End-to-end predict-and-optimize dynamic predictive maintenance planning integrating prognostics - the case of short-range electric aircraft with Lithium-ion batteries

Simon van Oosterom[1*], Mihaela Mitici[2]

1: Air Traffic Management and Airports, Royal Netherlands Aerospace Centre (NLR), Anthony Fokkerweg 2, 1059 CM, Amsterdam, The Netherlands
2: Faculty of Science, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
** Corresponding author: simon.van.oosterom@nlr.nl*

Abstract

Modern assets are continuously monitored by sensors. As a result, large datasets on the health condition of these systems are often available. Using supervised machine learning, recent studies have leveraged such data to generate remaining useful life (RUL) prognostics. Here, the focus of the machine learning regressors is on achieving prognostics of high accuracy. Once obtained, in a second stage, these prognostics are usually integrated into maintenance planning optimisation models. However, aiming for high accuracy prognostics in a first stage does not guarantee that the maintenance costs are also minimized in a second, maintenance planning stage. To address this, we propose an end-to-end, dynamic framework for the predictive maintenance problem that integrates the planning stage into the prediction stage. For this, the maintenance costs are directly estimated from sensor data, instead of being derived based on RUL prognostics. We apply our end-to-end framework for the maintenance planning of a fleet of electric Vertical Take-Off and Landing (eVTOL) aircraft equipped with Lithium-ion batteries. We show that, when compared to state-of-the-art prognostics-based maintenance planning, the proposed framework reduces the number of battery failures by 24% and the total maintenance costs by 9.4%. Overall, our framework proposes an effective, data-driven paradigm for an end-to-end predictive maintenance planning.

Keywords: predictive maintenance planning, end-to-end maintenance, electrical Vertical Take-off and Landing aircraft, prognostics, predict-and-optimize

1. Introduction

Modern assets are continuously monitored by an increasing number of sensors that generate large amounts of data. Think of monitoring emerging technologies, for example, wind turbines, high-speed railway systems and trains, or aircraft [1, 2]. Such data is used to foresee faults and failures, or to obtain Remaining Useful Life (RUL) prognostics. This information enables the optimization of maintenance planning of assets [3, 4, 5, 6]. This is often referred to as predictive maintenance planning.

Recent studies on predictive maintenance propose two-stage approaches where the RUL prognostics and maintenance planning are decoupled [7, 8, 9, 10, 11, 12]. In the first stage, condition monitoring measurements are used to estimate the Remaining Useful Life (RUL) of each individual asset using, for example, supervised machine learning [11]. The RUL prognostics are typically estimated as a point values,

for example for turbofan engines using a convolution neural network [13, 11]. More recent studies have also focussed on developing probabilistic estimates of the RUL, by e.g. [4]. In the second stage, these RUL prognostics are integrated into maintenance planning for a set of assets, with limited maintenance capacity. This is implemented as e.g., a threshold-based approach [9], an integer linear program [11], a Markov Decision Process [14, 15], a Model Predictive Control [16], a Reward-Renewal process [17]. The advantage of such a two-stage approach is that is allows for a modular combination of data-driven, asset-specific RUL prognostics and advanced maintenance planning models.

In the first stage of such approaches, the focus is on obtaining RUL prognostics with a high accuracy such that the difference between the actual RUL and the estimated RUL is minimal. However, in the second (planning) stage, an improvement in the accuracy of RUL estimates does not guarantee an improvement in the maintenance planning decisions and the associated costs. The performance gap in terms of these costs is measured by the *decision regret*: the difference between the true cost obtained with an optimal decision based on perfect RUL prognostics (Oracle) and the true cost obtained with decisions based on the data-driven generated RUL estimates. The reason behind this gap is the fact that the RUL prognostics, even when their accuracy is high, are obtained with machine learning models that are oblivious of the objectives of the maintenance planning models. Overcoming this problem requires feedback from the maintenance planning model (second stage) towards the RUL regressor (first stage).

For general optimization problems with uncertain variables, end-to-end frameworks have been recently introduced to account for the shortcomings of two-stage planning approaches [18, 19, 20]. This class of problems is referred to as Predict-Then-Optimize (PTO) problems. In end-to-end optimization, similar to the two-stage approaches, a machine learning regressor is trained to estimate the model parameters from measured data. In such end-to-end approaches, however, the regressor is trained with knowledge of the optimization problem itself. This is performed by integrating the decision regret in the training of the regressor. Several end-to-end algorithms have been developed for this, such as the *Smart Predict-then-Optimize* (SPO+) loss [21, 22, 23], the *differentiable black-box solver* [24], and the *differentiable perturbed optimizer* (DPO) [25]. For an overview of end-to-end optimization frameworks, see Tang and Khalil [26].

In this paper, we propose an end-to-end framework for data-driven, dynamic predictive maintenance planning of a set of assets. Our approach uses health-monitoring measurements to estimate maintenance planning costs, while incorporating the decision regret into a loss function which expands the existing SPO+ loss. This approach is distinct from existing studies on predictive maintenance that use a two-stage approach where data is used in the first stage to estimate the best possible RUL estimates, while these estimates are further used for maintenance planning in a second stage [4, 11, 13]. In this paper, instead, *decision regret-minimizing prognostics are made*. Compared to standard end-to-end optimization frameworks, our approach differs in two ways. Firstly, we consider the case when the sensor measurements are uncorrelated across assets. This is a relevant consideration in practice, where assets (in our case, batteries) are used independently of each other. This consideration is leveraged by designing and training the machine learning algorithm to generate maintenance costs for each asset separately. This approach, however, is different from conventional end-to-end frameworks, where all unknown parameters are estimated by a single machine learning model, allowing faster training. Secondly, we assume that the maintenance planning decisions can be re-evaluated over time. This is leveraged by formulating the problem as a rolling horizon model, instead of the conventional single stage decision problem used for end-to-end optimization frameworks. Lastly, for

every stage in the rolling horizon, we formulate the maintenance planning problem as an Integer Linear Program (ILP).

To the best of our knowledge, this is an innovative approach for maintenance planning problems specifically designed for predictive maintenance. When compared with a Markov Decision Process, this approach scales well for a larger number of assets, and does not require the formulation of states and transition probabilities. Compared to an end-to-end unsupervised learning approach to optimize maintenance planning, such as reinforcement learning [27], the model requires less data to be trained on, and provides more robust maintenance policies. Concluding, this approach offers a method to perform predictive maintenance for applications with large instances, with limited training data, and without the need to artificially construct states and transition probabilities. Additionally, the use of an ILP provides transparency to the overall planning approach.

We illustrate our approach for battery maintenance of a fleet of electric Vertical Takeoff and Landing (eVTOL) aircraft. These aircraft are an emerging technology considered for passenger transport, delivery, or emergency support. We compare our approach with conventional two-stage maintenance planning approaches, which make use of RUL prognostics. The results show that our end-to-end framework leads to a 21% reduction in the total number of unforeseen Li-ion battery failures when compared to the (best-case) two-stage maintenance planning framework. In this context, we also show that our approach reduces the decision regret (of maintenance costs) by 30%.

The main contributions of this paper are:

- We develop an end-to-end, dynamic framework for predictive maintenance of a set of assets, where the health condition measurements of the assets are directly informing the maintenance scheduling model. This is in contrast with the majority of studies on predictive maintenance that use a two-stage approach where in the first stage data-driven RUL estimates are obtained, which are informing the maintenance scheduling model in a second, independent stage [28, 9, 11].

- We show that predictive maintenance planning for a set of assets can be formulated as a general end-to-end optimization problem [26]. Distinct from general end-to-end optimization problems, we *decompose* the end-to-end optimization problem into identical maintenance planning problems for each individual asset. This is enabled by the fact that the sensor measurements are uncorrelated across assets.

- We apply our framework to plan maintenance for Li-ion batteries of a fleet of eVTOL aircraft. We show that this method increases the reliability of the maintenance schedules while reducing costs compared to a two-stage predictive maintenance approach.

The remainder of the paper is structured as follows. In Section 2, the maintenance planning problem for a set of assets is introduced. In Section 3 we discuss the predictive maintenance planning model formulation. We discuss the conventional two-stage predictive maintenance planning models, based on estimates of the Remaining Useful Life of the assets. After this, we introduce our proposed end-to-end predictive maintenance planning framework. This method is compared with other stochastic predictive maintenance planning frameworks. In Section 4, we apply our framework for the case of maintenance planning of Li-ion batteries of a fleet of eVTOL aircraft. The results of this case study are presented in Section 5. Conclusions are given in Section 6.

2. Predictive maintenance planning for a set of assets

We consider a set $\mathcal{A}$ of assets and a maintenance planning horizon of days $\mathcal{D}$. The state-of-health of each asset degrades over time until an asset failure (the end-of-life is reached). When this occurs, the asset is immediately maintained to a good-as-new condition and a large penalty $c_{\text{unscheduled}}$ is incurred.

Each asset is continuously monitored by sensors: let $\mathbf{x}_a \in \mathbb{R}^n$ denote the measurements of asset $a \in \mathcal{A}$ at the start of day $d \in \mathcal{D}$, where n denotes the number of sensor measurement features. These measurements provide information on the health condition of the assets.

To avoid failures, this data may be leveraged to preemptively maintain assets. The costs of preventive maintenance are $c_{\text{replace}} \ll c_{\text{unscheduled}}$. At most H assets can be replaced per day. For both working and failed assets, maintenance takes an entire day.

At the start of the current day $d_0 \in \mathcal{D}$, maintenance is planned for days $d_0 + 1, d_0 + 2, ...$ Maintenance that needs to be performed exactly on the current day d_0 is fixed. Next, we shift to the new current day $d_0 + 1$ and the maintenance planning horizon $d_0 + 2, d_0 + 3, ...$

We are interested in determining which assets should be scheduled for replacement by leveraging the measurements $(\mathbf{x}_a)_{a\in\mathcal{A}}$. We aim to determine an optimal replacement day for each assets such that (1) failures are avoided due to high failure costs, while (2) assets are used as long as possible, or equivalently the wasted life of the assets is minimized.

3. Predictive maintenance planning model formulation

In this section we introduce four modeling paradigms for the dynamic predictive maintenance planning problem from Section 2. Firstly, we consider the maintenance planning model for the idealized case in which the *actual* RUL of each asset is known in advance (an oracle perspective). We next consider the same planning problem, but for the case when the RUL is not known in advance (the realistic perspective). For this, we propose three algorithms where: (i) only a *point estimate* of the actual RUL of the assets is obtained at various moments in time based on the sensor measurements of these assets; these estimates then become the input of a maintenance planning model (Section 3.2.1), (ii) the *distribution* of the actual RUL of the assets is obtained at various moments in time; these estimated distributions then become the input of a maintenance planning model (Section 3.2.2), and (iii) the sensor measurements are directly used to estimate the cost coefficients of the same maintenance planning model (Section 3.3) without the intermediate step of estimating the RUL of the assets. This section concludes with a conceptual comparison between this method and other stochastic predictive maintenance planning methods.

Algorithm 1 shows the rolling horizon predictive maintenance framework. We consider planning maintenance of a set $\mathcal{A}$ of assets. At the start of the current day $d_0 \in \mathcal{D}$, the (unknown) Remaining Useful Life (RUL), i.e., the days until the end-of-life, of asset a is denoted by r_a. The assets which failed on $d_0 - 1$ and the assets which are scheduled for maintenance on d_0 are denoted by $\mathcal{A}^f$ and $\mathcal{A}^s$, respectively. At the start of the current day d_0 we plan replacements of the assets $a \in \mathcal{A}' = \mathcal{A} \setminus (\mathcal{A}^f \cup \mathcal{A}^s)$ within a time window $\mathcal{D}_{d_0} = \{d_0 + 1, ..., d_0 + k\}$, $k \geq 1$. The replacements planned exactly on d_0 are fixed. After planning and performing maintenance, we shift to the next day $d_0 + 1$, the next planning window $\mathcal{D}_{d_0+1} := \{d_0 + 2, ..., d_0 + k + 1\}$, and re-compute the maintenance schedule, see also Figure 1.

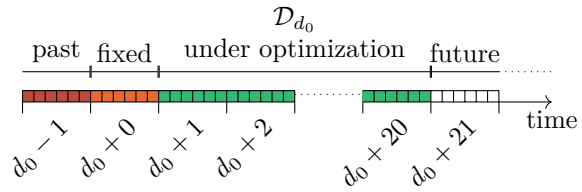


Figure 1: Illustration of the maintenance planning time window at current day d_0, with $k = 20$, $\mathcal{D}_{d_0} = \{d_0+1, d_0+2, ...d_0+20\}$.

For maintenance planning, we consider a linear programming formulation. At d_0, the decision variables considered are:

$$y_{ad} = \begin{cases} 1, & \text{if maintenance for asset } a \text{ is scheduled on day } d_0 + d, \\ 0, & \text{otherwise,} \end{cases} \tag{1a}$$

$$y_a^{\text{postpone}} = \begin{cases} 1, & \text{if maintenance for asset } a \text{ is not scheduled in } \mathcal{D}_{d_0}, \\ 0, & \text{otherwise,} \end{cases} \tag{1b}$$

with $d \in \{1, ..., k\}$. For any asset $a \in \mathcal{A}'$, let $\mathbf{y}_a = (y_{a1}, ..., y_{ak}, y_a^{\text{postpone}})^T$. Let $\mathbf{y} = (\mathbf{y}_a)_{a\in\mathcal{A}'}$ denote the vector of all decision variables.

Algorithm 1: Rolling horizon predictive maintenance planning for a set of assets.

Data: Set of assets $\mathcal{A}$, days of operations $\mathcal{D}$, hangar capacity H, maintenance costs $c_{\text{unscheduled}}$ and c_{replace}.
Result: Maintenance schedule at total cost C

Initialize $C = 0$;
Initialize scheduled maintenance days for assets for $a \in \mathcal{A}$: $d_a \leftarrow$ `None` ;
for *days* $d_0 \in \mathcal{D}$ **do**
  Obtain the condition measurements $\mathbf{x}_a$ from the assets $a \in \mathcal{A}$;
  Obtain the failed assets $\mathcal{A}^f \subset \mathcal{A}$, and the to-be-maintained assets $\mathcal{A}^s = \{a \in \mathcal{A} : d_a = d_0\}$;
  Set $\mathcal{A}' = \mathcal{A} \setminus (\mathcal{A}^f \cup \mathcal{A}^s)$;
  Plan maintenance for assets $\mathcal{A}'$, obtain the decision variables y and y^{postpone} (Section 3.1, 3.2, 3.3) ;
  for *asset* $a \in \mathcal{A}'$ **do**
    Set $d_a \leftarrow d_0 + d$ if $y_{ad} = 1$ or $d_a \leftarrow$`None` if $y_a^{postpone} = 1$;
  end
  Use assets $\mathcal{A}'$ for operations;
  Perform maintenance on assets $\mathcal{A}^f \cup \mathcal{A}^s$;
  Set $C \leftarrow C + c_{\text{unscheduled}}|\mathcal{A}^f| + c_{\text{replace}}|\mathcal{A}^s|$;
end

3.1. Oracle maintenance planning

We first consider the situation in which the *actual* RUL of the assets is known in advance. As before, let r_a denote the actual RUL of asset $a \in \mathcal{A}'$ at d_0. We aim to determine an optimal maintenance schedule for all assets $a \in \mathcal{A}'$. Clearly, the optimal moment to replace asset a is at $d_0 + r_a$. The cost to maintain an

asset $a \in \mathcal{A}'$ in d days such that $d + d_0 \in \mathcal{D}_{d_0}$, given RUL r_a is:

$$c(d, r_a) = \frac{c_{\text{replace}}}{L}(r_a - d)^+ + \frac{c_{\text{unscheduled}} - c_{\text{replace}}}{L}(r_a - d)^-, \tag{2}$$

where c_{replace} and $c_{\text{unscheduled}}$ are the costs of a planned replacement, and of an unscheduled replacement due to a failure, respectively (see Section 2). Also, L denotes a nominal useful life of the assets. In case the replacement is postponed to the next planning window, the following cost c^{postpone} is incurred:

$$c^{\text{postpone}}(r_a) = \frac{c_{\text{unscheduled}} - c_{\text{replace}}}{L}(r_a - k - 1)^-. \tag{3}$$

For any asset $a \in \mathcal{A}'$, let:

$$\mathbf{c}_a = \mathbf{h}(r_a) := (c(r_a, 1), ..., c(r_a, k), c^{\text{postpone}}(r_a))^T \in \mathbb{R}^{k+1}, \tag{4}$$

Let $\mathbf{c} = (\mathbf{c}_a)_{a \in \mathcal{A}'}$ denote the vector of all maintenance costs.

We consider the following optimization model to plan maintenance for the set $\mathcal{A}$ of assets:

$$\min_{y} \quad \mathbf{c}^T \mathbf{y}, \tag{5a}$$

$$\text{s.t.} \quad \mathbb{1}^T \mathbf{y}_a = 1 \qquad \forall a \in \mathcal{A}', \tag{5b}$$

$$\sum_{a \in \mathcal{A}} y_{ad} \leq H \qquad \forall d \in \{1, ..., k\}, \tag{5c}$$

$$\mathbf{y}_a \in \{0, 1\}^{k+1} \qquad \forall a \in \mathcal{A}'. \tag{5d}$$

Here, equation (5a) gives the cost of the planned maintenance. Constraints (5b) ensures that for each asset, maintenance is planned or postponed, where $\mathbb{1}$ denotes the all-ones vector. Constraints (5c) ensure that the hangar capacity is not exceeded. Last, constraints (5d) ensure that the variables take binary values.

3.2. Two-stage algorithms for the predictive maintenance problem

The RUL r_a of an asset $a \in \mathcal{A}$ is in fact not known in advance. By extension, the cost vector $\mathbf{c}_a$ is also unknown. What is often available are sensor measurements $\mathbf{x}_a$ continuously recorded for an asset a. As such, predictive maintenance planning is a Predict-Then-Optimize (PTO) problem [26]. Many studies leverage this data $\mathbf{x}_a$, using for example machine learning regressors $g(\mathbf{x}_a, \theta)$ to estimate the RUL of asset $a \in \mathcal{A}'$. Subsequently, the estimated RUL is considered in maintenance planning models to generate maintenance cost estimates $\hat{\mathbf{c}}$ [11]. We note that this approach is a *two-stage* PTO approach in the sense that the training of the machine learning regressors for RUL estimation, and the generation of the RUL estimates is performed independent of the maintenance planning models and without knowledge of c_{replace}, $c_{\text{unscheduled}}$, and the average asset lifetime at the moment of RUL generation.

In the following, we distinguish between i) a two-stage maintenance planning where g generates a point estimate of the RUL (2S-P algorithm, Section 3.2.1), and ii) a two-stage maintenance planning where g generates a distribution of the RUL (2S-D algorithm, Section 3.2.2). All algorithms are summarized in Table 1.

Table 1: The Oracle, the two-stage maintenance planning with RUL Point estimates $\hat{r}_a$ (2S-P, Sec. 3.2.2), the 2Stage maintenance planning with RUL Distribution estimates $\hat{p}_a$ (2S-D, Sec. 3.2.2), and the End-to-end Maintenance (E2E-M, Sec. 3.3) planning for asset a in a rolling horizon framework. Here, $\mathbf{x}_a$ is the sensor data of asset a, g, and f are machine learning regressors with parameters θ, and h are cost functions.

	Oracle	Two-stage planning		E2E-M planning
		2S-P	2S-D	
Data-driven estimated variable	n.a.	RUL point estimate $\hat{r}_a$	RUL distribution estimate $\hat{p}_a^{\mathrm{R}}$	Maintenance cost coefficients $\hat{\mathbf{c}}_a$
RUL estimate	actual RUL r_a	$\hat{r}_a = g_{\mathrm{2SP}}(\mathbf{x}_a, \theta_{\mathrm{2SP}})$ (first stage)	$\hat{p}_a^{\mathrm{R}} = g_{\mathrm{2SD}}(\mathbf{x}_a, \theta_{\mathrm{2SD}})$ (first stage)	none
Maintenance costs estimate	$\mathbf{c}_a = \mathbf{h}(r_a)$	$\hat{\mathbf{c}}_a = \mathbf{h}(\hat{r}_a)$ (second stage)	$\hat{\mathbf{c}}_a = \mathbf{h}_{\mathrm{2SD}}(\hat{p}_a^{\mathrm{R}})$ (second stage)	$\hat{\mathbf{c}}_a = f(\mathbf{x}_a, \theta_{\mathrm{E2E}})$

3.2.1. ***Two Stage*** *predictive maintenance planning with RUL* ***Point*** *estimates (2S-P)*

Conventionally, a two-stage approach for the predictive maintenance problem uses RUL point estimates (a 2S-P approach) [5]. Figure 2 shows such an 2S-P approach.

Stage 1:

A machine learning regressor g_{2SP} with parameters θ_{2SP} generates point estimates of the RUL of an asset $a \in \mathcal{A}'$: $\hat{r}_a = g_{\mathrm{2SP}}(\mathbf{x}_a, \theta_{\mathrm{2SP}})$, where $\mathbf{x}_a$ are sensor measurements. It estimates the RUL by minimizing a loss function $\mathcal{L}_{\mathrm{2SP}}$, given by the 2-norm:

$$\mathcal{L}_{\mathrm{2SP}}(r, \hat{r}) = \|r - \hat{r}\|_2. \tag{6}$$

We note that the training g is performed independently from the subsequent maintenance planning problem. The model g is both trained and tested in this phase.

Stage 2:

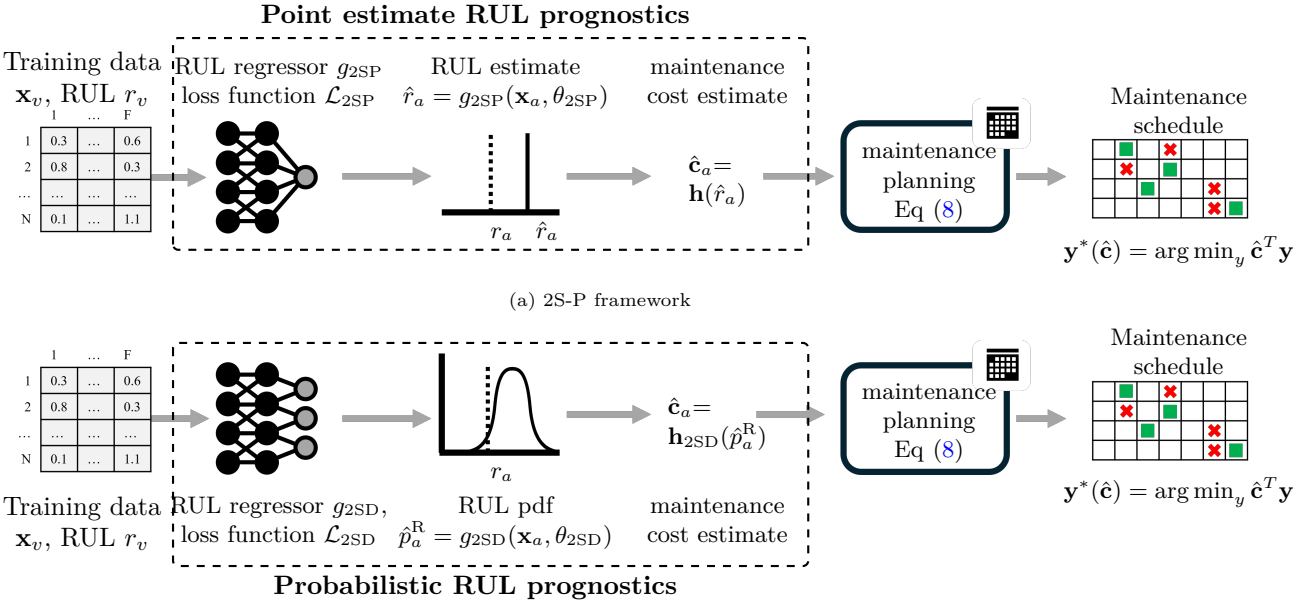
After g_{2SP} is trained, it is deployed for maintenance (Algorithm 1, line 7). First, the regressor g_{2SP} generates RUL point estimates $\hat{r}_a$ for asset $a \in \mathcal{A}'$ based on the current asset measurements $\mathbf{x}_a$. These RUL estimates are now integrated in the maintenance planning model, by applying $\mathbf{h}(\cdot)$ to the RUL estimates (see also Eq. (4) in the Oracle planning model where the actual RUL r_a is assumed known):

$$\hat{\mathbf{c}}_a = \mathbf{h}(\hat{r}_a). \tag{7}$$

The combined cost estimates for all assets are denoted by $\hat{\mathbf{c}} := (\hat{\mathbf{c}}_a)_{a \in \mathcal{A}'}$. We now consider the maintenance problem $\min_{\mathbf{y}} \hat{\mathbf{c}}^T \mathbf{y}$ subject to the constraints (5b)-(5d). Let $\mathbf{y}^*(\hat{\mathbf{c}})$ denote the optimal maintenance decisions for this problem:

$$\mathbf{y}^*(\hat{\mathbf{c}}) = (\mathbf{y}_a^*(\hat{\mathbf{c}}))_{a \in \mathcal{A}'} := \arg\min_{\mathbf{y}} \hat{\mathbf{c}}^T \mathbf{y} \quad \text{s.t.} \quad (5b) - (5d). \tag{8}$$

With these maintenance decisions $\mathbf{y}^*(\hat{\mathbf{c}})$, maintenance is planned in the next step of Algorithm 1.



(a) 2S-P framework

(b) 2S-D framework

Figure 2: Two-stage approach for predictive maintenance planning problem, using RUL point estimates (2S-P) or probabilistic estimates (2S-D). A machine learning (ML) regressor is trained on a dataset $\mathcal{D}$ of sensor measurements to predict the RUL by minimizing a loss function $\mathcal{L}_{2SP}$ or $\mathcal{L}_{2SD}$. After training, the RUL estimates are used to specify a maintenance cost function $h(\hat{\mathbf{c}}_a)$ to be minimized.

*3.2.2. **Two Stage** predictive maintenance planning with RUL **Distribution** estimates (2S-D)*

More recent studies have explored the possibility of using probabilistic RUL prognostics for maintenance planning, i.e., the distribution of the RUL is estimated in the first phase [8]. This approach is also shown in Figure 2.

Stage 1:

Instead of generating RUL point estimates, a machine learning regressor g_{2SD} is trained to generate a probability density of the RUL of an asset $a \in \mathcal{A}'$. This model has parameters θ_{2SD}, and the estimated RUL probability density is denoted by $\hat{p}_a^R = g_{2SD}(\mathbf{x}_a, \theta_{2SD})$ for an asset a. The model is trained to minimize a loss function $\mathcal{L}_{2SD}$. This loss function depends on the used model, an example of $\mathcal{L}_{2SD}$ being the negative log-loss function [29].

Stage 2:

In the second stage, after training the model, g_{2SD} is applied in maintenance planning (Algorithm 1, line 7). The regressor generates for an asset $a \in \mathcal{A}'$ the pdf of its RUL $\hat{p}_a^R$ for the current measurements $\mathbf{x}_a$. Using $\hat{p}_a^R$, the expected maintenance costs $\hat{\mathbf{c}}_a$ are computed. To perform maintenance on day $d_0 + d$, the associated cost estimate is given by:

$$\hat{c}_{\text{dist}}(d, \hat{p}_a^R) = \sum_{r \geq 0} c(d, r)\hat{p}_a^R(r), \tag{9}$$

where $c(d, r)$ is given in Equation (2), r denotes the RUL, and $\hat{p}_a^{\mathrm{R}}(r)$ is the probability that the RUL of asset a is r days. The expected costs of postponing maintenance are given by:

$$\hat{c}_{\mathrm{dist}}^{\mathrm{postpone}}(\hat{p}_a^{\mathrm{R}}) = \sum_{r=0}^{k} c^{\mathrm{postpone}}(r)\hat{p}_a^{\mathrm{R}}(r), \tag{10}$$

where $c^{\mathrm{postpone}}(r)$ is given in Equation (3). The estimated maintenance costs $\hat{\mathbf{c}}_a$ are given by:

$$\hat{\mathbf{c}}_a = \mathbf{h}_{\mathrm{2SD}}(\hat{p}_a^{\mathrm{R}}) := (\hat{c}_{\mathrm{dist}}(1, \hat{p}_a^{\mathrm{R}}), \hat{c}_{\mathrm{dist}}(2, \hat{p}_a^{\mathrm{R}}), ..., \hat{c}_{\mathrm{dist}}(k, \hat{p}_a^{\mathrm{R}}), \hat{c}_{\mathrm{dist}}^{\mathrm{postpone}}(\hat{p}_a^{\mathrm{R}}))^T. \tag{11}$$

Using these cost $\hat{\mathbf{c}}_a$, the maintenance planning problem in (8) is considered and maintenance decisions $\mathbf{y}^*(\hat{\mathbf{c}})$ are obtained. With these maintenance decisions $\mathbf{y}^*(\hat{\mathbf{c}})$, maintenance is planned in the next step of Algorithm 1.

3.3. ***End-to-end*** *predict-and-optimize predictive* ***maintenance*** *planning (E2E-M)*

The merits of the two-stage approaches for maintenance planning in Sections 3.2.1 and 3.2.2 have been discussed in detail in several studies such as Consilvio et al. [11] and Lee et al. [28]. However, these two-stage approaches do not guarantee that the estimates $\hat{\mathbf{c}}$ are as close to $\mathbf{c}$ as possible, or that the actual maintenance cost is minimized. Specifically, these approaches do not optimize for the decision regret, i.e., the cost gap between the true optimal solution and the one acquired using $\hat{\mathbf{c}}$, which is defined as:

$$\mathcal{L}_{\mathrm{regret}}(\hat{\mathbf{c}}, \mathbf{c}) = \sum_{a \in \mathcal{A}} \mathbf{c}_a^T \mathbf{y}_a^*(\hat{\mathbf{c}}) - \mathbf{c}_a^T \mathbf{y}_a^*(\mathbf{c}), \tag{12}$$

with

$$\mathbf{y}^*(\hat{\mathbf{c}}) = (\mathbf{y}_a^*(\hat{\mathbf{c}}))_{a \in \mathcal{A}'} := \arg\min_{\mathbf{y}} \left\{ \hat{\mathbf{c}}^T \mathbf{y} \quad \text{s.t.} \quad (5b) - (5d) \right\},$$
$$\mathbf{y}^*(\mathbf{c}) = (\mathbf{y}_a^*(\mathbf{c}))_{a \in \mathcal{A}'} := \arg\min_{\mathbf{y}} \left\{ \mathbf{c}^T \mathbf{y} \quad \text{s.t.} \quad (5b) - (5d) \right\}.$$

To address this, we propose a novel, end-to-end PTO algorithm to address the rolling horizon predictive maintenance problem by directly minimizing the actual maintenance costs based on the sensor measurements, without the need to generate RUL estimates. As opposed to the two-stage approaches 2S-P and 2S-D, the machine learning regressor in this framework is trained to minimize the decision regret $\mathcal{L}_{\mathrm{regret}}(\hat{\mathbf{c}}, \mathbf{c})$.

Figure 3 shows the proposed end-to-end framework. During training, the regressor $f(\mathbf{x}_a, \theta_{\mathrm{E2E}})$ generates maintenance cost estimates. Using these estimates, the model receives feedback on the decision regret from the maintenance planning problem. Based on the feedback on the regret, the parameters θ_{E2E} of the regressor f are updated. We note that this feedback loop is not present for the 2-stage approaches, where after the RUL is estimated in the first stage, there is no further feedback between the first stage and the second, planning stage.

We note that the regressor generates maintenance cost estimates for a single asset $\hat{\mathbf{c}}_a$, instead of being trained on the ILP with multiple assets (5a)-(5d). As only a subset of the ILP is considered, this potentially limits the performance of this approach. This choice is motivated by two reasons. First, developing these regressors poses tractability issues. During training, the number of possible scenario's for which it needs to

be applied increases exponentially with the number of assets. For a larger number of assets, training the regressor thus becomes intractable. Additionally, the fact that the number of assets $\mathcal{A}'$ which need to be scheduled for maintenance varies in size between days $d \in \mathcal{D}$. Thus, generating maintenance cost estimates for multiple assets requires training separate regressors for each possible size of $|\mathcal{A}'|$. As such, even though training the regressor on the ILP with multiple assets should generate better cost estimates, we develop the method for a single-asset cost regressor.

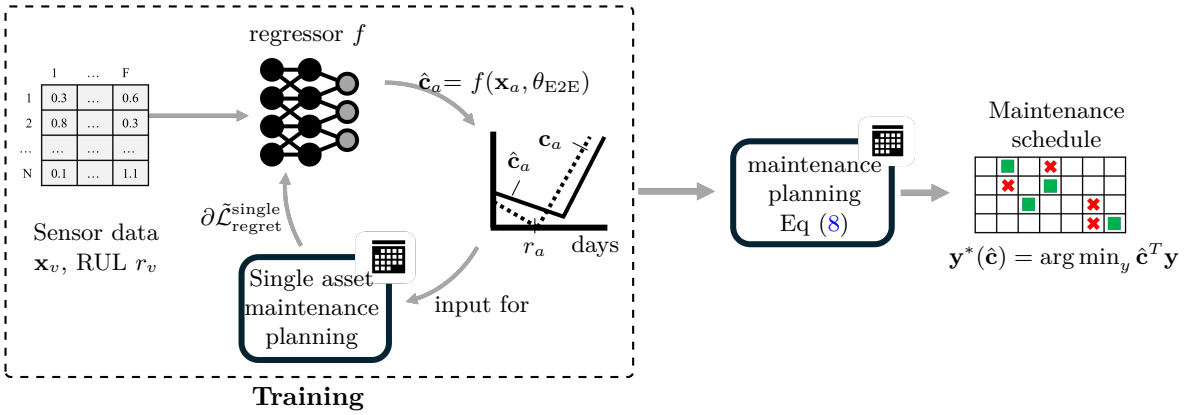


Figure 3: *End-to-end maintenance* framework for the predictive maintenance planning problem. A machine learning regressor f is trained on sensor data, with regret of the maintenance planning problem in a feedback loop. f generates maintenance cost estimates, using the loss function $\tilde{\mathcal{L}}^{\text{single}}_{\text{regret}}$, a proxy for $\mathcal{L}_{\text{regret}}$. After training, the cost estimates are used to plan maintenance.

Training the maintenance cost regressor

A machine learning regressor f with parameters θ_{E2E} is trained to generate estimates for the maintenance costs for a single asset a: $\hat{\mathbf{c}}_a = f(\mathbf{x}_a, \theta_{\text{E2E}})$. We consider the following maintenance problem for a single asset:

$$\mathbf{y}_a^{*,\text{single}}(\mathbf{c}_a) = \arg\min_{y_a} \mathbf{c}_a^T\mathbf{y}_a \qquad \text{s.t.} \quad \mathbf{y}_a \in B_{k+1} := \{\mathbf{y}_a : \mathbb{1}^T\mathbf{y}_a = 1, \quad \mathbf{y}_a \in \{0,1\}^{k+1}\}. \tag{13}$$

We note that eq. (5c) in the maintenance model is not mentioned here since we consider a single asset.

To train f, information of the decision regret of this problem is received in a feedback loop, with the regret for a single asset being defined as:

$$\mathcal{L}^{\text{single}}_{\text{regret}}(\hat{\mathbf{c}}_a, \mathbf{c}_a) = \mathbf{c}_a^T\mathbf{y}_a^{*,single}(\hat{\mathbf{c}}_a) - \mathbf{c}_a^T\mathbf{y}_a^{*,single}(\mathbf{c}_a), \tag{14}$$

where $\mathbf{y}_a^{*,single}(\hat{\mathbf{c}}_a)$ and $\mathbf{y}_a^{s*}(\mathbf{c}_a)$ denote the optimal decisions for (13) for the estimated and true costs, respectively.

The model f is trained by applying the gradient descent algorithm to $\mathcal{L}^{single}_{\text{regret}}$, see Algorithm 2. However, $\mathcal{L}^{single}_{\text{regret}}$ is not differentiable. To use the gradient descent algorithm, $\mathcal{L}^{single}_{\text{regret}}$ is substituted by an approximation

$\tilde{\mathcal{L}}^{single}_{\text{regret}}(\hat{\mathbf{c}}_a, \mathbf{c}_a)$ introduced in Elmachtoub et al. [21]:

$$\mathcal{L}^{single}_{\text{regret}}(\hat{\mathbf{c}}_a, \mathbf{c}_a) \approx \tilde{\mathcal{L}}^{single}_{\text{regret}}(\hat{\mathbf{c}}_a, \mathbf{c}_a) := -\min_{\mathbf{y}_a \in B_{k+1}} \{(2\hat{\mathbf{c}}_a - \mathbf{c}_a)^T \mathbf{y}_a\} + 2\hat{\mathbf{c}}_a^T \operatorname*{arg\,min}_{\mathbf{y}_a \in B_{k+1}} \{\mathbf{c}_a^T \mathbf{y}_a\} - \min_{\mathbf{y}_a \in B_{k+1}} \{\mathbf{c}_a^T \mathbf{y}_a\}, \tag{15}$$

$$= -\min\{2\hat{\mathbf{c}}_a - \mathbf{c}_a\} + 2\hat{\mathbf{c}}_a^T \operatorname*{arg\,min}_{\mathbf{y}_a \in B_{k+1}} \{\mathbf{c}_a^T \mathbf{y}_a\} - \min\{\mathbf{c}_a\},$$

which is differentiable w.r.t. $\hat{\mathbf{c}}_a$, convex and has the following subgradient:

$$2 \operatorname*{arg\,min}_{\mathbf{y}_a \in B_{k+1}} \{\mathbf{c}_a^T \mathbf{y}_a\} - 2 \operatorname*{arg\,min}_{\mathbf{y}_a \in B_{k+1}} \{(2\hat{\mathbf{c}}_a - \mathbf{c}_a)^T \mathbf{y}_a\} \in \frac{\partial \tilde{\mathcal{L}}^{\text{single}}_{\text{regret}}(\hat{\mathbf{c}}_a, \mathbf{c}_a)}{\partial \hat{\mathbf{c}}_a}, \tag{16}$$

which can easily be evaluated. This subgradient is substituted in:

$$\frac{\partial \mathcal{L}^{\text{single}}_{\text{regret}}(\hat{\mathbf{c}}_a)}{\partial \theta_{\text{E2E}}} \approx \frac{\partial \tilde{\mathcal{L}}^{\text{single}}_{\text{regret}}(\hat{\mathbf{c}}_a)}{\partial \hat{\mathbf{c}}_a} \frac{\partial \hat{\mathbf{c}}_a}{\partial \theta_{\text{E2E}}},$$

such that the gradient descent algorithm (Algorithm 2) can be used.

Algorithm 2: Batch gradient descent algorithm for training the machine learning regressor used in the E2E predict-and-optimize maintenance framework.

Data: Machine learning algorithm f, batched dataset of RUL-labeled sensor data $(\mathbf{x}_a, r_a)$, number of iterations i, learning rate α
Result: Trained f

```
Initialize parameters θ_E2E;
for iterations i do
    for batch of data {(x_a, r_a)} do
        Sample data (x_a, r_a) from the batch;
        Compute the cost coefficients c_a = (c(r_a, 1), ..., c(r_a, k), c^postpone(r_a))^T (Eq. (2) and (3));
        Predict the cost coefficients ĉ_a = f(x_a, θ_E2E);
        Determine y*_a(c_a) and y*_a(ĉ_a);
        Compute ∂L̃^single_regret / ∂θ_E2E using Eq. (16);
        Let θ_E2E ← θ_E2E − α * ∂L̃^single_regret / ∂θ_E2E;
    end
end
```

Maintenance scheduling

Once regressor f is trained, it is deployed for maintenance planning (Algorithm 1, line 7). The estimated costs $\hat{\mathbf{c}}_a$ are computed for each asset from the measurements $\mathbf{x}_a$. The combined estimates are denoted by $\hat{\mathbf{c}} := (\hat{\mathbf{c}}_a)_{a \in \mathcal{A}'}$. Using these cost estimates, the maintenance planning problem Eq. (8) is solved and maintenance decisions $\mathbf{y}^*(\hat{\mathbf{c}})$ are obtained. With these maintenance decisions $\mathbf{y}^*(\hat{\mathbf{c}})$, maintenance is planned in the next step of Algorithm 1.

Relation to alternative stochastic maintenance planning methods

The method described above has a number of advantages when compared to other stochastic predictive

maintenance frameworks, besides the two-stage approaches, such as Markov Decision Processes (MDPs) Reinforcement Learning (RL).

Markov Decision Processes have been applied in the context of predictive maintenance [14, 15], and have been found to be well-suited for low-dimensional problems. However, the method becomes impractical when considering a large number of states. Additionally, using an MDP requires the specification of the states and transition probabilities with limited connection to the actual degradation process of the asset once the system becomes more complex. Formulating a regressor f to generate maintenance cost estimates does not require model simplifications to define these transition probabilities.

Reinforcement Learning has also been successfully used in the context of predictive maintenance [27, 30], where it is noted that the algorithm works well in partially observable systems. However, it is noted that RL requires to be trained on large datasets with a variety of failure paths to obtain robust results. This limits the number of applications for which it can be used. By including the optimization model (13) during the training phase of the algorithm, we provide structure to the optimization model, requiring smaller training datasets. Additionally, by training and deploying the regressor f on a single asset, the proposed model also scales well.

Overall, the approach presented in this section offers a method to perform predictive maintenance for applications with large instances, with limited training data, and without the need to artificially construct states and transition probabilities.

4. Case study: predictive maintenance planning for eVTOL Lithium-ion batteries

In this section, we will apply our end-to-end predictive maintenance planning framework to a fleet of electric Vertical Take-Off and Landing (eVTOL) aircraft.

eVTOLs are short-range electric aircraft which fulfill a need for greener and quieter flights and address increasing concerns about urban traffic congestion. These aircraft are designed to carry a payload up to 800 kg over a distance of up to 100 km. Currently, several legacy manufacturers as well as start-ups are developing eVTOLs [31]. Envisioned applications are (among others) on-demand urban passenger transportation and emergency response services [32, 33, 34].

Battery health management is one of the most important obstacles for eVTOL operations [35]. Currently, Lithium-ion batteries are the most frequently considered batteries for eVTOLs, due to their relatively good performance and affordability. However, due to the high performance requirements for eVTOL operations, these batteries are prone to quick degradation, which may lead to safety concerns. As such, monitoring the health of the battery in combination with predictive maintenance scheduling, as we propose, is crucial.

4.1. Dataset on condition monitoring of Li-ion batteries of eVTOLs

We consider the condition dataset for the Sony-Murata 18650 VTC-6 Li-Ion batteries [36]. These batteries are cycled through an eVTOL mission simulation in the A^3 Vahana eVTOL, designed by Acubed/Airbus. Each mission consists of a charging phase and a flight phase. The charging phase consists of: constant current charging (CC), constant voltage charging (CV) and a rest period. The flight phase consists of: the takeoff, cruise and landing phase.

A total of 22 batteries are considered. Each battery is cycled though missions with different parameters, referred to as the mission profile (MP) or Vahana (VAH). The parameters of these profiles can be seen in

Table 2. Three of these batteries are cycled according to the baseline mission profile (MP1, MP13, and MP14). The other mission profiles cover a wide range of operational conditions, with varying cruise time, battery power during the different flight stages, charging rates, and battery surface temperature.

Table 2: Mission profile characteristics of the Sony-Murata 18650 Li-ion batteries used for Vahana operations simulation [36].

	Cruise time	Power Take-off	Power Cruise	Power Landing	CC	CV	Temperature	Vahana	#Missions	#Capacity tests
MP1	**800s**	**54W**	**16W**	**54W**	**1C**	**4.2V**	**25°C**	**VAH01**	**847**	**17**
MP2	**1000s**	54W	16W	54W	1C	4.2V	25°C	VAH02	625	13
MP3	800s	90% of 54W	90% of 16W	90% of 54W	1C	4.2V	25 °C	VAH05	1615	29
MP4	800s	54W	16W	54W	**0.5C**	4.2V	25°C	VAH06	9290	20
MP5	800s	54W	16W	54W	1C	**4V**	25°C	VAH07	339	6
MP6	800s	54W	16W	54W	1C	4.2V	**20°C**	VAH09	8527	27
MP7	800s	54W	16W	54W	1C	4.2V	**30°C**	VAH10	1431	28
MP8	800s	80% of 54W	80% of 16W	80% of 54W	1C	4.2V	25 °C	VAH11	2249	45
MP9	**400s**	54W	16W	54W	1C	4.2V	25°C	VAH12	2349	46
MP10	**600s**	54W	16W	54W	1C	4.2V	25°C	VAH13	1042	20
MP11	**1000s**	54W	16W	54W	1C	4.2V	25°C	VAH15	554	11
MP12	800s	54W	16W	54W	**1.5C**	4.2V	25°C	VAH16	559	11
MP13	**800s**	**54W**	**16W**	**54W**	**1C**	**4.2V**	**25°C**	**VAH17**	**1002**	**20**
MP14	800s	54W	16W	54W	**1.5C**	4.2V	25°C	VAH20	611	12
MP15	**1000s**	54W	16W	54W	1C	4.2V	25°C	VAH22	579	11
MP16	800s	54W	16W	54W	1C	**4V**	25°C	VAH23	697	14
MP17	800s	54W	16W	54W	**0.5C**	4.2V	25°C	VAH24	801	16
MP18	800s	54W	16W	54W	1C	4.2V	**20°C**	VAH25	554	11
MP19	**600s**	54W	16W	54W	1C	4.2V	25°C	VAH26	1164	23
MP20	**800s**	**54W**	**16W**	**54W**	**1C**	**4.2V**	**25°C**	**VAH27**	**587**	**12**
MP21	800s	90% of 54W	90% of 16W	90% of 54W	1C	4.2V	25°C	VAH28	1182	23
MP22	800s	54W	16W	54W	1C	4.2V	**35°C**	VAH30	919	18

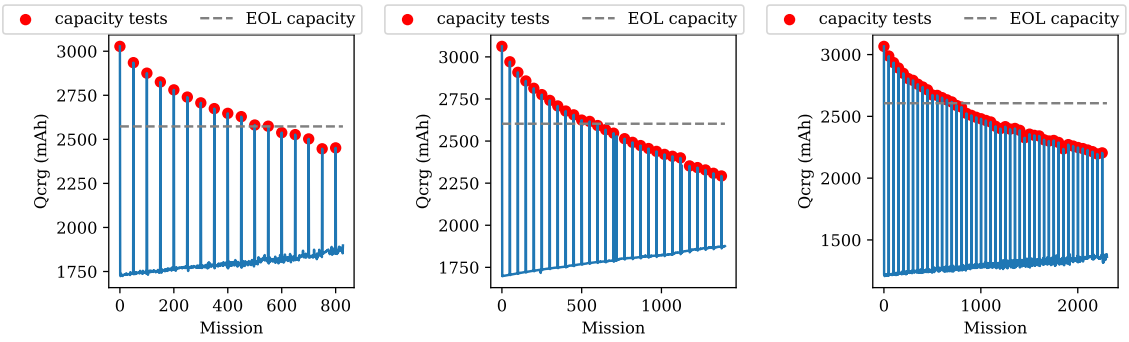
4.1.1. Sensor measurements

During the missions, the battery is monitored with a number of sensors. Every 10 seconds, the following are recorded: the cell voltage, the current, the energy and charge supplied during the charging phase, the energy and charge supplied by the battery during the flight phase, and the surface temperature.

Over time, the charge capacity of the batteries decrease. However, the true capacity cannot be measured without fully discharging the battery. In order to do this, a capacity test is performed every 50 missions. During this test, the battery is discharged until the voltage drops below 2.5V. This is followed by a cooldown period. After this, the battery is recharged until it is at full capacity.

The eVTOL batteries can only be used if its charge capacity is high enough. We say that the battery reaches its end-of-life (EOL) once its capacity is 85% of the original capacity.

Figure 4 shows an example of how the battery capacity decreases with each mission (on MP1/VAH01). The total capacity drops from 3000 mAh on the first capacity test/mission to 2450 mAh on the 17th capacity test (800th mission). The battery EOL is reached at the 12th capacity test (750th mission).



(a) VAH01 (b) VAH10 (c) VAH12

Figure 4: Battery capacity for batteries VAH01, VAH10, and VAH12. The values measured at the capacity tests are shown as dots. Batteries VAH01, VAH10, and VAH12 have a useful life of 600, 600, and 750 cycles, respectively.

4.1.2. Feature engineering

The sensor measurements have been processed into 33 features. Let $\mathcal{F} = \{MP1, MP2, ..., MP22\}$ denote the set of mission profiles. For each mission profile $f \in \mathcal{F}$, let M_f denote the number of missions performed by f. For each $f \in \mathcal{F}$ and mission number $1 \leq m \leq M_f$, we consider the following features:

Charging phase features. We store the amount of charge supplied $Qcrg^{f,m}$, the duration of each charging segment (CC, CV, Rest) $\Delta t^{segment,f,m}$, and the last measured charge capacity $Qlast^{f,m}$.

Flight phase features. For each mission segment during flight (take-off, cruise, and landing), we store the duration $\Delta t^{segment,f,m}$. Additionally, we store the maximum, minimum, mean, and variance of the battery voltage in each phase, denoted by $V_{max}^{segment,f,m}$, $V_{min}^{segment,f,m}$, $V_{mean}^{segment,f,m}$, and $V_{var}^{segment,f,m}$, respectively. Additionally, we store the maximum, minimum, and variance of the extracted charge during each segment, denoted by $Qdis_{max}^{segment,f,m}$, $Qdis_{min}^{segment,f,m}$ and $Qdis_{var}^{segment,f,m}$, respectively. Last, for each segment, we record the maximum cell temperature $T_{max}^{segment,f,m}$.

Time feature. Finally, for each mission, we store the number of cycles since the last capacity test $\Delta C^{f,m}$. This measures the uncertainty within the difference between the current capacity and the last measured capacity $Qlast^{f,m}$.

4.1.3. Feature selection

From the generated features, we select the most important half to generate prognostics for maintenance planning. This is done based on the Shapley values [37], which are given in Table 3. The features $V_{min}^{take-off}$ through $Qdis_{max}^{take-off}$ are selected.

Last, these features are subjected to normalization. This is done with z-score normalization, which scales the data to a mean of 0 and a variance of 1 [38]. In order to capture the degradation of the batteries, the features from the current mission, as well as 50 missions (1 capacity test) back are used. This is the data that shall be used in order to perform predictive maintenance for the eVTOL batteries.

4.2. Predictive maintenance planning for a fleet of eVTOLs

In this section, we illustrate the predictive maintenance planning for eVTOL batteries using (i) machine learning to predict point estimates of the battery RUL, and integrate this in a integer linear program to

Feature	Shapley	Feature	Shapley	Feature	Shapley	Feature	Shapley
$V_{min}^{take-off}$	**95.4**	$V_{var}^{landing}$	**59.3**	$Qdis_{max}^{cruise}$	45.9	$\Delta^{landing}$	19.4
$V_{mean}^{take-off}$	**94.7**	$V_{mean}^{landing}$	**57.4**	T_{max}^{cruise}	45.4	Δ^{CC}	12.9
$Qlast$	**93.4**	$V_{max}^{take-off}$	**57.2**	$T_{max}^{landing}$	41.1	$Qdis_{var}^{cruise}$	3.5
$V_{var}^{take-off}$	**92.4**	ΔC	**56.1**	$T_{max}^{take-off}$	38.8	$Qdis_{max}^{landing}$	2.4
V_{max}^{cruise}	**87.8**	$V_{min}^{landing}$	**51.6**	Δ^{rest}	36.5	$Qdis_{mean}^{cruise}$	2.3
$Qcrg$	**87.1**	$Qdis_{var}^{take-off}$	**47.7**	$V_{max}^{landing}$	35.5	V_{var}^{cruise}	2.2
Δ^{CV}	**86.5**	$Qdis_{mean}^{take-off}$	**46.5**	$\Delta^{take-off}$	23.4	$Qdis_{var}^{landing}$	1.4
V_{min}^{cruise}	**78.8**	$Qdis_{max}^{take-off}$	**45.9**	Δ^{cruise}	19.7	$Qdis_{mean}^{landing}$	1.2
V_{mean}^{cruise}	**63.8**						

Table 3: SHAP values (importance) for the 33 considered features; top 50% of the features are selected for maintenance planning prognostics (in **bold**).

plan maintenance (2S-P), (ii) machine learning to predict the distribution of the battery RUL, and integrate this in a integer linear program to plan maintenance (2S-D), and (iii) machine learning to directly estimate the maintenance cost coefficients of an integer linear program that plans maintenance (E2E-M).

4.2.1. Parameters of the eVTOL fleet operations and maintenance

We consider maintenance planning for a fleet of eVTOLs $\mathcal{A}$. Each eVTOL is equipped with one of the batteries from Section 4.1. The fleet consists of 25 eVTOLs. Each eVTOL performs 10 trips per day, five round trips to and from the hub, such that a capacity test is performed every five days.

Each eVTOL battery is monitored by a number of sensors, from which 17 features are deduced, as described in Section 4.1. Based on this battery health data, maintenance may be planned. Replacing a battery costs $c_{\text{replace}} = 100$. To perform scheduled battery maintenance, $H = 1$ hangar is available. When a breakdown occurs, at the EOL, the eVTOL battery is immediately replaced at a cost of $c_{\text{unscheduled}} = 1000$. As described in Section 2, maintenance takes one day and may be planned one day in advance. We use a rolling horizon of $k = 10$ days to plan maintenance. A total planning horizon of $\mathcal{D} = 10$ years is considered.

A six-fold cross validation is used to train and test the regressors. For each fold, six eVTOLs are randomly used for testing, while ensuring that one baseline mission profile (VAH01, VAH17 or VAH27) is used. The other eVTOLs are used for training. The testing eVTOLs for each fold are given in Table 4.

Table 4: eVTOL batteries in the test set of each fold.

Fold	**Testing baseline battery**	**Testing other batteries**
1	VAH01	VAH02, VAH13, VAH20, VAH28, VAH30
2	VAH01	VAH05, VAH06, VAH13, VAH15, VAH16
3	VAH17	VAH19, VAH11, VAH22, VAH23, VAH25
4	VAH17	VAH02, VAH06, VAH20, VAH26, VAH30
5	VAH27	VAH05, VAH12, VAH15, VAH16, VAH24
6	VAH27	VAH10, VAH12, VAH22, VAH24, VAH25

4.2.2. Two stage predictive maintenance planning with RUL point estimates (2S-P)

For the 2S-P algorithm, as in Section 3.2.1, we use a Long Short-Term Memory (LSTM) regressor $g_{\text{2SP}}(\cdot, \theta_{\text{2SP}})$ to generate RUL point estimates of the eVTOL batteries. As before, θ_{2SP} denotes the param-

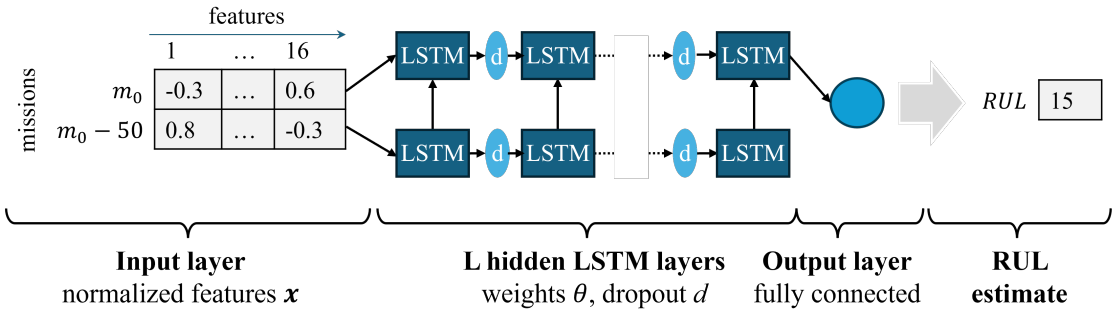


Figure 5: LSTM architecture for the RUL regressor used in the two-stage predict-then-optimize algorithm for predictive maintenance planning. It is used i) without random dropout during testing, to obtain RUL point prognostics $g_{2\text{SP}}$, and ii) with random dropout during testing, to obtain probabilistic RUL prognostics $g_{2\text{SD}}$.

eters of the LSTM.

The LSTM is a recurrent neural network (RNN), capable of learning long-term dependencies [39]. The LSTM solves the vanishing and exploding gradient problems by adding a memory cell and gate mechanism. A generic architecture of the LSTM and the architecture of a LSTM unit can be found in Yu et al. [40]. This architecture has been shown to be particularly well suited for identifying the degradation over time and estimating the RUL for Li-ion batteries [41].

Figure 5 shows the architecture of our LSTM network. The input of the network consists of the normalized battery sensor measurements from the current mission m and fifty missions $m-50$ (one capacity test) ago. This data (see Section 4.1) is fed through L layers of 2 LSTM units. To avoid overfitting, a Monte Carlo dropout layer is added after every LSTM layer except the last. After this, a fully connected layer is applied, which outputs the RUL point estimates of the batteries. The loss function of the network minimizes the deviation from the true RUL (Eq. (6)).

4.2.3. Two stage predictive maintenance planning with RUL distribution estimates (2S-D)

To obtain RUL distribution estimates for the 2S-D algorithm, as in Section 3.2.2, we use two regressors: a LSTM with Monte Carlo dropout in the testing phase [42], and a Mixture Density Network (MDN) [29].

LSTM with Monte Carlo dropout in the testing phase:

The LSTM network from Section 4.2.2 with Monte Carlo dropout in the testing phase is used to estimate the distribution of the RUL of the batteries (probabilistic RUl prognostics).

Mixture Density Network:

Mixture Density Networks (MDNs) are feed forward neural networks that generate the distributions of non-Gaussian unknowns [29]. The MDN estimates the RUL distribution as a linear combination of several Gaussian distributions. We have trained an MDN to obtain probabilistic RUL prognostics, as shown in Figure 6. Specifically, the sensor measurements $\mathbf{x}_a$ are fed through L fully connected hidden layers. After this, it is fed through an MDN output layer to J normal distributions, consisting of a three-fold output. These are the means $\mu_j(\mathbf{x}_a, \theta_{2\text{SD}})$, the standard deviations $\sigma_j(\mathbf{x}_a, \theta_{2\text{SD}})$ and the mixture coefficients $\alpha_j(\mathbf{x}_a, \theta_{2\text{SD}})$, with $j = 1, ...J$. The probability density of the output (RUL) is given by:

$$\hat{p}_a^{\mathrm{R}}(r_a|\mathbf{x}_a,\theta_{\mathrm{2SD}}) = \sum_{j=1}^{J} \alpha_j(\mathbf{x}_a,\theta_{\mathrm{2SD}})\phi(r_a|\mu_j(\mathbf{x}_a,\theta_{\mathrm{2SD}}),\sigma_j(\mathbf{x}_a,\theta_{\mathrm{2SD}})), \tag{17}$$

where ϕ denotes the pdf of a normal distribution. The network is trained on the loss function:

$$\mathcal{L}_{MDN}(r_v,\mathbf{x}_v) = -\log \hat{p}_a^{\mathrm{R}}(r_v|\mathbf{x}_v,\theta_{\mathrm{2SD}}), \tag{18}$$

which is the negative log-likelihood of r_v given the probability density function $\hat{p}_a^{\mathrm{R}}$.

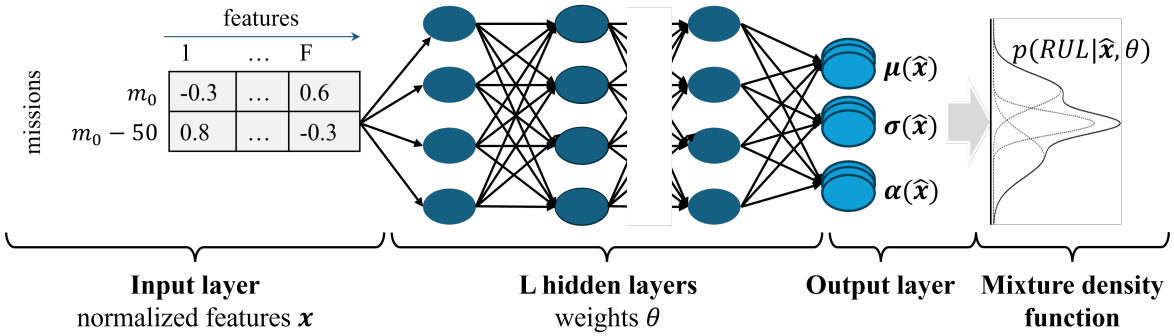


Figure 6: MDN architecture for the RUL regressor used in the two-stage predict-then-optimize algorithm for predictive maintenance planning with probabilistic RUL prognostics, g_{2SD}.

4.2.4. End-to-end predictive maintenance planning (E2E-M)

For the E2E-M algorithm, see Section 3.3, we directly obtain estimates of the maintenance costs in the planning model using an LSTM regressor. This LSTM $f(\cdot,\theta_{\mathrm{E2E}})$ has parameters θ_{E2E}, see also Figure 5. The architecture is similar to the network used for the 2S-P algorithm, but has a different output layer, i.e. the cost coefficients of the maintenance planning ILP model.

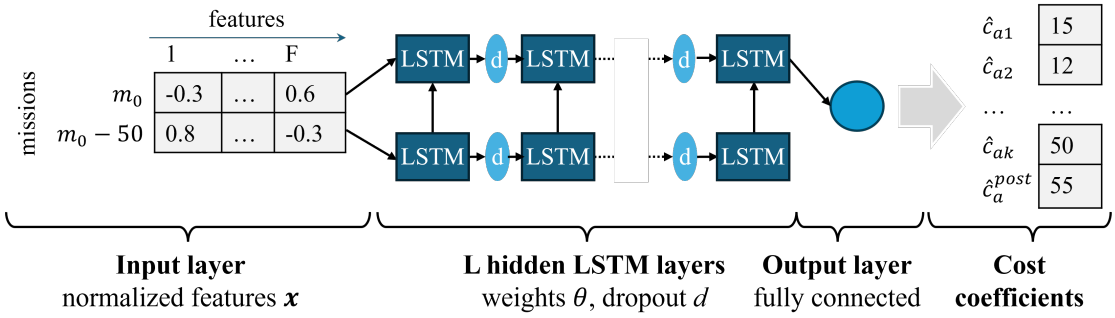


Figure 7: LSTM architecture used for the *end-to-end maintenance* framework.

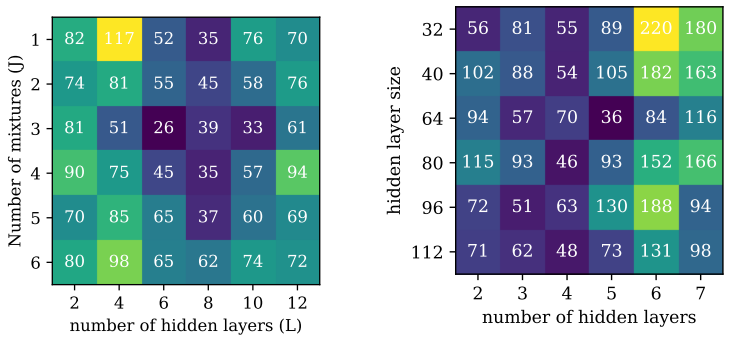
4.3. Hyperparameter tuning

A hyperparameter tuning has been performed on each algorithm from Section 4.2. A grid-based search was used to perform this.

For the LSTM used to obtain point estimate and probabilistic RUL prognostics (Section 4.2.2, 4.2.3), the following parameters are used. We consider $L = 4$ hidden LSTM layers. The layers have 176, 126, 116, and 110 units, respectively. During training and testing, a dropout rate of 0.5 is applied after the second layer. The LSTM is optimized using the RMSprop algorithm [43] with a learning rate of 0.01.

For the MDN used to obtain probabilistic RUL prognostics (Section 4.2.3), the following parameters are used. We consider $L = 6$ dense hidden layers. The layers have 24, 122, 116, 116, 90, and 38 units, respectively. The first five layers use the ReLu activation function, and the last one a tanh activation function. The output of the MDN consists of a mixture of $J = 3$ normal distributions. The MDN is optimized using the RMSprop [43] algorithm with a learning rate of 0.01. Figure 8a shows the results of the hyperparameter tuning grid search. It shows the CRPS (Continuous Ranked Probability Score), for the different configurations of the MDN during this phase.

For the LSTM used to obtain maintenance cost estimates (Section 4.2.4), the following parameters are used. We consider $L = 5$ hidden layers. The size of the output of each hidden layer, including the last, is 64 units. As the LSTM architecture is relatively large compared with the number of batteries, overfitting of the model is avoided by using a dropout rate of 0.2 during the testing phase. The LSTM is optimized using the Adam algorithm [43] with a learning rate of 0.005. Figure 8b shows the results of the hyperparameter tuning grid search, expressed as the decision regret. The figure shows a large decision regret for 6 and 7 hidden layers, indicating that this region is prone to overfitting.



(a) Continuous Ranked Probability Score for the different configurations of the MDN used to obtain RUL estimates.

(b) Regret score (%) for the different configurations of configurations of the LSTM used to obtain maintenance cost estimates.

Figure 8: Results of the hyperparameter tuning grid search for the MDN used to obtain RUL estimates, and the LSTM used to obtain maintenance cost estimates.

5. Results

5.1. Comparing end-to-end (E2E-M) and two-stage (2S-P, 2S-D) maintenance planning algorithms

Each algorithm (Oracle planning, two-stage with RUL point (2S-P) and distribution (2S-D) estimates, and end-to-end maintenance (E2E-M)) has applied to 100 simulations for eVTOL fleet maintenance planning for a planning period of $\mathcal{D} = 10$ years. Table 5 shows the average results of each algorithm over all 100 runs. It shows the number of batteries replaced as well, the lifetime of the batteries once they were replaced,

and the accompanying costs. The used batteries and costs have been split into scheduled and unscheduled maintenance.

Table 5: Maintenance performance indicators for the planning algorithms and regressors. The number of maintained batteries (on-time, broken down, and total) are given, together with the average used lifetime of the batteries, the maintenance costs, and the maintenance cost regret w.r.t. the Oracle.

Method		Maintained batteries (#)			Battery lifetime used [%]	costs	regret [%]
	Regressor	on time	broken down	total			
Oracle	-	1659.9	4.5	1664.4	97.9	170490.0	0
2S-P	LSTM	1125.2	594.6	1719.8	94.8	707120.0	315
2S-D	LSTM	1692.9	107.7	1800.6	88.9	276990.0	62
2S-D	MDN	1740.1	79.2	1819.3	90.2	253210.0	49
E2E-M	LSTM	1699.7	60.2	1759.9	91.2	230170.0	35

Table 5 shows that the Oracle planning algorithm requires (on average) 1664.4 batteries during the 10 years of operations. Of these, only a small number is not replaced before breakdown (4.5), leading to a total maintenance cost of 170490. It shows that form the other PTO methods, the End-to-End Maintenance algorithm leads to the fewest breakdowns (60.2) and the lowest regret: 35%. The two stage method with RUL distribution prognostics leads, with both the MDN and LSTM, to a larger number of breakdowns, with a corresponding regret of 49% and 62%, respectively. The two stage method with RUL point estimates uses the lowest amount of batteries (with a 94.8% utilization) but incurs a large number of breakdowns.

To explain the results, we use the preferred maintenance day. The preferred maintenance day for an eVTOL, d^* is defined as the day with the lowest maintenance costs:

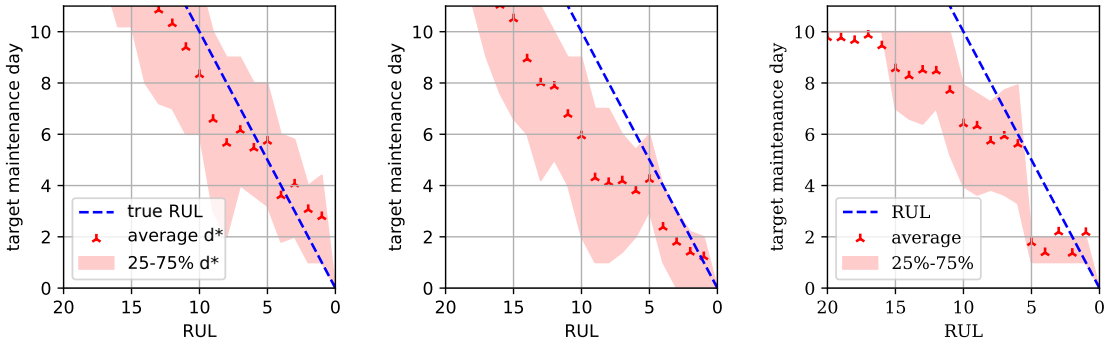
$$d^* = \begin{cases} \arg\min_d \hat{\mathbf{c}}_{ad}, & \text{if } \min_d \hat{\mathbf{c}}_{ad} \leq \hat{\mathbf{c}}_a^{\text{postpone}} \\ \text{post} & \text{else,} \end{cases} \tag{19}$$

such that $d^* \in \{1, ..., k-1, \text{post}\}$, where 'post' corresponds to recommending postponing maintenance. For the 2S method with point distributions d^* is given by the RUL estimate.

Figure 9 shows the preferred maintenance days for three PTO algorithms: 2S with LSTM point estimates (9a), 2S with MDN distribution estimates (9b), and E2E with LSTM estimates (9c). Each figure shows the target maintenance days as a function of the RUL. The target maintenance days are shown for all eVTOLs of all six folds. For each RUL between 0 and 20, the average target maintenance day is given, as well as the 25 and 75 percentile values. The dashed line indicates the actual RUL value.

For the End-to-End Maintenance algorithm (Figure 9c), the figure shows how conservative the target maintenance days are. The 75-percentile value is almost never larger than the actual RUL (shown as a dashed line). Furthermore, the effect of the capacity tests can be seen, with large changes around RUL values 5, 10, and 15. Below a RUL of 5 days, the target maintenance day is almost always 1 (recommending immediate replacement of the batteries).

Figures 9a and 9b show the target maintenance days for the 2S methods. It can be seen that the former gives target days closest to the actual RUL, but does quite often overestimate the RUL. On the other hand, the target days of the latter are still close to the actual RUL, but they are almost always smaller than the RUL. When compared to the End-to-End Maintenance method with LSTM, there is a notable difference



(a) 2S RUL point algorithm with LSTM regressor.

(b) 2S RUL distribution algorithm with MDN regressor.

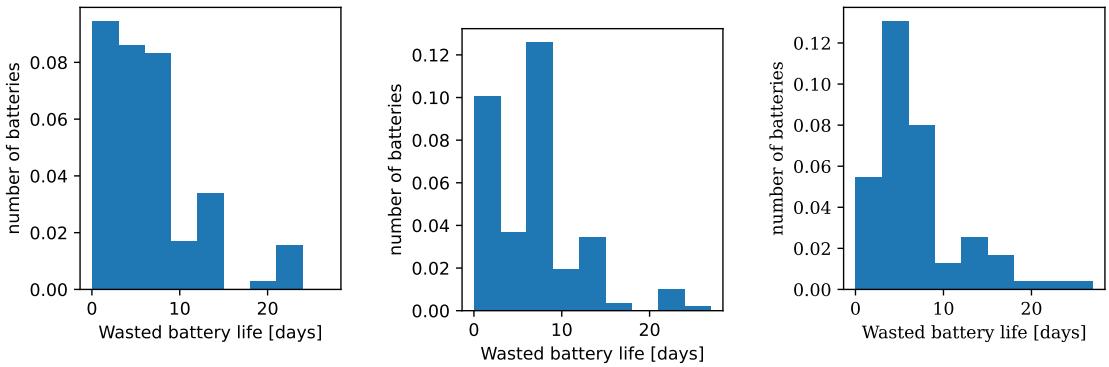
(c) End-to-End Maintenance algorithm with LSTM regressor.

Figure 9: Maintenance target days d^* for different RUL values over all battery types (Table 6). The average target day (triangles), as well as the 25th and 75th percentile (shaded area) are given. The true target day, or the actual RUL, is given by a dashed line.

in the last five days before EOL, as the End-to-End Maintenance algorithm almost always recommends immediate maintenance, whereas the two stage algorithm proposes to wait another day.

Overall, we observe that the target days of the 2S-P algorithm are very aggressive, risking a large number of failures. We also observe that when the actual RUL is low (smaller than 5), the target days recommended by the E2E-M algorithm are more conservative than for the other algorithms. This reduces the time the batteries are used, but also reduces the risk of a battery failure.

These target days are translated into the utilization of the batteries, which is shown in Figure 10. The figure shows a histogram of the RUL of the batteries at the moment they were maintained (and restored to as-good-as-new condition). It can be seen that the RUL of the batteries is larger for the End-to-End Maintenance method, which are sometimes replaced 10 days in advance. For the two stage method with RUL point estimates, there is a large number of batteries with either 1 or 0 days of RUL left.



(a) 2S-P algorithm with LSTM regressor.

(b) 2S-D algorithm with MDN regressor.

(c) E2E-M algorithm with LSTM regressor.

Figure 10: Histogram of the RUL (in days) of the maintained batteries at the moment of their maintenance.

5.2. Results for the End-to-End Maintenance (E2E-M) planning algorithm

For the remainder of the section, we shall discuss the results of our variant of the End-to-End Maintenance algorithm.

Table 6 gives the performance of the LSTM for each fold. It gives the average Root Mean Square Error (RMSE) of the maintenance costs as well as the regret for single eVTOL maintenance planning. The results show that the typical regret is around 40%.

Table 6: Performance of the End-to-End Maintenance algorithm. For each fold, the average RMSE of the cost estimates and the decision regret (Equation 12) are given.

Fold#	RMSE [-]	$\mathcal{L}_{\text{regret}}$ [%]
1	9.82	31.6
2	12.90	29.1
3	6.21	48.6
4	10.61	42.9
5	5.92	49.2
6	4.31	38.2
ALL	8.29	39.9

5.2.1. Analysis of the E2E-M performance

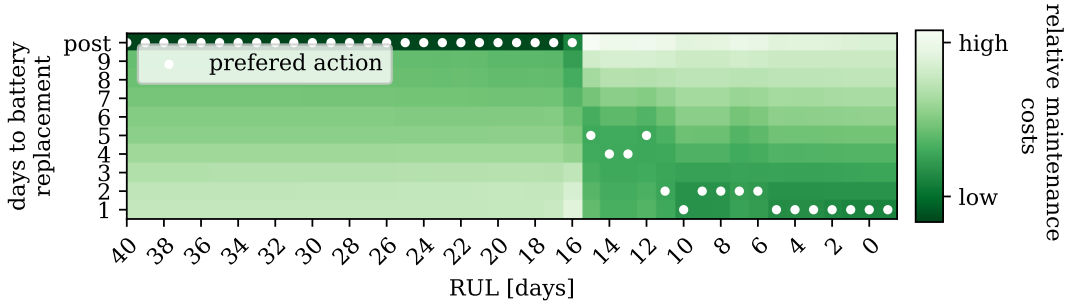
As shown in Section 5.1, integrating the decision regret into the training of the regressor provides an increase in performance when compared to the two-stage algorithms, even though the regressor is not trained on the ILP with multiple assets (as discussed in Section 3.3). This can be explained by the fact that, even while training on a single asset, enough of the structure of the problem is conserved.

First, training on a single asset does still ensure that the regressor explicitly learns the costs of under- and overestimating the RUL. This is not affected by the omission of the interaction between assets during training. The regressor also learns the true costs of maintenance, instead of using an estimated RUL as a proxy-indicator (as in the two-stage algorithms). Second, the inclusion of the hangar capacity H and constraints (5c) during maintenance planning, which was omitted during training, does not affect the priorities with which assets are scheduled to be maintained. These priorities are still based on the risk of an asset reaching its end-of-life before a given date. As such, the E2E-M algorithm maintains the advantages of the end-to-end predict-and-optimize framework.

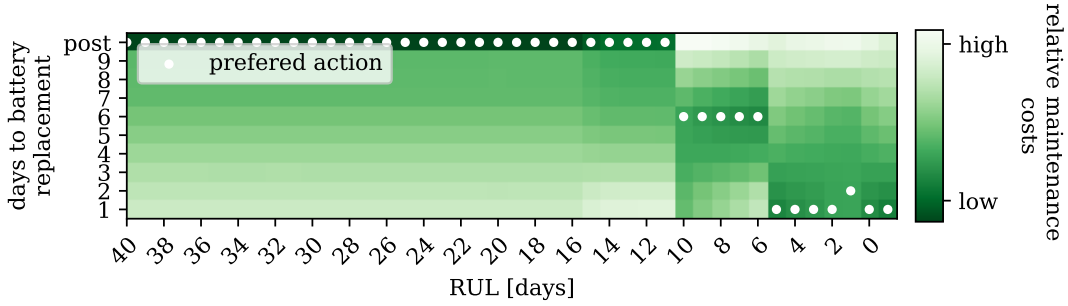
5.2.2. E2E-M predict-and-optimize maintenance cost estimates

Using the trained LSTM networks, the maintenance costs of each eVTOL for each RUL have been estimated. These estimates are discussed in this section.

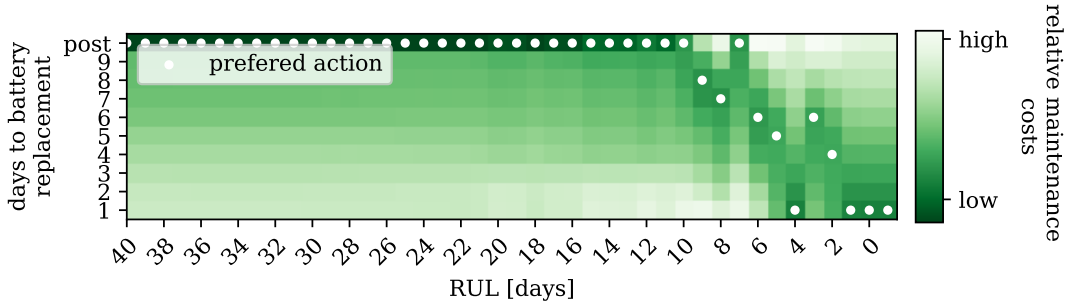
Figure 11 shows three examples of the estimated maintenance costs: VAH01 in fold 1 (11a), VAH10 for fold 3 (11b), and VAH24 for fold 5 (11c). The figure shows a heatmap of the maintenance costs $\hat{\mathbf{c}}_{ad}$ and $\hat{\mathbf{c}}_a^{\text{postpone}}$ for each value of the actual RUL (horizontal, from 40 to 0) and maintenance options (vertical): repairing in $d = 1$ up to $d = 9$ days, and postponing maintenance. The costs are normalized for each RUL, with dark colours corresponding to low costs, and light colours to high costs.



(a) VAH01, fold 1



(b) VAH10, fold 3



(c) VAH24, fold 5

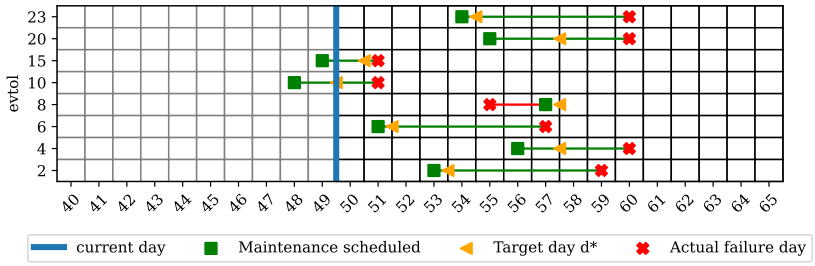
Figure 11: Heatmap of the maintenance cost estimates obtained by the End-to-End Maintenance algorithm (E2E-M) with the LSTM network (Section 4.2.4) for three batteries. The estimates are shown from a RUL of 40 days (400 cycles) to the EOL. The vertical axis gives the maintenance options: repairing in $d = 1$ up to $d = 9$ days, and postponing maintenance. The estimates shown are normalized for each RUL value.

In Figure 11, the target day is shown as a white dot. This target maintenance day is defined, as in Figure 9, as the day with the lowest estimated costs given the current RUL.

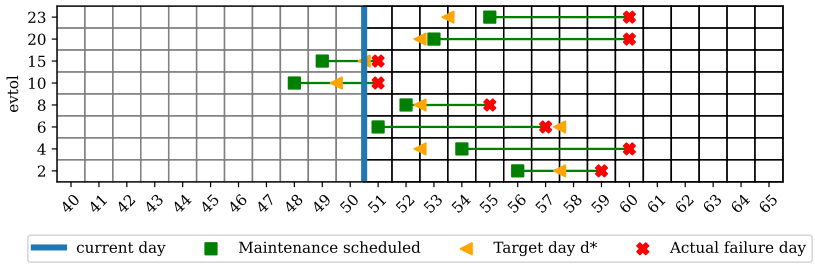
The cost estimates for VAH01 are more conservative than those for VAH10 and VAH24: the target maintenance day starts converging to 1 around a RUL of 16 days (160 cycles). Furthermore, for VAH01, the target maintenance day is already imminent, 10 days before the end-of-life. Last, one can notice the effects of the capacity tests every 5 days on the estimated costs for VAH01 and VAH10, where distinct differences are visible around 15, 10, and 5 days of RUL. This can be explained by the fact that at every capacity test, a new capacity measurement (from which the RUL is derived) is performed, which the LSTM takes into account when estimating the maintenance costs.

5.2.3. End-to-End predict-and-optimize maintenance - example maintenance schedule

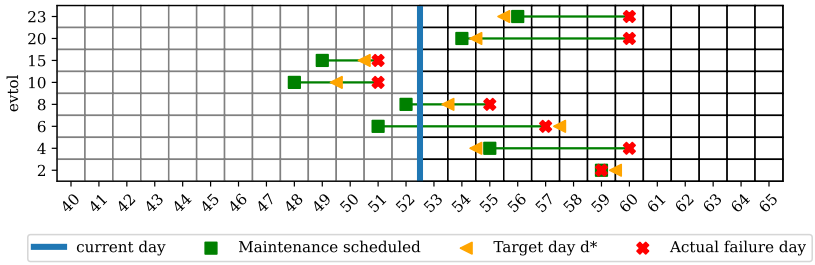
Last, we show an illustrative example where the estimated maintenance costs are integrated into maintenance planning. Figure 12 shows how a maintenance schedule evolves over six days. Figures 12a, 12b, and 12c show the schedules at days 50, 51, and 53, respectively. From day 45, battery maintenance is planned for eVTOLs 2, 4, 6, 8, 10, 15, 20 and 23 shown as rows in each figure. For each eVTOL with planned maintenance, we show the EOL (red cross), target maintenance (yellow arrow) and scheduled maintenance (green square) days. In case maintenance is planned before the EOL, a green solid line is shown. In case it is planned after the EOL, a red dashed line is shown.



(a) current day 50



(b) current day 51



(c) current day 53

Figure 12: Illustration of the evolution of the maintenance schedule, generated with the End-to-End Maintenance algorithm (E2E-M), for three days (51, 53, and 53). Days 40 through 65 and eVTOLs which have an EOL within this window are shown. For each eVTOL, the EOL is shown as a (red) cross, together with the current target day as a (yellow) triangle. The current scheduled maintenance day is shown as a (green) square, connected to the EOL day.

Using the rolling horizon approach, the maintenance decisions of the previous days are performed on

the next day. Before the start of day 50, maintenance has already been performed on eVTOLs 10 (on day 48) and 15 (on day 49). At the start of day 50, maintenance is planned for eVTOL 6 on day 51. This is performed on day 51, 6 days before the EOL, while the other maintenance decisions are reevaluated.

At the start of day 51, a conflict arises between eVTOLs 4, 8, and 20. Using the latest available SOH data, the target days for all three eVTOLs have been set to 52. Maintenance for these three eVTOLs is scheduled for the next three opportunities, at days 52, 53 and 54. Because of this, maintenance for eVTOL 23 cannot be planned before its target day, 53, and is scheduled for day 55. This issue is resolved in the next iteration, at the start of day 53, when the target days of all eVTOLs are reevaluated. The target days for eVTOLs 4 and 20 have been updated to day 54. Using this information, maintenance for the two eVTOLs is planned for the next two available slots and is performed on these days.

5.2.4. Computational performance

Table 7 shows the total computational time required to obtain the optimized maintenance schedules for $\mathcal{D} = 10$ years of operations. These results have been obtained with the Gurobi Optimizer 13.0, using an AMD Ryzen 7 4800H. Here, eVTOL fleet sizes varying from 5 to 250 aircraft have been considered, with hangar capacity scaling linearly. Additionally, we have considered a $k = 20$ day horizon. For a 10 day horizion and fleet of 250 aircraft, the maintenance schedule is obtained in 1566.7 seconds, corresponding to 0.429 seconds per day of operations. For a 20 day horizon, the schedule is obtained in 4200 seconds, or 1.15 seconds per day of operations.

Number of eVTOLs	5	10	25	50	75	100	150	200	250
Hangar capacity H	1	1	1	2	3	4	6	8	10
10 day horizon [s]	40.3	76.0	176.0	313.3	453.3	526.7	700.0	1153.4	1566.7
20 day horizon [s]	107.4	188.8	443.0	812.0	1266.4	1524.1	1624.2	3349.2	4200.6

Table 7: Running time of the maintenance schedule optimization in the E2E-M framework, for various fleet sizes. A period of 10 years of operations is considered.

6. Conclusions

This paper proposes an end-to-end, dynamic framework for the maintenance planning problem for a set of assets. Sensor continuously monitor the state of health of the assets. Based on these sensor measurements, maintenance for each asset is planned, with the objective of planning maintenance as close near the end-of-life of the asset as possible. For this, a limited maintenance capacity is available. In case the asset fails unexpectedly, a large penalty is incurred. Using an End-to-End Maintenance framework, we train a machine learning regressor to generate estimates for the costs of maintaining the assets. Compared to existing studies on end-to-end approaches for general predict-then-optimize problems, we leverage the fact that the health measurements of different assets are uncorrelated. By assuming this, we are able to estimate the maintenance costs for each asset separately, allowing for faster and more effective training. Secondly, we incorporate the dynamic aspect of the maintenance planning problem by developing a rolling horizon algorithm. Thirdly, when compared to unsupervised learning algorithms, our method is more suited for applications where limited datasets are available, and scales more effectively to larger instance sizes. Also,

compared with the majority of studies on predictive maintenance, which make use of two-stage approaches, our approach proposed a way of planning maintenance by including feedback from the maintenance planning problem while training the maintenance cost estimation regressor. When compared to MDPs, which can be used in this two-stage framework, our model does not require the specification of the states and transition probability matrix, limiting the scalability of the MDP framework.

The proposed framework is applied in a case study for a fleet of 25 electric vertical take-off and landing (eVTOL) aircraft. These perform round trips to and from a hub airport. Each eVTOL is equipped with a battery which degrades across time. A dataset of sensor measurements related to the temperature, voltage, and the current of the batteries is used to inform the maintenance planning of the batteries and associated costs [36]. The proposed End-to-End Maintenance framework is implemented with a Long-Short Term Memory (LSTM) regressor. When compared with the optimal maintenance decisions obtained by using a RUL-Oracle, 35% extra maintenance costs are incurred, with only 6 battery failures per year (3.6%). This is a significant improvement when compared to a conventional two-stage maintenance planning model. In this case, 49% extra maintenance costs are incurred, with 8 battery failures per year.

Overall, this study lays the groundwork for data-driven, end-to-end, dynamic predictive maintenance for a set of assets. As future work, we plan to extend our proposed end-to-end framework for the case of more complex maintenance planning cases, with increasing more dependencies between the maintenance needs of the assets and the operational practices specified for these assets. Additionally, future research could be dedicated to extending the maintenance cost regressor to be trained on multiple assets simultaneously, and comparing the results to using a single-asset cost regressor. Last, the E2E-M algorithm can be further validated by applying it to a variety of state-of-health datasets, in combination with using different network types.

References

[1] V. Badea, Z. Alin, R. Boncea, Big Data in the Aerospace Industry, Informatica Economica 22 (2018) 17–24. doi:10.12948/issn14531305/22.1.2018.02.

[2] J. Tautz-Weinert, S. J. Watson, Using SCADA data for wind turbine condition monitoring – a review, IET Renewable Power Generation 11 (4) (2017) 382–394, _eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-rpg.2016.0248. doi:10.1049/iet-rpg.2016.0248.
URL http://onlinelibrary.wiley.com/doi/abs/10.1049/iet-rpg.2016.0248

[3] L. Ren, L. Zhao, S. Hong, S. Zhao, H. Wang, L. Zhang, Remaining Useful Life Prediction for Lithium-Ion Battery: A Deep Learning Approach, IEEE Access 6 (2018) 50587–50598, conference Name: IEEE Access. doi:10.1109/ACCESS.2018.2858856.
URL https://ieeexplore.ieee.org/document/8418374

[4] L. Biggio, A. Wieland, M. A. Chao, I. Kastanis, O. Fink, Uncertainty-Aware Prognosis via Deep Gaussian Process, IEEE Access 9 (2021) 123517–123527, conference Name: IEEE Access. doi:10.1109/ACCESS.2021.3110049.
URL https://ieeexplore.ieee.org/document/9530487

[5] I. de Pater, A. Reijns, M. Mitici, Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics, Reliability Engineering & System Safety 221 (2022) 108341. doi:10.1016/j.ress.2022.108341.
URL https://www.sciencedirect.com/science/article/pii/S0951832022000175

[6] A. Jamshidi, S. Hajizadeh, Z. Su, M. Naeimi, A. Núñez, R. Dollevoet, B. De Schutter, Z. Li, A decision support approach for condition-based maintenance of rails based on big data analysis, Transportation Research Part C: Emerging Technologies 95 (2018) 185–206. doi:10.1016/j.trc.2018.07.007.
URL https://www.sciencedirect.com/science/article/pii/S0968090X18309859

[7] X. Liu, Q. Sun, Z.-S. Ye, M. Yildirim, Optimal multi-type inspection policy for systems with imperfect online monitoring, Reliability Engineering & System Safety 207 (2021) 107335. doi:10.1016/j.ress.2020.107335.
URL https://www.sciencedirect.com/science/article/pii/S0951832020308279

[8] J. Lee, M. Mitici, Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics, Reliability Engineering & System Safety 230 (2023) 108908. doi:10.1016/j.ress.2022.108908.
URL https://www.sciencedirect.com/science/article/pii/S0951832022005233

[9] C. Chen, G. Zhu, J. Shi, N. Lu, B. Jiang, Dynamic Predictive Maintenance Scheduling Using Deep Learning Ensemble for System Health Prognostics, IEEE Sensors Journal PP (2021) 1–1. doi:10.1109/JSEN.2021.3119553.

[10] K. T. P. Nguyen, K. Medjaher, A new dynamic predictive maintenance framework using deep learning for failure prognostics, Reliability Engineering & System Safety 188 (2019) 251–262. doi:10.1016/j.ress.2019.03.018.
URL https://www.sciencedirect.com/science/article/pii/S0951832018311050

[11] A. Consilvio, A. D. Febbraro, N. Sacco, A Rolling-Horizon Approach for Predictive Maintenance Planning to Reduce the Risk of Rail Service Disruptions, IEEE Transactions on Reliability 70 (3) (2021) 875–886, conference Name: IEEE Transactions on Reliability. doi:10.1109/TR.2020.3007504.
URL https://ieeexplore.ieee.org/abstract/document/9144460

[12] Z. Allah Bukhsh, A. Saeed, I. Stipanovic, A. G. Doree, Predictive maintenance using tree-based classification techniques: A case of railway switches, Transportation Research Part C: Emerging Technologies 101 (2019) 35–54. doi:10.1016/j.trc.2019.02.001.
URL https://www.sciencedirect.com/science/article/pii/S0968090X18309057

[13] I. d. Pater, M. Mitici, Novel Metrics to Evaluate Probabilistic Remaining Useful Life Prognostics with Applications to Turbofan Engines, PHM Society European Conference 7 (1) (2022) 96–109, number: 1. doi:10.36001/phme.2022.v7i1.3320.
URL http://www.papers.phmsociety.org/index.php/phme/article/view/3320

[14] P. C. Lopes Gerum, A. Altay, M. Baykal-Gürsoy, Data-driven predictive maintenance scheduling policies for railways, Transportation Research Part C: Emerging Technologies 107 (2019) 137–154. doi:10.1016/j.trc.2019.07.020.
URL https://www.sciencedirect.com/science/article/pii/S0968090X18314918

[15] S. Sharma, Y. Cui, Q. He, R. Mohammadi, Z. Li, Data-driven optimization of railway maintenance for track geometry, Transportation Research Part C: Emerging Technologies 90 (2018) 34–58. doi:https://doi.org/10.1016/j.trc.2018.02.019.
URL https://www.sciencedirect.com/science/article/pii/S0968090X18302444

[16] Z. Su, A. Jamshidi, A. Núñez, S. Baldi, B. De Schutter, Integrated condition-based track maintenance planning and crew scheduling of railway networks, Transportation Research Part C: Emerging Technologies 105 (2019) 359–384. doi:10.1016/j.trc.2019.05.045.
URL https://www.sciencedirect.com/science/article/pii/S0968090X18304339

[17] M. Mitici, I. de Pater, A. Barros, Z. Zeng, Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines, Reliability Engineering & System Safety 234 (2023) 109199. doi:10.1016/j.ress.2023.109199.
URL https://www.sciencedirect.com/science/article/pii/S095183202300114X

[18] T. Vanderschueren, T. Verdonck, B. Baesens, W. Verbeke, Predict-then-optimize or predict-and-optimize? An empirical evaluation of cost-sensitive learning strategies, Information Sciences 594 (2022) 400–415. doi:10.1016/j.ins.2022.02.021.
URL https://www.sciencedirect.com/science/article/pii/S0020025522001542

[19] J. Kotary, V. D. Vito, J. Christopher, P. V. Hentenryck, F. Fioretto, Predict-Then-Optimize by Proxy: Learning Joint Models of Prediction and Optimization, arXiv:2311.13087 [cs] (Nov. 2023). doi:10.48550/arXiv.2311.13087.
URL http://arxiv.org/abs/2311.13087

[20] J. Zhang, E. Shan, L. Wu, J. Yin, L. Yang, Z. Gao, An End-to-End Predict-Then-Optimize Clustering Method for Stochastic Assignment Problems, IEEE Transactions on Intelligent Transportation Systems 25 (9) (2024) 12605–12620, conference Name: IEEE Transactions on Intelligent Transportation Systems. doi:10.1109/TITS.2024.3385029.
URL https://ieeexplore.ieee.org/abstract/document/10504737

[21] A. N. Elmachtoub, P. Grigas, Smart "Predict, then Optimize", arXiv:1710.08005 [cs, math, stat] (Nov. 2020). doi:10.48550/arXiv.1710.08005.
URL http://arxiv.org/abs/1710.08005

[22] J. Mandi, E. Demirovi, P. Stuckey, T. Guns, Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems,

Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 1603–1610. doi:10.1609/aaai.v34i02.5521.

[23] X. Chang, J. Wu, H. Sun, X. Yan, A Smart Predict-then-Optimize method for dynamic green bike relocation in the free-floating system, Transportation Research Part C: Emerging Technologies 153 (2023) 104220. doi:10.1016/j.trc.2023.104220.
URL https://www.sciencedirect.com/science/article/pii/S0968090X23002097

[24] M. Vlastelica, A. Paulus, V. Musil, G. Martius, M. Rolínek, Differentiation of Blackbox Combinatorial Solvers, arXiv:1912.02175 [cs] (Feb. 2020). doi:10.48550/arXiv.1912.02175.
URL http://arxiv.org/abs/1912.02175

[25] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, F. Bach, Learning with Differentiable Pertubed Optimizers, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 9508–9519.
URL https://papers.nips.cc/paper/2020/hash/6bb56208f672af0dd65451f869fedfd9-Abstract.html

[26] B. Tang, E. B. Khalil, PyEPO: a PyTorch-based end-to-end predict-then-optimize library for linear and integer programming, Mathematical Programming Computation (Jul. 2024). doi:10.1007/s12532-024-00255-x.
URL https://doi.org/10.1007/s12532-024-00255-x

[27] C. P. Andriotis, K. G. Papakonstantinou, Managing engineering systems with large state and action spaces through deep reinforcement learning, Reliability Engineering & System Safety 191 (2019) 106483. doi:10.1016/j.ress.2019.04.036.
URL https://www.sciencedirect.com/science/article/pii/S0951832018313309

[28] J. Lee, M. Mitici, An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri nets, Reliability Engineering & System Safety 202 (2020) 107052. doi:10.1016/j.ress.2020.107052.
URL https://www.sciencedirect.com/science/article/pii/S0951832020305536

[29] C. M. Bishop, Mixture density networks, Monograph, Aston University, Birmingham, UK, num Pages: 438543 Place: Birmingham Publisher: Aston University (1994).
URL https://publications.aston.ac.uk/id/eprint/373/

[30] O. Ogunfowora, H. Najjaran, Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization, Journal of Manufacturing Systems 70 (2023) 244–263. doi:https://doi.org/10.1016/j.jmsy.2023.07.014.
URL https://www.sciencedirect.com/science/article/pii/S0278612523001462

[31] L. Granado, M. Ben-Marzouk, E. Solano Saenz, Y. Boukal, S. Jugé, Machine learning predictions of lithium-ion battery state-of-health for eVTOL applications, Journal of Power Sources 548 (2022) 232051. doi:10.1016/j.jpowsour.2022.232051.
URL https://www.sciencedirect.com/science/article/pii/S037877532201028X

[32] T. Hagspihl, R. Kolisch, S. Schiffels, Planning an airport shuttle network with air taxis using choice-based optimization, OR Spectrum (2025) 1–35Company: Springer Distributor: Springer Institution: Springer Label: Springer Publisher: Springer Berlin Heidelberg. doi:10.1007/s00291-024-00801-y.
URL https://link-springer-com.tudelft.idm.oclc.org/article/10.1007/s00291-024-00801-y

[33] L. A. Garrow, B. J. German, C. E. Leonard, Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research, Transportation Research Part C: Emerging Technologies 132 (2021) 103377. doi:10.1016/j.trc.2021.103377.
URL https://www.sciencedirect.com/science/article/pii/S0968090X21003788

[34] M. Waltz, O. Okhrin, M. Schultz, Self-organized free-flight arrival for urban air mobility, Transportation Research Part C: Emerging Technologies 167 (2024) 104806. doi:10.1016/j.trc.2024.104806.
URL https://www.sciencedirect.com/science/article/pii/S0968090X24003279

[35] M. Mitici, B. Hennink, M. Pavel, J. Dong, Prognostics for Lithium-ion batteries for electric Vertical Take-off and Landing aircraft using data-driven machine learning, Energy and AI 12 (2023) 100233. doi:10.1016/j.egyai.2023.100233.
URL https://www.sciencedirect.com/science/article/pii/S2666546823000058

[36] A. Bills, S. Sripad, L. Fredericks, M. Guttenberg, D. Charles, E. Frank, V. Viswanathan, A battery dataset for electric vertical takeoff and landing aircraft, Scientific Data 10 (1) (2023) 344, publisher: Nature Publishing Group. doi:10.1038/s41597-023-02180-5.
URL https://www.nature.com/articles/s41597-023-02180-5

[37] S. Lundberg, S.-I. Lee, A Unified Approach to Interpreting Model Predictions, arXiv:1705.07874 [cs] (Nov. 2017). doi:10.48550/arXiv.1705.07874.

URL http://arxiv.org/abs/1705.07874

[38] J. Urbano, H. Lima, A. Hanjalic, A New Perspective on Score Standardization, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1061–1064. doi:10.1145/3331184.3331315.
URL https://dl.acm.org/doi/10.1145/3331184.3331315

[39] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536, publisher: Nature Publishing Group. doi:10.1038/323533a0.
URL https://www.nature.com/articles/323533a0

[40] Y. Yu, X. Si, C. Hu, J. Zhang, A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures, Neural Computation 31 (7) (2019) 1235–1270. doi:10.1162/neco_a_01199.
URL https://doi.org/10.1162/neco_a_01199

[41] L. Li, Z. Zang, R. Geng, C. Liu, H. Song, Y. Chen, Optimized-architecture long short-term memory and particle swarm optimization integration for lithium-ion batteries remaining useful life prediction, Journal of Energy Storage 152 (2026) 120875. doi:https://doi.org/10.1016/j.est.2026.120875.
URL https://www.sciencedirect.com/science/article/pii/S2352152X26005396

[42] M. Wei, H. Gu, M. Ye, Q. Wang, X. Xu, C. Wu, Remaining useful life prediction of lithium-ion batteries based on Monte Carlo Dropout and gated recurrent unit, Energy Reports 7 (2021) 2862–2871. doi:10.1016/j.egyr.2021.05.019.
URL https://www.sciencedirect.com/science/article/pii/S2352484721002973

[43] S. Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747 [cs] (Jun. 2017). doi:10.48550/arXiv.1609.04747.
URL http://arxiv.org/abs/1609.04747