The International Journal of Advanced Manufacturing Technology (2023) 128:4169–4188
https://doi.org/10.1007/s00170-023-12123-4

ORIGINAL ARTICLE

Hierarchical ensemble deep learning for data-driven lead time prediction

Ayse Aslan[1] · Gokula Vasantha[1] · Hanane El-Raoui[2] · John Quigley[2] · Jack Hanson[3] · Jonathan Corney[3] · Andrew Sherlock[4]

Received: 8 December 2022 / Accepted: 4 August 2023 / Published online: 28 August 2023

Abstract
This paper focuses on data-driven prediction of lead times for product orders based on the real-time production state captured at the arrival instants of orders in make-to-order production environments. In particular, we consider a sophisticated manufacturing system where a large number of measurements about the production state are available (e.g. sensor data). In response to this complex prediction challenge, we present a novel ensemble hierarchical deep learning algorithm comprised of three deep neural networks. One of these networks acts as a generalist, while the other two function as specialists for different products. Hierarchical ensemble methods have previously been successfully utilised in addressing various multi-class classification problems. In this paper, we extend this approach to encompass the regression task of lead time prediction. We demonstrate the suitability of our algorithm in two separate case studies. The first case study uses one of the largest manufacturing datasets available, the Bosch production line dataset. The second case study uses synthetic datasets generated from a reliability-based model of a multi-product, make-to-order production system, inspired by the Bosch production line. In both case studies, we demonstrate that our algorithm provides high-accuracy predictions and significantly outperforms selected benchmarks including the single deep neural network. Moreover, we find that prediction accuracy is significantly higher in the synthetic dataset, which suggests that there is complexity (i.e. subtle interactions) in industrial manufacturing processes that are not easily reproduced in artificial models

Keywords Smart manufacturing · Lead time · Sensors · Deep neural networks · Hierarchical ensemble learning · Make-to-order production

1 Introduction

Lead time, the duration between initiation and completion of a production process, is a vital performance indicator in manufacturing systems. Irrespective of the product being made, engineers constantly aim to optimise lead time estimations—a factor vital for customer satisfaction and real-time decision-making—while simultaneously minimising the length of lead times to improve productivity. However, predicting lead times in complex systems, particularly make-to-order systems with diverse product demands, is challenging. Lead times are influenced by the real-time state of the production system, including machine status, which is prone to uncertainties and disruptions such as variable resource availability. Consequently, lead times can vary considerably among products.

According to [1], current manufacturing environments are too complex and dynamic for traditional analytical methods to be effective in production planning and control. While analytical modelling can provide average lead time performance, it is less effective in predicting lead times for individual product orders. However, [2] have demonstrated that data-driven predictions outperform analytical approaches in *lead time prediction*. This suggests that in order to predict order-specific lead times, systems must dynamically monitor the production state and adopt a data-driven approach. With the prevalence of sensor-rich machines and real-time data collection technology in modern smart production systems [3], *data-driven lead time prediction* is now feasible. In this paper, we propose a novel *Hierarchical Ensemble Deep Learning Algorithm* (HEDA) for data-driven lead time prediction in complex make-to-order manufacturing environments. The

✉ Ayse Aslan
a.aslan@napier.ac.uk

Extended author information available on the last page of the article

proposed HEDA utilises real-time production state data, when new orders arrive, to estimate their lead times and provide initial information to customers regarding order waiting times.

We evaluate the proposed HEDA's performance in two case studies using three prediction quality measures. The first case study utilises the Bosch production line industrial dataset, which is one of the largest manufacturing datasets available and was released as part of a Kaggle competition in 2016 (see https://www.kaggle.com/competitions/bosch-production-line-performance/overview). Our algorithm achieves high-quality predictions in this dataset across all measures. By comparing our algorithm's performance with other conventional supervised learning methods like random forest and k-nearest neighbours, as well as the individual base learners of our ensemble algorithm, we demonstrate the effectiveness of our ensemble framework and the superiority of our algorithm as a supervised learning approach. Notably, our ensemble algorithm outperforms other algorithms significantly in terms of the mean absolute error measure.

In addition to predicting lead times for the Bosch dataset, we conduct a second case study using synthetic datasets. These datasets are generated by simulating a reliability-based model of a multi-product make-to-order production line with 11 stations, similar to the structure of the Bosch dataset. To introduce unpredictability, we incorporate daily, weekly, and seasonal patterns that make machine processing rates and product arrival rates non-stationary. We use workload, machine health state, and performance-related features as input data, commonly employed in studies using synthetic datasets for time-related production performance prediction [4, 5]. These features are recorded at product arrival times to predict lead times once products enter the system. This additional study serves two purposes: (1) to validate the performance of the proposed HEDA and compare prediction quality between synthetic datasets and the real-world Bosch dataset, and (2) to gain insights into the impact of features and stations on prediction quality. Note that exploration of the second purpose with the Bosch dataset is limited due to anonymised features. Our investigations using the synthetic datasets confirm the superior prediction quality achieved by our algorithm, as observed in the Bosch dataset. However, compared to the results from the Bosch dataset, prediction quality appears to be higher in the synthetic datasets. Analysing the relative importance of features in the synthetic datasets reveals that monitoring bottleneck stations is crucial for lead time prediction.

The rest of the paper is organised as follows. Section 2 reviews the related literature and outlines our contributions (Sect. 2.3). In Sect. 3, we formulate the problem of data-driven lead time prediction and describe our methodology that is based on hierarchical ensemble deep learning. Section 4 provides an overview of the Bosch dataset we use for lead time prediction. In Sect. 5, we investigate the performance of our algorithm with the Bosch dataset. Section 6 presents our second case study for data-driven lead time prediction with synthetic datasets. Finally, we conclude the paper in Sect. 7. The datasets and scripts used for predictions and simulation modelling can be found at http://researchrepository.napier.ac.uk/Output/2912661 upon being accepted for publication.

2 Related works

In Sect. 2.1, we focus on the literature relating to the data-driven lead time prediction with machine learning (ML) algorithms, including deep learning and hierarchical ensemble algorithms. Secondly, since we use the Bosch dataset to demonstrate the suitability of our algorithm, in Sect. 2.2, we review the studies that have used this dataset and describe how our handling of this large dataset differs from these earlier studies. In Sect. 2.3, we summarise the contributions of this work within the context of the existing literature.

2.1 Data-driven lead time prediction with machine learning

Machine learning algorithms are commonly used in smart manufacturing to enable data-driven approaches [1], including lead time prediction [6]. Various supervised ML algorithms such as decision trees [7, 8], random forests [9], support vector machines [5], neural networks [4, 10], and belief networks [11] have been applied for lead time prediction. Neural networks, in particular, are often selected for these tasks [6]. However, there are important gaps in the literature, as most studies use synthetically generated data from analytical production models instead of real-world production systems [4, 5, 12]. In this paper, we propose ML-based data-driven lead time prediction using real-world data from a large-scale production system with the proposed HEDA.

Hierarchical ensemble prediction algorithms are commonly used for multi-class classification problems [13], such as cancer classification [14], image recognition [15], and Alzheimer's disease classification [16]. These algorithms leverage the hierarchical structure among classes by dividing the prediction problem into subproblems (frequently implemented using deep learning algorithms [17, 18]). In such systems, multiple "base learners" are employed to address subproblems (e.g. specific classification tasks), and their predictions are combined using a fusion scheme, to generate the overall predictions. In this paper, we extend the hierarchical ensemble learning methodology to the regression task of lead time prediction in manufacturing. We achieve this by categorising the training samples into two groups based on their lead times.

Each of the base learners has to handle large amounts of input data from a complex manufacturing environment, a task at which traditional machine learning algorithms have proved to be ineffective [3]. Instead, deep neural networks with their intricate architecture, consisting of multiple hidden layers between the input and output layers, are more suitable. Sophisticated deep neural networks are increasingly employed for prediction tasks in smart manufacturing [1]. For instance, [19] employed an 8-layer deep neural network to predict the remaining useful life of rolling bearings. Fang et al. [10] proposed a network with 3 hidden layers, incorporating dropout and normalisation layers, to predict job remaining times. Huang et al. [4] demonstrated the superiority of a long short-term memory network over conventional ML algorithms in predicting product completion times in a serial line. Shajalal et al. [20] introduced a deep neural network with 4 hidden layers and a dropout layer to predict product backorder occurrences.

Our two-layer hierarchical ensemble deep learning algorithm comprises three base learners, each trained with a deep neural network that has 5 hidden and one dropout layer.

2.2 The Bosch production line dataset

The Bosch dataset, introduced as part of the data challenge at the 2016 IEEE International Conference on Big Data, aimed to classify faulty products using categorical, numerical, and date input features. The dataset includes a binary variable labelled as “Response”, which takes the value 1 for faulty products. Since the dataset’s release, several classification algorithms have been proposed for fault prediction in the Bosch dataset [21–25], with random forests showing good performance in this task [25]. Previous studies often focused on numeric features, neglecting categorical features due to their sparsity, and rarely incorporating date features that provide timestamps for measurements [24]. In this paper, we utilise the timestamps from the date features to determine the lead times of products, considering the time between the first and last measurements. We then use the numeric features as input data for predicting lead times. To our knowledge, this paper is the first to employ the Bosch dataset for lead time prediction. Additionally, our data exploration reveals a correlation between lead times and faults, suggesting that lead times could be an important input feature for fault prediction.

The Bosch dataset represents a production line with 51 stations and contains measurements from over 1000 features, posing a big data challenge. To address the high dimensionality of this dataset, the literature proposes techniques such as feature selection based on feature importance measures calculated with the XGBoost classification algorithm [21, 23]. Despite the large number of features in the Bosch dataset, the data they provide is sparse when considering all products together. This sparsity arises since products follow different paths through the stations, and features related to unvisited stations offer no information. It is likely that the anonymised dataset represents multiple product types, as indicated by repeated paths among products [21]. Although previous studies have not adopted an approach that models each product type separately, [21] suggest that the best predictions can be achieved by fitting models to each type. In this paper, we cluster the products in the Bosch dataset based on their unique paths among the stations and treat each type individually for predictions. This clustering significantly reduces dimensionality, making feature selection optional.

2.3 Contributions

This work makes four contributions:

- **Novel ML architecture:** The extension of hierarchical ensemble learning methods to a regression problem (i.e. lead time prediction) by using deep learning algorithms to enable high-accuracy base learners.
- **Validation insight:** Assessment of the proposed prediction algorithm using both industrial and model-generated synthetic datasets, revealing how the prediction using real-world, rather than artificial data, is a more demanding method of assessment.
- **Benchmark dataset for data-driven lead time prediction research:** Use of the time features of the Bosch dataset to derive lead times of products passing through the production line demonstrates a new application (i.e. lead time prediction) and the availability of an industrial benchmark for future research.
- **Effective partitioning strategy for Bosch dataset:** The research demonstrates how the “big data challenges” of the Bosch production line dataset can be mitigated by identifying product types and clustering data accordingly before using prediction algorithms.

3 Problem description and methodology

3.1 Problem description

Consider a make-to-order production system consisting of M production stations $i \in I = \{1, 2, .., M\}$. The system is a multi-product manufacturing environment that processes N different types of products $j \in J = \{1, 2, ..., N\}$. Each product order of type j is to be processed in stations $I^j \subseteq I$ of the system. The state of the production system is being monitored/recorded in real-time with measurements $k \in K$. For example, these could be obtained from sensors placed on machines. Each measurement k supplies state information $s_k(t) \in \mathbb{R}$ at any time t. Measurements $k \in K_i \subset K$ provide state information concerning station i such as vibration signals coming from the machinery at the station. Based on the information obtained from measurements K, real-time production state $\boldsymbol{s}(t) = (s_k(t))_{k \in K}$ is available at any given time t. When an order is placed for a type-j product and its production process is to start at time t, a prediction on its lead time τ should be made to inform its customer based on $\boldsymbol{s}(t)$. Thus, the aim of this research is to predict lead times for orders of every type j at the time of their arrivals based on the observed real-time production state information.

3.2 Methodology

For systems that already acquired considerable amounts of historical data $\mathcal{H}^j$ on the processing of past orders relating to each product type j, predictions can be entirely made with a data-driven approach, without the need for analytical models or simulations. When this data also contains the actual lead times of the past orders, along with their real-time production state information at the time of their arrivals, supervised learning (SL) algorithms such as decision trees and neural networks can be used for predicting lead times. In this paper, we assume that this is the case, and we consider that for every type j, $\mathcal{H}^j$ contains a large number of past type-j orders (n^j) and for each such past order l, $\mathcal{H}^j$ contains the arrival time of the order (t_l), the state information at the time of their arrival ($\boldsymbol{s}(t_l)$), and its actual lead time (τ_l) passed in the system, i.e. $\mathcal{H}^j = (\boldsymbol{s}(t_l), \tau_l)_{l=1}^{l=n^j}$. Hence, an SL algorithm can be trained with $\mathcal{H}^j$ to predict lead times of a number ($\tilde{n}^j$) of new type-j products ($\mathcal{N}^j = (\boldsymbol{s}(t_l))_{l=1}^{l=\tilde{n}^j}$). In this paper, we propose the following ensemble deep learning algorithm for this purpose.

3.2.1 Data-driven lead time prediction with hierarchical ensemble deep neural networks

In this paper, we propose a hierarchical ensemble algorithm for lead time prediction per product type j by dividing the training samples $\mathcal{H}^j$ structurally into two categories: those with short lead times $\underline{\mathcal{H}}^j$ and those with long lead times $\bar{\mathcal{H}}^j$. In this separation, we use the mean lead time of all training instances $\bar{\tau}^{\text{train}} = \frac{1}{n^j} \sum_{l \in \mathcal{H}^j} \tau_l$. Specifically, we let $\underline{\mathcal{H}}^j = \{l \in \mathcal{H}^j | \tau_l \leq \bar{\tau}^{\text{train}}\}$ and $\bar{\mathcal{H}}^j = \{l \in \mathcal{H}^j | \tau_l > \bar{\tau}^{\text{train}}\}$. This division is motivated by the observation that products with longer than average lead times can be structurally different from others, and this might be because these products are exposed to specific manufacturing conditions which do not allow for more timely production. Under this separation of samples, our algorithm uses three base learners. The first one, which we call *the generalist*, is trained with all samples $\mathcal{H}^j$. The remaining two learners are called *the specialists*. The first specialist is trained with short lead time samples $\underline{\mathcal{H}}^j$, whereas the second one is trained with long lead time samples $\bar{\mathcal{H}}^j$. We illustrate how our algorithm, which we call it HEDA, produces predictions in Fig. 1. HEDA is a hierarchical ensemble deep learning approach as we use deep neural

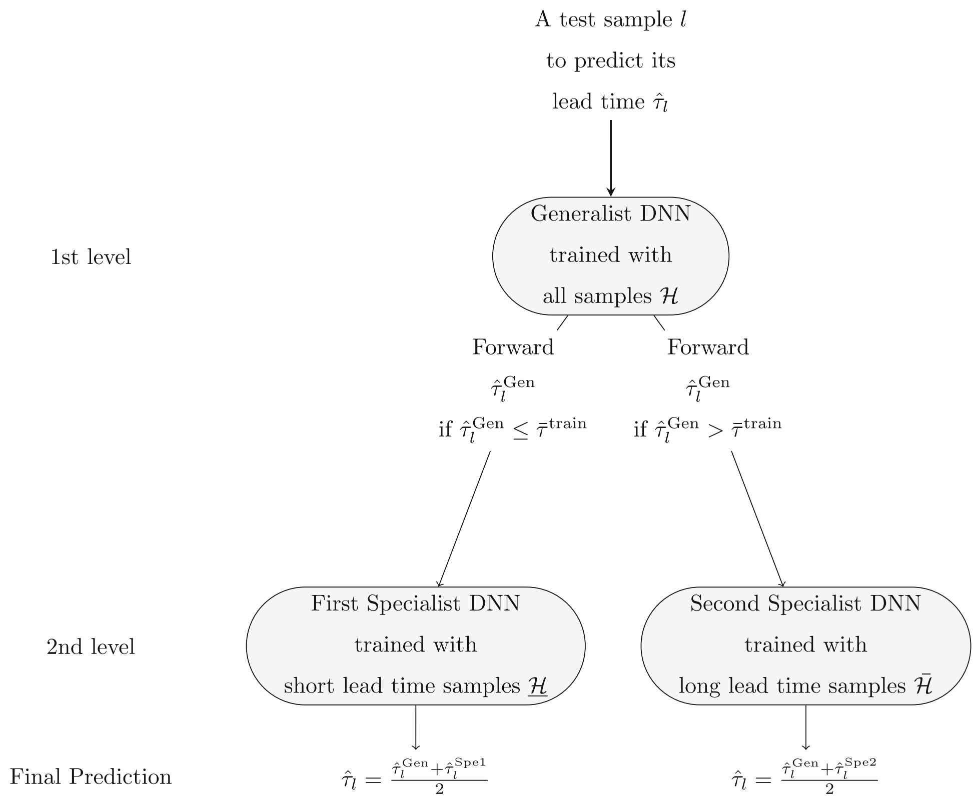
A test sample l to predict its lead time $\hat{\tau}_l$
1st level
Generalist DNN trained with all samples $\mathcal{H}$
Forward $\hat{\tau}_l^{\text{Gen}}$ if $\hat{\tau}_l^{\text{Gen}} \leq \bar{\tau}^{\text{train}}$
Forward $\hat{\tau}_l^{\text{Gen}}$ if $\hat{\tau}_l^{\text{Gen}} > \bar{\tau}^{\text{train}}$
2nd level
First Specialist DNN trained with short lead time samples $\underline{\mathcal{H}}$
Second Specialist DNN trained with long lead time samples $\bar{\mathcal{H}}$
Final Prediction
$\hat{\tau}_l = \frac{\hat{\tau}_l^{\text{Gen}} + \hat{\tau}_l^{\text{Spe1}}}{2}$
$\hat{\tau}_l = \frac{\hat{\tau}_l^{\text{Gen}} + \hat{\tau}_l^{\text{Spe2}}}{2}$

Fig. 1 Illustration of lead time prediction with HEDA

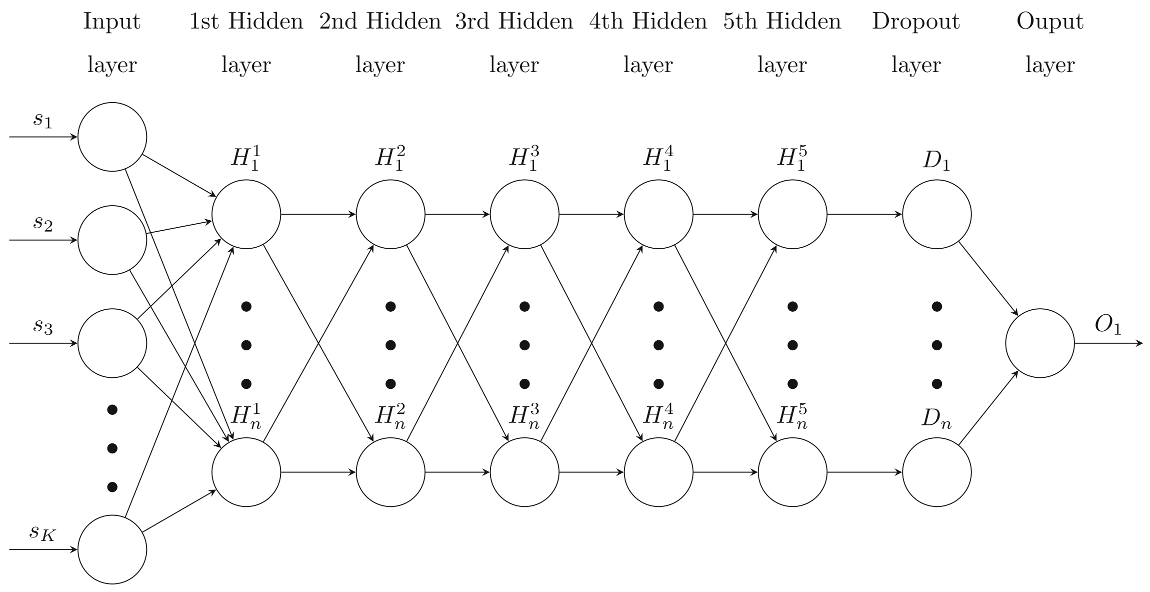

Fig. 2 The architecture of DNN base learners

networks (DNN) for each of its base learners. Figure 2 shows the network architecture used in these learners that consist of five hidden layers and a dropout layer.

The generalist DNN operates at the first level. For a test instance l, it predicts its lead time as $\hat{\tau}_l^{\text{Gen}}$. When the generalist's prediction is not above $\bar{\tau}^{\text{train}}$ such that it classifies l to be one of the short lead time products, this prediction is forwarded to be processed by the first specialist who is trained with short lead time samples. On the other hand, when the generalist predicts that the lead time will be above $\bar{\tau}^{\text{train}}$, it forwards its prediction to the second specialist who is trained with long lead time samples. Each specialist produces its own predictions for l, $\hat{\tau}_l^{\text{Spe1}}$, and $\hat{\tau}_l^{\text{Spe1}}$, respectively. Finally, predictions fuse the generalist's and the chosen specialist's predictions through averaging, as in [14]. When the actual lead time of an order is misjudged by the generalist and forwarded to the wrong specialist, the specialists' prediction can be too off from its actual lead time. Considering this disadvantage of relying totally on the specialists, through averaging, we incorporate the generalist's prediction in the final predictions to increase the robustness of our predictions.

3.2.2 Benchmark algorithms and performance measures

To assess the performance of HEDA, we use several conventional SL algorithms as benchmarks. These are ridge regression (Ridge), random forests (RF), k-nearest neighbours algorithm (kNN), and artificial neural networks (ANN). The advantage of SL algorithms with linear models such as ridge regression is that they can be trained very fast. However, for complex datasets with many input features, these simple models are often inadequate. We include Ridge along with other three non-linear models to be able to demonstrate how inferior the predictions can be with a naive SL model. We include random forests because of their proven efficiency in predicting faults in the Bosch dataset [25]. To predict lead times in production lines, neural networks is the most commonly used SL algorithm in the literature [6]. The ANN considered in this paper contains a single hidden layer.

We use HEDA and the benchmark algorithms in two case studies and evaluate their prediction performance. For this, we employ three commonly used prediction quality measures: coefficient of determination (R^2), root mean square error ($RMSE$), and mean absolute error (MAE). Given the predicted lead times $\hat{\boldsymbol{\tau}}^j$ for type-j products on the dataset $\mathcal{N}^j = (\boldsymbol{s}(t_l))_{l=1}^{l=\tilde{n}^j}$ and their actual lead times $\boldsymbol{\tau}^j$, these measures are calculated as follows:

$$R^2 = 1 - \frac{\sum_{l=1}^{\tilde{n}^j}(\tau_l^j - \hat{\tau}_l^j)^2}{\sum_{l=1}^{\tilde{n}^j}(\tau_l^j - \bar{\boldsymbol{\tau}}^j)^2}, \quad \bar{\boldsymbol{\tau}}^j = \frac{1}{\tilde{n}^j}\sum_{l=1}^{\tilde{n}^j}\tau_l^j \tag{1}$$

$$RMSE = \sqrt{\frac{1}{\tilde{n}^j}\sum_{l=1}^{\tilde{n}^j}(\tau_l^j - \hat{\tau}_l^j)^2} \tag{2}$$

$$MAE = \frac{1}{\tilde{n}^j}\sum_{l=1}^{\tilde{n}^j}|\tau_l^j - \hat{\tau}_l^j| \tag{3}$$

4 Overview of the Bosch dataset

The Bosch dataset provides high-dimensional data records on the production process of products as they are processed

Table 1 Lead time distribution statistics for faulty and non-faulty samples

	Mean	Std	Q1	Median	Q3
Faulty samples	13.91	21.96	1.95	4.93	16.30
Non-faulty samples	10.69	16.98	1.71	3.69	11.75

from station to station. Input features are split into three categories: categorical, numeric, and date features. All of the features are anonymised. The only information provided on a feature, which is indicated by its label that is of the form L#_S##_F####, is the production line and station at which the feature is measured. For example, $L3\_S36\_F3939$ is a feature measured on line 3, station 36, and its feature number is 3939.

Many studies using this dataset have focused on numeric features. Categorical features are often totally excluded due to their extreme sparsity [22], and only few [24] took date features into account. In this paper, we also exclude categorical features and consider the training dataset of numeric features (provided with the file train_numeric.csv on Kaggle) as our sensor data, having in total 1,183,747 samples of product orders passing through 4 production lines and 51 stations, while being monitored by nearly 970 sensors. The competition launched by Bosch on Kaggle in 2016 was intended for predicting faults in the products. Since then, various studies have proposed various SL-based approaches to classify whether the products being made are going to be faulty or not [25]. Here, we focus on lead times of products instead of faults.

We derive lead times of products from date features (i.e. training dataset) by considering the time difference between the date features measured at the first and last stations in the production process. We find that lead time over all products has mean of 10.71 with a standard deviation of 17.01. We must note that Bosch has also anonymised date features; they are not given as timestamps, instead presented in a converted form. So, in these features, we do not know exactly what a time unit corresponds to. However, according to the autocorrelation analysis conducted by [21], 16.75 units should correspond to a week.

Even though fault detection is outside the scope of our study, it is of interest to explore the relationship between the fault occurrence and lead times of products in the Bosch dataset. For this, we obtain statistics for the lead times of products which are labelled as faulty and non-faulty separately (Table 1). These statistics reveal that there is a positive correlation between the fault occurrence and longer lead times. The reason for this in a production environment such as Bosch's could be about the disruptions that faulty products pose to the production process.

Reported studies of the Bosch dataset have so far avoided distinguishing products and consequently used data of all stations for all products in their predictions. However, as also been argued by [21], there are several product types in the Bosch dataset, and in order to achieve best predictions, models are needed for each type separately. When we group products according to the unique set of stations that process them, we find in total 7928 product types in the dataset. This means that there are certain production patterns that are repeated in 1,183,747 samples that can potentially pass through 51 stations (with 2^{51} possible patterns).

In this paper, we restrict our attention to the 6 most frequently manufactured product types, since for types with smaller amounts of data available prediction through SL will not be feasible. In Fig. 3, we provide box and whisker plots for lead times per product type and also for all the samples in the

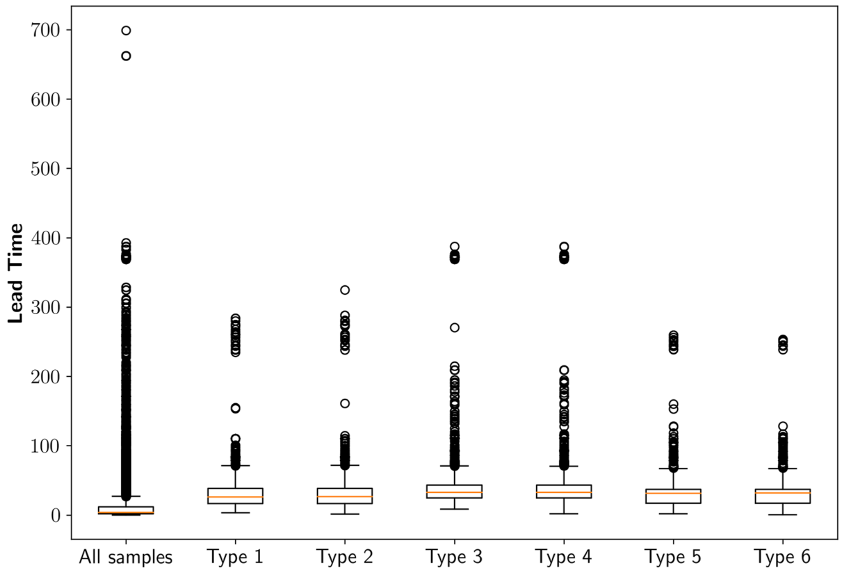

Fig. 3 Lead time distributions for the entire dataset and the six most frequently manufactured products

Table 2 Selected product types and their data dimensions

Prod. Type (j)	Stations (I^j)	#Samples	#Input features (after preprocessing)
1	$L1\_S24$,$L2\_S26$, $L3\_S29$, $L3\_S30$, $L3\_S33$,$L3\_S34$, $L3\_S36$,$L3\_S37$	51,709	241
2	$L1\_S24$,$L2\_S26$, $L3\_S29$, $L3\_S30$, $L3\_S33$,$L3\_S34$, $L3\_S35$,$L3\_S37$	50,213	242
3	$L1\_S25$,$L2\_S26$, $L3\_S29$, $L3\_S30$, $L3\_S33$,$L3\_S34$, $L3\_S36$,$L3\_S37$	20,830	270
4	$L1\_S25$,$L2\_S26$, $L3\_S29$, $L3\_S30$, $L3\_S33$,$L3\_S34$, $L3\_S35$,$L3\_S37$	20,041	270
5	$L1\_S24$,$L2\_S27$, $L3\_S29$, $L3\_S30$, $L3\_S33$,$L3\_S34$, $L3\_S36$,$L3\_S37$	18,646	259
6	$L1\_S24$,$L2\_S27$, $L3\_S29$, $L3\_S30$, $L3\_S33$,$L3\_S34$, $L3\_S35$,$L3\_S37$	17,923	262

Bosch dataset. We can observe how this clustering of samples by their types helps in reducing the variety observed in lead times. In Table 2, we present these 6 selected types. Under the column "Stations", we provide labels for the stations that process them, while the column next to it shows their frequencies. Note that these types combined pass through 11 stations only. In Fig. 4, we illustrate the process flow of these types combined. The numbers between parentheses in the figure give the total number of features providing measurements on the corresponding stations before preprocessing applied. We see that first stations $S24$ and $S25$ are being monitored more aggressively than other stations.

We apply preprocessing on each of these type-based datasets. We replace all null values by zero. We also impute extremes. Imputation is necessary when sensors are taking measurements at different frequencies and/or their measurements are subject to noise. For every measurement k, we impute all values in the data that lie four standard deviations away from their means by their median values. Given that sensor data would usually contain a variety of measurements that do not necessarily share the same scale, scaling of input data is very important before SL algorithms are applied. In this paper, we apply standard scaling to map all input data to the range [0, 1]. Lastly, we apply a variance-based feature elimination for every feature k. Any feature k with a variance lower than 0.0001 is eliminated from the dataset. After the variance-based elimination procedure, the number of input features drops significantly (see the last column of Table 2). Given that the total number of features is nearly 970 in the Bosch dataset, this demonstrates how this type-based clustering can also be helpful in filtering out sensor data on unrelated stations and thus reducing the dimension of the data.

We next analyse individual stations involved in the selected product types with respect to their contribution to the lead times of products, namely to the amount of time that products spent at individual stations. We extract this data from the date features, in a similar manner used to extract lead times. In Fig. 5, we present the distributions of processing times at these stations. We observe that the first two stations that process product types, which are stations $S24$/$S25$ and $S26$/$S27$, take the longest. Furthermore, we recognise that processing times of stations $S24$ and $S25$, and likewise of stations $S26$ and $S27$, are very close. Given the process flow in Fig. 4, this similarity might be due to these stations being in parallel.

Having an understanding of the sensor features that are most important for an accurate lead time prediction will be useful for providing decision support to production systems. For example, the monitoring and maintenance of the sensors that provide the most important data for lead time prediction could be prioritised. Therefore, it is valuable to know exactly which sensor data are the most important for predicting the lead times in the Bosch dataset. Using the F value-based feature selection procedure, we measure the importance of features for explaining the lead times of products and present the 20 most important features in Table 3 for each type.

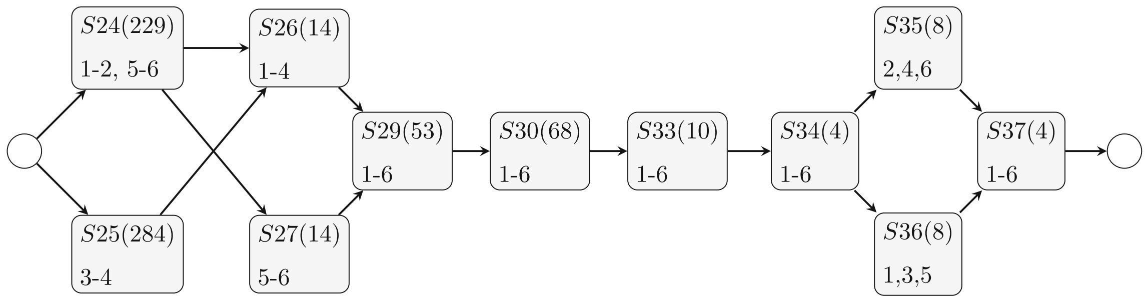

Fig. 4 Process flow for product types 1–6 (Numbers next to stations give the number of features relating to the station before preprocessing and underneath stated product types that pass through stations)

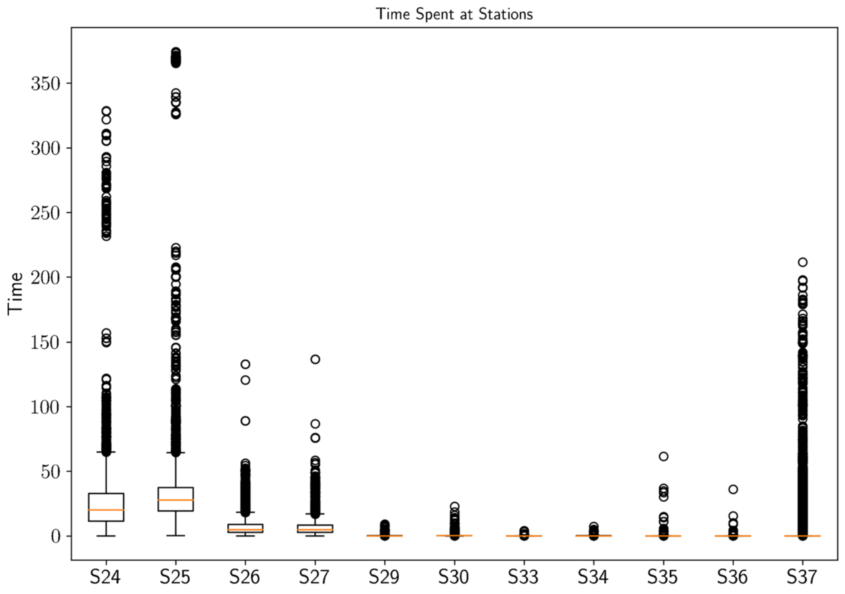

Fig. 5 Distributions of time spent per station in the entire dataset (considering involved stations in types 1–6)

Firstly, we observe that there are many common features that are among the most important for any product type. For example, the features which monitor stations 29 and 30 are significant in all product types. Note that all product types are processed at these stations (see Table 2). Station 25 is unique to types 3 and 4, and station 27 is unique to types 5 and 6 (see Table 2). We see in Table 3 that some features which monitor these stations are among the 20 most important features for the lead time prediction of the product types that relate to them. This is an indication that the importance of features might change in relation to the product types, namely the specific stations that process products, which suggests the unsuitability of an unified approach that does not distinguish product types.

Although fault detection falls outside the scope of our study, we are intrigued by the relationship between the fault

Table 3 The 20 most important features for predicting lead times of 6 product types (Feature importance is measured in terms of *F* value)

Rank	Type 1	Type 2	Type 3	Type 4	Type 5	Type 6
1	L3_S29_F3461	L3_S29_F3461	L3_S29_F3464	L3_S29_F3464	L3_S30_F3704	L3_S30_F3704
2	L3_S29_F3467	L3_S29_F3467	L3_S29_F3470	L3_S29_F3470	L3_S29_F3461	L3_S29_F3461
3	L3_S29_F3476	L3_S29_F3476	L3_S29_F3351	L3_S29_F3351	L3_S29_F3467	L3_S29_F3467
4	L3_S29_F3479	L3_S29_F3473	L3_S29_F3458	L3_S29_F3458	L2_S27_F3155	L2_S27_F3155
5	L3_S29_F3473	L3_S29_F3479	L2_S26_F3040	L2_S26_F3040	L3_S29_F3407	L2_S27_F3214
6	L3_S30_F3534	L3_S30_F3534	L3_S30_F3744	L3_S29_F3339	L3_S29_F3412	L3_S29_F3407
7	L3_S30_F3559	L3_S30_F3559	L3_S29_F3339	L3_S29_F3455	L2_S27_F3214	L3_S29_F3412
8	L2_S26_F3040	L2_S26_F3040	L3_S29_F3455	L3_S30_F3744	L2_S27_F3199	L1_S24_F1838
9	L3_S30_F3519	L3_S30_F3519	L2_S26_F3106	L2_S26_F3106	L1_S24_F1838	L1_S24_F1565
10	L1_S24_F1838	L1_S24_F1565	L3_S29_F3342	L3_S29_F3342	L1_S24_F1565	L3_S29_F3458
11	L1_S24_F1565	L1_S24_F1838	L3_S29_F3449	L3_S29_F3449	L3_S29_F3351	L3_S29_F3351
12	L2_S26_F3073	L2_S26_F3073	L1_S25_F2101	L1_S25_F2101	L3_S29_F3458	L2_S27_F3199
13	L2_S26_F3051	L2_S26_F3051	L2_S26_F3073	L3_S30_F3749	L1_S24_F1161	L1_S24_F1161
14	L3_S29_F3464	L3_S35_F3894	L3_S30_F3749	L2_S26_F3073	L1_S24_F1482	L1_S24_F1482
15	L3_S29_F3470	L1_S24_F1161	L3_S29_F3348	L3_S29_F3348	L1_S24_F1225	L1_S24_F1225
16	L3_S30_F3804	L1_S24_F1482	L3_S29_F3395	L3_S29_F3395	L1_S24_F1230	L1_S24_F1230
17	L2_S26_F3106	L2_S26_F3106	L1_S25_F2026	L1_S25_F2026	L3_S30_F3804	L3_S30_F3579
18	L3_S30_F3554	L1_S24_F1230	L1_S25_F1938	L1_S25_F1938	L3_S30_F3579	L3_S30_F3804
19	L1_S24_F1482	L1_S24_F1225	L1_S25_F2002	L1_S25_F2002	L3_S29_F3473	L3_S30_F3519
20	L1_S24_F1230	L3_S35_F3896	L1_S25_F2051	L1_S25_F2170	L3_S30_F3709	L3_S30_F3709

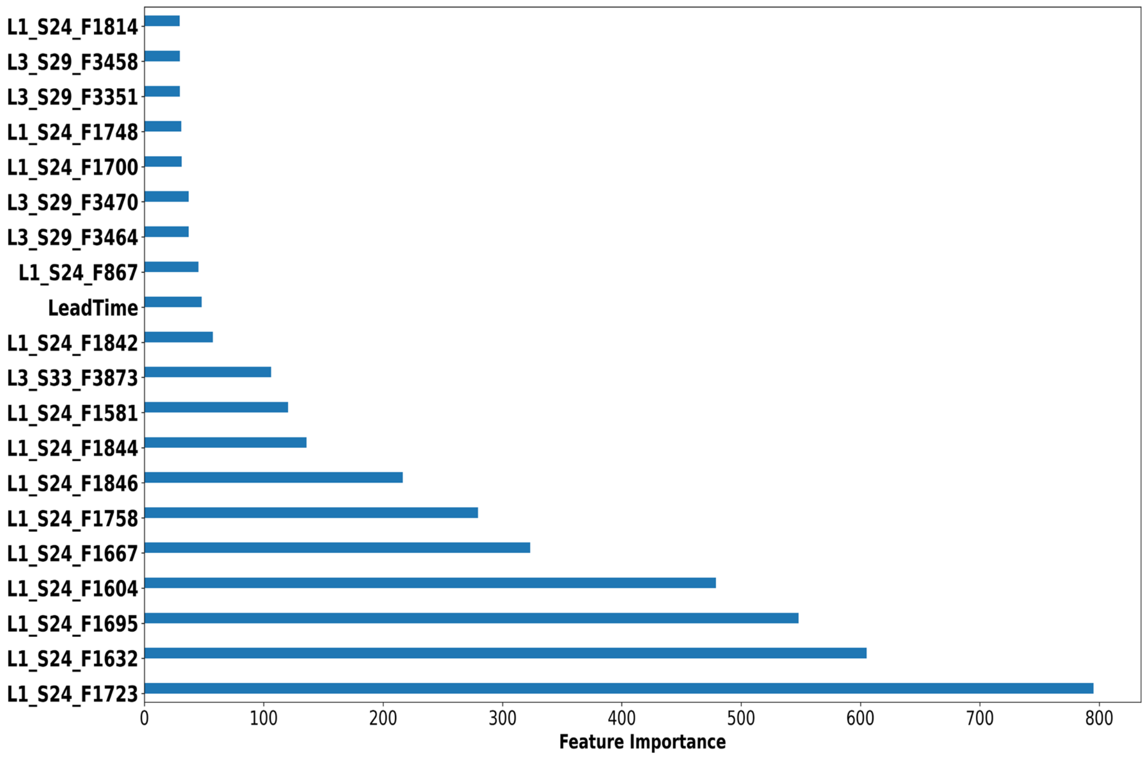

Fig. 6 The 20 most important features for predicting faults in type-1 products (Feature importance is measured in terms of F value)

occurrence and lead times, as it is explored in Table 1. To investigate how the importance of features changes when utilised for lead time prediction versus fault detection, we provide Fig. 6 for the fault detection case of type-1 products. In Fig. 6, it is observed that lead time, denoted as "LeadTime", ranks among the top 20 most important features. This suggests that lead times contain valuable information for detecting faults. This finding aligns with our earlier exploration in Table 1, where we discovered indications of a correlation between faulty occurrence and lead times. Consequently, it can be argued that prediction models targeting fault detection in the Bosch dataset should leverage lead time information. Comparing Fig. 6 and Table 3 (type 1), we observe a discrepancy in the importance of sensor features. While features associated with monitoring stations 29 and 30 dominate in lead time prediction, features related to station 24 take precedence in fault detection. Nevertheless, there are some features, such as 3464 and 3470, which monitor station 29, that rank among the top 20 most important features for both lead time prediction and fault detection.

5 Case study 1: the Bosch dataset

In this section, we investigate the performance of HEDA in an industrial case study using the datasets of 6 product types we derived from the Bosch dataset. As described in our "Methodology", we provide predictions for each type separately. For splitting the datasets into a training set $\mathcal{H}^j$ and a test set $\mathcal{N}^j$ while providing results on prediction quality that have statistical support, we perform 10-fold cross validation for every product type j. For implementing Ridge, RF, kNN, and ANN benchmark algorithms, we use scikit-learn Python machine learning library, and for HEDA, the TensorFlow library through keras deep learning interface is used for each of its three deep neural network components.

The parameters used with these algorithms are as follows: We use Ridge with its default setting, and for the other algorithms, we tune (some of) their hyper parameters through offline grid search. For kNN, its number of neighbours is set to 10. For RF, we tune several of its hyper parameters: the number of trees in the forest is set to 300, the minimum number of samples required at every node is set to 5, and the number of features considered when searching for the best split is fixed to be the square root of the total number of input features. For ANN, we consider a single hidden layer of 100 nodes with a ReLU activation function, and for its weight optimisation, the Adam solver is used with an iteration limit of 1000. Because of its stability advantages, ReLU is very commonly used in the literature [4, 11, 19]. For every deep neural network component of HEDA, we consider 5 fully connected (dense) hidden layers, each with a ReLU activation function. Similar to ANN, we fix the number of neurons per hidden layer to 100. To reduce overfitting, we add

Table 4 Performance of HEDA against benchmark SL algorithms with 10-fold cross validation for $n = 100$, $n = 200$, and $n = 270$ features selected

		$n = 100$					$n = 200$					$n = 270$(all features retained)				
Perf. measure	Prod. type	HEDA	Ridge	RF	kNN	ANN	HEDA	Ridge	RF	kNN	ANN	HEDA	Ridge	RF	kNN	ANN
R^2	1	0.93	0.64	0.90	0.89	0.89	0.96	0.67	0.91	0.89	0.90	0.96	0.67	0.91	0.87	0.90
	2	0.92	0.64	0.89	0.89	0.89	0.95	0.66	0.90	0.88	0.89	0.95	0.67	0.90	0.87	0.89
	3	0.86	0.62	0.83	0.83	0.81	0.87	0.64	0.83	0.82	0.78	0.88	0.65	0.83	0.80	0.75
	4	0.88	0.65	0.85	0.84	0.85	0.88	0.66	0.85	0.83	0.80	0.89	0.67	0.85	0.81	0.79
	5	0.92	0.74	0.91	0.89	0.88	0.93	0.77	0.92	0.91	0.87	0.93	0.77	0.91	0.89	0.85
	6	0.91	0.75	0.91	0.89	0.88	0.94	0.77	0.92	0.91	0.88	0.94	0.77	0.92	0.89	0.86
$RMSE$	1	3.86	8.64	4.64	4.74	4.69	2.90	8.24	4.37	4.87	4.44	2.89	8.22	4.43	5.15	4.54
	2	3.97	8.68	4.78	4.88	4.78	3.16	8.40	4.67	5.07	4.69	3.07	8.29	4.59	5.26	4.71
	3	5.47	9.10	6.01	6.13	6.37	5.31	8.82	6.03	6.33	6.93	5.16	8.75	6.13	6.51	7.31
	4	5.13	8.82	5.69	5.90	5.80	5.15	8.61	5.74	6.18	6.58	5.00	8.56	5.81	6.39	6.76
	5	4.37	7.55	4.21	4.85	5.04	3.81	7.17	4.28	4.51	5.42	3.83	7.15	4.32	4.87	5.69
	6	4.56	7.50	4.38	4.88	5.16	3.63	7.16	4.24	4.54	5.23	3.73	7.14	4.32	4.91	5.59
MAE	1	1.78	6.60	2.72	2.60	3.13	1.22	6.23	2.57	2.78	3.07	1.22	6.21	2.63	3.00	3.15
	2	1.80	6.58	2.77	2.61	3.15	1.32	6.31	2.73	2.85	3.21	1.27	6.22	2.68	3.01	3.24
	3	3.01	5.86	3.28	3.43	4.30	2.99	5.71	3.33	3.45	4.93	2.94	5.66	3.37	3.57	5.15
	4	2.89	5.80	3.22	3.49	4.02	2.96	5.61	3.25	3.36	4.68	2.88	5.58	3.29	3.52	4.80
	5	1.99	5.58	2.37	2.67	3.27	1.81	5.23	2.37	2.38	3.73	1.87	5.21	2.42	2.69	3.97
	6	2.19	5.55	2.42	2.81	3.37	1.78	5.23	2.34	2.40	3.62	1.85	5.21	2.40	2.72	3.92

a dropout layer with dropout rate 0.2 right after the fifth layer. For its output layer, we use the linear activation action. To initialise weights, we use HeNormal. Similar to ANN, we use the Adam solver to optimise and update the weights during training. We then train each deep neural network component for 100 epochs in batches of 50 samples.

In Table 4, we present R^2, $RMSE$, and MAE scores for HEDA and the benchmark Ridge, RF, kNN, and ANN algorithms over the 6 product types. The values in this table are the averages over 10 prediction instances from 10-fold cross validation. For high-dimensional datasets, feature selection is usually a necessary step to be able to make use of SL algorithms, since too many features might deteriorate their convergence and thus hinder their prediction abilities. Studies treating the Bosch dataset use various approaches for feature selection. The SL algorithms we propose, including HEDA, are able to produce predictions with all features included. Nevertheless, here we use feature selection to demonstrate how the performance of SL algorithms might worsen when fewer features are included. For this, we use an embedded method that ranks features based on their F values computed with a linear regression SL model and selects the top n features. To show how the number of features selected influences the scores, we present results for three different levels of n: 100, 200, and 270. Since after preprocessing, the total number of features remaining is at most 270 over all product types (see Table 2), letting $n = 270$ means keeping all input features.

We observe that HEDA consistently performs substantially better than all four benchmarks with respect to all three prediction quality metrics. This shows that with HEDA, significantly more accurate sensor data-based predictions on lead time can be obtained as opposed to some conventional SL algorithms. Moreover, we see that HEDA benefits from including more sensor features. This positive effect seems to be highly visible in some cases; $RMSE$ score improves by around 25% for product types 1–2 when all features are kept rather than only 100 features selected. This ability of HEDA which consists of three deep neural networks to generalise over many features and benefit from them is an expected outcome, given that the purpose of deep neural networks in general is to make predictions from complex data without needing feature engineering or expert knowledge [1, 3, 4]. On the other hand, we note that for RF, kNN, and ANN, having more features does not improve their prediction qualities.

When we compare the performance of HEDA and Ridge in Table 4, we see how inferior the predictions can be with a simple linear SL model like Ridge. While HEDA is able to achieve R^2 scores of above 0.9 most of the time, with Ridge, we can attain 0.77 at most. To illustrate this in detail, Fig. 7 shows how the predicted lead times by HEDA and Ridge correlate with the actual lead times using the first 200 test samples of a prediction instance of type-1 products. We can see in Fig. 7 that except for a number of instances, predicted lead time with HEDA is very close to the actual lead time.

We next assess the benefit of combining three deep neural networks in an ensemble approach by comparing HEDA to its generalist DNN component that uses a single deep neural network prediction model to train with all available training samples. We apply this generalist DNN for the results in Table 4 and show the percentage relative difference in the R^2, $RMSE$, and MAE scores under HEDA relative to the generalist DNN in Table 5. Overall, we see that with the ensemble approach, a higher R^2 score is achieved, and both $RMSE$ and MAE prediction errors decrease. However, we must note that the improvement that HEDA achieves by com-

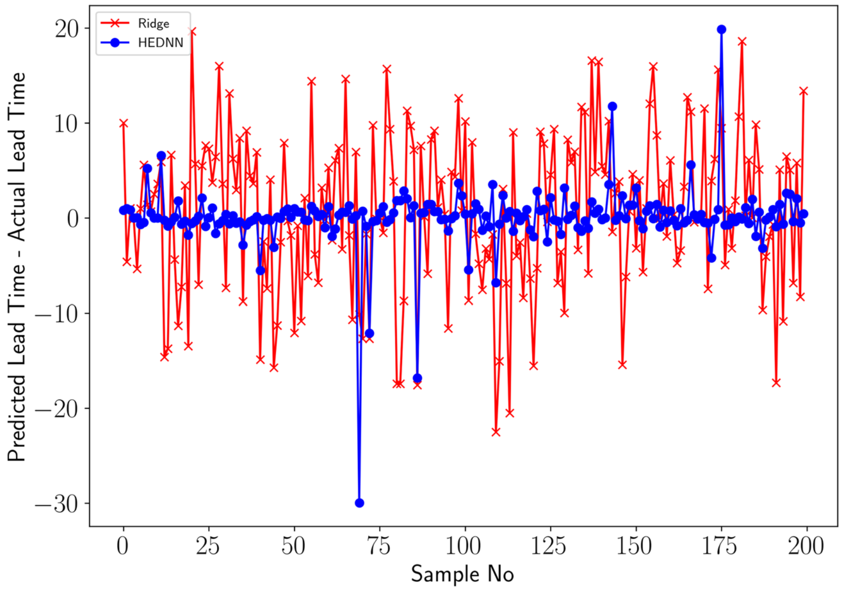

Fig. 7 Prediction quality (predicted lead time-actual lead time) with HEDA vs Ridge (for type-1 products with $n = 270$, presenting for 200 test samples)

Table 5 Performance of HEDA relative to its generalist DNN component (based on average scores from 10-fold cross validations)

Type	n	Change in R^2 (%)	Change in $RMSE$ (%)	Change in MAE (%)
1	100	0.32	−2.12	−9.67
	200	0.19	−2.31	−8.00
	270	0.19	−2.27	−7.85
2	100	0.30	−1.86	−8.17
	200	0.21	−2.09	−7.72
	270	0.16	−1.43	−7.49
3	100	−0.39	1.14	−4.30
	200	−0.23	0.89	−3.73
	270	−0.08	0.30	−4.11
4	100	0.27	−1.01	−6.40
	200	−0.16	0.74	−4.46
	270	0.12	−0.53	−6.33
5	100	0.42	−2.13	−8.87
	200	0.20	−1.51	−8.06
	270	0.26	−1.74	−5.06
6	100	0.46	−2.17	−9.57
	200	0.37	−3.00	−8.44
	270	0.15	−1.15	−6.11
Ave		0.15	−1.24	−6.91

Table 6 Performance of HEDA and generalist DNN under different number of hidden network layers (average scores from 10-fold cross validation using type-1 products with all features included)

	HEDA			DNN		
Num. layers	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE
2	0.94	3.53	1.99	0.93	3.70	2.26
3	0.95	3.14	1.60	0.95	3.28	1.80
4	0.96	3.00	1.35	0.96	3.03	1.48
5	0.96	2.89	1.22	0.96	2.96	1.32
6	0.96	2.90	1.16	0.96	2.96	1.27
7	0.96	2.99	1.19	0.95	3.07	1.31

bining the generalist DNN with two of the specialist DNNs is much more substantial in terms of reducing MAE, making nearly a 7% difference on average.

An important feature for deep neural networks is its depth, the number of hidden layers of neurons connected to each other in the network. Having many hidden layers provides the ability to capture many patterns from input features that would not be possible with few layers. However, having too many of them may lead to overfitting, due to the imposed ability of capturing even the smallest patterns from the training data that would not generalise over other unseen data (e.g. test data). We next investigate the effect of network depth on the performance of HEDA that is composed of three deep neural networks and on that of the generalist DNN. In Table 6, we showcase this using type-1 products. In doing so, we vary the number of hidden layers in the deep neural networks integrated in HEDA between 2 and 7, while keeping the number of neurons per layer fixed at 100, and present results for R^2, $RMSE$, and MAE prediction measures for each resulting setting. We see that having only a few layers (2–3 layers) performs the worst with respect to all prediction measures for both HEDA and DNN. Moreover, we observe that having too many layers (7 layers) detriments all three prediction scores when the generalist DNN is used. On the other hand, with HEDA, we see this negative effect only in $RMSE$; we see that having too many layers does not worsen the R^2 prediction quality score and improves MAE with HEDA. This indicates that our ensemble approach that combines three deep neural networks each trained with different subsets of training data has a potential to reduce the sensitivity to overfitting.

6 Case study 2: synthetic datasets

In this section, we build a discrete-event simulation of a multi-product make-to-order production system and investigate the performance of HEDA to predict lead times with real-time data recorded from simulations. In modelling this system, we get inspiration from the Bosch production process. As illustrated in Fig. 4, we consider a system with 11 stations ($i \in \{1, 2, .., 11\}$). With this additional study, we compare the performance of HEDA and benchmark prediction algorithms when they are used with synthetic datasets to their performance with the real-world Bosch dataset. This helps us to examine and compare the extent of prediction challenge for algorithms when they are used in synthetic datasets coming from models versus real-world datasets. Moreover, this study allows us to derive clearer insights into feature and station importance for lead time prediction that are not possible to derive from the Bosch dataset with its anonymised features.

Specifically, we consider that orders arrive to the system only for product types 1–6 ($j \in \{1, 2, .., 6\}$) and follow the paths in the system as defined in Table 2. We suppose that the state of the system is being monitored in real time periodically. We then use the state of the system to predict lead times of orders at the time of their arrival. We consider that each station i is a single machine in the simulated system. We assume that at the beginning of every period, an uncertain number of job arrivals for every product j will be observed, and by the end of the period, an uncertain number of jobs will be processed by each machine and directed to the next machine in their paths before the next period starts. We assume that jobs are processed according to the FIFO discipline at stations independent of their types. Poisson processes are used for modelling the number of arrivals and jobs processed at stations. For these, we use non-stationary rates. More specifically, for the arrivals, we consider a weekly pattern in which the arrival rate is much higher during weekends. As for the processing rates, we implement a daily pattern and consider that rates are much slower during the last period of the day.

For machines, we consider a reliability model such that they can break and stop processing jobs and get back to processing once they are repaired. In modelling breakdown likelihoods of machines, we use a degradation model in which machines become degraded once they have worked for some time since their last repair. In this model, degraded machines have a much higher likelihood of breaking. Machines do not break only independently; machines can also fail due to system-wide failure events. In our model, these events have a likelihood of occurring depending on the workload in the system and the system's tolerance for it. For the system's tolerance, we also incorporate the seasonal effects and suppose that the tolerance will be lower during summer. To model the system workload in a period, we use the total number of jobs in the system. When a system-wide failure occurs, the system switches to a protected mode for some time which protects the system from repeated system-wide failures and allows machines to be repaired. Finally, the lead times of a product is the number of periods spent in the system till the product is processed at the final station of its path plus a small duration of time spent for preparing and picking up the product from the system. We model the pick up time as a random parameter for each product independent of the system and the product types. This is for the purpose of including noise in the system.

To model this system, we use the following parameters:

- $\hat{\lambda}_j, \forall j$: arrival rate parameter for product type j. This parameter is the rate of the exponential distribution which models the stochastic interarrival times of orders. It is common to consider stochastic job arrivals in make-to-order production systems (see [26]).
- $\hat{\mu}_i, \forall i$: processing rate parameter for station i. The completion times of jobs in stations are also modelled using exponential distributions. These parameters are the rates of these distributions. Stochastic job completion times modelled with exponential distributions are commonly found in queueing models of production systems (see [27]).
- $\delta_i, \forall j$: degradation threshold parameter for station i. This parameter is the threshold for considering a healthy station with an uptime longer than this value as degraded. Because of this feature, our degradation model is not history independent and can be classified as non-Markovian [28].
- $\hat{\pi}^i_{20}$, $\hat{\pi}^i_{10}$: failure probability parameter for non-degraded and degraded station i. With these parameters, we model degradation-based breakdown probabilities for the stations in every period, as in [29].
- $\hat{\pi}^i_{02}$: repair probability parameter for station i. This is the state-independent probability that a failed station will be repaired in a period and be operational the next period. As in [4], repair times in our model are Markovian.
- $\hat{\pi}_0$: system-wide failure probability parameter. We include this to induce machine failures that do not only depend on the health states of individual stations but also on the state of all stations in the system. By considering this second mechanism causing failures, we render our model less predictable and noisier as a real-world system would be.
- α, β : workload tolerance parameter during and outside summer. Similar to the stress factor considered in [30], we use these workload-based parameters as factors that affect the machine health. We consider that the system will be under pressure when the total workload exceeds these values, which then will activate the possibility for system-wide failures.
- γ : repeated system-wide failure protection parameter. This is an auxiliary parameter to control the stability of the system by protecting it against system-wide failures occurring in short time intervals.
- ρ : job pick up time parameter. This is modelled as a state-independent random variable, bearing similarity to the way that setup times are modelled in [4].

Letting t denote a period, we now describe the time-dependent behaviour of the system. For this, we track several system variables. We let $Q_i(t)$ represent the total number of jobs waiting to be processed at station i at the beginning of period t. For station i, with $C_i(t)$, we track the time spent without breaking since its last repair, and with $F(t)$, we track the amount of time passed since the last system-wide failure observed in the system at period t.

- *Arrival rates:* Let $\lambda_j(t)$ denote the arrival rate of product type j in period t. If t is a weekend period, we then let $\lambda_j(t) = 2\hat{\lambda}_j$, and $\lambda_j(t) = \hat{\lambda}_j/2$ otherwise.
- *Processing rates:* Let μ_i denote the processing rate at station i in period t. If t is the last shift of a day, then $\mu_j(t) = \hat{\mu}_i/2$, and $\mu_j(t) = \hat{\mu}_i$ otherwise.
- *Number of jobs processed at operating stations:* The Poisson process with rate $\mu_i(t)$ is used for determining the total number of jobs that can be processed at operational station i during period t. If this number is above $Q_i(t)$, we let the number of jobs to be processed be $Q_i(t)$.

- *Breakdown probabilities:* Let $p_0^i(t)$ denote the breakdown probability of operational station i at the beginning of period t. This depends on the station's own breakdown probability, which we denote by $\pi_0^i(t)$, and the probability for system-wide failure, which we denote by $\pi_0(t)$. In period t, $\pi_0^i(t)$ depends on whether the machine is degraded or not, which is dependent on the time spent without breaking since the last repair $C_i(t)$. On the other hand, $\pi_0(t)$ depends on the total workload $Q(t) = \sum_i Q_i(t)$ and the amount of time passed since the last system-wide failure observed in the system $F(t)$. Given that system-wide failure is independent from the degradation-based failure of machine i, we have the following relationships:

$$p_0^i(t) = \pi_0(t) + (1 - \pi_0(t))\pi_0^i(t) \tag{4}$$

$$\pi_0^i(t) = \begin{cases} \hat{\pi}_{10}^i, & C_i(t) > \delta_i \\ \hat{\pi}_{20}^i, & otherwise \end{cases} \tag{5}$$

$$\pi_0(t) = \begin{cases} \hat{\pi}_0, & F(t) > \gamma,\ Q(t) > \alpha,\ t \text{ is a summer period} \\ \hat{\pi}_0, & F(t) > \gamma,\ Q(t) > \beta,\ t \text{ is not a summer period} \\ 0, & otherwise \end{cases} \tag{6}$$

- *Breakdown and repair events:* If machine i was operational during period $t-1$, we use $p_0^i(t)$ to generate breakdown events for machine i at the beginning of period t. If this event is observed, then the machine moves to the failed state immediately and therefore is not able to process jobs during period t. For machines that were at the failed state during period $t-1$, we use the repair probabilities π_{02}^i to generate repair events at the beginning of period t which will repair the machine to the non-degraded operational state such that they will be able to process jobs during period t.

To be used as an input feature in predicting the lead time of a job arriving at the beginning of period t, we record the following real-time state information from the system. The column labels used for these features in the datasets are shown in parentheses.

- *Workload-related features:* We provide $Q_i(t)$, $\forall i$ as input features. Moreover, we provide $Q(t) = \sum_i Q_i(t)$ (under the column "QSumall") and the total workload at stations that are along the path of the arrived job (under the column "QSumroute").
- *Machine state–related features:* We report the heath state of each machine i as 0, 1 or 2 (under the columns "M1state, M2state,..,"). State 0 encodes the failed state, while state 1 and state 2 encode the degraded and non-degraded operational states. Along with health state indicators, we also give $C_i(t)$, $\forall i$ as input features (under the columns "M1activesince, M2activesince,..,"). Similar to as we do with the workload features, we incorporate the type information of the jobs and we report the total number of failed, non-degraded and degraded machines along the path of the arrived job as features (under the columns "Mfailedroute", "Mhealthyroute", and "Mdegradedroute").
- *Performance-related features:* We use statistics relating to the lead time and throughput of the most recent jobs that left the system prior to period t as input features. The statistics include the minimum, maximum, and average for throughput (columns starting with "Min_t", "Max_t" and "Ave_t") and lead time measures (columns starting with "Min_lt", "Max_lt", and "Ave_lt"). Half of the features are type specific such that they consider the jobs that share the same type with the arrived job (columns with "type" in their labels), and the other half take into account all jobs without distinguishing their types (columns with "all" in their labels). For capturing the very recent performance trend, we compose features by considering the jobs that left the system in the last five periods (columns whose labels end with "short"), and for capturing the longer trend, we also look at the last twenty periods to calculate performance-related features (columns whose labels end with "long"). In total, we use 24 performance-related features.

To generate samples for lead time prediction, we simulate this system for 10,000 periods, while considering that a day corresponds to 5 periods. We let the pick up time parameter $\rho \sim U\{0, 2\}$ (a discrete uniform distribution). For the arrival rate parameters $\hat{\lambda}_j$s, as an inspiration from the Bosch production line, we use the sample frequencies of product types observed in Table 2 to fit them proportionally. In doing so, we fix $\hat{\lambda}_6 = 1$. To investigate the robustness of our prediction algorithms, we generate six interesting datasets in which we vary the processing rate and degradation threshold parameters. We fix the remaining parameters as follows: $\alpha = 100$, $\beta = 150$, $\gamma = 35$, $\hat{\pi}_0 = 0.01$, $\hat{\pi}_{02}^i = 0.3, \forall i$, $\hat{\pi}_{20}^i = 0.05, \forall i$, and $\hat{\pi}_{10}^i = 0.1, \forall i$. The parameter settings for $\hat{\mu}_i$ and δ_i used in these datasets are given in Table 7. In the first two datasets, all stations process fast with a rate of 18. In the datasets encoded with FSF, the first six stations process fast, while the remaining stations process slower with a rate of 12, and in the ones encoded with LSF, the later stations process fast. In setting the degradation parameter, each time we consider two cases depending on whether it is the fast processing stations that degrade fast (encoded with FDF) or not (encoded with FDS).

To illustrate the behaviour of the production system in these six datasets, we present Fig. 8 that shows the lead time distributions of the products made in each dataset in the form of a box and whisker plot. We note that lead times in datasets ASF_FDF and ASF_FDS have relatively lower

Table 7 Parameters of synthetic datasets

Datasets	$\hat{\mu}_i$	δ_i
ASF_FDF	$\hat{\mu}_i = 18, \forall i$	$\delta_i = 20, \forall i$
ASF_FDS	$\hat{\mu}_i = 18, \forall i$	$\delta_i = 70, \forall i$
FSF_FDF	$\hat{\mu}_i = 18$ if $i \leq 6$, $\hat{\mu}_i = 12$, otherwise	$\delta_i = 20$ if $i \leq 6$, $\delta_i = 70$, otherwise
FSF_FDS	$\hat{\mu}_i = 18$ if $i \leq 6$, $\hat{\mu}_i = 12$, otherwise	$\delta_i = 70$ if $i \leq 6$, $\delta_i = 20$, otherwise
LSF_FDF	$\hat{\mu}_i = 12$ if $i \leq 6$, $\hat{\mu}_i = 18$, otherwise	$\delta_i = 70$ if $i \leq 6$, $\delta_i = 20$, otherwise
LSF_FDS	$\hat{\mu}_i = 12$ if $i \leq 6$, $\hat{\mu}_i = 18$, otherwise	$\delta_i = 20$ if $i \leq 6$, $\delta_i = 70$, otherwise

variances compared to other datasets in which stations differ with respect to their processing rates. It seems that distinguishing stations with respect to their processing rates, and degradation thresholds, has a complicating effect on the stability of the simulated production system.

Next, we investigate the performance of HEDA in predicting the lead times of these six simulated datasets in Table 8. To compare its performance, we focus on RF, which is the best performing algorithm among the benchmarks proposed in the first case study, and Ridge, which is the most simplest of the benchmarks. Also, to understand the usefulness of our hierarchical ensemble approach with deep neural networks, we compare HEDA to the generalist DNN, as we do in the first case study. To evaluate the robustness of results regarding the relative performance of different prediction algorithms for lead time prediction in the Bosch dataset, here we test the prediction algorithms under the same parameter settings that we use in obtaining Table 4 and similarly apply a 10-fold cross validation and provide their averages. We must note that here we do not apply feature selection since the total number of features is rather small. Also, unlike in Sect. 5, here we do not provide prediction models separately for each type; a single prediction model is produced by combining training samples of all product types. By comparing the performance of algorithms and ranking them, we observe consistent results as demonstrated in the first case study using the Bosch dataset. These results further support the superiority of our deep learning approach, HEDA, over conventional SL algorithms like RF and Ridge. Also, as in the first case study, we see that HEDA provides significantly better predictions than the generalist DNN, which confirms the suitability of our hierarchical ensemble approach to refine deep learning even more. Similarly to the first case study, here we also observe that the superiority of HEDA is the most prevalent in terms of reducing MAE. Previously, in Fig. 8, we note that datasets ASF_FDF and ASF_FDS demonstrate a more stable behaviour than other datasets. In Table 8, it can be seen how this difference results in lower $RMSE$ values for these instances compared to others.

To understand which input features contribute more to predict lead times in the simulated datasets, we measure the importance of features with their F values, as we do in obtaining Table 3, and present them in Fig. 9. The feature with label “Qsumroute”, which records for a newly arrived job of a certain type the total number of jobs in the stations which the job has to be processed, is found to be the most important feature in all datasets. Among station-based workload features ($Q_i(t)$), tracking stations 3 and 5 seems more important in

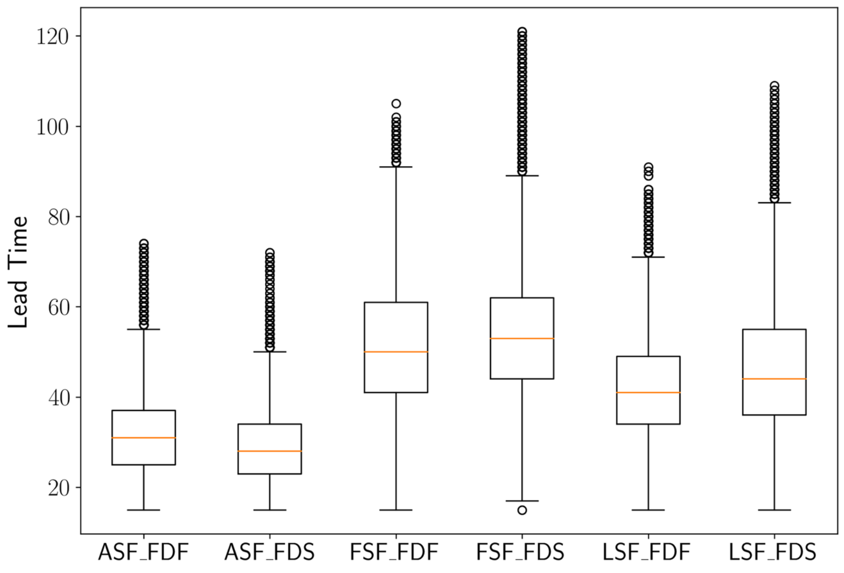

Fig. 8 Lead time distributions of the synthetic datasets

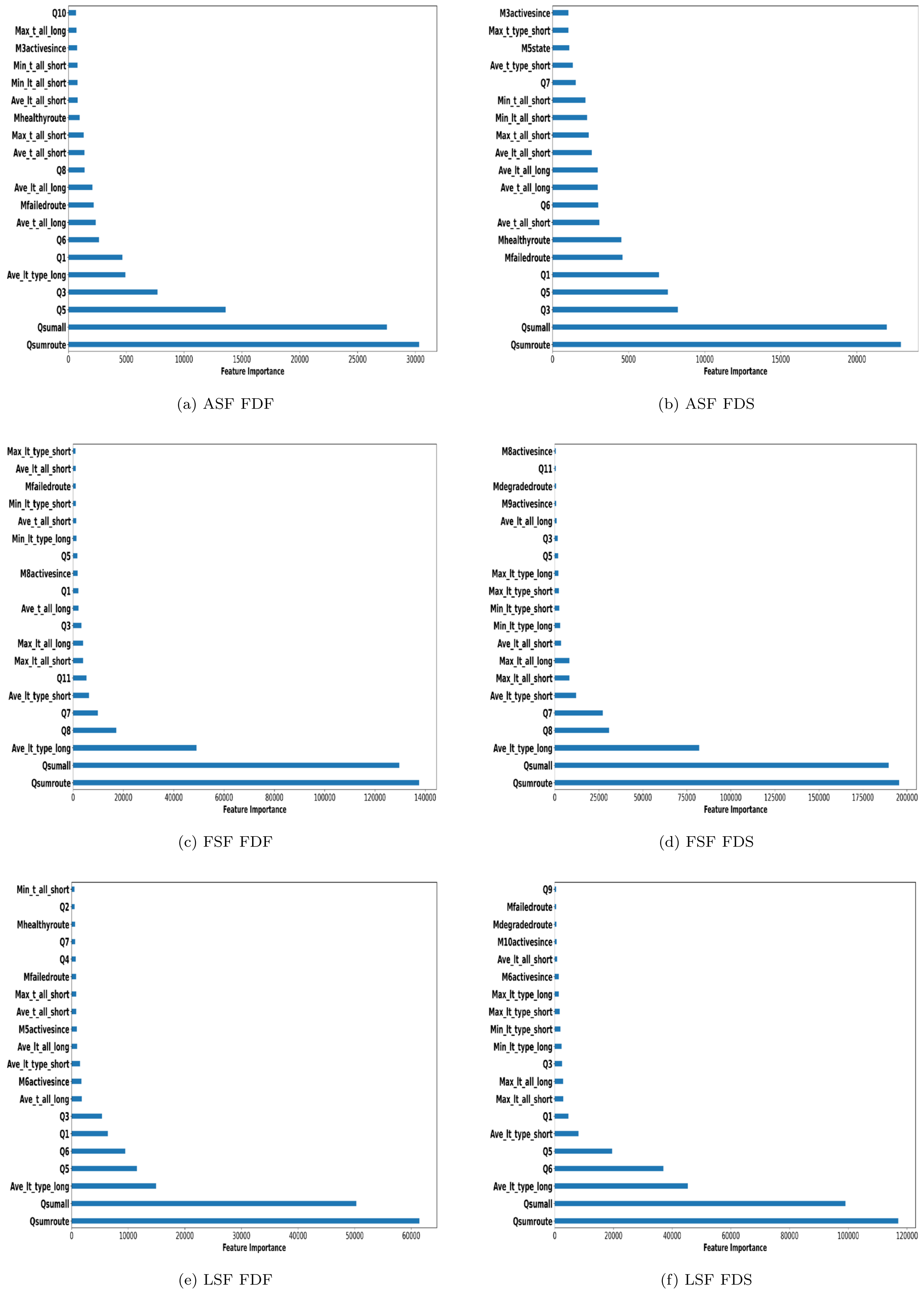

Fig. 9 The 20 most important features for lead time prediction in synthetic datasets (Feature importance is measured in terms of F value)

Table 8 Performance of HEDA against Ridge and RF benchmark SL algorithms and generalist DNN in synthetic datasets (average scores from 10-fold cross validation)

	Ridge			RF			DNN			HEDA		
Datasets	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE
ASF_FDF	0.50	6.29	4.90	0.91	2.63	1.81	0.93	2.35	1.63	0.94	2.21	1.45
ASF_FDS	0.44	6.15	4.72	0.90	2.54	1.73	0.93	2.22	1.55	0.93	2.10	1.40
FSF_FDF	0.71	7.82	6.11	0.96	2.75	1.88	0.97	2.49	1.77	0.98	2.29	1.57
FSF_FDS	0.80	7.26	5.75	0.97	2.70	1.83	0.94	3.21	1.81	0.93	3.22	1.63
LSF_FDF	0.63	6.94	5.51	0.93	2.92	1.99	0.95	2.61	1.81	0.95	2.43	1.61
LSF_FDS	0.74	7.42	5.79	0.96	2.92	1.98	0.97	2.62	1.84	0.97	2.42	1.64

Table 9 Performance of HEDA against Ridge and RF benchmark SL algorithms and generalist DNN in dataset ASF_FDF for different $\hat{\pi}_0$ values

	Ridge			RF			DNN			HEDA		
$\hat{\pi}_0$	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE
0.001	0.40	6.64	5.08	0.90	2.69	1.79	0.92	2.36	1.61	0.93	2.21	1.44
0.005	0.46	6.69	5.14	0.91	2.66	1.80	0.93	2.37	1.64	0.94	2.23	1.46
0.01	0.50	6.29	4.90	0.91	2.63	1.81	0.93	2.35	1.63	0.94	2.21	1.45
0.05	0.44	6.49	5.07	0.91	2.66	1.81	0.93	2.36	1.63	0.94	2.20	1.45
0.1	0.43	6.43	5.07	0.91	2.61	1.79	0.92	2.35	1.62	0.93	2.21	1.45

Table 10 Performance of HEDA against Ridge and RF benchmark SL algorithms and generalist DNN in dataset ASF_FDF for different $\hat{\pi}_{02}^i$ values

	Ridge			RF			DNN			HEDA		
$\hat{\pi}_{02}^i, \forall i$	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE
0.1	0.61	32.71	26.13	0.97	8.73	5.20	0.97	9.62	6.75	0.97	8.90	5.67
0.2	0.45	12.00	9.36	0.93	4.31	2.72	0.95	3.68	2.43	0.95	3.43	2.08
0.3	0.50	6.29	4.90	0.91	2.63	1.81	0.93	2.35	1.63	0.94	2.21	1.45
0.4	0.54	4.19	3.32	0.90	1.94	1.41	0.91	1.81	1.32	0.92	1.71	1.22
0.5	0.59	3.14	2.46	0.89	1.61	1.21	0.90	1.52	1.15	0.91	1.45	1.09

Table 11 Performance of HEDA against Ridge and RF benchmark SL algorithms and generalist DNN in dataset ASF_FDF for different $\hat{\mu}_i$ values

	Ridge			RF			DNN			HEDA		
$\hat{\mu}_i, \forall i$	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE	R^2	$RMSE$	MAE
12	0.78	7.74	6.21	0.97	2.84	1.95	0.97	2.81	2.03	0.97	2.61	1.82
15	0.54	7.11	5.62	0.92	2.87	1.98	0.94	2.57	1.80	0.95	2.40	1.58
18	0.50	6.29	4.90	0.91	2.63	1.81	0.93	2.35	1.63	0.94	2.21	1.45
21	0.40	6.05	4.69	0.90	2.51	1.72	0.92	2.21	1.54	0.93	2.10	1.40
24	0.46	2.88	2.23	0.86	1.48	1.12	0.87	1.44	1.09	0.87	1.39	1.05

datasets ASF_FDF and ASF_FDS. However, for datasets FSF_FDF and FSF_FDS, $Q7$ and $Q8$, and for datasets LSF_FDF and LSF_FDS, $Q5$ and $Q6$, seem to be the most important. This indicates that tracking slower machines could be more important since in datasets FSF_FDF and FSF_FDS, stations 7 and 8, which correspond to S33 and S34 in Fig. 4, and in datasets LSF_FDF and LSF_FDS, stations 5 and 6, which correspond to S29 and S30 in Fig. 4, are among the slower processing stations. The importance of slower machines is somewhat intuitive given that they are the bottlenecks in the system for lead times, which suggests the close monitoring of bottleneck stations in the production line for an accurate lead time prediction. When we look at features on machine states, we see that in datasets ASF_FDF and ASF_FDS, generic features such as “Mfailedroute” and “Mhealthyroute” that consider the stations along the route of the jobs are the most important. On the other hand, in other datasets in which stations differ with respect to their processing rates, station-dependent features such as “M8activesince” and “M6activesince” seem to be the most important. In case of performance-related features, we note that a variety of features are among the 20 most important features; we see both features relating to lead time and throughput, and to short-term and long-term trends, and to different statistics including averages, minimums, and maximums. This implies that including a diverse set of statistics on system performance might be useful.

The results we present on simulated datasets described in Table 7 confirms the superiority of HEDA to predict lead times from production state data. To understand how robust this performance is, we evaluate HEDA, the chosen benchmark SL algorithms and the generalist DNN with different simulated datasets which we generate by varying the model parameters. For this, we focus on the ASF_FDF dataset described in Table 7 and take a sensitivity analysis approach in which we vary one parameter at a time while keeping the rest of model parameters fixed. In Tables 9, 10 and 11, we present results from this investigation where $\hat{\pi}_0$, $\hat{\pi}^i_{02}$, and $\hat{\mu}_i$ are varied, respectively. We observe that HEDA is again the best performing prediction algorithm overall.

7 Discussion and conclusions

This paper introduces a novel hierarchical ensemble deep learning algorithm for data-driven lead time prediction. The approach enables real-time order-specific predictions by leveraging the state of production processes captured through sensor technologies. The effectiveness of our approach is demonstrated through two independent case studies. The first case study utilises the Bosch production line dataset which is one of the largest datasets available in the manufacturing literature, making this paper the first to predict lead times using this dataset. Unlike existing studies using this dataset, we cluster products into types based on the unique paths they follow amongst the stations. This approach is shown to be effective at addressing the dimensionality and sparsity issues inherent in the raw data. Our results highlight the superior performance of the proposed algorithm, which outperforms other prediction methods, notably reducing the mean absolute error.

In our second case study, we utilise knowledge of product types and their paths in the Bosch dataset to create a reliability-based make-to-order production system. We simulate outputs from this system to generate synthetic data for lead time prediction. By recording workload, machine health, and performance-related state information at order arrivals, we use these features to predict lead times. The superior performance of the hierarchical ensemble deep learning algorithm is validated on these synthetic datasets. Comparing predictions with the synthetic datasets to the Bosch dataset, we observe significantly higher prediction accuracy in the synthetic datasets, emphasising the importance of real-world data. Additionally, our study with the synthetic datasets allows us to gain insights into relative feature importance. This investigation reveals the potential significance of bottleneck stations’ features, suggesting that close monitoring of these stations can enhance system performance.

Given the high accuracy demonstrated by our hierarchical ensemble deep learning algorithm, future research can explore the development of similar advanced prediction algorithms. For instance, algorithms with more than two levels and additional base learners have the potential to further enhance predictions. Furthermore, alternative prediction algorithms beyond deep neural networks can be investigated. Future research can also focus on refining data-driven predictions that incorporate both lead time and fault information for product orders. This can involve accurately predicting lead times for non-faulty products and early detection of faults for faulty ones, leveraging the fault labels and lead time data available in the Bosch dataset. Another important research direction is the integration of lead time predictions with production system routing decisions. In systems with parallel machines or workstations, accurate lead time predictions based on real-time production data can enhance the utilisation of flexible routing and processing options.

Funding This work was supported by the Engineering and Physical Sciences Research Council, UK (grant number EP/V051113/1 -Productivity and Sustainability Management in the Responsive Factory). For the purpose of open access, the authors have applied a Creative

Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

Declarations

Competing Interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Usuga Cadavid JP, Lamouri S, Grabot B, Pellerin R, Fortin A (2020) Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. J Intell Manuf 31(6): 1531-1558. https://doi.org/10.1007/s10845-019-01531-7
2. Gyulai D, Pfeiffer A, Nick G, Gallina V, Sihn W (2018) Monostori L (2018) Lead time prediction in a flow-shop environment with analytical and machine learning approaches. IFAC-PapersOnLine 51(11):1029–1034. https://doi.org/10.1016/j.ifacol.2018.08.472. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM
3. Gao RX, Wang L, Helu M, Teti R (2020) Big data analytics for smart factories of the future. CIRP Annals 69(2):668–692. https://doi.org/10.1016/j.cirp.2020.05.002
4. Huang J, Chang Q, Arinez J (2020) Product completion time prediction using a hybrid approach combining deep learning and system model. Journal of Manufacturing Systems 57:311–322. https://doi.org/10.1016/j.jmsy.2020.10.006
5. Alenezi A, Moses SA, Trafalis TB (2008) Real-time prediction of order flowtimes using support vector regression. Computers & Operations Research 35(11):3489–3503. https://doi.org/10.1016/j.cor.2007.01.026. Part Special Issue: Topics in Real-time Supply Chain Management
6. Burggräf P, Wagner J, Koke B, Steinberg F (2020) Approaches for the prediction of lead times in an engineer to order environment - a systematic review. IEEE Access 8:142434–142445. https://doi.org/10.1109/ACCESS.2020.3010050
7. Öztürk A (2006) Kayalıgil, Özdemirel NE (2006) Manufacturing lead time estimation using data mining. European Journal of Operational Research 173(2):683–700. https://doi.org/10.1016/j.ejor.2005.03.015
8. Schuh G, Gützlaff A, Sauermann F, Kaul O (2020) Klein N (2020) Databased prediction and planning of order-specific transition times. Procedia CIRP 93(885–890):2020. https://doi.org/10.1016/j.procir.2020.04.026. 53rd CIRP Conference on Manufacturing Systems
9. Lingitz L, Gallina V, Ansari F, Gyulai D, Pfeiffer A, Sihn W (2018) Monostori L (2018) Lead time prediction using machine learning algorithms: a case study by a semiconductor manufacturer. Procedia CIRP 72:1051–1056. https://doi.org/10.1016/j.procir.2018.03.148. 51st CIRP Conference on Manufacturing Systems
10. Fang W, Guo Y, Liao W, Ramani K, Huang S (2020) Big data driven jobs remaining time prediction in discrete manufacturing system: a deep learning-based approach. International Journal of Production Research 58(9):2751–2766. https://doi.org/10.1080/00207543.2019.1602744
11. Wang C, Jiang P (2019) Deep neural networks based order completion time prediction by using real-time job shop RFID data. J Intell Manuf 30(3):1303–1318. https://doi.org/10.1007/s10845-017-1325-3
12. Bender J, Ovtcharova J (2021) Prototyping machine-learning-supported lead time prediction using AutoML. Procedia Computer Science, 180:649–655. https://doi.org/10.1016/j.procs.2021.01.287. Proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020)
13. Alshdaifat EAH (2015) Hierarchical ensemble classification: towards the classification of data collections that feature large numbers of class labels. PhD thesis, University of Liverpool
14. Liu KH, Zeng ZH, Ng VTY (2016) A hierarchical ensemble of ECOC for cancer classification based on multi-class microarray data. Information Sciences 349–350:102–118. https://doi.org/10.1016/j.ins.2016.02.028
15. Zhang L, Shah SK (2017) Kakadiaris IA (2017) Hierarchical multi-label classification using fully associative ensemble learning. Pattern Recognition 70:89–103. https://doi.org/10.1016/j.patcog.2017.05.007
16. Wang R, Li H, Lan R, Luo S, Luo X (2018) Hierarchical ensemble learning for Alzheimer's disease classification. In 2018 7th International Conference on Digital Home (ICDH), pages 224–229. https://doi.org/10.1109/ICDH.2018.00047
17. Postel M, Bugdayci B, Wegener K (2020) Ensemble transfer learning for refining stability predictions in milling using experimental stability states. Int J Adv Manuf Technol 107:4123–4139. https://doi.org/10.1007/s00170-020-05322-w
18. Manivannan S (2022) An ensemble-based deep semi-supervised learning for the classification of wafer bin maps defect patterns. Computers & Industrial Engineering 172:108614. https://doi.org/10.1016/j.cie.2022.108614
19. Ren L, Cui J, Sun Y, Cheng X (2017) Multi-bearing remaining useful life collaborative prediction: a deep learning approach. J Manuf Syst 43:248–256. https://doi.org/10.1016/j.jmsy.2017.02.013. High Performance Computing and Data Analytics for Cyber Manufacturing
20. Shajalal Md, Hajek P, Abedin MZ (2021) Product backorder prediction using deep neural network on imbalanced data. International Journal of Production Research 1–18. https://doi.org/10.1080/00207543.2021.1901153
21. Mangal A, Kumar N (2016) Using big data to enhance the Bosch production line performance: a Kaggle challenge. In 2016 IEEE international conference on big data (big data), pp 2029–2035. https://doi.org/10.1109/BigData.2016.7840826
22. Zhang D, Xu B, Wood J (2016) Predict failures in production lines: a two-stage approach with clustering and supervised learning. In 2016 IEEE international conference on big data (big data), pp 2070–2074. https://doi.org/10.1109/BigData.2016.7840832
23. Carbery CM, Woods R, Marshall AH (2019) A new data analytics framework emphasising preprocessing of data to generate insights into complex manufacturing systems. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 233(19–20):6713–6726. https://doi.org/10.1177/0954406219866867
24. Huang X, Zanni-Merk C, Crémilleux B (2019) Enhancing deep learning with semantics: an application to manufacturing time series analysis. Procedia Computer Science 159:437–446. ISSN 1877-0509. https://doi.org/10.1016/j.procs.2019.09.

198. Knowledge-based and intelligent information & engineering systems: proceedings of the 23rd international conference KES2019
25. Ge N, Li G, Zhang L, Liu Y (2021) Failure prediction in production line based on federated learning: an empirical study. J Intell Manuf. https://doi.org/10.1007/s10845-021-01775-2
26. Arredondo F, Martinez E (2010) Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. Computers & Industrial Engineering 58(1):70–83. https://doi.org/10.1016/j.cie.2009.08.005
27. Manitz M (2008) Queueing-model based analysis of assembly lines with finite buffers and general service times. Computers & Operations Research 35(8):2520–2536. https://doi.org/10.1016/j.cor.2006.12.016
28. Xi X, Chen M, Zhang H, Zhou D (2018) An improved non-Markovian degradation model with long-term dependency and item-to-item uncertainty. Mechanical Systems and Signal Processing 105:467–480. https://doi.org/10.1016/j.ymssp.2017.12.017
29. Ghaleb M, Taghipour S, Sharifi M, Zolfagharinia H (2020) Integrated production and maintenance scheduling for a single degrading machine with deterioration-based failures. Computers & Industrial Engineering 143:106432. https://doi.org/10.1016/j.cie.2020.106432
30. Deloux E, Castanier B, Bérenguer C (2009) Predictive maintenance policy for a gradually deteriorating system subject to stress. Reliability Engineering & System Safety 94(2):418–431. https://doi.org/10.1016/j.ress.2008.04.002

Authors and Affiliations

Ayse Aslan[1] · **Gokula Vasantha**[1] · **Hanane El-Raoui**[2] · **John Quigley**[2] · **Jack Hanson**[3] · **Jonathan Corney**[3] · **Andrew Sherlock**[4]

Gokula Vasantha
g.vasantha@napier.ac.uk

Hanane El-Raoui
hanane.el-raoui@strath.ac.uk

John Quigley
j.quigley@strath.ac.uk

Jack Hanson
jack.hanson@ed.ac.uk

Jonathan Corney
j.r.corney@ed.ac.uk

Andrew Sherlock
a.sherlock@strath.ac.uk

[1] School of Computing, Engineering and the Built Environment, Edinburgh EH10 5DT, UK

[2] Department of Management Science, University of Strathclyde, Glasgow G1 1XQ, UK

[3] School of Engineering, The University of Edinburgh, Edinburgh EH8 9YL, UK

[4] National Manufacturing Institute Scotland, University of Strathclyde, Glasgow PA4 9LJ, UK