

The results presented in Figure 5 below show that the mean square error of ARIMA (5a) is higher than LSTM (5b). In fact, if one looked to the final bar of the two histograms you can see that there are more time series with an error greater than 20 in the ARIMA model (5a) than the one of LSTM (5b). This is confirmed by the values shown by the ratio between the two mean square errors of the time series where it can be seen that the number of times in which ARIMA (5a) is greater than LSTM (5b) is more than the contrary. We have done this comparison over around 150 time series of products. The training of the ARIMA model is particularly slow and we could not perform the analysis over more products. In our interpretation, this indicates that concatenating the training of the time series increased the accuracy of the model in comparison to the more classical approach of ARIMA.

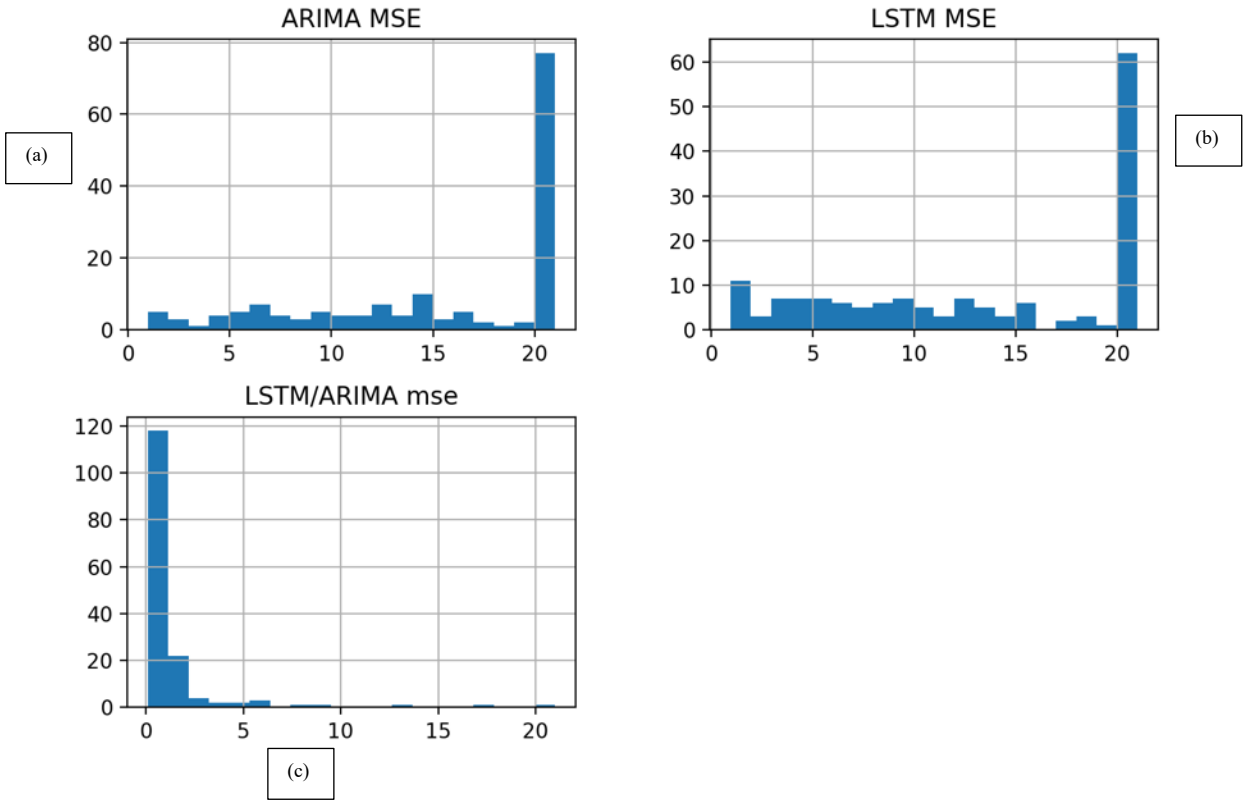


Figure 5 – Mean Square Error

(a) ARIMA Model: approx. 70 time series > 20
(b) LSTM Model: Mean Quare Error: 60 times series > 20
(c) Ratio between LSTM & ARIMA models: There are more than 100 time series with a value < 1, meaning LSTM is lower than ARIMA

4.4 Living Lab 4 Goals

Over the course of the project, Stockbooking identified the following as goals for Living Lab 4:

a. Increase warehouse Quality of Service
b. Improve the utilisation of the warehouse floor space

To order to achieve these goals, LL4 provided a dataset containing 5 years of activities conducted in one of their warehouses in France. IBM believe that prediction models can help achieve the desired goals of LL4. Effective planning of goods in arrival and departure can lead to a better usage of the available space in the warehouse and ultimately improve the quality of the services. A prediction model similar to the model trained for Living Lab 3 can be used in this context to forecast the goods in arrival and prepare in advance the space for the products. With the pontential introduction of a type of mapping system that can leverage the insights of the proposed predictive models we think this will help achieve the Living Lab goal of improving the utilisation of space within a warehouse.

Nowadays, it is possible to use methods of dimensionality reduction – such as self-organizing maps - in order to transform the representation of instances from several KPIs to specific bidimensional or three-dimensional maps. The dataset provided by Stockbooking shows a lot of similarities with the data provided by Sonae. In both cases there are several time series of movements of products. Although the context is different, it can be assumed that the statistical properties of the data are the same. This is because in both cases the goods in arrival and departure are related to the final retailer demands.

Therefore, a decision was made to apply the same type of machine learning algorithm, LSTM. We applied the addition of predicting not only the quantity in stocks but also the number of products leaving from and arriving into the warehouse and how many palettes would be involved. As a result, we predict 4 timeseries with a single model. Given its nature, the LSTM model will take into consideration any type of correlation between the 4 timeseries we are training to predict.

The normalization and the process of evaluation are the same as in the Living Lab 3 case. Following the approaches suggested in the previous version of this deliverable, we have also tried to apply self-organizing maps to this dataset. We think as interesting area for future work would be an evaluation of the clustering efficiency of self-organising maps over the forecasted time series.

Given that for this Living Lab we are going to predict a vector of 4 time series, we cannot compare the result of the prediction of LSTM with an ARIMA model. This is because ARIMA makes its forecasts only against a single value. In Econometrics domain, exists a model called Vector Autoregressive model (VAR) for these types of situations, but it will require a specific analysis of the stationarity of the time series vector. Given the training time required, we propose this as a future step. Thus, under the assumption of similarity between the two datasets, we decided to train an LSTM model, following process created for LL3.

© ICONET, 2020

4.5 Living Lab 4 Data

Stockbooking collected data of the movements of products in a warehouse from March 2015 to January 2020. We were particularly interested in the quantities and products leaving the warehouse. Table 2. below provides the description of the dataset.

Field	Description
maj	Indicates the date of movement
tmv	Indicates the type of movement. There are 5 types: arrival(EE), departure(SS), free space(LS), internal putting(DE), internal pushing(DS). In some cases, there are no records so not applicable (NA)
pbru	Indicates the weight moved
allee	Length of the warehouse
prof	Depth of the warehouse
niv	Level of the warehouse
ref	Unique id of the products
npalfm	Unique id of a palette
qte	Number of units of a palette moved.

Table 2 - Stockbooking Data Template

Table 3. below presents a summary of the key statistical features of the dataset. The dataset represents a series of records of movement of various quantities of products between palettes and outside of the warehouse. We focused our attention only on the movements of goods leaving the warehouse because we were interested in identifying the patterns in the demand of goods.

Field	Min	Max	Mean	Std. Dev	Median	Mode	Unique	Valid
maj	2015-01-06	2020-04-01	--	--	2017-08-28	2018-10-12	--	7191271
tmv	--	--	--	--	--	EE	6	7191271
pbru	0.000	1029.600	197.390	157.762	178.360	220.320	--	7191271
allee	1.000	999.000	253.922	243.926	176.000	160.000	--	7191271
prof	1.000	999.000	135.987	151.787	72.000	201.000	--	7191271
niv	1.000	10101.000	8.773	22.758	3.000	1.000	--	7191271
ref	140039.000	43932990.000	12120466.273	2184508.043	12263964.000	12310574.000	--	7191271

npalfm	--	--	--	--	--	--	--	7191271
qte	0.000	448.000	50.656	43.296	42.000	63.000	--	7191271

Table 3 - Stockbooking Dataset Statistics

Field	Outliers	Extreme	Action	Impute Missing	% Complete	Valid Records	Null Value	Empty String	White Space	Blank Value
maj	0	0	None	Never	99.999	7191271	58	0	0	0
tmv	--	--	--	Never	99.999	7191271	0	58	58	0
pbru	133796	420	None	Never	99.999	7191271	58	0	0	0
allee	1	0	None	Never	99.999	7191271	58	0	0	0
prof	58059	3503	None	Never	99.999	7191271	58	0	0	0
niv	1284	29	None	Never	99.999	7191271	58	0	0	0
ref	99657	54090	None	Never	99.999	7191271	58	0	0	0
npalfm	--	--	--	Never	99.999	7191271	0	58	58	0
qte	89855	26708	None	Never	99.999	7191271	58	0	0	0

Table 4 - Data Quality

Table 4. above shows the relative quality of the dataset. We processed this data for the purpose of having four timeseries; two for units and palettes in arrival and two for units and palette in departure. We therefore consider only 2 types of movements in respect of the 6 movement types shown in Table 5. above. This decision was made because we think that they are more closely related to the user demand.

4.6 Methods

To train the model, we focus on the movement of products, both in and out of the warehouse. Despite the dataset having granularity of days, it was decided to resample it to a weekly basis. This was to avoid: i) missing values and ii) too many days of zero activity. It is worth remembering, however, that there is not always daily movement of goods in a warehouse. Based on what we have learnt from Living Lab 3, we make the same assumption that there is heterogeneity between products in the LL4 dataset and so we normalised the time series using the same technique used in LL3. We have chosen to use a lag of 4 for the model because this represents a month of activity for the warehouse. After this pre-processing, we trained the LSTM model using the same approach as the one defined for LL3.

In this section, we present how we have used self-organising maps to cluster the products contained within the LL4 dataset. See Annex III - Overview of SOM (Self Organising Maps) for a brief explanation of what SOMs are and how they are used.

4.6.1 SOM Training

Training occurs in several steps and over many iterations:

1. Each node's weights are initialized.
2. A vector is chosen at random from the set of training data and presented to the lattice.
3. Every node is examined to calculate which one's weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).
4. The radius of the neighbourhood of the BMU is now calculated. This is a value that starts large, typically set to the 'radius' of the lattice but diminishes each time-step. Any nodes found within this radius are deemed to be inside the BMU's neighbourhood.
5. Each neighbouring node's (the nodes found in step 4) weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered.
6. Repeat step 2 for N iterations.

It was decided to leverage this technique to cluster the products in the warehouse and use the formed cluster to suggest the distribution of the products in the warehouse. For that, we created 3 different types of representations of the products available in LL4 which are listed below.

1. One in which we have considered how many units of a given product are leaving the warehouse every day with respect to the total leaving on any given day. We are therefore considering every day as a single order and studying its composition.
2. A representation of statistical properties of the time series (min, max, mean, median, length)
3. A set of normalized weekly aggregated time series.

© ICONET, 2020

5 Results & Discussion

5.1 Living Lab 3

We have considered the last 6 business days of records for each product and in order to perform an evaluation of the trained model we have considered training over each timeseries and testing over the last final part of each timeseries. This means that we created instances of the form length of time series per 6 lags. 60% of the initial length of the timeseries was in included in the training set and the rest was used for testing.

Figure 6 below shows the distribution of the mean squared error of the prediction in training and testing. It is evident that testing creates more errors of training than should be the case.

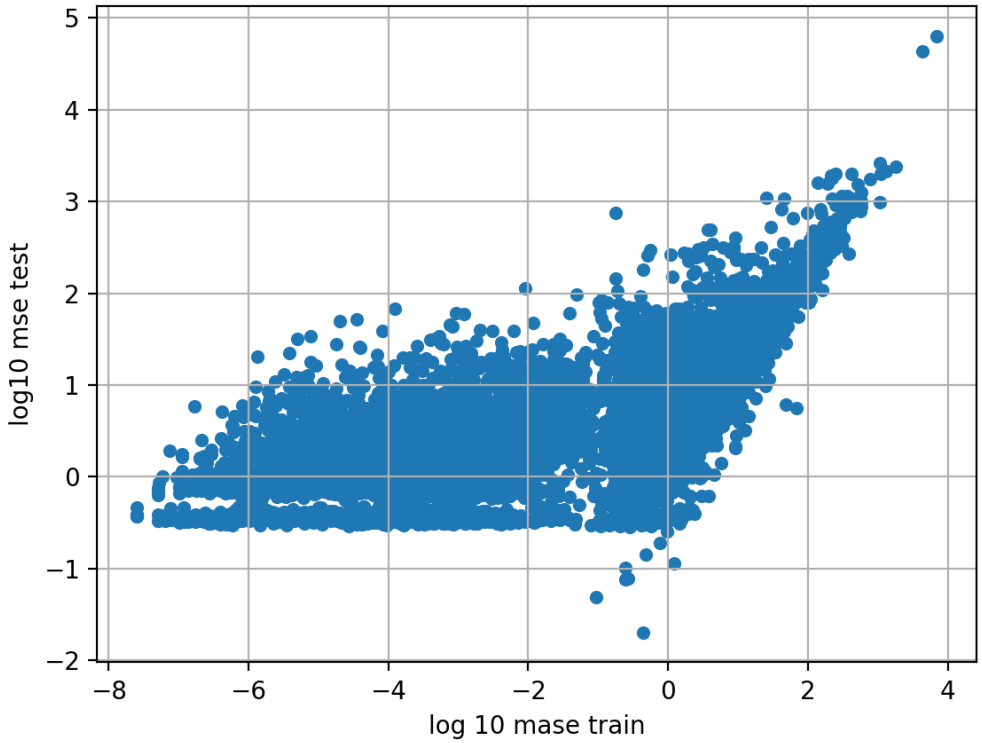


Figure 6 - Mean Square Error Scatterplot for the distribution of prediction of each time series

It is also evident however that, when viewed as an overall distribution, the model seems to be overfitted. In reality if we look at the scatter plot in Figure 6 we can see that, for the majority of the timeseries, the error is under 10 which makes it a good prediction. It is evident that we cannot impose a perfect prediction of the amount, due to the statistical nature of the prediction model. In absence of specific indication from the Living Lab, given the mean value of a time series overall and the better performance in respect of ARIMA we can be satisfied. It is also evident that for some timeseries there is a significant error both in training and testing as can be seen in Figure 7 below, but this can be improved in future work.
Another argument that we think will need more attention is the metric of evaluation. We have considered a simple mean average of the mean squared error for each prediction. It would be a good and more logical way to evaluate if the model was able to capture the future pattern.

© ICONET, 2020

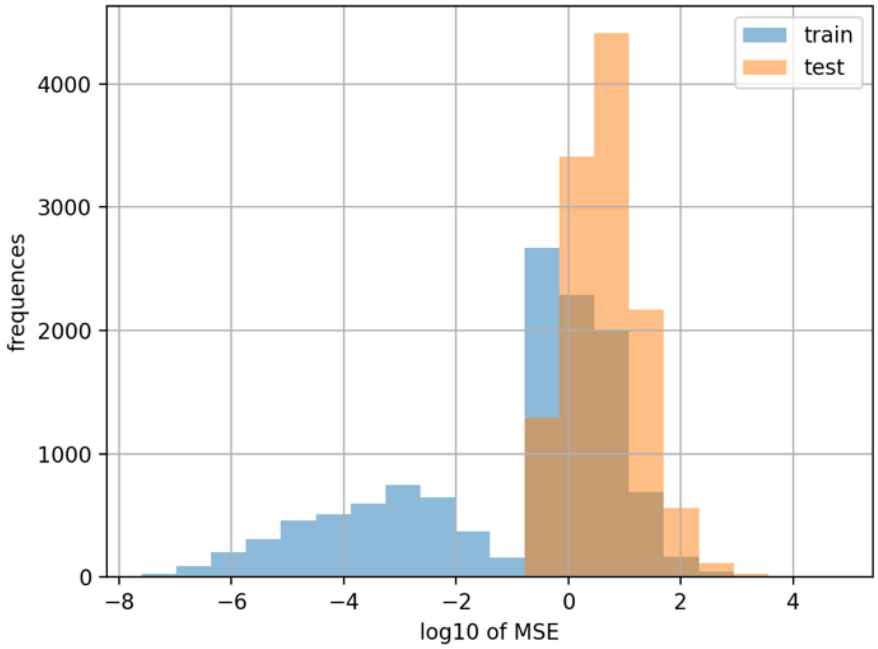


Figure 7 - Mean Squared Error - Training vs Test

5.2 Living Lab 4

In this section we show the result of two analyses of the Living Lab 4 data to increase the efficiency of storing products in the warehouse. This result is derived from the application of the self-organizing map technique to the real dataset. It may be possible to apply the same approach to the forecasted time series to predict the future distribution of the goods in the warehouse.

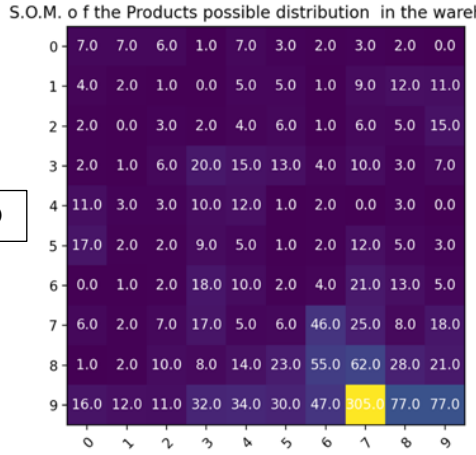
The heatmap shown in Figure 8(a) below shows the distribution of the products according to their weekly exit frequency. It has been built with data only related to a quarter of the year 2019. The heatmap figure 8(b) shows the distribution according to the quantities involved in daily departures. In other words, the timeseries were aggregated in order to identify which percentage of a given product, on a daily basis, is present. This type of analysis is similar to the analysis proposed by (Davis, 2017) where the author applies SOM to product orders, in the context of Living Lab 4 such a dataset is difficult to archive so we decided to apply the same approach but instead of the orders we have used the daily departures records for each product. It is interesting to notice that even with such a different aggregation, the SOM is still able to capture a topological structure in the dataset leading to the insights that the relation determinated by the demands of the users is present even in the daily records. We think this is a sign of the order flow i.e. the arrival of products to a warehouse. In future studies, it would be interesting to see if such a trace is present even with different datasets leading to the conclusion the it is true that we can use flow of the data trought a PI-Node to optimize operations within the node.

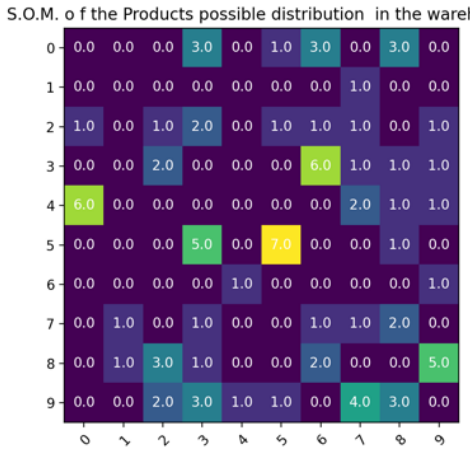
We considered an order each day and a structure can be viewed in the dataset. It is possible to use this technique to plan the zones of a warehouse or a store and if this technique is applied to the forecasted time series, it can be planned in advance.
In order to see if we can generate such a prediction from the dataset provided by Stockbooking, we trained an LSTM model similar to the one generated for Living Lab 3, with the addition that we are going to forecast 4 timeseries together. We made this choice in order to take into the consideration the level of

correlation between the four selected values: the units in departure, units in arrival and the palettes involved in arrival and in departure.
There are also several other advantages in knowing in advance the values of these time series. Given the goals of Living Lab 4 (e.g. better space utilization of the warehouse) it is evident that knowing in advance the flow of goods in arrival can lead to a better usage of the warehouses controlled by Stockbooking.

The Scatterplot in Figure 9 below shows the distribution of the errors of the forecasted time series in Living Lab 4's LSTM. In contrast to the results obtained on LL3 dataset, there is no difference in the distribution of the errors between training and testing, however the error is higher. Given that the error is computed by averaging the error of all four timeseries forecasted by the model, we do not know which one is determining the highest error. A deeper analysis of this result is required to improve the accuracy of the model for Living Lab 4.

(a)





(b)

Figure 8 – SOM Heat Maps
Left Map (a): A heat map based on weekly quanity in departure
Right Map (b): A heat map based over the daily distribution of units in departure

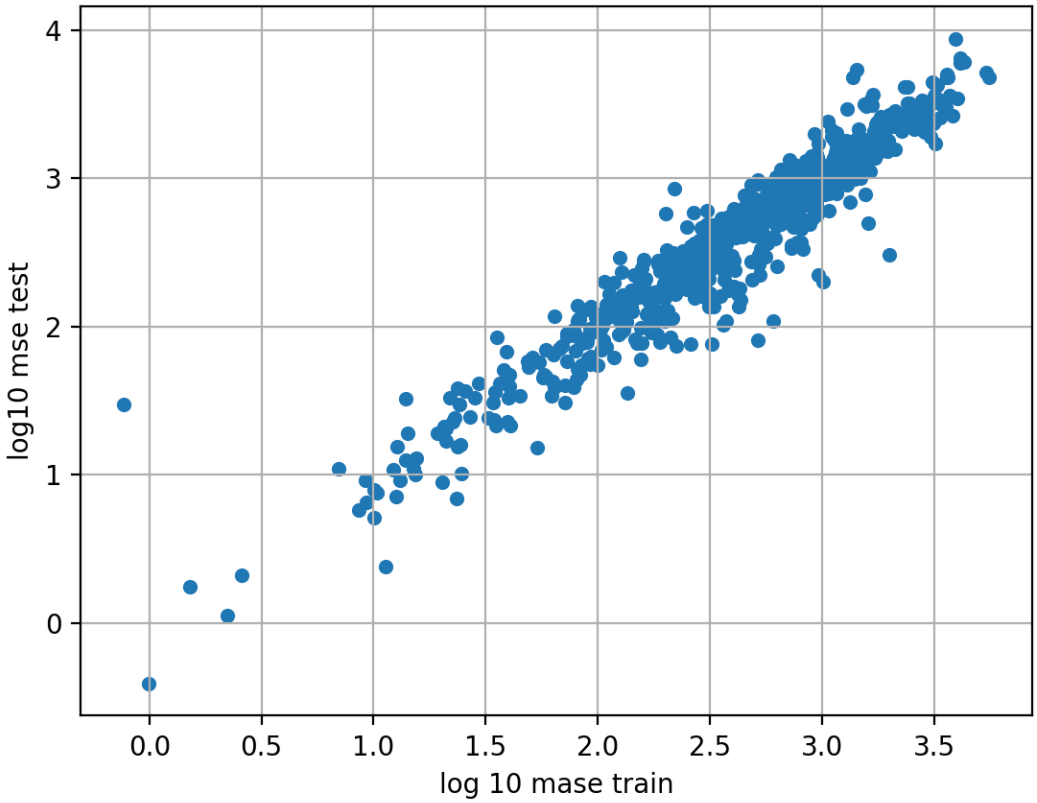


Figure 9 - Scatter Plot: Mean Square Error Training vs Test

6 Optimisation PI WebServices & Integration with other PI Services

In this section, we briefly describe the development work involved in updating the Optimisation PI Web Services described in the previous version of this deliverable, D2.13. The objective of the Optimisation Service is to allow users to create an optimised loading plan to convey PI-Containers by rail-based PI Means in an efficient manner.
The Optimisation Service retains the use of Representational State Transfer (ReST)[1] APIs, the Swagger[2] interface and the use of Docker for quick and easy deployment to cloud platforms e.g. IBM Cloud[3]. The previous version of the service consisted of a single rail wagon loading function for creating an optimised loading plan of PI-Containers in two-dimensional space (2D) i.e. loading un-stackable PI-Containers to a flat-bed rail wagon. Here we describe how the service was augmented with a function for creating an optimised loading plan of PI-Containers in three-dimensional space (3D) i.e. loading stackable PI Containers to a flat-bed rail wagon or an enclosed rail wagon. In addition to this, we describe an added train loading service that processes a rail schedule in a PI-Node and allocates incoming rail wagons to appropriate shunting yard lines for forwarding from the PI-Node.

6.1 Web Service Extension

The Optimisation Service is a ReST API, served using the Python Flask web server and packaged for deployment as a Docker image. The implementation details for this technology stack remains the same as those described in the previous version of this deliverable, D2.13. The following sections describe the restructuring of the original version of the service to accommodate the two new functions and the implementations of the 3D Wagon Loading API and the Train Loading API.

[1] Representational State Transfer
[2] Swagger
[3] IBM Cloud

© ICONET, 2020

6.1.1 Restructuring the Optimisation ReST API

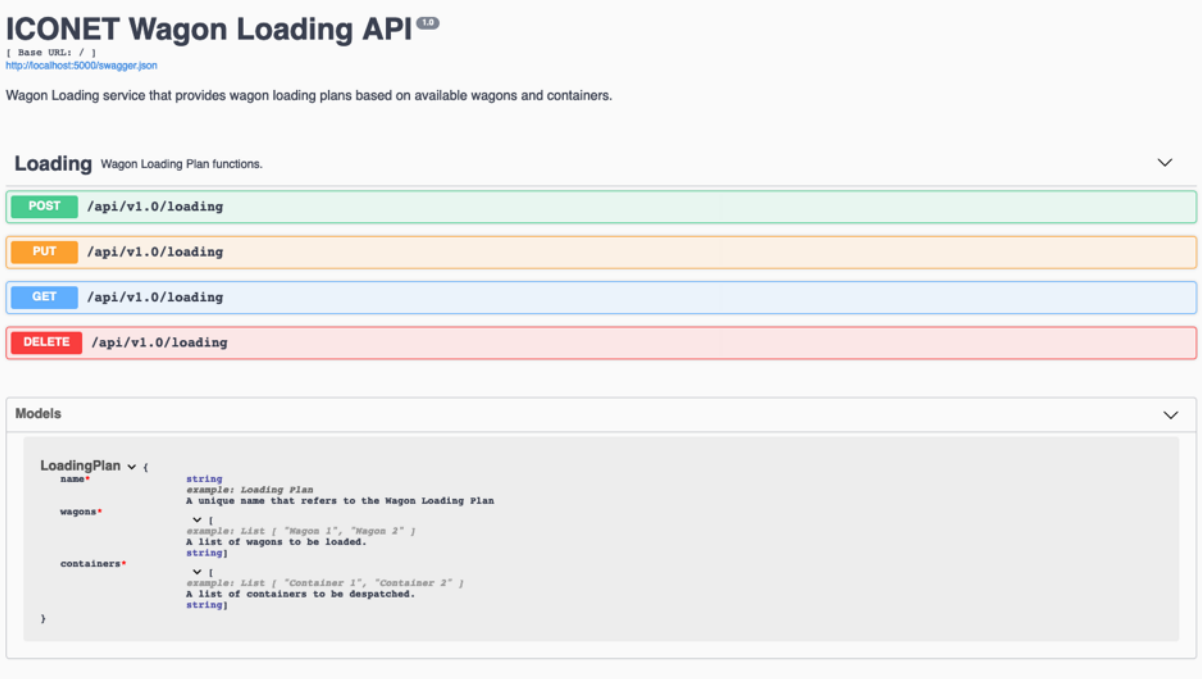


Figure 10 - The original Optimisation Service API

The original Optimisation Service was named the ICONET Wagon Loading API and consisted of a single function, called Loading. This Loading function ReST API was implemented with HTTP GET, DELETE, POST and PUT methods. The Loading function accepted a list of rail wagons to be loaded with PI-Containers and the list of PI-Containers to be despatched. The Swagger interface to the original Optimisation Service API can be seen in figure above. The Loading function optimises the PI-Containers in 2D space across the input list of rail wagons. Optimisation considers both the spatial restrictions of the rail wagon and the maximum weight restriction of the rail wagon. If any PI Containers are unassigned due to all available space on the input rail wagons being consumed, one or more new rail wagons with the same dimensions as the input rail wagons are created and the remaining PI Containers are assigned to them. The Loading function output consists of an optimised JSON loading plan. This loading plan lists the rail wagons with their assigned PI Containers. A sample of the input and output JSON formats for this API are shown below in 11 below.

© ICONET, 2020

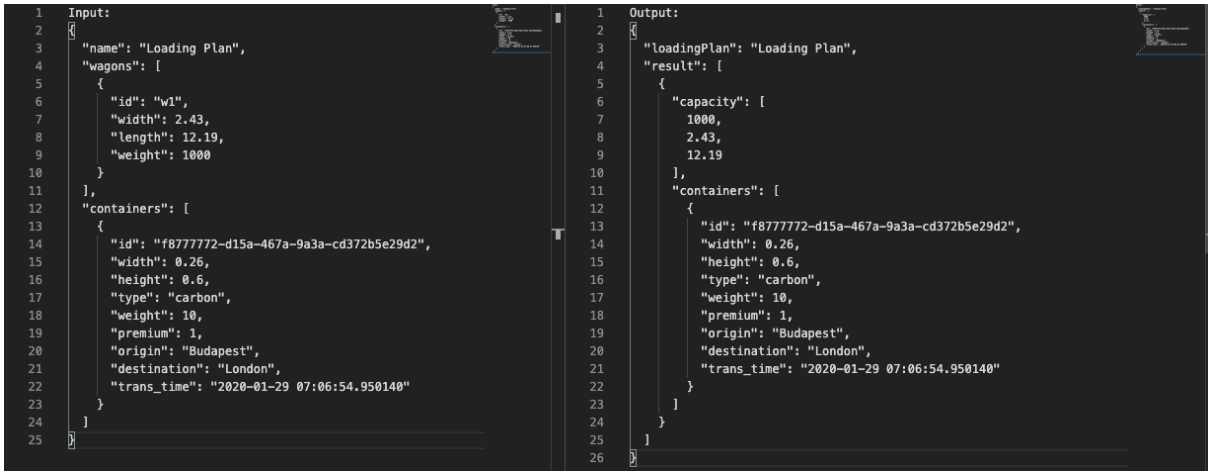


Figure 11 - Loading function API inputs and output

Functionally, the Loading API does not need to be changed for this new version of the Optimisation Service. However, as we are extending the scope of the service to include a 3D version of the rail wagon loading function and the train loading function, we have re-organised the API naming and structure.

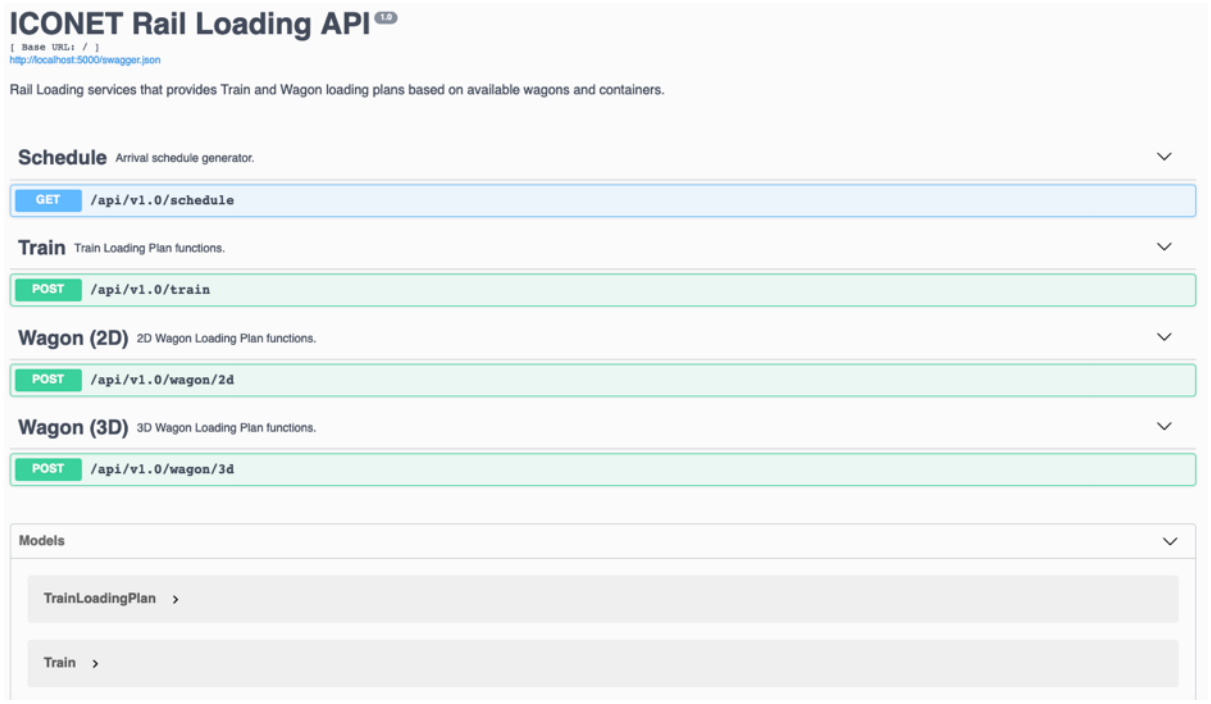


Figure 12 - The ICONET Rail Loading Swagger interface

The previous Loading function API has now been renamed the Wagon (2D) function API. This newly named version operates in the same way as before: accepting lists of rail wagons to be loaded and the PI Containers to be despatched and producing an optimised loading plan for 2D space. As the focus for the Optimisation Service was centred on producing optimised loading plans for rail wagons or trains it was decided that the additional HTTP methods (GET, DELETE, PUT) that may have been used for management operations were no longer needed. The API functions now employ HTTP POST methods only. The Optimisation Service naming has been updated to include the expanded scope of optimising

train loading for shunting yard lines and there are new API functions for producing optimised rail wagon loading plans in 3D space. The following sections describe these new functions in more detail.

6.1.2 Rail wagon loading in Three Dimensions

The first addition to the Optimisation Service for this deliverable, D2.14, is the Wagon (3D) function. PI-Containers that are to be despatched from a PI-Node may be stackable. This means that PI-Containers could be loaded onto a flatbed, or enclosed, rail wagon in 3D space. The Wagon (3D) function API was implemented in a similar style to the Wagon (2D) function API to ensure that usage of the Optimisation Service is consistent for users. The Wagon (3D) function API accepts a list of rail wagons to be loaded and a list of PI-Containers to be despatched as input. The input JSON format is the same as that of the 2D version but with the addition of the third dimension (height) for both rail wagons and PI Containers. The three-dimensional PI-Containers are optimised across the input list of rail wagons. Optimisation considers both the spatial restrictions of the rail wagon and the maximum weight restriction of the rail wagon. If there are unassigned PI-Containers after the input rail wagons space has been consumed, one or more new rail wagons with the same dimensions as the input rail wagons are created and the remaining PI-Containers are assigned.

The optimisation of the PI-Containers in 3D space is implemented using a Python package called py3dbp[4]. This package is distributed as open source software and it implements optimisation in 3D space based on the techniques described in the research paper "Optimizing Three-Dimensional Bin Packing through Simulation"[5] by Erick Dube & Leon Kanavathy (2006). Py3dbp employs heuristic bin packing algorithms to perform the optimisation of the loading arrangement. This package was integrated into the Wagon (3D) function API Python implementation and the input JSON rail wagon list and PI Containers list are adapted to the specification of the py3dbp interface.
The Wagon (3D) function API produces an optimised loading plan in a JSON format similar to that of Wagon (2D), again, with the added third dimension (height) for both rail wagons and PI Containers. A sample of the input and output JSON formats for this API are shown below in Figure 13.

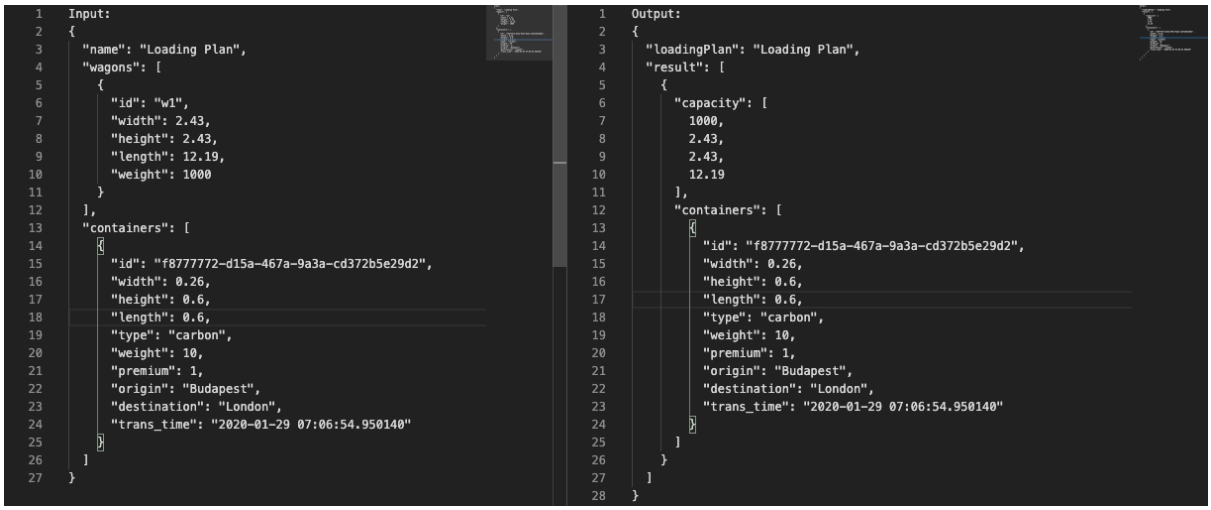


Figure 13 - Wagon (3D) JSON input and output

[4] Py3dbp
[5] Optimizing Three-Dimensional Bin Packing through Simulation

6.1.3 Train loading

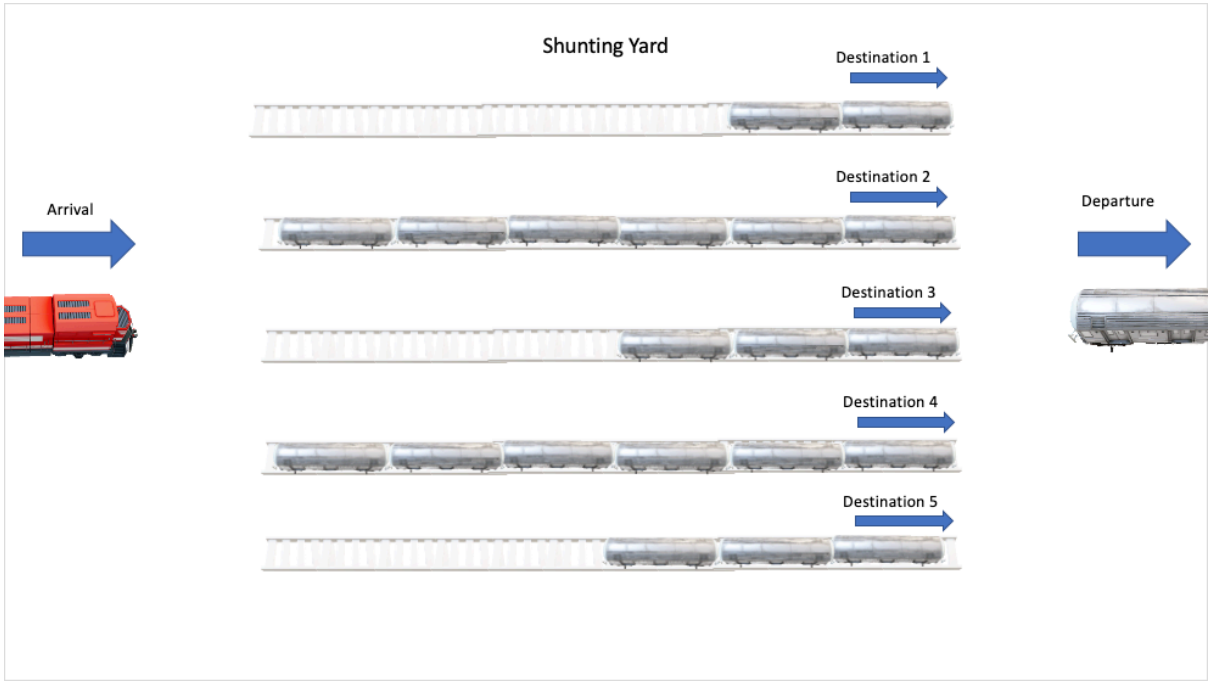


Figure 14 - Wagon (3D) JSON input and output

The second addition to the Optimisation Service is the Train loading function API. This function consists of two new APIs, Train and Schedule (see Figure 12), and aims to assist users with rail shunting yard operations in PI-Nodes. PI-Containers to be despatched may be loaded to rail wagons as part of the rail shunting yard process. As shown in 14, Shunting yards consist of several parallel rail lines in a yard that are used to sort the incoming trains and wagons for loading and forwarding to their ultimate destination. The process of sorting incoming trains at the shunting yard is driven by the arrival schedule for the PI-Node. The Optimisation Service provides a Schedule generator function API. This API employs the HTTP GET method and produces a randomly generated train arrival schedule with the following characteristics:

- Daily arrival period (24hrs)
- Segmented by hour with a maximum of 6 train arrivals in 1 hour
- Trains have a randomly assigned number of wagons (maximum 15)
- Wagon destinations are randomly assigned from a list of EU cities

The generated schedule is output in a JSON format as a list of trains with associated wagons and sorted by arrival time in ascending order. The Schedule API is provided as a convenience feature for testing the Train loading API.
The Train loading function API accepts the arrival schedule JSON as input along with the number of available shunting lines. The API produces a rail wagon and shunting line allocation table in a JSON format. This output shows the user how to allocate rail wagons to the shunting lines as they arrive.
We believe adding this functionality to the Optimisation Service complements the rail wagon loading functions, Wagon (2D) and Wagon (3D), by generating loading plans for the end-to-end rail loading process in the PI Node.

© ICONET, 2020

6.2 Integration with other Services

Under the paradigm of PI, centralized activities of a supply chain network may become autonomous but will continue cooperating with each other to reach a global optimization goal. The scope of optimisation service is to optimize the operations within the PI-Hub. In the previous sections, we highlighted how optimization service works, and in this section, we provide details on how the results of the optimization service are consumed by other service or vice versa. We show the interactions between optimization and other services in Figure 15.

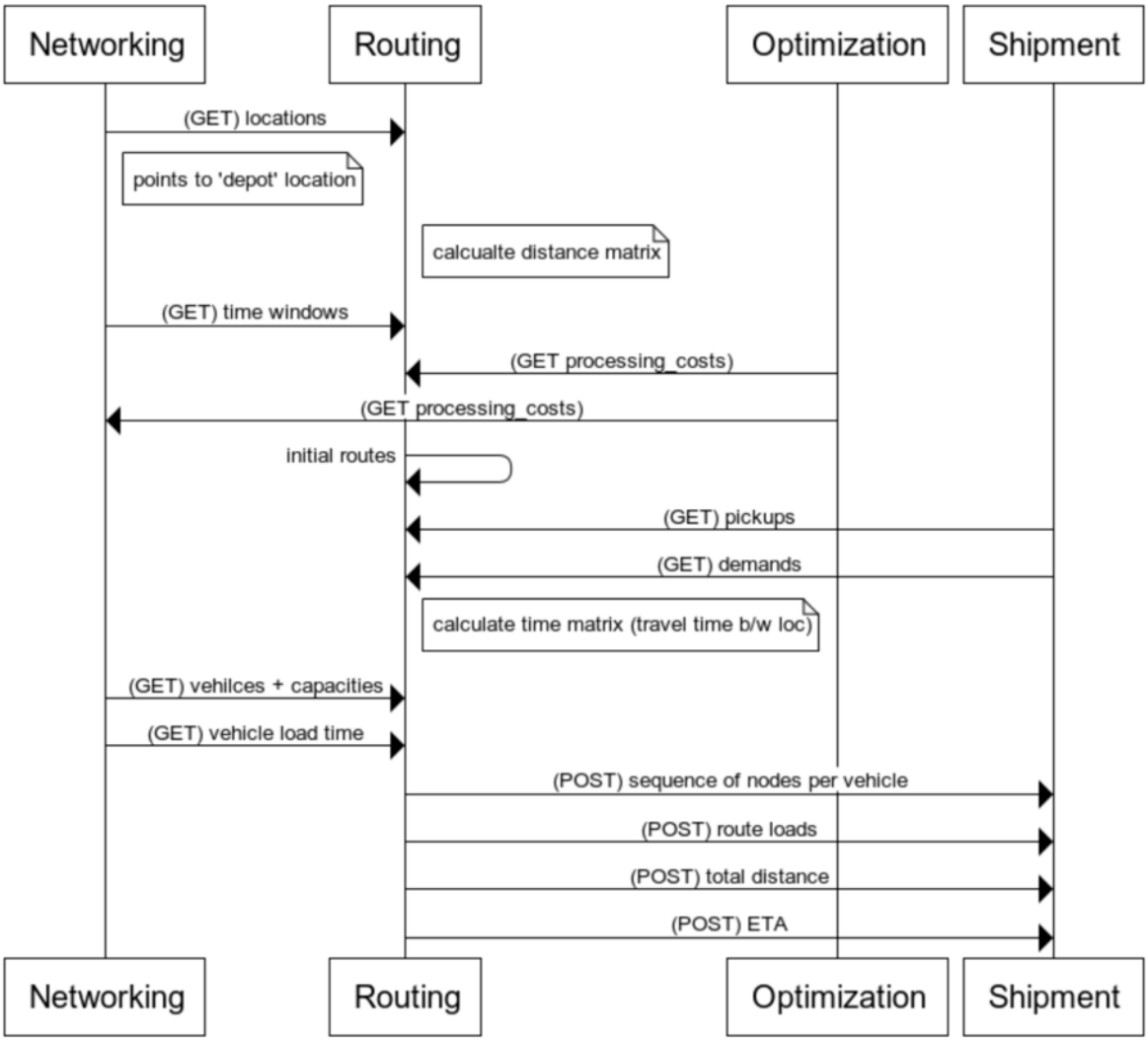


Figure 15 - Interaction between optimization and other services

6.3 Integration with Optimization and Routing

One straight-forward consumer of the optimisation service's results is the routing service. As described in deliverables (D2.4 & D2.5), the routing service determines the optimal route from a source to a destination through some intermediary hubs. Routing service is formulated Travelling Salesman Problem (TSP) with constraints guided by LLs. One of the constraints that the routing service considers is the cost of processing PI-Containers at the intermediary hub. For the LL3 and LL4, the cost of processing at the PI-Hub is represented as availability of product stock and the pallet storage respectively.

For example, in the network shown in Figure 16, there are link costs as well as the processing costs for each PI-hub (shown in green). Now if routing service is required between nodes A and Z, when finding the path, the total cost is calculated as: $cost = link\_cost + processing\_cost$, where processing cost is coming from optimization service. In the LL3, this cost represents the predicted stock per store. Based on this predicted stock, routing service decides on order fulfilment strategy by routing the vehicle to pick up items from stores which have the available stock considering the delivery date of the order.

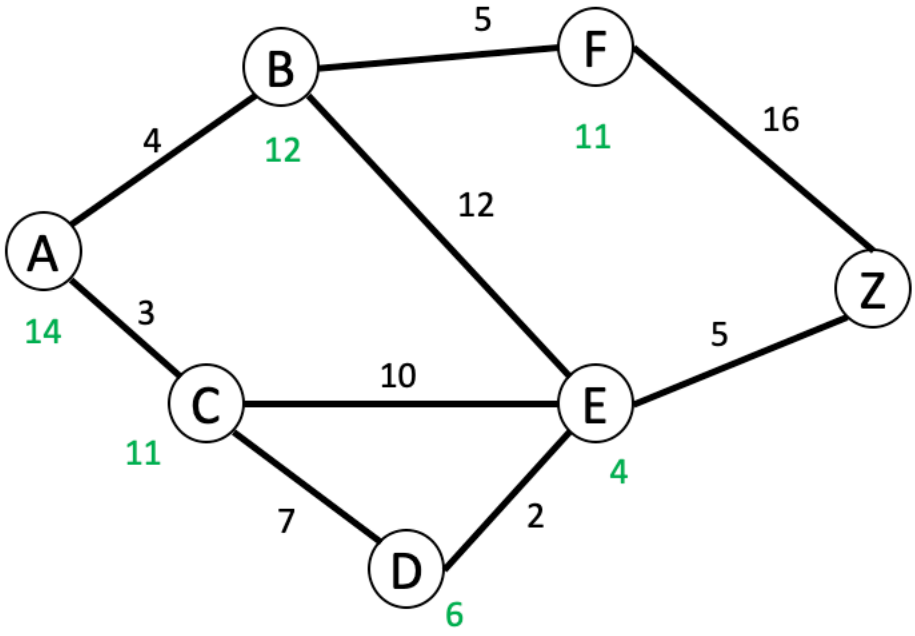


Figure 16 - Example network

7 Conclusion

After months of intense work and collaboration with key operators in the logistics chain, we concluded that the Physical Internet could drive significant global change in the delivery of goods. It is also envisaged that research will continue into how to optimize these services along a decentralized stack, such as the digital internet; we have also seen evidence that modern machine learning techniques can considerably help towards this direction.

We have shown that the LSTM network has the power to predict better future evolution of stocks rather than the classic econometrics model. It is also clear that the volume of data allowed for the creation of accurate, deployable models for forecasting, as well as heuristic algorithms, can be a simple yet extremely efficient solution to an irreducible issue of computation for rail operations.

The creation of this community of operators, those in the logistics field and those like IBM in software engineering, is one of the key elements that will allow us to produce more advanced solutions in future. The level of accuracy shown by our LSTM model and its persistent superiority in respect of the standard econometric approach not only represent hope for the future but also continue to show the incredible power of machine learning solutions. With the increased effort demonstrated by IBM in its journey in cloud services as well as the level of knowledge acquired during this project, we will able to deploy more accurate solutions. Leveraging the power of edge computing could feasibly result in increased speed and also real time optimisation of operations.

We have demonstrated how our solutions presented in this deliverable can bring value to the Living Labs in the ICONET project. In the case of LL1, our partner PoA, the multi-dimensional heuristic algorithm presented and implemented, is a simple and efficient solution. If offers an increase in number of transports per day and an increase in the amount of units loaded per train. This is a direct consequence of the full utilization of the wagons in the rail network. If all possible containers are loaded on a train as it departs, this meana there will be more slots available for other movements, something that the random association performed nowadays is not doing due to the backlog of containers to be moved.

In Living Lab 4, our partner Sockbooking has seen the value of their dataset in the application of self-organized maps and LSTM neural network: as our analysis demonstrates how a better usage of warehouse floor can be archived by clustering the products more effectively. Also, we have seen that a sign of the users order flow could be seen in recorded data of activities of warehouses. Finally, we have seen how accurate, flexible and easily deployable an LSTM model can be. LL3 Sonae, would benefit from the usage of an accurate predictive model which could potentially lead to a decentralization of the operations of order fulfilment. The integration with the routing service also brings a better utilization of the Movers used within the Living Lab.

© ICONET, 2020

8 Bibliography

Davis, C. J. (2017). *Using Self-Organizing Maps to Cluster Products for Storage Assignment in a Distribution Center.* Ohio University.

Dube, E. K. (2006). Optimizing Three-Dimensional Bin Packing Through Simulation. *Sixth IASTED International Conference Modelling, Simulation, and Optimization.*

Fazili, M. V. (2017). Physical Internet, conventional and hybrid logistic systems: a routing optimisation-based comparison using the Eastern Canada road network case study. *International Journal of Production Research*, 55(9),2703-2730.

Frederick Ducatelle, J. L. (2001). Optimisation for Bin Packing and Cutting Stock Problems,. *Proceedings of the UK Workshop on Computational Intelligence.* Edinburgh.

Göbel, C. L. (2015). Cutting food waste through cooperation along the food supply chain. *Sustainability*, 7(2), 1429-1445.

Guo, Z. Z. (2020). An Online Learning Collaborative Method for Traffic Forecasting and Routing Optimization. *IEEE Transactions on Intelligent Transportation Systems.*

Halevy, Y. (2015). Time consistency: Stationarity and time invariance. *Econometrica*, 83(1), 335-352.

Hochreiter, S. &. (1997). Long Short Term Memory. *Neural Computation*, 9(8), 1735-1780.

Hu, H. e. (n.d.).

Kantasa-Ard, A. N. (2019). Dynamic Clustering of PI-Hubs Based on Forecasting Demand in Physical Internet Context. *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing* (pp. 27-99). Chan: Springer.

Kwon, H. H. (2007). Stochastic simulation model for nonstationary time series using an autoregressive wavelet decomposition: Applications to rainfall and temperature. *Water Resources Research*, 43(5).

Montreuil, B. (2011). Toward a Physical Internet: meeting the global logistics sustainability grand challenge. *Logistics Research 3.2-3*, 71-87.

Qiao, B. P. (2019). Dynamic Pricing for Carriers in Physical Internet with Peak Demand Forecasting. *IFAC-PapersOnLine*, 52(13), 1663-1668.

Ranco, G. B. (2016). Coupling news sentiment with web browsing data improves prediction of intra-day price dynamics. *PLoS one*, 11(1), e0146576.

Zhong, R. Y. (2017). Big data analytics for physical internet-based intelligent manufacturing shop floors. *nternational journal of production research.*, 55(9), 2610-2621.

9 Annex I – Pseudocode

```
if (binWidth is smaller than binHeight
and binDepth) then
{
      packByWidth=true
      packByHeight=false;
}
else if (binDepth is smaller than
binHeight and binWidth) then
```

```
{
      packByWidth=false
      packByHeight=false  //both  false
      implies pack by depth
}
else  if  (binHeight  is  smaller  than
binWidth and binDepth) then
{
      packByWidth=false
      packByHeight=true
}

notPacked=Items

[a]do
{
  toPack=notPacked
  notPacked={} //clear notPacked

  Create  a  new  bin  called  currentBin
  and check whether the item toPack[0]
  is  able  to  fit  in  this  bin  at
  position (x,y,z)=(0,0,0).
  if  toPack[0]  does  not  fit  then
  rotate  it  (over  the  six  rotation
  types)  until  it  fits  and  pack  it
  into this bin at postion (0,0,0).
  [b]for i=1 to (size of toPack-1) do
  {
   currentItem=toPack[i]
   fitted=false

   [c]for p=0 to 2 do
   {
    k=0
   [d]  while  (k  <  number  of  items  in
currentBin) and (not fitted)
    {
     binItem=k^th item in currentBin
      if (packByWidth) then
             pivot=p
      else if (packByHeight) then
        switch (p)
        {
            compute pivot p for height
        }
      else  //pack by depth
        switch (p)
        {
            compute pivot p for depth
        }

 switch (pivot)
  {
    case 0 : Choose (pivotX, pivovY,
pivotZ ) as the back lower
              right corner of binItem
             break
    case 1 : Choose (pivotX, pivovY,
pivotZ ) as the front lower
```

```
            left corner of binItem
             break
    case 2 : Choose (pivotX, pivovY,
pivotZ ) as the back Upper
             left corner of binItem
            break
  }

     if (currentItem can be packed
     in currentBin at
         position(pivotX,    pivotY
     ,pivotZ ) ) then
     {
      Pack currentItem into
     currentBin at position
      (pivotX, pivotY ,pivotZ).
      fitted=true
     }
      else
    { // try rotating item
     do
      Rotate currenItem
     while (currentItem cannot be
     packed in currentBin at

     position(pivotX,pivotY) )
             and (not all
     rotations for currentItem
     checked)

     if (currentItem can be packed
     in currentBin at
         position(pivotX, pivotY ,
     pivotZ) ) then
     {
       Pack currentItem into
     currentBin at position
       (pivotX, pivotY ,pivotZ).
       fitted=true

     }else
         Restore   currentItem   to
     its original rotation type
   }

     if (not fitted) then
       Add currentItem to the list
notPacked

    }
   }
  }

} while notPacked has at least one
Item in it (*i.e. notPacked is
                non-empty *)
```

10 Annex II – Overview of LSTM (long short-term memory)

In this annex, we seek to aid the understanding of the machine learning model used in this deliverable, by leveraging an excellent informative blog by Christopher Olah for LSTM.

Long short-term memory (LSTM) networks are a special kind of recurrent neural networks (RNNs). Recurrent neural networks address the issue of making prediction according to a sequence of inputs rather than only by looking at a single one. They are networks with loops in them, allowing information to persist.

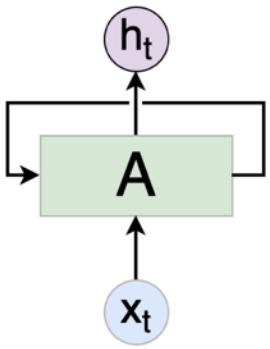


Figure 1 - Neural Network Chunk

In Figure 1, a chunk of a neural network, A, looks at some input x_t and outputs at value h_t. A loop allows information to be passed from one step of the network to the next. These loops make recurrent neural networks seem quite complex. However, upon further analysis, it turns out that they are not very different from an ordinary feedforward neural network. A recurrent neural network can be thought of as xmultiple copies of the same network, each passing a message to a successor. Consider what happens in Figure 2 when a loop is unrolled:

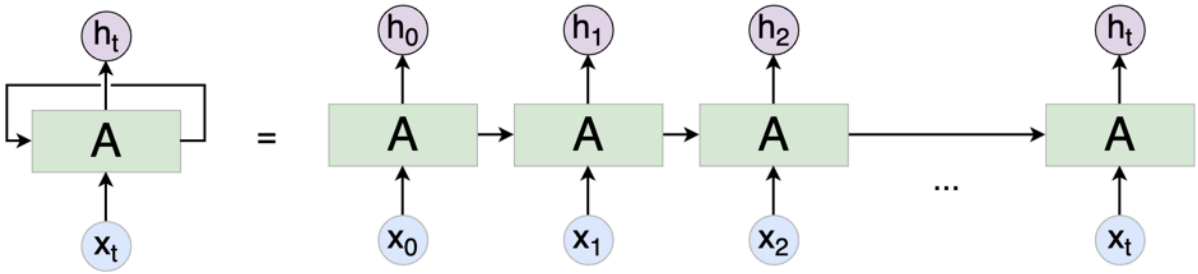


Figure 2 - Neural Network Loop

Long Short-Term Memory networks are capable of learning long-term dependencies and are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is inherent to their design. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

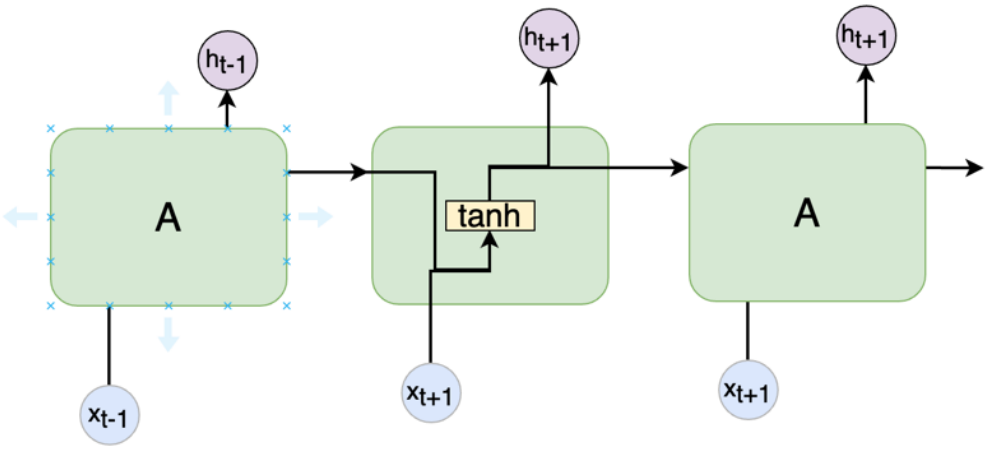
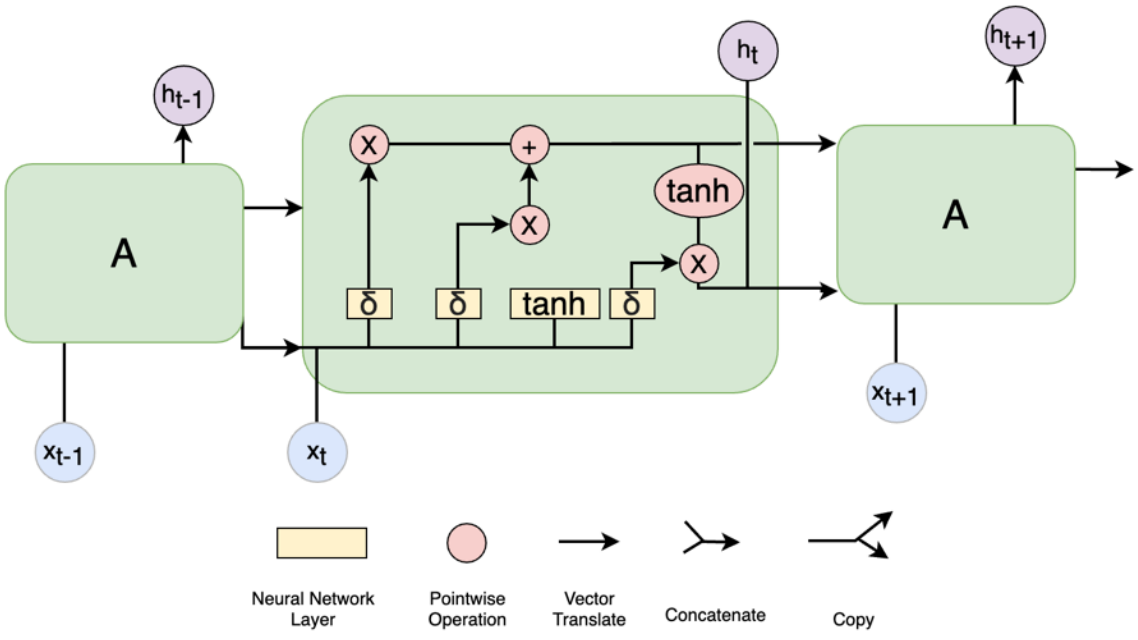


Figure 3 - Tanh Layer

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four interacting layers.



x

Figure 4 - Detailed LSTM

In Figure 4, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

11 Annex III – Overview of SOM (Self Organising Maps)

Self-Organising Feature Maps, or SOMs, provide a way of representing multidimensional data in much lower dimensional spaces - usually one or two dimensions. The process of reducing the dimensionality of vectors is essentially a data compression technique known as *vector quantisation*. In addition, the Kohonen technique creates a network that stores information in such a way that any topological relationships within the training set are maintained.

A common example used to help teach the principals behind SOMs is the mapping of colours from their three-dimensional components - red, green and blue, into two dimensions. Figure 10. below shows an example of a SOM trained to recognize the eight different colours shown on the right. The colours have been presented to the network as 3D vectors - one dimension for each of the colour components - and the network has learnt to represent them in the 2D space you can see. Notice that in addition to clustering the colours into distinct regions, regions of similar properties are usually found adjacent to each other.

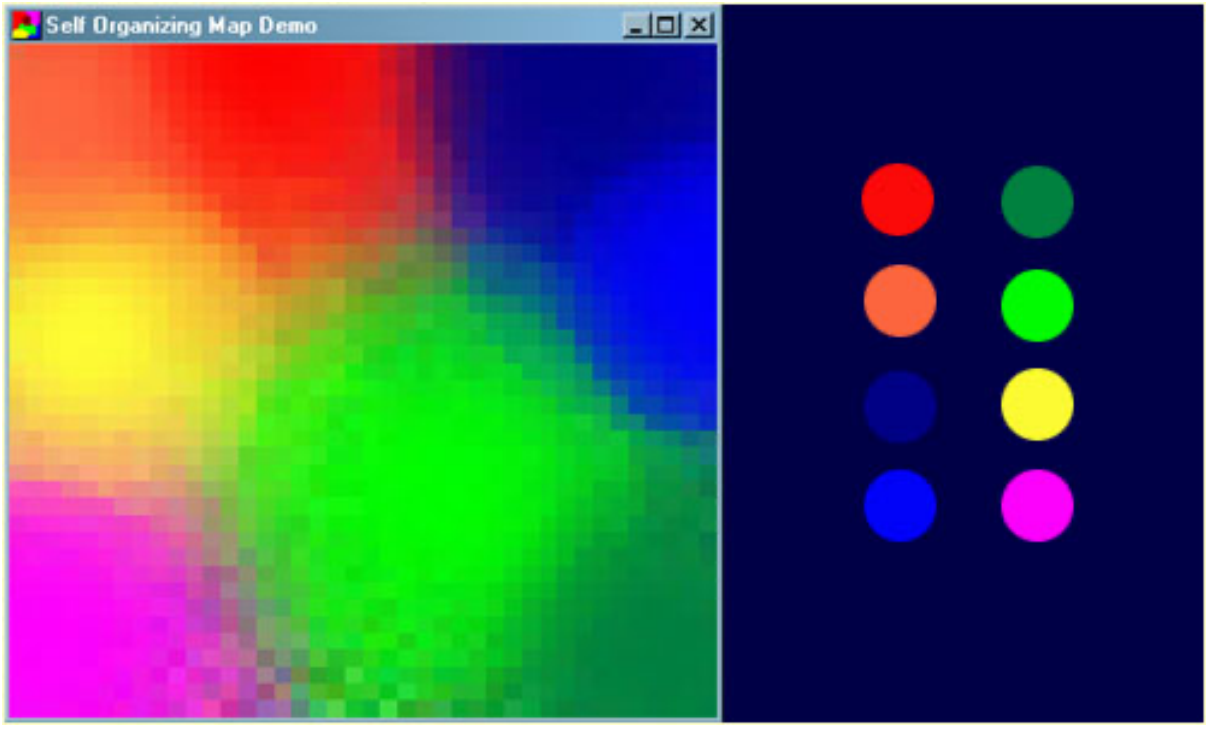


Figure 1 - Screenshot of the demo program (left) and the colours it has classified (right)

One of the most interesting aspects of SOMs is that they learn to classify data *without supervision.* Training a SOM, however, requires no target vector. A SOM learns to classify the training data without any external supervision whatsoever.

11.1.1.1 Network Architecture

For the purposes of this deliverable, the discussion centres on a two-dimensional SOM. The network is created from a 2D lattice of 'nodes', each of which is fully connected to the input layer. The figure below shows a very small Kohonen network of 4 X 4 nodes connected to the input layer (shown in green) representing a two-dimensional vector.

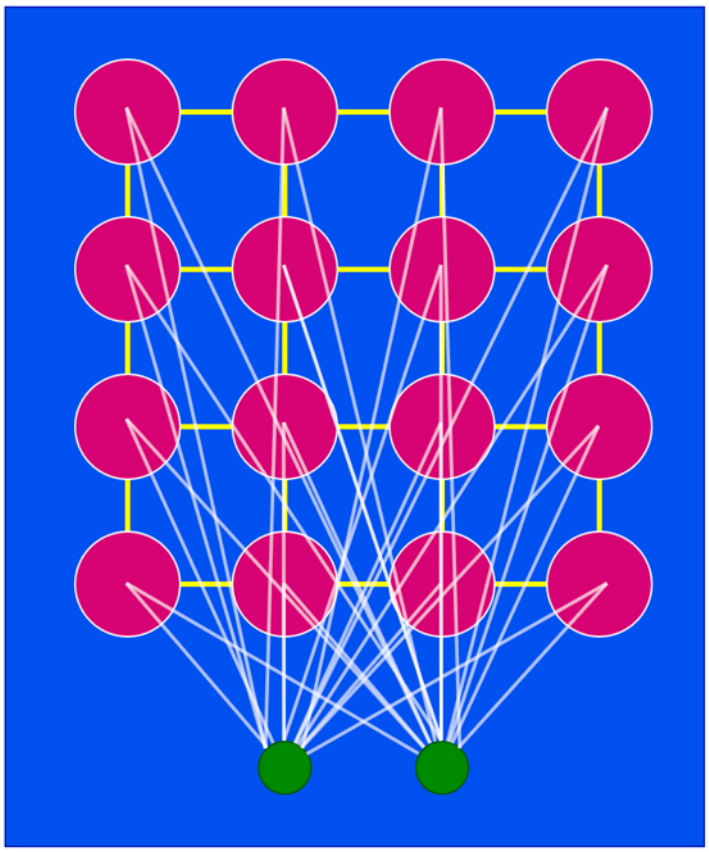


Figure 2 - A simple Kohonen network

Each node has a specific topological position (an x, y coordinate in the lattice) and contains a vector of weights of the same dimension as the input vectors. That is to say, if the training data consists of vectors, *V*, of *n* dimensions:

$$V_1, V_2, V_3 \ldots V_n$$

Then each node will contain a corresponding weight vector W, of n dimensions:

$$W_1, W_{2,3} \ldots W_n$$

The lines connecting the nodes in the above figure are only there to represent adjacency and do not signify a connection as normally indicated when discussing a neural network. There are no lateral connections between nodes within the lattice.

The SOM shown in Figure 12 below has a default lattice size of 40 X 40. Each node in the lattice has three weights, one for each element of the input vector: red, green and blue. Each node is represented by a rectangular cell when drawn to your display. Figure 12 also shows the cells rendered with black outlines so you can clearly see each node.

© ICONET, 2020

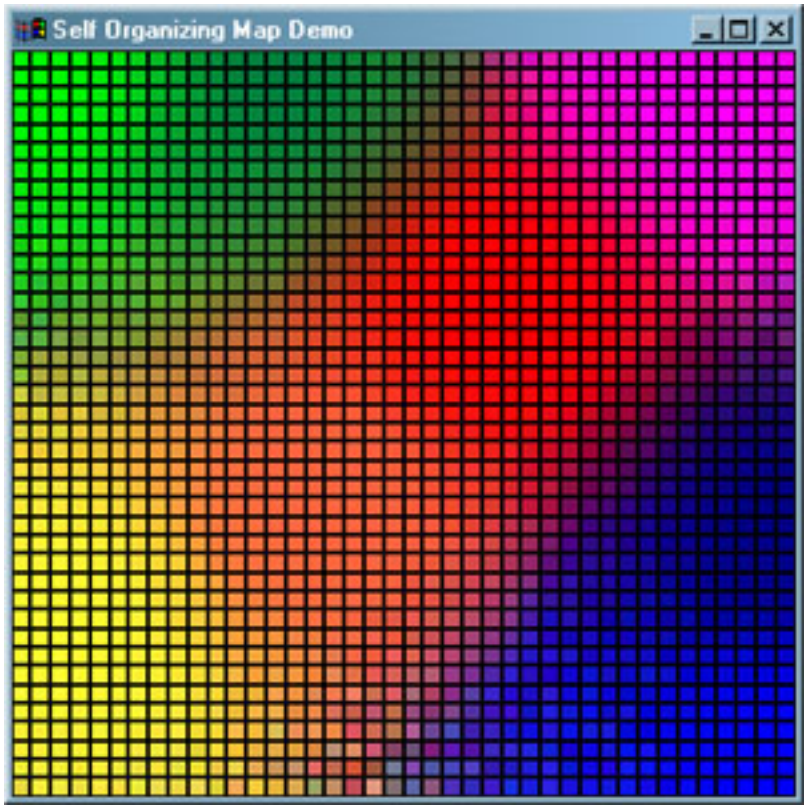


Figure 3 - Each cell represents a node in the lattice (size 40 x 40)

11.1.1.2 Learning Algorithm Overview

A SOM does not need a target output to be specified unlike many other types of network. Instead, where the node weights match the input vector, that area of the lattice is selectively optimized to more closely resemble the data for the class that the input vector is a member of. From an initial distribution of random weights, and over many iterations, the SOM eventually settles into a map of stable zones. Each zone is effectively a feature classifier, so you can think of the graphical output as a type of feature map of the input space. If you take another look at the trained network shown in Figure 12 above, the blocks of similar colour represent the individual zones. Any new, previously unseen input vectors presented to the network will stimulate nodes in the zone with similar weight vectors.