MEED: An Unsupervised Multi-Environment Event Detector for Non-Intrusive Load Monitoring

Daniel Jorde, Matthias Kahl, and Hans-Arno Jacobsen
Chair for Application and Middleware Systems
Technische Universität München, Germany
Email: daniel.jorde@tum.de

***Abstract*—The accurate detection of transitions between appliance states in electrical signals is the fundamental step that numerous energy conserving applications, such as Non-Intrusive Load Monitoring, rely on. So far, domain experts define rules and patterns to detect changes of appliance states and to extract detailed consumption information of individual appliances subsequently. Such event detectors are specifically designed for certain environments and need to be tediously adapted for new ones, as they require in-depth expert knowledge of the environment. To overcome this limitation, we propose a new unsupervised, multi-environment event detector, called MEED, that is based on a bidirectional recurrent denoising autoencoder. The performance of MEED is evaluated by comparing it to two state-of-the-art algorithms on two publicly available datasets from different environments. The results show that MEED improves the current state of the art and outperforms the reference algorithms on a residential (BLUED) and an office environment (BLOND) dataset while being trained and used fully unsupervised in the heterogeneous environments.**

I. Introduction

One of the challenges humanity is facing nowadays is the depletion of natural energy resources, while the overall energy demand keeps increasing, especially the demand for electrical energy [1]. For this reason, researchers are striving to find solutions to improve the way limited energy resources are used. By making both industrial and residential consumers aware of their detailed electricity consumption, one aims to reduce the waste of energy [2]. Several surveys indicate that appliance-level information can reduce energy consumption by raising consumer awareness [3]. The consumption of individual appliances can be acquired using Non-Intrusive Load Monitoring (NILM) methods with a low-cost single-sensor approach to record an aggregated signal, measured only at the mains of a building or industrial plant [2]. After extracting relevant signal segments using an event detection algorithm, the appliances that had caused the events can be identified and the signal can be decomposed into the individual appliances. Besides using appliance-level information to save energy, it enables other applications, such as detecting malfunctions in appliances to reduce maintenance costs [2].

Most of the electrical data used in NILM is collected by smart meters, which usually sample the signals at a low rate (< 1 kHz). As a result, only some of the major devices can be detected [2]. Data that are sampled using higher rates increase the probability for successful NILM [4] and allow to detect more devices, especially low consumption ones [2]. Although high-sampling-rate data contains a high amount of information, most machine learning algorithms can not be used on it as they suffer from the curse-of-dimensionality caused by the sampling rate [5]. Hence, it is necessary to reliably extract relevant segments from the overall signal, i.e., appliance-state transitions, that can be used to identify appliance-level behavior.

In the past, researchers focused on methods for low-sampling-rate data, driven by the high costs (for metering, storage, and processing) associated with the acquisition of high-sampling-rate data. As there is an evident lack of methods and because of the advantages of high-sampling-rate data, we focus on this domain. Detecting events and distinguishing them from signal noise is particularly challenging and prone to errors. So far, researchers focused mainly on residential buildings and their appliances. Thus, there are multiple residential datasets publicly available [6]. Recent work also investigates industrial and office settings and new datasets are published [7].

Existing event detection algorithms exhibit one important disadvantage: Most rely on customized, expert-made event definitions. This prohibits such approaches to generalize well to a setting they were not designed for. Subsequently, the hyperparameters and the algorithms themselves need to be tediously fine-tuned for being used in a new setting.

The main contribution of this paper is to present a new multi-environment event detector (MEED) that does not rely on a dedicated event definition while being trained and used fully unsupervised. Hence, MEED can be used in different environments without the need to preset additional hyperparameters, enabling new possibilities for NILM and energy-related applications in general. Furthermore, MEED detects events more reliable than the existing state of the art, inducing fewer errors into subsequent analysis steps. We compare MEED with two state-of-the-art algorithms on two publicly available datasets from different environments. By doing so, we improve the current state of the art on the residential BLUED [6] and the office environment BLOND [7] datasets.

The rest of the paper is organized as follows: In Section II we give an overview on event detection in NILM and relevant metrics. Section III summarizes related work, followed by the description of MEED in Section IV. Subsequently, we detail our experimental setup in Section V and discuss the corresponding results in Section VI. Section VII then concludes this paper.

978-1-5386-8099-5/19/$31.00 ©2019 IEEE

II. Background

Based on the foundational work on energy disaggregation by George Hart [8], multiple new algorithms that use events to extract relevant signal segments have been proposed.

A. Event Detection

As the majority of researchers use different, specific definitions of events that reduce their capability to generalize to new settings, we use a general event definition. In particular, we define events to be transitions between states of individual appliances. Event detection algorithms can be divided into ones using supervised- and unsupervised learning, depending on the use of labeled data during training (supervised) or not (unsupervised). Supervised algorithms are less flexible in tasks like event detection than unsupervised algorithms, as they are highly dependent on the event definition used. Thus, we propose an unsupervised event detector.

Another possibility to classify event detection methods are the three categories introduced by Anderson et al. [9], namely Expert Heuristics (EH), Probabilistic Models (PM), and Matched-Filters (MF). Rule-based approaches, including simple threshold-based ones, and methods using machine learning are considered to be EH, whereas approaches using statistical metrics to determine events belong to the PM category. MF approaches match learned event masks with the signal to detect events [9].

B. Metrics

The following commonly used metrics are based on the scores of the confusion matrix, namely the amount of records that are True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}} \quad \text{FPP} = \frac{\text{FP}}{\text{TP}+\text{FN}}$$

$$\text{F1-Score} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

Another common metric is the True Positive Percentage (TPP), which is defined as $\frac{\text{TP}}{\text{E}} = \frac{\text{TP}}{\text{TP}+\text{FN}}$ [9]. As this is equal to the recall metric, we omitted the TPP in our evaluation. It is of particular importance to define how the single scores of the confusion matrix are computed to ensure the comparability of the results [10]. Ground truth events are often generated by human experts, and thus can be imprecise with respect to their exact location in time. Hence, it is common to define a tolerance limit τ for the matching of detected e_{det} and ground truth events e_{gt} [10]–[12]. A detected event e_{det} is a TP if there exists a ground truth event within an interval of $\pm\tau$, i.e., if $\exists e_{gt} : e_{det} - \tau \leq e_{gt} \leq e_{det} + \tau$.

III. Related Work

In the following, we summarize the current state of the art in event detection in NILM on high-sampling-rate data and identify gaps. The essential characteristics of the related algorithms are listed in Table I, with the scores being rounded to the second decimal digit. In case the exact value for a particular metric is unclear from the publication, we declared the result to be approximate ($\approx$). We further categorize the related work according to the following criteria. The "Setting" column indicates the environment the algorithms are designed for and evaluated in, namely Residential (R) or Industrial (I) environments. The cross-validation criterion (CV) reports whether cross-validation was used in the evaluation, as suggested by Makonin and Popowich [25] to improve the reliability of the results. Most of the publications exhibit no information about the tolerance limit τ used to calculate the metrics. Hence, the scores can not be directly, but only approximately, compared.

Valovage and Gini introduce a PM [11] that relies on a Bayesian detection method at its core. The algorithm tries to partition the signal into run sequences, followed by declaring the transitions between such sequences as events.

Pereira [12] introduces another PM. The algorithm makes use of a log likelihood ratio detector to estimate a detection statistic. The algorithm then searches the signal for extreme values to determine events.

In contrast to the other PM, Wild et al. [17] use a kernel Fisher discriminate analysis to detect start and end times of event and non-event segments in a supervised way.

Alcala et al. [13] use the signal's envelop of the normalized current and voltage RMS values and a threshold to detect events. Another approach using the signal's envelope is based on a Hilbert Transform [14]. It further applies an average and a derivative filter to obtain a set of spikes to detect transitions.

The algorithm proposed by de Baets et al. [15] transforms the signal into the frequency domain and applies a threshold on the computed Cepstrum components to detect events.

As part of an unsupervised NILM system, Barsim et al. [16] developed a three-step unsupervised event detection algorithm. First, they separate steady and transient states by applying the mean-shift clustering algorithm to logarithmized real and reactive power values. Afterward, they use expectation-maximization clustering and Gaussian mixture models to detect the time limits of the events. In the last step, the algorithm verifies the detected events according to expert-defined constraints for the specific setting.

The approach by Trung et al. [20] makes use of the cumulative sum statistics (CUMSUM) and an adaptive threshold to detect the start and the end of the transient segments in the signal. Furthermore, Zhu et al. [22] use CUMSUM to detect events in a residential setting.

In contrast to methods designed for residential settings, Cox et al. [24] use the voltage distortion to detect events in industrial data.

Yang et al. [23] compare an EH and a goodness-of-fit (GOF) event detector, concluding that the latter performs better.

Barsim and Yang [18] use the unsupervised DBSCAN clustering algorithm. Zheng et al. [19] also show the applicability of DBSCAN to detect events. The clustering algorithm detects various appliance states and their transitions before they are post-processed with dedicated thresholds on power and time to

TABLE I
RELATED WORK (ACRONYMS ARE EXPLAINED IN THE TEXT)

Ref.	Learning	Setting	Type	Dataset	CV	Recall	Precision	FPR	FPP	F1-Score	Limit τ
[11]	Unsup.	R	PM	BLUED A | B	No	-	-	-	-	$\approx$ 0.98 | $\approx$ 0.80	2 s
[12]	Unsup.	R	PM	BLUED A | B	No	-	-	-	-	$\approx$ 0.96 | $\approx$ 0.72	1 s
				REDD	No	-	-	-	-	$\approx$ 0.80	-
[13]	Unsup.	R	EH	BLUED A | B	No	0.94 | 0.88	-	0.88e-3 | 0.12	-	-	-
[14]	Unsup.	R	EH	REDD subset	No	0.93	-	0	0	-	-
[15]	Sup.	R	EH	BLUED A | B	Var.	-	-	-	-	0.98 | 0.80	-
[16]	Unsup.	R	EH	BLUED A | B	No	0.99 | $\approx$ 0.70	-	-	$\approx$ 0.55 | $\approx$ 0.09	-	-
[17]	Sup.	R	PM	BLUED A | B	No	0.99 | 0.86	0.99 | 0.92	-	-	0.99 | 0.89	-
[18]	Unsup.	R	EH	BLUED A | B	No	0.97 | 0.68	0.99 | 0.93	-	0.78e-2 | 0.49e-1	0.98 | 0.79	-
[19]	Unsup.	R	EH	BLUED A | B	No	0.99 | 0.88	-	0.99 | 0.69	0.71e-2 | 0.4	0.99 | 0.77	3 s
[20]	Unsup.	R	EH	REDD subset	No	0.94	-	-	-	-	-
[21]	Unsup.	R	PM	Non Public	-	-	-	-	-	-	-
[22]	Unsup.	R	EH	Non Public	-	-	-	-	-	-	-
[23]	Unsup.	R	PM, EH	Non Public	-	-	-	-	-	-	-
[24]	Unsup.	I	EH	Non Public	-	-	-	-	-	-	-

filter out duplicates and FPs. For its transparent evaluation and its state-of-the-art performance on BLUED, we have selected the latter approach as one of the two algorithms to benchmark the performance of MEED.

In addition to the first reference algorithm, we have selected the GOF based approach by Jin et al. [21] as it has multiple advantages over other probabilistic event detectors, despite not being evaluated on a public dataset. In contrast to other algorithms, this GOF approach provides a guideline for the selection of the event window hyperparameter and a closed form for the decision threshold. Furthermore, the authors show the superiority of their algorithm over the widely used generalized log likelihood detector [21].

Based on the related work, the base requirements for MEED are as follows: MEED has to be designed independent of any specific event definition, allowing it to generalize better, even to different settings. Furthermore, this multi-environment event detector has to identify events unsupervised to avoid the need for costly human labeled events.

IV. EVENT DETECTION APPROACH

Based on the introduced requirements, we design a new window-based EH event detector. MEED applies a two-step process to the pre-processed input data to detect events, as shown in Figure 1. The first step is based on the mean-squared reconstruction error (MSE) of the signal window, which is computed by an autoencoder. Subsequently, we apply a peak detection algorithm to find the exact timestamps within the windows that exceed an absolute MSE value.

A. Data Pre-Processing

One of the features that is used in multiple event detection methods is CUMSUM [20], [22] as it has shown to reveal trends and changes in time-series data successfully. Small fluctuations are suppressed in CUMSUM, while substantial changes in the data are revealed. The input to the first step of MEED is the CUMSUM of the five-period root-mean-square (RMS) of the current signal. The CUMSUM at time t_i is computed by accumulating the deviations of the RMS values from the window's mean value for all $t \leq t_i$. Therefore, CUMSUM reveals trends in the signal, while suppressing small fluctuations. During training, we scale the CUMSUM input signal to a value range between -1 and 1.

B. Coarse Event Detection Autoencoder

The first step of MEED is to apply a denoising autoencoder to the input windows. In doing so, we seize the rarity of events to build a model that can detect signal windows that contain transitions. A similar idea is used to solve classical anomaly detection tasks, as presented in the summary on this topic by Chandola et al. [26]. The authors present several methods to identify unexpected events. One of these methods applies a neural network, in particular, an autoencoder, to learn a hidden representation h of the input x to reveal events [26].

Autoencoders are neural networks that are typically used in representation learning tasks. They consist of an encoder $h = f(x)$ and a decoder function $r = g(h)$ [5]. As events are rare compared to non-events, they constitute the minority class, making it hard for the model to learn a proper event representation h. Instead, the model learns to represent and reconstruct the non-event majority class. Hence, a large reconstruction error is produced when the model faces a deviation from normal behavior, i.e., an event. MEED's hyperparameter settings can generalize well between environments, as events rarely occur in all electrical environments. Denoising autoencoders, a special case of the general autoencoder, minimize a loss function $L(x, g(f(\widetilde{x})))$, while trying to reconstruct the input x from intentionally corrupted input data $\widetilde{x}$. The corrupted input $\widetilde{x}$ is obtained by adding white Gaussian noise ($\mu = 0$, $\sigma = 0.25$) to x.

Each of the three hidden layers of MEED are bidirectional long short-term memory (LSTM) cells that have shown to be able to learn long term dependencies and patterns from sequential data [5]. Briefly, a LSTM cell is a memory cell with non-linear gating units (i_t, o_t, f_t and c_t) that control its information flow, as summarized in the following equations,

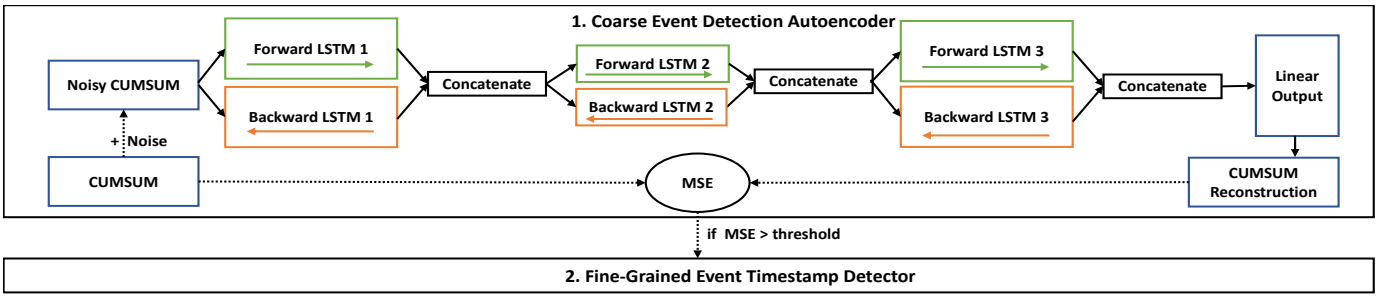

Fig. 1. MEED Architecture, with step 1 (the coarse detection autoencoder) and step 2 (the fine-grained event timestamp detector)

with sig being the logistic sigmoid, $tanh$ the hyperbolic tangent function, and $\circ$ the element-wise product [5]:

$$f_t = sig(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = sig(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = sig(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ tanh(c_t)$$

The input window to MEED has a fixed size, thus, we use bidirectional LSTMs to make use of future context information to improve the overall performance [27], as depicted in the autoencoder step in Figure 1. The bidirectional networks consist of one LSTM to process the data in the forward direction, i.e., from the beginning of the window to the end, and one separate LSTM to do so in the opposite direction [27]. The outputs of both directions are concatenated into a single output to compute the MSE to optimize the network. On top of the LSTM layers, the linear output layer ensures that the in- and output have the same dimensionality. We evaluated several hyperparameters and different architectural components using a grid search on the training data, resulting in the final optimal settings shown in Table II.

TABLE II
AUTOENCODER HYPERPARAMETERS, IMPLEMENTED WITH KERAS [28]

Parameter	Value	Parameter	Value
learning rate	0.001	LSTM 1 cell size	216
optimizer	Adam [29]	LSTM 2 cell size	108
initializer	Glorot uniform [30]	LSTM 3 cell size	216
merge mode	concatenate	window size	10 s

C. Fine-Grained Event Timestamp Detector

In case the coarse event detector determines an input window to be an event window, i.e., the reconstruction error threshold is exceeded, we apply the fine-grained detection procedure to determine the exact number and timestamps in the window. The algorithm aims to detect significant state-changes, while supressing ones that are caused by noise. We compute the RMS values over five periods of the raw input signal. Then, we convert the RMS signal into a binary one, setting values higher than the mean of the values to 1 and the smaller ones to 0. Afterward, all transitions between the binary values are determined. We use two hyperparameters to distinguish between relevant and noise related transitions, namely the min_time and the $fluctuation\_threshold$ parameter. The first one is used to ensure a minimum time between transitions. In particular, consecutive transitions are suppressed that belong to the same event, therefore it is set to a value of 2. The latter parameter filters out events that are caused by noise in the standardized signal. It suppresses small fluctuations that amount to an RMS value smaller than 1.

V. DATASETS AND EXPERIMENTAL SETUP

We evaluated our approach on two distinct publicly available datasets from a residential and an office environment. Furthermore, we compare our algorithm to the re-implemented approaches by Jin et al. [21] and Zheng et al. [19]. We use cross-validation to evaluate all algorithms. In the coarse detection step, we selected a threshold on the MSE of 2, guided by the reconstruction errors produced during training. In general, a wide range of thresholds is feasible, as non-event windows produce MSE errors close to zero, while events result in high errors.

For the evaluation, we set the tolerance limit for the calculation of the performance metrics to $\tau = 1$ s, as proposed by Pereira [12], ensuring a minimum precision in time that is necessary for the majority of the NILM algorithms. The first dataset we used, namely BLUED, contains the voltage and current, sampled at 12 kHz, of a house with a two-phase connection [6]. The first phase (Phase A), has only a few devices attached to it, producing over-optimistic detection results [12]. Phase B, on the other hand, resembles the actual consumption of typical households, as it contains more diverse devices and a higher, more realistic noise level than Phase A. Consequently, we use Phase B to obtain a performance benchmark in a realistic setting. The BLUED dataset contains a few minutes of corrupted data, reducing the number of events for Phase B by 4 to 1574. In particular, the events on 26.10 from 01:22:00 to 01:24:00 are removed. As BLUED is divided into 16 folders, we apply 16-fold cross-validation with one day allocated for training in each iteration.

Additionally, we use an office environment dataset, namely BLOND [7]. The aggregated electrical signals are sampled at 50 kHz, making BLOND the only publicly available office

dataset that is sampled with high frequency. The appliance composition in office environments is substantially different from the one in households, as they contain more devices like, for example, laptops that induce noise into the signals. The results are 5-fold cross-validated, with the first five days of November 2016 being used for training and the subsequent ten days used for testing. BLOND is the only office environment dataset that is publicly available, but as it does not provide ground truth labels, we sampled a subset of the detected events for each approach separately to calculate the TPs and FPs and thus the precision to compare the algorithms. The appropriate sample size n is determined by assuming a binomial probability distribution for the TPs and FPs, using a Wilson score interval to estimate n [31]. We use the scores of MEED on the BLUED dataset as a prior guideline for the success probability p of the distribution. We expect the number of the TPs to be slightly lower, due to the noise level of the BLOND dataset. Hence, we account for this by subtracting 0.1 from p. The sample size is then calculated using an α value of 0.1 and a confidence interval with a conservative width of 0.4 around the estimate for p. This results in a minimum sample size per day of $n = 10$ for MEED, of $n = 13$ for the EH by Zheng et al. [19], and of $n = 14$ for the PM by Jin et al. [21].

VI. Experimental Results

The following results are obtained by running the input windows through the coarse event detection autoencoder and the fine-grained timestamp detector that constitute MEED. The cross-validated results of the detection performance benchmark on BLUED are shown in Figure 2, with MEED clearly outperforming the other algorithms. Furthermore, Figure 2 shows that the cross-validation scores are within a narrow range for all algorithms, hence the influence of the individual training folds in BLUED is limited.

TABLE III
Mean Scores achieved on BLUED Phase B, with $\tau = 1$ s

Algorithm	F1-Score	Recall	Precision	FPR	FPP
MEED	**0.75**	0.69	**0.83**	0.03e-3	0.14
Jin et al. [21]	0.51	**0.87**	0.36	0.04e-2	1.55
Zheng et al. [19]	0.51	0.41	0.69	0.046e-3	0.18

Averaging the scores over the folds of the cross-validation, one obtains a stable result for the overall performance of the algorithms, as shown in Table III. The autoencoder clearly outperforms the two reference algorithms with regard to the F1-Score. The algorithm by Zheng et al. [19] performs worse compared to the results presented in the original paper. This can be explained by the use of a different tolerance limit τ. When using $\tau = 3$ s, like Zheng et al. [19] in the original paper, MEED achieves an F1-Score of 0.79, hence, also outperforming the F1-Score that is reported by Zheng et al. [19]. The scores achieved show that MEED is more resilient to noise than the two reference algorithms, resulting in higher overall precision. The PM by Jin et al. [21] achieves a higher recall, but at the cost of a high amount of FP events. When

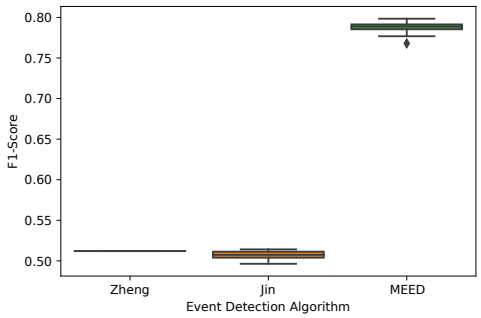

Fig. 2. Cross-validated F1-Scores for the algorithms of Zheng et al. [19], Jin et al. [21] and MEED on BLUED Phase B

trading off precision against recall, precision should be favored as FP events propagate through the entire analytical pipeline of NILM, leading to problems in the subsequent steps. In this context, missing out single events is not as severe as detecting multiple FPs. Regarding the FPs that MEED detected on BLUED, the majority of them corresponds in fact to true events that are not labeled accordingly in the data. Despite the use of BLUED as the de-facto standard dataset for the evaluation of event detection algorithms, one can see that the ground truth has some flaws, as also claimed by Zheng et al. [19]. As there are no labels for the BLOND dataset, our sampling procedure only allows us to compute the TP and FP events, thus, no F1-Score can be calculated. Looking at the results on the BLOND dataset in Table IV, one has to account for human labeling bias and the uncertainty caused by the sampling process. Despite this, MEED achieves a precision that significantly outperforms the other two algorithms on all three channels (1, 2, 3) of BLOND.

TABLE IV
Best Scores achieved on BLOND, with $\tau = 1$ s

	TP			FP			Precision		
Algorithm	1	2	3	1	2	3	1	2	3
MEED	86.8	69	89.6	64.4	41.6	46	**0.58**	**0.62**	**0.66**
Jin et al. [21]	53.6	39.2	80	138.4	113.4	116	0.28	0.26	0.40
Zheng et al. [19]	49	31	60	126	68	69	0.28	0.31	0.47

The amount of samples differs between the channels and the algorithms, as some of the evaluation days are no work days. Due to the low activity on such days, the number of detected events is smaller than the sample size n per day, resulting in an overall lower amount of samples. The variation of the scores between the MEED models as shown in Figure 3 indicates that an intelligent selection of typical work days for training can improve the model performance. In conclusion, one can see that MEED detected events with high precision in both settings. Consequently, MEED substantially reduces the amount of FPs that are induced into the analysis pipeline, while reliably detecting the majority of the events.

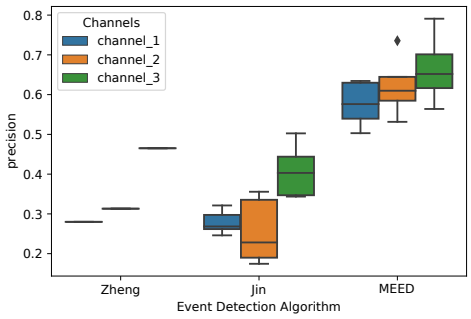

Fig. 3. Cross-validated precision scores for the algorithms of Zheng et al. [19], Jin et al. [21] and MEED on BLOND

VII. Conclusion

We propose MEED, a new multi-environment event detector, that does not rely on a dedicated event definition and that generalizes well to different environments without the need to adapt the model. In contrast to the present state of the art, MEED requires no dedicated expert knowledge about the environment it is used in. We compare MEED, a fully unsupervised bidirectional recurrent denoising autoencoder, to two reference event detectors on a residential (BLUED) and an office environment dataset (BLOND). In doing so, MEED achieves state of the art results with an F1-Score of 0.75 on BLUED and clearly outperforms the reference algorithms on BLOND, while using a tolerance limit of one second.

Acknowledgment

This research was supported by the Federal Ministry For Economic Affairs and Energy based on a decision by the German Bundestag.

References

[1] European Commission, "EU Energy in Figures," 2018.

[2] C. Armel, A. Gupta, G. Shrimali, and A. Albert, "Is Disaggregation the Holy Grail of Energy Efficiency? The Case of Electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013.

[3] J. Kelly and W. J. Knottenbelt, "Does Disaggregated Electricity Feedback reduce Domestic Electricity Consumption? A Systematic Review of the Literature," *CoRR*, vol. abs/1605.00962, 2016. [Online]. Available: http://arxiv.org/abs/1605.00962

[4] R. Dong, L. Ratliff, H. Ohlsson, and S. Sastry, "Fundamental Limits of Nonintrusive Load Monitoring," *HiCoNS 2014 - Proceedings of the 3rd International Conference on High Confidence Networked Systems*, Oct. 2013.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts and London, England: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[6] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, "BLUED: A Fully Labeled Public Dataset for Event-based Non-Intrusive Load Monitoring Research," *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability*, pp. 1–5, Jan. 2012.

[7] T. Kriechbaumer and H.-A. Jacobsen, "BLOND, a Building-Level Office Environment Dataset of Typical Electrical Appliances," *Scientific Data*, vol. 5, 2018.

[8] G. W. Hart, "Nonintrusive Appliance Load Monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[9] K. D. Anderson, M. E. Berges, A. Ocneanu, D. Benitez, and J. M. F. Moura, "Event Detection for Non Intrusive load monitoring," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct. 2012, pp. 3312–3317.

[10] L. Pereira and N. Nunes, "An Experimental Comparison of Performance Metrics for Event Detection a Algorithms in NILM," in *4th International Workshop on NILM*, 2018.

[11] M. Valovage and M. Gini, "Label Correction and Event Detection for Electricity Disaggregation," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 990–998.

[12] L. Pereira, "Developing and Evaluating a Probabilistic Event Detector for Non-Intrusive Load Monitoring," in *2017 Sustainable Internet and ICT for Sustainability (SustainIT)*, Dec. 2017, pp. 1–10.

[13] J. M. Alcalá, J. Ureña, and Á. Hernández, "Event-Based Energy Disaggregation Algorithm for Activity Monitoring From a Single-Point Sensor," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 10, pp. 2615–2626, Oct. 2017.

[14] ——, "Event-based Detector for Non-Intrusive Load Monitoring based on the Hilbert Transform," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sept. 2014, pp. 1–4.

[15] L. de Baets, J. Ruyssinck, D. Deschrijver, and T. Dhaene, "Event Detection in NILM using Cepstrum Smoothing," in *3rd International Workshop on NILM*, 2016, pp. 1–4.

[16] K. S. Barsim, R. Streubel, and B. Yang, "Unsupervised Adaptive Event Detection for Building-Level Energy Disaggregation," *Proceedings of power and energy student summit (PESS)*, 2014.

[17] B. Wild, K. S. Barsim, and B. Yang, "A new Unsupervised Event Detector for Non-Intrusive Load Monitoring," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec. 2015, pp. 73–77.

[18] K. S. Barsim and B. Yang, "Sequential Clustering-Based Event Detection for Non-Intrusive Load Monitoring," in *Computer Science & Information Technology*, vol. 6, 2016, pp. 77–85.

[19] Z. Zheng, H. Chen, H. Xiaowei, and L. Xiaowei, "A Supervised Event-Based Non-Intrusive Load Monitoring for Non-Linear Appliances," *Sustainability*, vol. 10, p. 1001, March 2018.

[20] K. N. Trung, E. Dekneuvel, B. Nicolle, O. Zammit, C. N. Van, and G. Jacquemod, "Event Detection and Disaggregation Algorithms for NIALM System," in *2nd International Workshop on NILM*, 2014.

[21] Y. Jin, E. Tebekaemi, M. Berges, and L. Soibelman, "Robust Adaptive Event Detection in Non-Intrusive Load Monitoring for Energy Aware Smart Facilities," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 4340–4343.

[22] Z. Zhu, S. Zhang, Z. Wei, B. Yin, and X. Huang, "A Novel CUSUM-Based Approach for Event Detection in Smart Metering," *IOP Conference Series: Materials Science and Engineering*, vol. 322, no. 7, 2018.

[23] C. C. Yang, C. S. Soh, and V. V. Yap, "Comparative Study of Event Detection Methods for Non-intrusive Appliance Load Monitoring," *Energy Procedia*, vol. 61, pp. 1840 – 1843, 2014.

[24] R. Cox, S. B. Leeb, S. R. Shaw, and L. K. Norford, "Transient Event Detection for Nonintrusive Load Monitoring and Demand Side Management using Voltage Distortion," in *21st Annual IEEE Applied Power Electronics Conference and Exposition, 2006. APEC '06.*, March 2006.

[25] S. Makonin and F. Popowich, "Nonintrusive Load Monitoring (NILM) Performance Evaluation," *Energy Efficiency*, vol. 8, pp. 809–814, Dec. 2014.

[26] V. Chandola, A. Banerjee, and V. K. Vipin, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, July 2009.

[27] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[28] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[30] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proceedings of the 13. International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[31] W. W. Piegorsch, "Sample Sizes for Improved Binomial Confidence Intervals," *Computational Statistics and Data Analysis*, vol. 46, no. 2, pp. 309–316, June 2004.