Machine Learning-based Update-time Prediction for Battery-friendly Passenger Information Displays

Peter Herrmann, Ergys Puka
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
{herrmann,puka}@ntnu.no

Tor Rune Skoglund
FourC AS
Trondheim, Norway
trs@fourc.eu

***Abstract*—Personal Information Displays (PID) at bus stops help making the usage of public transport more attractive. If no electric grid is nearby, however, the installation of PIDs is very expensive due to the high wiring costs. To resolve this issue, the partners of the R&D project IoT-STOP develop a novel PID system that will be independent from the access to power lines. The system uses e-papers as displays that can be accessed using a cellular network. To prevent long, energy-intensive idle listening, the network receiver operates only when the passenger information, in particular, the Expected Times of Arrival (ETA) of the buses, is updated. Between two updates, the receiver is switched off such that adjustments after sudden events are not possible. Therefore, the update periods have to be carefully selected. In this paper, we introduce a predictor that estimates time intervals between updates. Our method is based on linear regression using samples of previous bus rides to forecast arrival times. Its predictions are applied by an algorithm to detect areas during the journey of a bus at which its ETA at a later stop changes with a certain probability. The forecasted times for passing such areas are then selected to update the PID at this stop. In addition, we present a number of tests of the predictor carried out at some bus stops in Bergen, Norway. The results show that the proposed method indeed predicts sensible update times of the PID systems.**

***Index Terms*—battery-driven passenger information display; e-paper; update prediction algorithm; linear regression;**

I. Introduction

Providing public transportation users with sufficient real-time information is helpful to make this mode of transport more attractive [1]. That holds particularly for major disruptions like heavy delays of a bus[1] [2]. Nowadays, many Public Transport Authorities (PTA) offer mobile phone applications and dedicated web-pages that, amongst others, inform a user about the Expected Time of Arrival (ETA) of a bus at a certain stop [3]. Many users, however, prefer fixed real-time Passenger Information Displays (PID) at the stops to both, mobile solutions and classical paper-based timetables [4]. The likely reason is that it is easy and quick to figure the next arriving buses and their ETAs out on the display. In contrast, starting an application takes time and may cause some trouble in harsh weather conditions resp. when the user has no hands free or wears gloves.

On the other side, passengers are not ready for a significant increase of the ticket fare to finance the use of PIDs and other means of information. A study presented in [5] reveals that only 30% of the interviewed passengers are willing to pay 0.60€ more for a ticket in order to get better traffic information. This is contrary to the fact that the installation of PIDs is often quite expensive. We interviewed several PTAs in Norway and found out that providing a bus stop with a traditional monitor-based PID affords an investment of around 2,500€. In addition, the connection for the power supply has to be arranged which can be as much as 100,000€ when there are no electric power lines nearby. Further, the maintenance cost of a system has to be considered as well. Therefore, it is no surprise that only about 2% of the approximately 115,000 bus stops in Norway are equipped with fixed PIDs.

The consortium of the R&D project IoT-STOP[2] intends to alleviate the connection costs by developing a PID system that is independent from the electrical grid. It is based on batteries and uses e-papers [7] as displays. This novel screen technology has the great advantage that power is only used when the display is changed but not in between.

Another major consumer of power is the idle listening for incoming network messages [8]. By switching the network receiver off between two transmissions, however, we can disburden this power usage significantly. Yet, the temporary shutdowns of the receiver pose the risk that, in the meantime, sudden issues like heavy delays might occur about which the passengers cannot be promptly informed. Thus, the selection of useful update times is crucial. To tackle this problem, we developed a machine learning-based predictor of update times for PIDs, that is the contribution of this paper.

The article is structured as follows: We sketch the developed PID system in Sect. II. Thereafter, we introduce the basic structure of the predictor for update intervals in Sect. III. This is followed by a discussion of linear regression, the selected machine learning technology, and its use to forecast ETAs in Sect. IV. Next, we present an algorithm applied to determine useful update times of the PIDs in Sect. V. Using

This work is supported by Innovation Norway within the R&D project IoT-STOP.

[1]Throughout this paper, we refer to buses but, of course, our method works also for other modes of transport like trains and trams.

[2]The IoT-STOP consortium consists of the software development company FourC [6] (project leader), the university partner NTNU, the major communication provider Telenor, as well as the Norwegian PTAs Skyss and Kringom. The project is supported by Innovation Norway within the Environment Technology (Miljøteknologiordningen) program.

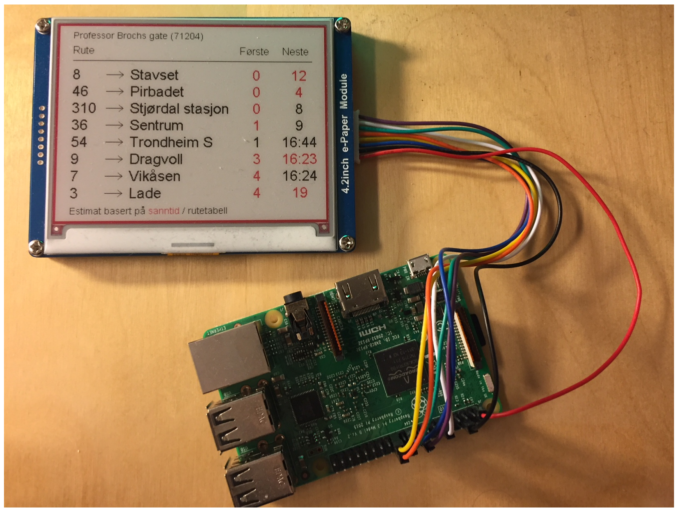


Fig. 1. Demonstrator of an e-paper-based PID.

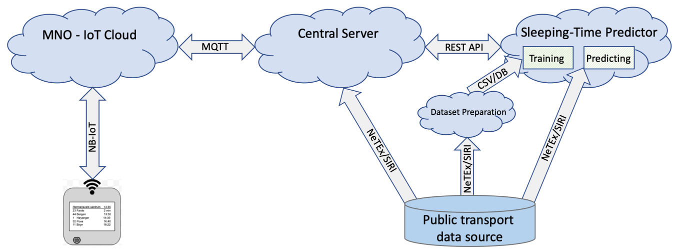


Fig. 2. Architecture of the planned PID-system.

a simulator for the PID system, we tested different variants of the update time prediction functions. The results of these tests are introduced and discussed in Sect. VI. The article is completed by a look at related work in Sect. VII and a conclusion with an outlook about future activities in Sect. VIII.

II. A Sketch of the PID System

Modern e-papers [7] are realized using electrophoretical particles that are dispersed in a highly viscous liquid. For each pixel, a cell containing this alloy is placed behind a transparent panel. By imposing an electrical field on the alloy, the particles are either sponged towards the panel or away from it. In consequence, the optical refraction is changed and the pixel appears black or white. One can also use two different kinds of electrophoretical particles that react with a different intensity on the electrical field. That makes it possible to display a third color, mostly red. Thanks to the high viscosity of the liquid, the particles remain stationary if no electrical field is imposed. Thus, an e-paper only needs power when changes are made. That makes it suited to the purposes of the R&D project IoT-STOP. Figure 1 depicts a small e-paper used as a demonstrator of a PID system in a pre-project of IoT-STOP. It shows the arrival times of various buses at a bus stop in Trondheim, Norway. ETAs in red indicate real-time-based arrival times while those based on the timetable are outputted in black.

TABLE I
Power consumption experiments

Component	Activity	Consumption
Controller (5.0 V)	normal	46 mA
Controller (3.3 V)	normal	25 mA
E-paper	sleeping	0 mA
	update (mostly)	2 mA
	update (peek)	3 mA
Receiver (5.0 V)	initialization	120 mA
	normal	85 mA
	power-down	88 mA
Receiver (4.3 V)	initialization	140 mA
	normal	100 mA
	power-down	100 mA

The architecture of the overall system is depicted in Fig. 2. The local PID-system at a bus stop is linked with an IoT Cloud service offered by Telenor using Narrowband-Internet-of-Things (NB-IoT) [9], a variant of the cellular network standard LTE. Due to the good cellular network coverage of most public roads in Norway (see [10], [11]), NB-IoT allows us to place PID systems nearly everywhere. A central server prepares the data to be displayed as well as the planned time for the next update and sends them to the PID controller via the IoT cloud service. The link to this service is performed using a standard IoT protocol like MQTT [12].

The component introduced in this paper is the predictor that calculates the update intervals. It will be connected with the central server using a REpresentational State Transfer (REST) API. Both, the server and the predictor fetch the necessary ETA data of relevant buses from the open data source provided by Entur [13]. This service offers real-time traffic data from most public transport providers in Norway.

The PID system residing locally at a bus stop, consists of at least one e-paper, a chipset controlling the system, and an NB-IoT receiver. For the power supply, high capacity batteries are used but we also consider power generation technologies like solar panels or wind-based generators (see [14]).

As already mentioned, the idle listening of network receivers, which is necessary to fetch the input from remote peers, is an important consumer of the scarce energy supplies of small devices [8], [15]. To get evidence, we conducted tests for a version of the demonstrator using the type of e-paper shown in Fig. 1, the well-known Arduino board, and an NB-IoT device that is based on a SIMCOM SIM7000E chipset. Both boards were tested with different voltages. The results are listed in Tab. I. One can easily see that the energy consumption of the e-paper is negligible. More relevant is the energy consumption of the board. Instead of using an Arduino or the Raspberry Pi, we applied for our functional tests in the pre-project, the final system will therefore be based on a more energy-efficient board from Atmel.

By far the most dominating power user, however, is the SIMCOM7000E chipset that, depending on the used voltage, requires between 85 and 100 mA in normal operation. Powering this chipset down needs nearly no additional power while the (re-)initialization, that lasts for approximately 12 seconds,

TABLE II
UNNECESSARY WAKE UP PERCENTAGES USING A SIMPLE ALGORITHM

Bus stop name	Updates per day	Unnecessary updates
Hesthagen	1160	1.5%
Flakkråa	666	37.2%
Solbakken bru	242	54.2%

demands around 40% more. That makes it worthwhile to power the device down if the waiting time between two updates is at least 40 seconds. Technically, this is done by the controller which starts the shutdown directly after completing a transfer. Using the update interval received with the previous update data, the controller then re-initializes the receiver around 15 seconds before the next update is planned, such that it will be ready in due time.

III. PREDICTING SLEEPING TIMES

Determining suitable update times of a Passenger Information Display (PID) system needs to address two main concerns:

- Passengers want to be promptly informed about changes of the Expected Times of Arrival (ETA) of their buses, in particular, if the adjustment is major. We consider an update as *major* if the ETA changes by at least two minutes. Most critical, however, is to prevent that overly large delays are displayed until shortly before the bus arrives. This is necessary to avoid that passengers leave the bus stop temporarily, since they expect a major delay, and thereby miss the bus.
- Public Transport Authorities (PTA) like to minimize the power consumption in order to avoid replacing the batteries too often. To achieve that, it is crucial to minimize the number of unnecessary wake ups, i.e., restarting the receiver for a new message that does not contain any ETA adjustments at all.

The two aspects are conflicting since by using shorter update intervals, the promptness of the display is increased but usually at the cost of more unnecessary wake ups.

During the pre-project, we tested three bus stops in Trondheim, Norway, for the percentage of unnecessary wake ups. For that, we used a simple algorithm which determined the update time intervals t_u depending on the ETA t_{fb} of the first bus, as follows (in minutes):

$$t_u = \begin{cases} 1, & \text{if } t_{fb} \leq 25 \\ 5, & \text{if } 25 < t_{fb} \leq 65 \\ 15, & \text{if } 65 < t_{fb} \leq 135 \\ 60, & \text{otherwise.} \end{cases}$$

The results are depicted in Tab. II. Hesthagen is an extremely busy station close to the city center that, besides of many residential areas and businesses, serves the main campus of NTNU. In average, we measured 1160 wake ups per day which shows that buses arrive nearly continuously. Having to deal with so many buses leads to constant adjustments of the ETAs which explains the tiny fraction of unnecessary wake ups. In contrast, Flakkråa is a station outside the city with moderate traffic of one or two buses an hour. In consequence, the number of updates is significantly lower but more than a third of them were unnecessary since they did not show any ETA changes. Finally Solbakken bru is a remote bus stop in a rural area from which only few buses leave every day. Our algorithm led to far too many wake ups since more than every second one was unnecessary.

The test results show that the algorithm is adequate for busy bus stops but leads to an excessive number of unnecessary wake ups at stops with moderate or low traffic. However, since such stations are often in rural areas without easy access to electric grids and therefore predestined for our PID system, a better method to determine sleeping time intervals has to be found.

We use a method based on machine learning to detect when major changes of the displayed ETAs are to be expected. Unfortunately, we could not resort to the delay predictor used in the real-time service of Entur since it is a public nationwide service with limited options to reveal specifics of their internals to third parties. Therefore, we developed an own solution to forecast ETAs. For that, we use the planned and effective arriving and leaving times of a selected line operated by our project partner Skyss in Bergen, Norway.

We utilize around 70% of the collected data samples for a route towards a bus stop s_{PID}, at which the PID system is installed, to train models using the relatively simple machine learning technique linear regression. The created models correspond to functions f_i that describe the forecasted ETA at stop s_{PID} when the bus passed on its way towards s_{PID} all stops from its starting point s_1 to s_i. For that, f_i takes the effective leaving times from these bus stops as arguments. Assuming that stop s_{PID} is the $n+1$-st stop, i.e., $s_{PID} = s_{n+1}$, we produce the functions $f_1, \ldots, f_n$ by linear regression.

At next, we use the remaining 30% of the collected samples for analysis. For each sample, we carry out $n-1$ different analysis runs each reflecting the case that the sampled bus passed stop s_u but not yet s_{u+1} when the PID was last updated ($1 \leq u < n$). Thus, at this time, the result of function f_u corresponds with the ETA to be displayed on the PID for our bus. Now, we check for the particular sample also the results of all functions f_i with $u < i \leq n$. The values of f_i are then compared with f_u. The difference $f_i - f_u$ corresponds to the change of the displayed ETA if the next update is carried out after the bus has left stop s_i. If this difference is significant, i.e., by two minutes or more, a counter c_i^u is incremented.

This procedure is repeated for all ts-many testing samples such that c_i^u/ts corresponds to the share of all samples that would have been significantly changed, had the first update after bus stop s_u been at s_i. Thereafter, we search for the earliest stop $s_{i_{first}}$ for which this share exceeds a certain threshold. By doing that for all stops s_u, we create a mapping m with $m[u] = i_{first}$. This mapping reflects that the likelihood of significant ETA changes at all stations between s_u and $s_{m[u]}$ is sufficiently low such that we can abstain from carrying out updates when passing them. That predominantly happens if the route in between allows for a relatively uniform traffic such that gained or lost delays can be easily predicted. At stop $s_{m[u]}$, the likelihood of significant updates, however, is large enough such that a new update is recommended. This is usually the case in areas when obstacles like traffic jams sometimes slow a bus down but at other times not.

The mapping m can now be applied to predict the time for future updates. Whenever the central server plans to make a new update, for each bus b to be depicted on the PID, the predictor picks the stop s_{u_b} that b passed at last. Then the ETA of b should be updated when it passes bus stop $s_{m[u_b]}$. Using an ETA forecaster, the predictor determines the time $t_{m[u_b]}$, at which b likely reaches this stop. This procedure is repeated for all buses displayed on the PID, and the earliest of the calculated time points $t_{m[u_b]}$ is used as the time to schedule the next update. By that, we guarantee that this update takes place not after any of the buses b passed stop $s_{m[u_b]}$.

IV. Delay Prediction with Linear Regression

Using machine learning methods to create a numeric function $f : \mathbb{R}^n \to \mathbb{R}$, that forecasts the arrival times of buses at bus stops, is called *regression* [16]. The task is also *linear* since a delay of a bus b at the n-th bus stop s_n can be calculated as the sum of n different inputs:

- The initial delay with respect to the timetable at which the bus leaves the first bus stop s_1,
- the $n-1$ delay differences, i.e., the time interval by which the delay increases or decrases between the stops s_{i-1} and s_i with $1 < i \leq n$.

For instance, if a bus leaves s_1 with a delay of 10 minutes but makes up for a minute between two adjacent stops, it will reach the fourth stop s_4 with a delay of seven minutes.

The linear nature of our problem makes it possible to use the relatively simple and well-performing machine learning technique *linear regression*, see, e.g., [17]. It allows us to predict functions of the following form:

$$f(x_1, \ldots, x_n) = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

Here, the values $x_i \in \mathbb{R}$ refer to the various arguments of the function that are called *features*. By $w_i \in \mathbb{R}$, we describe parameters that weigh the features, while $b \in \mathbb{R}$ is a fixed constant called *bias*.

Like other machine learning techniques, linear regression uses samples of previously collected real-world data from which the parameters w_i and b can be identified by training. Usually, 70 to 80% of the samples are used to train the mechanism while the others are checked against the produced function to determine its quality. If we utilize ts samples for training and each sample contains n different features, we form a matrix $\mathbf{X}$ with ts rows and n columns such that $(\mathbf{X})_{k,i}$ contains the value of the i-th feature of the k-th sample. Further, for each sample, the result y to be reproduced by function f is used. From the results of the training samples, a vector $\mathbf{y}$ of the size ts is formed. Here, the value $(\mathbf{y})_k$ contains the result value of the k-th sample. As described, e.g., in [16], the vector $\mathbf{w}$ of the weights w_i can be computed as follows:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Here, $\mathbf{X}^\top$ describes the transposed matrix of $\mathbf{X}$ while $\mathbf{X}^{-1}$ refers to its inverse. The bias b can be computed by adding a new feature x_{n+1} that for each sample is set to the value 1. Then, b is equal to the computed weight w_{n+1}. Altogether, the linear regression algorithm corresponds to relatively simple matrix operations such that, in contrast to more complex machine learning algorithms, it performs well on computers.

Next, we introduce how linear regression is used to create a forecast function for ETAs. If te_i is the effective leaving time of a bus from stop s_i and tt_i the departure time at this stop according to the timetable, we define the features $x_1, \ldots, x_{n+1}$ for data based on n successive bus stops in the following way:

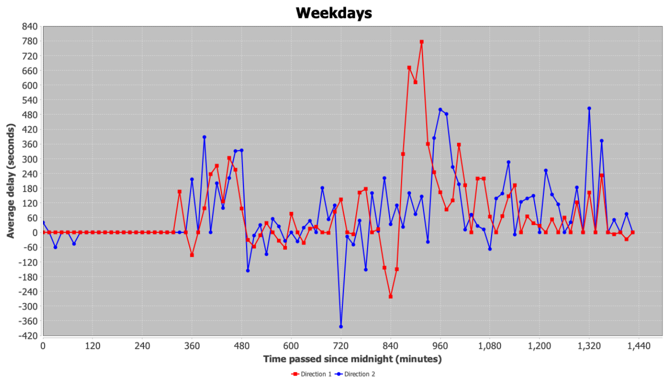
- The value of $x_1 \leftarrow te_1 - tt_1$ is set to the difference between the effective leaving time and the departure time according to the timetable at the initial stop s_1.
- For all bus stops s_i with $1 < i \leq n$, the feature x_i refers to the delay change between passing s_{i-1} and s_i, i.e., $x_i \leftarrow (te_i - tt_i) - (te_{i-1} - tt_{i-1})$.
- The feature $x_{n+1} \leftarrow 1$ is set to the value 1 in order to calculate the bias b.

Further, we take the different traffic situations under account. Of course, the likelihood of delays, e.g., during the morning rush hour tends to be different from that in the late evening. To consider that, we looked at patterns of the average delays collected for various bus stops in Bergen and could identify eight separate phases that depend on the time of the day. Since the behavior on all five weekdays is similar, we just differentiate between weekdays in general, Saturdays, and Sundays. The average daily delay for the different delays is depicted in Fig. 3. The graphs show that on the weekdays, we can, indeed, distinguish several separate phases, while the behavior on the weekend is more uniform. Thus, we define the following eight phases, the first six of which apply to weekdays only:

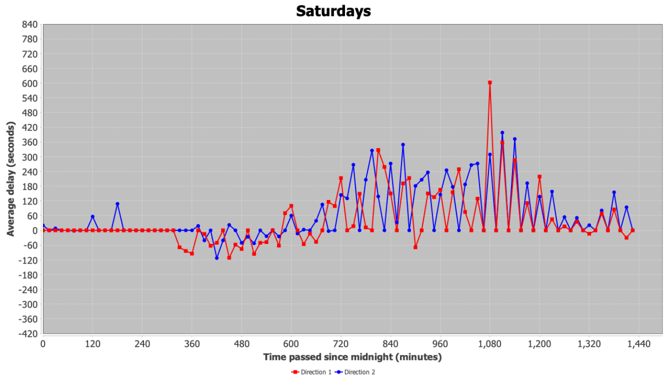
1) Early morning before 06:30,
2) morning rush hour between 06:30 and 09:00,
3) forenoon between 09:00 and 13:00,
4) light rush hour in the early afternoon between 13:00 and 14:30,
5) afternoon rush hour between 14:30 and 17:00,
6) evening after 17:00,
7) Saturdays,
8) Sundays.

To integrate also these phases to the linear regression, we extended the bias by adding eight features $x_{n+2}, \ldots, x_{n+9}$ each representing one phase. For the sample of a bus trip taking place in phase p, we assign the value 1 to feature x_{n+1+p} and 10^{-11} to the other seven features[3]. The vector $\mathbf{y}$ is produced by assigning each sample the difference between

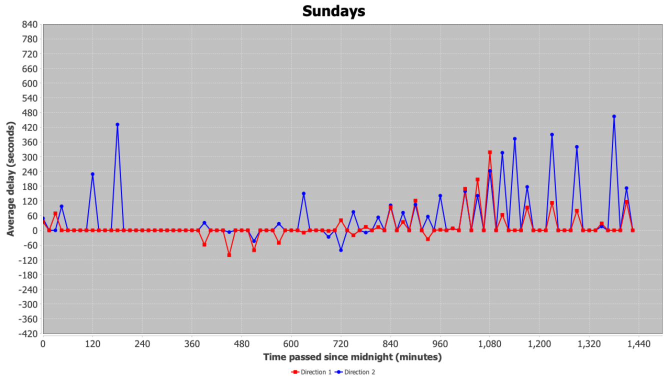
[3]We use the vary small value 10^{-11} instead of 0 to prevent singular matrices $\mathbf{X}$ which cannot be inverted.



(a)



(b)



(c)

Fig. 3. Average daily delay for (a) weekdays, (b) Saturdays, (c) Sundays

the effective arrival time at s_{PID} and the one according to the timetable.

Using the $n+9$ features, we can now produce our forecast function f_n by applying the linear regression method explained above. The function f_n can only predict ETAs if the departure delay data of the first n stops are available. As discussed in Sect. III, we cope with the increasing number of features due to passing further bus stops by creating n different functions $f_1, \ldots, f_n$. Thus, when a bus passes a stop s_i and its departure time t_i from this stop is recorded, we can thenceforth apply function f_i instead of f_{i-1} such that the newly collected leaving time t_i can also be considered.

As a test, we used around 30,000 samples of bus line 5 operating in Bergen. 70% of them were applied to train the linear regression to produce the forecast functions f_1 to f_n and the rest predominantly for the prediction algorithm introduced in Sect. V. But, of course, the testing samples can also be used to analyze the quality of the created functions. Not surprisingly, the forecasts are more precise if the departure data of more intermediate bus stops are available. For instance, we checked the bus stop Sandvikstorget of line 5 towards Loddefjord terminal. It is the 28^{th} stop of the line such that we produced the functions $f_1, \ldots, f_{27}$ by linear regression using 21,012 training samples. We define a forecast to be correct, if the predicted time is within a minute to the effective arrival. The function f_1, that considers only the initial delay towards the timetable at stop s_1, prognosticated 35.9% of the 9,005 testing samples at Sandvikstorget properly. For f_9, this percentage was 52.7% while f_{18} produced 70.0% and f_{27}, that considered all stops ahead of Sandvikstorget, 99.9% correct results. The apparent reason for this improvement is that the potential for disruptions due to, e.g., traffic jams is diminishing as closer the bus comes to the bus stop s_{PID}.

V. Determining Update Time Intervals

As already discussed in Sect III, the predictor functions f_1 to f_n are applied to find out the likelihood that a change after the bus passes a stop s_i leads to a significant change of its ETA at the PID system. We consider two types of ETA changes as significant:

- A *major change* occurs when progressing from function f_{i-1} to f_i leads to a change of the ETA at stop s_{PID} by at least two minutes.
- A *critical change* takes place if by the switch between f_{i-1} and f_i a displayed delay is reduced to a point in time that already elapsed.

We use Algorithm 1 to produce the mapping m that was introduced in Sect. III. The mapping specifies that, after updating the PID when the bus passes stop s_u, the following update shall be made at latest when the bus reaches stop s_i. However, the values of this mapping heavily depend on the current traffic situation. Therefore, we create an own mapping m_p for each of the eight phases p introduced in Sect. IV.

Algorithm 1 uses two parameters c_m and c_c that allow us to determine the thresholds used to detect bus stops at which significant changes happen:

- c_m determines the threshold for the major changes. It depends on the percentage cs_u^i/ts of changes of two minutes or more between s_u and s_i and the duration d_i from s_i to s_{PID}. A higher value leads to a stricter updating policy.
- c_c is used to define the threshold for critical changes. It depends on the share cs_u^i/ts of such changes. Higher values model a less restrictive updating policy.

We discuss typical settings for the two parameters in Sect. VI.

In the first part of Algorithm 1, we specify the creation of the statistics variables cs_u^i and cc_u^i that are initialized in the first two lines. With cs_u^i, we count all samples for which an

Algorithm 1: Using the samples of set Sam_p in phase p to compute the mapping m_p relating a bus stop s_u, after which an update was made, to the next stop s_i after which the next adjustment should be made.

```
Parameters: $c_m \in \mathbb{R}$ and $c_c \in \mathbb{R}$
 1: $\forall u \in \{0,\dots,n-1\} \forall i \in \{u,\dots,n\} : cs_u^i \leftarrow 0$
 2: $\forall u \in \{0,\dots,n-1\} \forall i \in \{u,\dots,n\} : cc_u^i \leftarrow 0$
 3: for all $sam \in Sam_p$ do
 4:   for $u = 0$ to $n-1$ do
 5:     for $i = u+1$ to $n$ do
 6:       if $|f_i(sam) - f_u(sam)| > 120$ then
 7:         $cs_u^i \leftarrow cs_u^i + 1$
 8:       end if
 9:       if $f_u(sam) - f_i(sam) > d_i$ then
10:         $cc_u^i \leftarrow cc_u^i + 1$
11:       end if
12:     end for
13:   end for
14: end for
15: for $u = 0$ to $n-1$ do
16:   $m_p[u] \leftarrow 0$
17:   $i \leftarrow u+1$
18:   while $i \le n$ and $m_p[u] = 0$ do
19:     if $\frac{d_i \cdot |Sam_p|}{cs_u^i} < c_m$ or $\frac{cc_u^i}{|Sam_p|} > c_c$ then
20:       $m_p[u] \leftarrow i$
21:     else
22:       $i \leftarrow i+1$
23:     end if
24:   end while
25: end for
26: return $m_p$
```

update at stop s_i leads to a major change, while cc_u^i is used to tell the upcoming critical changes. The outer loop from line 3 to 14 specifies that all elements sam of the set Sam_p containing testing samples from phase p are considered. By the middle loop between lines 4 and 13 and the inner one from line 5 to 12, we guarantee that the counters cs_u^i and cc_u^i are created for all pairs between a bus stop s_u and a stop s_i reached later by our bus. We use the value $u = 0$ for the case that no update was made yet and the PID shows the timetable-based ETA. The first if-statement between lines 6 and 8 checks if an update causes a major change. For convenience, we note by $f_i(sam)$ that all features of the sample sam until stop s_i are used. The function $f_0(sam)$ refers to the arriving time of the sampled bus at s_{PID} according to the timetable. By means of the second if-statement from line 9 to 11, we do the same for the critical cases. Here, d_i describes the average journey time of buses between stops s_i and s_{PID}.

The second part of the algorithm describes the creation of our mapping m_p. For each bus stop passed on the way, we search the first succeeding stop that exceeds at least one of the two thresholds c_m and c_c. The outer loop between lines 15 and 25 assures that all the bus stops s_u are considered, after which an update can be made. By $u = 0$, we specify the case that the PID still depicts the timetable-based ETA. It is used since we also want to find out after which bus stop the first real-time value should be displayed. In the lines 16 and 17, we initialize the mapping m_p and the counter i.

As expressed by the while-loop from line 18 to 24, we analyze the stops s_i following s_u until we find the first one exceeding one of the two thresholds expressed by the parameters c_m and c_c. That is specified by the condition of the if-statement between lines 19 and 23. The then-part models the case that one of the two conditions is fulfilled. Here, $m_p[k]$ is set to the current bus stop s_i and the while-loop is terminated. Otherwise, the counter i is incremented. The while-loop terminates at latest if i is set to $n + 1$. In this case, the value of $m_p[k]$ remains 0 describing that no further update of the ETA has to be made before reaching bus stop s_{PID}. After finishing the outer loop, the algorithm terminates by returning the mapping m_p.

When an update shall be made, our predictor determines the sleeping time interval as described in Sect. III. Using the mapping m_p for the current phase p, the current positions s_u of all the buses are found out and the candidates $s_{m_p[u]}$ for the next update are computed. Thereafter, we use either the real-time predictor of Entur [13] or our own forecasting functions f_i introduced in Sect. IV to determine the time $t_{m_p[u]}$, a bus is supposed to pass $s_{m_p[u]}$. After doing that for all relevant buses, the first of these time points is selected as the new update time.

VI. Testing and Discussion

To test our prediction method, we developed a simulator of the PID system. It predicts update time intervals based on the results of Algorithm 1. The real-time predictor provided by Entur is used to find out the proposed times of passing the bus stops assigned by mapping m_p since it will also be applied in the final realization of the PID system. We set the time interval for the next update 30 seconds later than the point of time computed by our predictor in order to enable the Entur system processing the incoming departure time of stop $s_{m_p[u]}$ first. The simulator collects various statistical data like the percentage of updates without changing ETAs which corresponds to unnecessary wake ups of the network receiver in the PID system. In addition, it records also the numbers of ETA changes of a certain magnitude.

Using the simulator, we carried out three tests using Algorithm 1 with different settings of the parameters c_m and c_c introduced in Sect. V:

- *A:* This is the most conservative test using $c_m = 6,000$ and $c_c = 0.001$. The setting of c_m means that a bus stop is selected for an update if the percentage of major updates in the testing samples exceeds the journey time to s_{PID} in minutes. Thus, if the journey time is 20 minutes, the share of major updates must be larger than 20% to select the stop for an update, while it is only 5% if the remaining travelling time is just five minutes. Alternatively, a stop

TABLE III
NUMBER OF PID UPDATES PER DAY AND PERCENTAGE OF UNNECESSARY UPDATES FOR SOME SELECTED BUS STOPS IN BERGEN

Bus stop	**Test**	**Updates per day**	**Unnecessary updates**
Sandvikstorget	*A*	251.5	57.0%
	B	211.8	62.8%
	C	145.7	44.5%
Torget	*A*	224.6	46.6%
	B	179.9	52.8%
	C	130.5	32.5%
Lillevågen	*A*	391.9	62.6%
	B	328.4	65.8%
	C	202.3	43.6%
Møhlenpris	*A*	107.3	31.0%
	B	79.8	34.3%
	C	61.4	8.6%

is selected for an update if more that 0.1% of all delay-reducing updates measured for the testing samples were critical changes.

- *B:* In this intermediate test, we apply the values $c_m = 4,000$ and $c_c = 0.0015$.
- *C:* This test is the most progressive one since, here, we use the settings $c_m = 2,000$ and $c_c = 0.002$.

We utilize the collected samples of bus line number 5 operated by Skyss in Bergen to carry out our tests. It connects the city center with two suburban areas at its edges. We selected the following stops:

- Sandvikstorget is the 28^{th} bus stop in the direction Loddefjord terminal of altogether 39 stops, that line 5 approaches. It was arbitrarily selected.
- Torget is the 33^{rd} stop in the same direction. It was chosen since it is close to the terminus with a journey time of around 32 minutes from the starting point. Moreover, it is sufficiently close to Sandvikstorget which allows us to make comparisons between the two stops.
- Lillevågen is the 27^{th} stop in the other direction Åsane terminal. It was selected since the journey time from the starting point is about the same as for Sandvikstorget in the other direction making it possible to compare similar travelling times over different routes.
- Møhlenpris is the fifth bus stop in the direction Åsane terminal. We use it since we wanted also to look at stations with only short journey times from the starting point.

The three tests lasted for around nine days each.

The average number of updates every day as well as the percentage of unnecessary updates for the four bus stops and three tests are listed in Tab. III. It is hardly surprising that the mean number of updates goes down if we use a more progressive test since, in this case, Algorithm 1 tends to select later stops for the next update which leads to larger update time intervals.

Astonishing, however, is the fact that, for all four stops, the number of updates for test *B* is higher than those for test *A*. One would expect that the larger intervals leads to a larger likelihood, that at least one ETA has to be updated. While this effect is clearly visible in test *C*, it is apparently not the case for the other ones. We see two potential reasons for that: One reason is that there can be a simple testing error. Since we wanted to collect as many data as possible in a relatively short amount of time, we did not look at the days of the week, the tests were carried out. Therefore, we did not recognize that test *A* covered only one but test *B* two weekends. While we use separate phases for Saturdays and Sundays (see Sect. IV) which should mitigate this effect, it can be that the algorithm handles these phases more conservatively than the various weekday phases. That might have lead to the larger number of unnecessary updates in test *B*.

The other reason is a potential accordion effect with respect to delays. Let us assume that s_i, s_j, and s_k are three arbitrary bus stops on a line with $i < j < k$, and there is a tendency that delays grow between s_i and s_j but are reduced between s_j and s_k. Then, a more conservative setting might lead the algorithm to select both, s_j and s_k for updates. Thus, the displayed ETA increases after passing s_j and again decreases after s_k such that no unnecessary updates take place. In contrast, a more relaxed setting of the mapping m_p might lead to the decision that the update at s_j is not necessary but that s_k or another stop close to it is the location of the next update. The two delay effects may cancel each other out, and the majority of buses have the same supposed ETA as at stop s_i such that the PID does not need to be updated.

Another interesting effect is that the average of daily updates for the bus stop Torget is lower than the one of nearby stop Sandvikstorget. The likely reason is a different degree of ambiguity on the routes directly ahead the two stops. A general property of Algorithm 1 is that it uses shorter update intervals if a bus is nearby. This is reasonable since the displayed ETAs have to be more precise when the arrival time is close. If, e.g., bus lanes and synchronized traffic lights ahead of Torget allow smoother rides, updates will be less often than in areas where traffic jams and traffic lights might cause varying delays. Torget is in the center of the city with a lot of traffic calmed roads such that this effect might be the reason.

The simulation for bus stop Lillevågen uses significantly more updates than for Torget and Sandvikstorget. Besides the impact of the immediate route ahead of a stop discussed above, the whole bus traffic in the direction Åsane terminal seems to be more ambiguous than in direction Loddefjord terminal. That leads to the selection of more stops for making updates and, in consequence, shorter update intervals. In addition, there are two daily buses more in this direction which operate in the late evening but that does not explain the magnitude of the additional updates.

The next aspect, we like to discuss about Tab. III, is that the general number of updates at Møhlenpris is much lower than for the other stops. This is easy to explain. Except for the rush hour times, in which line 5 operates every ten minutes, buses are just going every 20 or 30 minutes. The journey time of a bus from the starting point to Møhlenpris is only 11 minutes according to the timetable. Thus, after passing this stop, there will be no further updates for nine resp. 19 minutes until the

TABLE IV
MAGNITUDE OF DISPLAY CHANGES FOR THE SELECTED BUS STOPS IN DIRECTION LODDEFJORD TERMINAL

Sandvikstorget

Test	**Time to bus stop**	**0**	**Increase**			**Decrease**		
			1	**2**	**> 2**	**1**	**2**	**> 2**
A	Start	48.1	19.8	16.0	11.4	4.8	0.0	0.0
	Over 20	71.7	17.9	4.6	2.2	3.2	0.0	0.0
	15 to 20	73.6	15.7	2.5	1.0	7.0	0.0	0.2
	10 to 15	72.5	15.1	3.8	0.6	7.2	0.6	0.3
	5 to 10	71.1	18.4	2.1	0.1	7.5	0.4	0.4
	Under 5	78.0	10.4	0.3	0.1	9.8	0.9	0.4
B	Start	55.2	16.9	13.7	8.1	6.1	0.0	0.0
	Over 20	82.1	10.6	3.8	1.9	1.7	0.0	0.0
	15 to 20	74.9	15.2	4.0	0.9	4.5	0.5	0.0
	10 to 15	77.4	11.6	4.7	1.0	4.8	0.5	0.0
	5 to 10	71.5	16.0	2.4	0.3	8.4	1.2	0.2
	Under 5	81.0	8.8	0.6	0.1	8.9	0.4	0.3
C	Start	43.4	29.0	15.2	7.0	5.2	0.0	0.0
	Over 20	60.9	24.5	7.8	3.1	3.6	0.0	0.0
	15 to 20	67.5	22.4	3.5	2.0	4.3	0.4	0.0
	10 to 15	62.6	18.9	8.7	2.5	7.3	0.0	0.0
	5 to 10	58.7	20.8	7.6	3.3	8.4	1.3	0.0
	Under 5	68.8	15.2	1.8	0.0	12.2	1.7	0.4

Torget

Test	**Time to bus stop**	**0**	**Increase**			**Decrease**		
			1	**2**	**> 2**	**1**	**2**	**> 2**
A	Start	33.2	16.3	8.1	6.4	22.2	13.4	0.2
	Over 20	73.8	15.0	3.3	1.2	6.1	0.5	0.2
	15 to 20	69.3	16.7	2.9	1.9	8.2	0.8	0.3
	10 to 15	70.1	16.5	4.3	0.9	6.8	1.1	0.3
	5 to 10	62.8	18.9	1.8	0.7	13.4	1.8	0.7
	Under 5	73.0	9.0	0.2	0.1	15.5	1.8	0.3
B	Start	37.1	11.6	6.1	5.3	26.2	13.6	0.2
	Over 20	80.0	11.1	2.8	1.7	4.1	0.1	0.0
	15 to 20	76.3	10.1	3.5	2.3	7.4	0.4	0.0
	10 to 15	71.1	12.7	6.7	1.7	6.3	1.3	0.2
	5 to 10	68.5	14.0	3.2	0.2	10.6	3.0	0.4
	Under 5	76.4	7.1	0.3	0.3	13.5	2.0	0.3
C	Start	27.3	14.6	5.6	3.0	27.5	21.0	1.0
	Over 20	66.6	20.4	3.8	1.1	7.4	0.6	0.0
	15 to 20	69.7	9.5	3.5	2.0	14.4	0.5	0.5
	10 to 15	63.3	16.7	6.4	2.7	9.8	0.8	0.4
	5 to 10	41.4	23.3	15.8	11.6	4.7	2.3	0.9
	Under 5	60.7	11.7	1.0	0.0	22.1	3.4	1.2

TABLE V
MAGNITUDE OF DISPLAY CHANGES FOR THE SELECTED BUS STOPS IN DIRECTION ÅSANE TERMINAL

Lillevågen

Test	**Time to bus stop**	**0**	**Increase**			**Decrease**		
			1	**2**	**> 2**	**1**	**2**	**> 2**
A	Start	34.3	21.3	16.9	22.6	4.5	0.2	0.2
	Over 20	82.3	10.7	1.6	0.6	4.4	0.4	0.1
	15 to 20	80.1	13.2	2.0	0.3	4.2	0.1	0.0
	10 to 15	83.0	9.9	0.8	0.1	5.8	0.4	0.0
	5 to 10	83.3	9.4	0.6	0.1	6.3	0.3	0.1
	Under 5	79.4	13.0	0.9	0.1	6.1	0.5	0.0
B	Start	41.8	19.9	14.4	18.9	4.5	0.3	0.2
	Over 20	85.9	7.6	1.7	1.2	3.4	0.1	0.0
	15 to 20	84.3	9.6	1.9	0.3	3.6	0.3	0.0
	10 to 15	80.2	9.6	1.6	1.0	6.8	0.9	0.0
	5 to 10	83.4	8.4	0.5	0.2	7.2	0.3	0.0
	Under 5	81.0	11.4	1.2	0.3	5.6	0.3	0.1
C	Start	25.8	32.6	20.7	14.3	6.2	0.0	0.4
	Over 20	77.5	12.3	2.6	0.9	6.1	0.4	0.2
	15 to 20	71.9	14.1	4.4	0.5	8.4	0.5	0.2
	10 to 15	69.8	12.6	5.0	1.3	11.0	0.3	0.0
	5 to 10	64.6	15.2	2.4	0.3	16.7	0.9	0.0
	Under 5	65.5	19.0	2.4	0.6	10.6	1.5	0.3

Møhlenpris

Test	**Time to bus stop**	**0**	**Increase**			**Decrease**		
			1	**2**	**> 2**	**1**	**2**	**> 2**
A	Start	15.6	0.5	0.2	3.5	1.6	10.1	68.4
	Over 5	61.8	1.5	2.9	33.8	0.0	0.0	0.0
	Under 5	48.6	32.0	13.4	1.3	4.5	0.0	0.2
B	Start	24.6	0.5	0.0	3.1	1.0	10.3	60.6
	Over 5	87.8	0.0	0.0	12.2	0.0	0.0	0.0
	Under 5	52.8	26.8	16.5	2.6	1.3	0.0	0.0
C	Start	5.7	1.0	0.2	1.5	0.8	15.7	75.1
	All	25.8	22.6	12.9	38.7	0.0	0.0	0.0

next bus leaves the starting point. That decreases the number of updates significantly.

Finally, we like to compare the results of our tests in Tab. III with those created by the initial algorithm depicted in Tab. II. The number of daily tours at line 5 in Bergen is somewhere in between the overall traffic of bus stops Flakkråa and Hesthagen in Trondheim. Compared to that, we did not manage to reduce the number of unnecessary updates in our linear regression-based approach. The reason for that is that updates have often to be done for all buses since some few of them would lead to significant changes. For instance, in test *A*, an update is already made when just 0.1% of the samples would have led to a critical change. Since we do not know the situation in advance, we therefore conduct updates also for many buses for which the ETA is not changed. However, our algorithm reduces the overall number of daily updates significantly compared with the original algorithm. Below, we show that we can use the values of test *B* for Torget and *C* for the other bus stops such that there are not more than around 200 updates a day. That is less than a third compared with the stop Flakkråa and a fifth of the updates at Hesthagen.

Besides the average number of updates and the share of unnecessary ones, it is also relevant how our algorithm influences the adjustments of the ETAs displayed. In Tabs. IV and V, we depict the nature of the ETA adjustments at the PIDs for the four bus stops and three tests. Since it can be relevant, in which distance to bus stop s_{PID} an update takes place, we created several statistics. This allows us to separate initial from later updates and consider different remaining journey times to the stop s_{PID} carrying the PID system. The first statistics is denominated as "Start". It contains all cases in which the timetable-based ETA is replaced by one showing real-time data. The other statistics refer to updates changing from one real-time-based ETA to another one. Each one represents a five minute slot, e.g., "15 to 20" states that the remaining travelling time to s_{PID} is between 15 and 20 minutes. In column "0", we show the percentage that an update did not lead to an adjustment of the ETA for a bus. The other columns list the percentages of increasing resp. decreasing the ETA by one minute, two minutes, or more than two minutes.

For the bus stops Sandvikstorget, Torget, and Lillevågen, the majority of the updates does not lead to ETA adjustments which, as discussed above, holds particularly for test *B*. Factual increases or, to a lower degree, decreases are by only one minute. An exception is just the starting phase, in which, some larger increases are sensed. The reason is that buses sometimes arrive late at their terminus such that they start already delayed to their next trip as well. Other larger changes by two minutes or more are mostly delay increases that result from being stuck in unexpected traffic jams, etc. In contrast, major ETA decreases are rare. The only exception is the stop Torget which quite often leads to a reduction of two minutes when the timetable-based ETA is replaced by a real-time value for the first time. The reason may be similar to the one discussed later for Møhlenpris but since the journey time from the starting point is 32 minutes according to the timetable, we do not see a major issue here.

Important is to check the number of critical ETA reductions close before the bus reaches the bus stop s_{PID} which might lead to passengers missing it since they saw overly long delay displays before. In test *A*, we had one case for both, Sandvikstorget and Torget in which the reduction was more than ten minutes. Our guess is that a bus was so heavily delayed that it was replaced by another vehicle on the route which sometimes happen. Neither Entur nor our approach based on linear regression are able to handle such extreme situations, at least not for the first bus stops after the replacement. Otherwise, Tabs. IV and V reveal that ETA decreases close to the stop are rare and nearly all of the cases were by exactly three minutes. Of course, with less than five minutes to go, this is borderline but it happened only once every three or four days. The Public Transportation Authority (PTA) has to decide if this low likelihood is acceptable.

Interestingly, the use of the very progressive test *C* did not significantly increase the number of critical ETA reductions for the stops Sandvikstorget and Lillevågen such that the corresponding setting of Algorithm 1 seems appropriate for the two stops. However, changing from test *B* to *C* quadrupled the number of delay reductions by three minute reductions close to the stop Torget. Thus, with 1.2%, the corresponding share is far too large. It is further striking that the percentage of increases by one minute or more in the period between five and ten minutes until the stop is much higher than in the other two tests. Thus, it seems that the ETA is often increased too much in this time interval which has then to be corrected in the next adjustment even closer before reaching the bus stop. Here, it is advisable to use the settings of the constants from test *B* or to apply at least the constant c_c for the critical cases.

We separate the discussion of bus stop Møhlenpris since the simulator results for it are digressive. In around three quarters of all cases in all three tests, the ETA is reduced by exactly three minutes during the initial adjustment. That effect holds not only for the Entur-based real-time prediction but also when using function f produced by linear regression. Mostly, this ETA is thereafter kept until the bus passes the stop. The reason for that could be revealed by looking at some of the samples, we collected. According to the timetable, the journey time to Møhlenpris from the preceding stop Lyngbø riksveg is six minutes while the one from Møhlenpris to Festplassen, the next stop, is just one minute. In nearly all checked samples, however, the time from Lyngbø riksveg to Møhlenpris was around 3.5 minutes and the one to Festplassen 2.5 minutes. Altogether, the trip from the initial stop Loddefjord terminal to Møhlenpris needs mostly eight minutes instead of the eleven minutes it would take according to the timetable. Our algorithm handled this case correctly by assigning the initial stop for the first update immediately after leaving the starting point. That allows the PID system to correct the displayed ETA as fast as possible. That holds for all tests and all phases of the day. The apparent solution is to correct the timetable such that the correct ETAs is already displayed to the passengers from Møhlenpris even before a bus leaves the first stop.

This case is also a good example to discuss the accordion effect. The error of the timetable did not effect the behavior for the stop Lillevågen since all three test cases are sufficiently progressive to miss the untimeliness that is corrected by increasing delays at the stops following Møhlenpris. A more conservative setting of the algorithm, however, might have detected it, which would have led to more but unproductive corrections of the ETAs. This example gives evidence that it is an important task to select suitable settings of the parameters c_m and c_c such that the accordion effect is prevented. That must not come, however, at the cost of a too large number of critical cases as experienced in the test *C* for the bus stop Torget.

VII. Related Work

To our best knowledge, there is, yet, no other work about predicting the time for updates at PIDs mounted at bus stops with machine learning technology in order to preserve power. Nevertheless, there are quite some contributions about using machine learning for the prediction of real-time ETAs.

In [18], the authors evaluate the influence of both upstream signalized intersection and surrounding traffic flow on the prediction of arrival times at bus stops. The change of the traffic density at different locations is used to detect the average bus speed, which is constantly updated according to the traffic density at different locations. The comparison between the forecasted and the actual arrival times at two bus stops in Jinan, China, shows that there is a low mean relative error between them. The authors of [19] used historical data collected by automated passenger counters (APC) in order to develop models for predicting bus arrival times at stops along a route. The computation effort of the proposed model to predict the arrival time of a bus at a certain stop is low, making it a potential model also for providing real-time bus-arrivals.

The authors of [20] use GPS data to create a central server that receives and stores real-time GPS data provided by Android-based smartphones which are installed in the buses. The implementation of an arrival time bus stop predictor using the collected data and the real-time information will be the next step of the authors. In [21], Kalman filters are applied to integrate the information about delays into the prediction of bus travelling times. This technique is also utilized in [22]. Here, the authors divide the bus routes into segments and use static transit data that represent routing information like timetables. Moreover, they apply recorded arrival and departure information at selected transit stops together with real-time updates and collected data from mobile apps. This includes anonymous information about the location, when the passengers get on/off the buses, their walking distances to bus stops, etc. The authors use the data to predict arrival time delays using clustering analysis and the Kalman filters.

The position of buses is also considered by the authors of [23] who apply a kernel regression model to represent the dependencies between position updates and the arrival times at bus stops. The data set consists of a collection of real-time GPS measurements. Furthermore, in order to reduce the size of historical data sets, the authors propose a strategy based on interpolation points. They show that their strategy outperforms the linear regression model and predictions based on the K-nearest neighbor algorithm.

In [24], a real-time prediction model is proposed which exploits long-range dependencies across multiple time steps and heterogeneous measurements. In order to encode heterogeneous measurements into a vector space, the authors introduced the one-hot coding technique. Further, they use Recurrent Neural Networks (RNN) [25] with Long Short-Term Memory (LSTM) to exploit long-range dependencies across multiple time steps. The results of this method were compared with other state-of-the-art machine learning techniques like linear regression, kernel regression, and support vector machine. The authors showed that their method offers the best performance on the considered data set.

Other work discusses the general usefulness of installing PIDs mounted to bus stops. In [1], [4], the authors present the results of a study about providing public transport passengers in Dublin, Ireland, with various kinds of information. The results show that the use of fixed PIDs is highly appreciated. However, as mentioned in the introduction, another study by [5] revealed that the desire for PIDs is far lower if that leads to a significant increase of the ticket fee by 0.60€ or more. Another study was made about mobile passenger information systems [3]. It makes clear that the layout of the user interface has a significant impact on the popularity.

Some papers introduce actual passenger information systems using fixed PIDs at the bus stops. The approaches in [26]–[28] present various information systems containing PIDs that, however, all use power from the electrical grid. More interesting in the context of the project IoT-STOP is the work introduced in [29]. Like us, the authors apply a wireless PID system using e-papers and batteries, albeit based on LoRaWAN instead of NB-IoT. Unlike us, however, they concentrate less on power reduction since a more power-intensive Raspberry-Pi is used for the controller. Further, the article does not mention any methods to reduce the impact of idle listening. In consequence, using a battery of 24,000 mAh, their system can only operate for around 76 hours which is not useful in practice. An early trial of using e-papers for PIDs in Birmingham, UK, is introduced in [30]. Here, solar panels are used as power source and Near Field Communication (NFC) for data transfer. The system seems to be used for changing timetable information but not for real-life ETA updates.

VIII. Concluding Remarks

In this article, we reported on a method to predict update times. That allows the control units of Passenger Information Displays (PID) to switch off their network receivers temporarily without spoiling the value of the displayed Expected Times of Arrival (ETA). In particular, we use linear regression to forecast arrival times that are used by our predictor to detect places which call for the update of a PID.

The tests presented in Sect. VI are highly encouraging but, of course, the predictor still provides potential for improvement. We will continue to test bus stops with varying settings of the parameters c_m and c_c in Algorithm 1. Further, we will test other strategies to determine the thresholds which are used to decide if an update should be made when a bus passes a certain bus stop.

The results from [23], [24] indicate that also the machine learning algorithm used to forecast the delays can be improved. A major obstacle to be overcome is the state-based nature of forecasting ETAs based on the leaving times from intermediate bus stops. When a bus passes a stop and the corresponding leaving time is recorded, the number of features to be considered is increasing. Many machine learning methods are not suited to cope with changing numbers of features (see [16]). That holds also for linear regression but, at least, it has the advantage to be well-performing such that a large number of runs can be executed. So, it is possible to use an own forecasting function for each bus stop that is passed over the journey of a bus. Other techniques that sometimes afford hours or even days for a single run, are too slow to produce the dozens of models needed to solve our problem. Recurrent Neural Networks (RNN) [25], however, are suited to systems with varying spatiotemporal properties. That fits well to ETA forecasting, and promise the use of only a single model for Algorithm 1 instead of a whole set of functions $f_1, \ldots, f_n$. Therefore, we work also on an RNN-based system and will report about the results of our experiments in the future.

Another issue to be considered is the scalability of our approach. The tests revealed that each bus stop is different and needs a particular setting for the constants used in Algorithm 1. For instance, test *C* showed good results for Sandvikstorget and Lillevågen while Torget should be better operated with the constants used in test *B*. For a handful of demonstrators, the correct setting of the mapping m_p for a stop can be done manually but that would not be practical if the PID systems shall be used at hundreds of different bus stops. This scalability problem can be solved by using digital twins [31] of the PID systems. A simulator not unlike the one applied for our tests can run in parallel to the real physical system and record important properties like the magnitude of ETA changes that were taken. Moreover, various update policies, i.e., mappings m_p, are produced for each PID system with varying degrees of progressiveness. The simulator can now regularly check the collected numbers for the quality and use a more progressive mapping if the numbers seem to be too conservative while the cumulation of critical results might lead to use a more conservative policy. Thus, over time, each PID system will automatically adapt to a policy which is suited to its particular situation. We plan to develop such a system which also allows us to update the mappings from time to time to reflect events like timetable changes or altered traffic routings.

Acknowledgment

We like to show our gratitude to the colleagues from FourC for their valuable support. Moreover, we thank the other project partners from Telenor, Skyss, and Kringom for their contributions as well as Innovation Norway for providing us with the necessary funding.

References

[1] B. Caulfield and M. O'Mahony, "An Examination of the Public Transport Information Requirements of Users," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 21–30, 2007.

[2] K. Papangelis, S. Sripada, D. Corsar, N. Velaga, P. Edwards, and J. D. Nelson, "Developing a Real Time Passenger Information System for Rural Areas," in *Human Interface and the Management of Information (HIMI)*, ser. LNCS 8017. Las Vegas, NV, USA: Springer-Verlag, 2013, pp. 153–162.

[3] S. Beul-Leusmann, C. Samsel, M. Wiederhold, K.-H. Krempels, E.-M. Jakobs, and M. Ziefle, "Usability Evaluation of Mobile Passenger Information Systems," in *Design, User Experience, and Usability (DUXU)*, ser. LNCS 8517. Heraklion, Greece: Springer-Verlag, 2014, pp. 217–228.

[4] B. Caulfield and M. O'Mahony, "Stated Preference Evaluation of Passenger Information in Dublin," in *37th Annual Conference of the Universities Transport Study Group*, Bristol, UK, 2005, p. 12 pages.

[5] I. Politis, P. Papaioannou, S. Basbas, and N. Dimitriadis, "Evaluation of a Bus Passenger Information System from the Users' Point of View in the City of Thessaloniki, Greece," *Research in Transportation Economics*, vol. 29, no. 1, pp. 249–255, 2010.

[6] FourC, "FourC Homepage," https://www.fourc.eu/, 2020, accessed: 2020-08-29.

[7] M. Wang, C. Lin, H. Du, H. Zang, and M. McCreary, "Electrophoretic Display Platform Comprising B, W, R Particles," *SID Digest*, vol. 45, no. 59.1, pp. 857–860, 2014.

[8] K. R. Lai, P. K. Sahoo, C. Y. Chang, and C. C. Chen, "Reduced Idle Listening Based Medium Access Control Protocol for Wireless Sensor Networks," in *International Conference on Communications and Mobile Computing*, vol. 3, 2010, pp. 329–333.

[9] R. Ratasuk, N. Mangalvedhe, Y. Zhang, M. Robert, and J.-P. Koskinen, "Overview of Narrowband IoT in LTE Rel-13," in *IEEE Conference on Standards for Communications and Networking (CSCN)*. Berlin, Germany: IEEE, 2016, pp. 1–7.

[10] NPerf, "3G / 4G / 5G Coverage Map," https://www.nperf.com/, 2020, accessed: 2020-09-02.

[11] E. Puka, P. Herrmann, T. Levin, and C. B. Skjetne, "A Way to Measure and Analyze Cellular Network Connectivity on the Norwegian Road System," in *10th International Conference on Communication Systems & Networks (COMSNETS)*. Bengaluru, India: IEEE Computer, Jan. 2018, pp. 595–600.

[12] MQTT.org, "Message Queuing Telemetry Transport," http://mqtt.org/, 2020, accessed: 2020-09-02.

[13] Entur AS, "Real-Time Data," https://developer.entur.org/pages-real-time-intro, 2020, accessed: 2020-09-02.

[14] K. Axelsson, T. Ekblom, and A. Olsson, "How to Supply Bus Stops with Electricity without Connecting them to the Electricity Grid," Bachelor's Thesis, Uppsala University, Jun. 2013.

[15] A. Azari and G. Miao, "Energy Efficient MAC for Cellular-based M2M Communications," in *Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2014, pp. 128–132.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[17] X. Yan and X. G. Su, *Linear Regression Analysis — Theory and Computing*. World Scientific Publishing, 2009.

[18] H. Zhang, S. Liang, Y. Han, M. Ma, and R. Leng, "A Prediction Model for Bus Arrival Time at Bus Stop Considering Signal Control and Surrounding Traffic Flow," *IEEE Access*, vol. 8, pp. 127 672–127 681, 2020.

[19] Shaowu Cheng, Baoyi Liu, and Botao Zhai, "Bus Arrival Time Prediction Model based on APC Data," in *6th Advanced Forum on Transportation of China (AFTC)*. Beijing, China: IEEE, 2010, pp. 165–169.

[20] Q. Zhang, Y. Zhang, and J. Li, "EasyComeEasyGo: Predicting Bus Arrival Time with Smart Phone," in *9th International Conference on Frontier of Computer Science and Technology*. Dalian, China: IEEE, 2015, pp. 268–273.

[21] R. P. S. Padmanaban, L. Vanajakshi, and S. C. Subramanian, "Automated Delay Identification for Bus Travel Time Prediction towards APTS Applications," in *2nd International Conference on Emerging Trends in Engineering & Technology*. Nagpur, India: IEEE, 2009, pp. 564–569.

[22] F. Sun, Y. Pan, J. White, and A. Dubey, "Real-Time and Predictive Analytics for Smart Public Transportation Decision Support System," in *IEEE International Conference on Smart Computing (SMARTCOMP)*. St. Louis, MO, USA: IEEE, 2016, pp. 1–8.

[23] M. Sinn, J. W. Yoon, F. Calabrese, and E. Bouillet, "Predicting Arrival Times of Buses using Real-time GPS Measurements," in *15th International IEEE Conference on Intelligent Transportation Systems*. Anchorage, AK, USA: IEEE, 2012, pp. 1227–1232.

[24] J. Pang, J. Huang, Y. Du, H. Yu, Q. Huang, and B. Yin, "Learning to Predict Bus Arrival Time From Heterogeneous Measurements via Recurrent Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3283–3293, 2019.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, pp. 533–536, 1986.

[26] K. Ganesh, M. Thrivikraman, J. Kuri, H. Dagale, G. Sudhakar, and S. Sanyal, "Implementation of a Real Time Passenger Information System," *Computing Research Repository (CoRR)*, vol. abs/1206.0447, 2012.

[27] R. K. Megalingam, N. Raj, A. L. Soman, L. Prakash, N. Satheesh, and D. Vijay, "Smart, Public Buses Information System," in *International Conference on Communication and Signal Processing*. Melmaruvathur, India: IEEE, 2014, pp. 1343–1347.

[28] D. Vakula and B. Raviteja, "Smart Public Transport for Smart Cities," in *International Conference on Intelligent Sustainable Systems (ICISS)*. Palladam, India: IEEE, 2017, pp. 805–810.

[29] T. Boshita, H. Suzuki, and Y. Matsumoto, "Smart Bus Stop using LoRaWAN and e-Paper," in *8th Global Conference on Consumer Electronics (GCCE)*. Osaka, Japan: IEEE, 2019, pp. 278–282.

[30] G. Scott, C. Wilson, and G. Tyler, "E-paper Public Transport Information System," in *Road Transport Information and Control Conference (RTIC)*. London, UK: IET, 2014, pp. 1–6.

[31] S. W. Loke, S. Smanchat, S. Ling, and M. Indrawan, "Formal Mirror Models: An Approach to Just-in-Time Reasoning for Device Ecologies," *International Journal of Smart Home*, vol. 2, no. 1, pp. 15–31, 2008.