Energy & Buildings 270 (2022) 112282

Contents lists available at ScienceDirect

Energy & Buildings

journal homepage: www.elsevier.com/locate/enb

Uncertainty quantification for appliance recognition in non-intrusive load monitoring using Bayesian deep learning ☆

Lorin Werthen-Brabants *, Tom Dhaene, Dirk Deschrijver

Ghent University - imec, Technologiepark-Zwijnaarde 126, 9052 Ghent, Belgium

ARTICLE INFO

Article history:
Received 20 April 2022
Revised 9 June 2022
Accepted 26 June 2022
Available online 04 July 2022

Keywords:
Deep learning
Uncertainty quantification
NILM
NIALM
SGHMC
Neural Networks
Appliance classification
Bayesian neural networks

ABSTRACT

Non-Intrusive Load Monitoring (NILM) can be used to detect, recognize, and classify switching events of individual electrical appliances from an aggregate power signal that is measured at the main line of the grid connection. A limitation of existing solutions is that deep learning models tend to overfit the data and do not express their uncertainty when making predictions. This paper shows that uncertainty information can be obtained in a natural way by making use of Bayesian Neural Networks. Having this information is very valuable, because it supplies relevant information about potential misclassifications of the model to an end-user. The source of these misclassifications can be attributed to ambiguous data, or the model requiring more examples to learn from. In this work, an increase in generalization performance is observed when making use of Stochastic Gradient Hamiltonian Monte Carlo over Stochastic Gradient descent, and the usefulness of uncertainty in a NILM context is discussed.

1. Introduction

In the past years, several advanced classification models for NILM have been proposed to identify individual electrical appliances from a measured, aggregate power signal (a.k.a. appliance identification) [1–10]. This leads to a higher cost-effectivity as compared to installing a single meter for every appliance or socket in a household.

The introduction of *Deep Learning* techniques has improved the accuracy of appliance identification. However, NILM often deals with non-stationary conditions. New appliances are connected, more complex appliances may exhibit unseen behaviors, and sensor quality may degrade over time, which are all sources of *concept drift*. As a consequence, these state-of-the-art deep learning techniques may misclassify certain events without having the ability to express their *uncertainty*.

This work shows that *Bayesian Neural Networks* (BNNs) are able to quantify different types of uncertainty, on which the developer of a NILM system may put a threshold, signifying the user of the system of a likely misclassified event, as well as providing information on the root cause of misclassification. The method is validated on the VI trajectories [11] of all appliances in the PLAID [1] benchmark dataset. Additionally, an increase in accuracy is observed by using Bayesian techniques, as they have an inherent regularizing behaviour.

2. Types of uncertainty

Uncertainty of a prediction consists of two major components: the *epistemic* uncertainty u_e and the *aleatoric* uncertainty u_a [12]. The epistemic uncertainty encodes "how many" different parameter weights $\boldsymbol{\theta}$ of a model fit the dataset $\mathcal{D}$ optimally. Aleatoric uncertainty refers to the amount of inherent uncertainty in the data.

Epistemic Uncertainty The epistemic uncertainty u_e displayed by a model parameterized by $\boldsymbol{\theta}$ and trained on dataset $\mathcal{D}$ is measured by the posterior probability distribution: $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, as derived from Bayes' Rule. This posterior distribution exhibits high entropy if a model was not trained on enough data to explain the observation. This type of uncertainty is therefore also referred to as *reducible*, as more training data reduces this uncertainty. A concrete example for Appliance Identification would be as follows: say a classifier has been trained on a given dataset $\mathcal{D}$. If a new appliance is introduced,

☆ This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

* Corresponding author.
E-mail addresses: lorin.werthenbrabants@ugent.be (L. Werthen-Brabants), tom.dhaene@ugent.be (T. Dhaene), dirk.deschrijver@ugent.be (D. Deschrijver).

https://doi.org/10.1016/j.enbuild.2022.112282
0378-7788/

or an existing appliance exhibits never before seen behavior, the epistemic uncertainty would generally be high, as multiple parameterizations of the model fit the novel observation equally well (or rather equally bad).

Aleatoric Uncertainty The aleatoric uncertainty u_a expressed by a model is encoded in the conditional distribution of a class, given an input and a set of weights: $p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$. This describes the inherent noise present in the data and cannot be explained away by introducing more data. The aleatoric uncertainty typically rises when a model learns that some VI trajectories are inherently hard to classify (to its own ability). In this case, the epistemic uncertainty can be low, signifying many parameterizations leading to the same result, but with high aleatoric uncertainty. In other words, there is certainty of inherent ambiguities in the data in this case.

Predictive Uncertainty The predictive uncertainty is the sum of the epistemic and aleatoric uncertainty $u_p(\mathbf{x}) = u_e(\mathbf{x}) + u_a(\mathbf{x})$. This describes the total uncertainty present when making a prediction on a certain input, accounting for both the uncertainty the model has of its parameters, and the inherent ambiguity present in the data.

It must be noted that epistemic and aleatoric uncertainty are not absolute notions. For small, simple models — such as linear or logistic regression — epistemic uncertainty might be low, but aleatoric uncertainty high. This may be the case when the simple model has no better parametrization fitting the given dataset $\mathcal{D}$ [12].

A common way of quantifying the uncertainty of the distributions that encode the epistemic and aleatoric uncertainties, is to sample the distribution and calculate its entropy:

$$H[p(x)] = -\sum_{x\in\mathcal{X}} p(x)\log_2 p(x). \tag{1}$$

Eq. (1) describes how "surprising" the distribution $p(x)$ is. When dealing with ensembles, low (high) values of entropy indicate a high (low) agreement between different parameterizations $\boldsymbol{\theta}_n$.

3. Methodology - Stochastic Gradient Hamiltonian Monte Carlo

The parameters of a Bayesian Neural Networks (BNN) are not point estimates, but rather a probability distribution $p(\boldsymbol{\theta}|\mathcal{D})$. The entropy of the resulting predictions is a measure of the model uncertainty, or epistemic uncertainty. The method used to obtain a BNN is *Hamiltonian Monte Carlo* (HMC) [13,14], considered to be the gold standard to approximate [15] a predictive posterior distribution.

Hamiltonian Monte Carlo (HMC) is a *Metropolis–Hastings Markov Chain Monte Carlo* (MH-MCMC) [16] technique based on the idea of Hamiltonian Dynamics. MCMC is a class of algorithms to sample from probability distributions. Starting from a random guess of parameter values $\boldsymbol{\theta}$, MH-MCMC *walks* to another point in the parameter space $\boldsymbol{\theta}\prime$ that, given a loss function, is more likely than the previous parameters $\boldsymbol{\theta}$. All the parameters $\boldsymbol{\theta}$ that are visited are stored, and approximate the distribution of possible parameters. As a result, the output predictions also become distributions. A step-by-step overview of MH-MCMC techniques is as follows:

1. Start with a random initial guess $\boldsymbol{\theta}_1$
2. Sample a candidate around the last one $\boldsymbol{\theta}\prime \sim Q(\boldsymbol{\theta}\prime|\boldsymbol{\theta}_n)$. This could be a simple distribution, such as $Q(\boldsymbol{\theta}\prime|\boldsymbol{\theta}_n) = \mathcal{N}(\boldsymbol{\theta}_n, \sigma^2\mathbf{I})$.
3. If this is a more likely sample from the distribution we want to sample from than the previous one, choose this sample with a Bernoulli probability $p = \min\left(1, \frac{Q(\boldsymbol{\theta}\prime|\boldsymbol{\theta}_n)}{Q(\boldsymbol{\theta}_n|\boldsymbol{\theta}\prime)}\frac{p(\boldsymbol{\theta}\prime|\mathcal{D})}{p(\boldsymbol{\theta}_n|\mathcal{D})}\right)$. The posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ is often approximated in Deep Learning using the log-likelihood function.

Given sufficient time, the history of this Markov chain becomes an approximation of the target distribution $p(\boldsymbol{\theta}|\mathcal{D})$. As n approaches infinity, the average of the model output

$$\hat{f}(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x};\boldsymbol{\theta}_n) \tag{2}$$

converges to the true expectation

$$\lim_{n\to\infty}\hat{f}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]. \tag{3}$$

In practice, this set of parameters constitutes an ensemble of neural networks with the same architecture but different weights, each producing different predictions. These individual networks may be biased to certain portions of the training data, but as an ensemble are able to make accurate and well-calibrated predictions.

In the ordinary MH-MCMC algorithm stated above, the proposed weights are sampled directly from the proposal density. HMC modifies the regular MH-MCMC by introducing two components: a random momentum vector $\boldsymbol{\omega}$ and the Hamiltonian dynamics of the system. Betancourt [17] elaborates on this concept and provides the proper intuition of Hamiltonian Monte Carlo.

A further extension used in this work is *Stochastic Gradient Hamiltonian Monte Carlo* (SGHMC) [18] that uses Hamiltonian dynamics to improve convergence [13]. This method solves the infeasible calculation of a required gradient computation when simulating a Hamiltonian dynamical system with a large sample size, which is common in deep learning. SGHMC uses minibatches of data as a noisy estimate of the gradient of the entire dataset. This results in a method which has a correspondence to stochastic gradient descent with momentum, but with the added benefit that uncertainties can be calculated on the parameters resulting from evaluation of this technique.

The total predictive uncertainty of an ensemble is found by averaging the predictions of the ensemble, and calculating the resulting entropy [12]:

$$u_p(\mathbf{x}) = -\sum_{y\in\mathcal{Y}}\left(\frac{1}{N}\sum_{i=1}^{N} p(y|\boldsymbol{\theta}_i,\mathbf{x})\right)\log_2\left(\frac{1}{N}\sum_{i=1}^{N} p(y|\boldsymbol{\theta}_i,\mathbf{x})\right). \tag{4}$$

The resulting uncertainty estimate $u_p(\mathbf{x})$ also includes the epistemic uncertainty about the network weights $\boldsymbol{\theta}$. Fixing a single set of weights removes this uncertainty. Therefore, the expectation over the entropies of these distributions for an ensemble model

$$u_a(\mathbf{x}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{y\in\mathcal{Y}} p(y|\boldsymbol{\theta}_i,\mathbf{x})\log_2 p(y|\boldsymbol{\theta}_i,\mathbf{x}). \tag{5}$$

is a measure of aleatoric uncertainty.

Finally, the epistemic uncertainty is simply calculated as the difference

$$u_e(\mathbf{x}) = u_p(\mathbf{x}) - u_a(\mathbf{x}). \tag{6}$$

4. Application – NILM

The methodology is applied to the PLAID dataset [19,1] where an increase of generalization performance is observed. Additionally, a qualitative analysis of aleatoric and epistemic uncertainty is applied to the dataset, showcasing the the use and usefulness for these quantities.

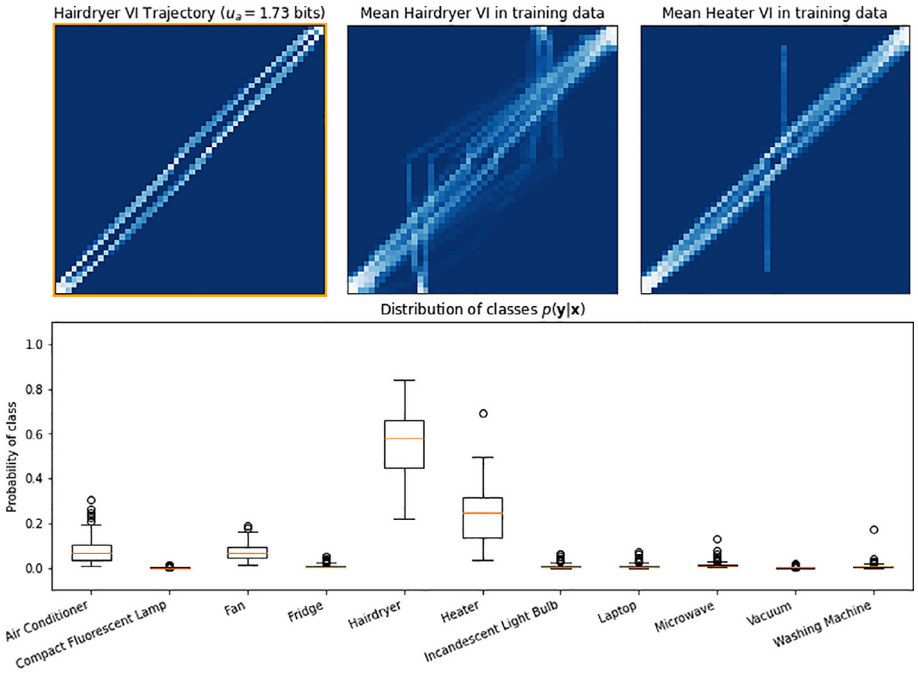

Fig. 1. A visualisation of aleatoric uncertainty in NILM. Top: VI trajectory of example data point (Hairdryer), and means of VI trajectories of the Hairdryer and Heater classes. Bottom: boxplot of class probability outputs over all parametrizations $\boldsymbol{\theta}_i$.

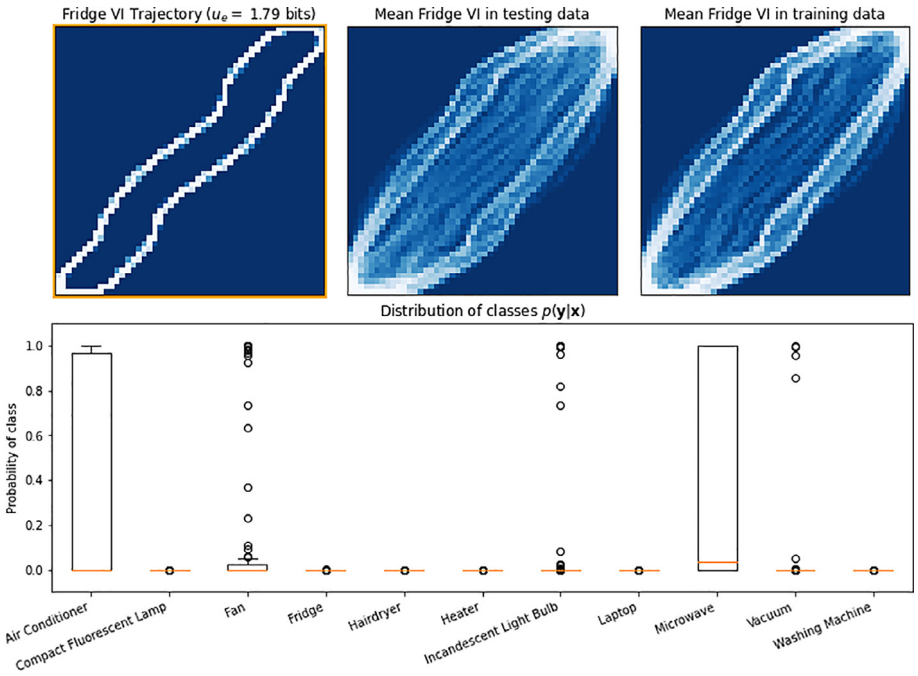

Fig. 2. A visualisation of epistemic uncertainty in NILM. Top: VI trajectory of example data point (Fridge), and means of VI trajectories of the Air conditioner and Microwave classes. Bottom: boxplot of class probability outputs over all parametrizations $\boldsymbol{\theta}_i$.

4.1. Dataset

PLAID [19,1] is a public dataset including sub-metered voltage (V) and current (I) measurements sampled at 30 kHz for 11 different appliance types. More than 200 individual appliances are available, captured in 55 households. For each appliance, at least 5 start-up events are measured, resulting in a total of 1074 measurements. The time series are transformed into VI trajectories as follows [2,11]: First, a transient in the current needs to be located at time t, signifying an *on*-event. Then, making use of the corresponding AC voltage time series, 20 cycles are skipped to allow for the current to fall into a steady state at time $t\prime$. From then onwards, both current and voltage are graphed so the voltage is on the x-axis and the current on the y-axis for C cycles. The result-

ing graph is then discretized by assigning bins at regular intervals in the x and y axes and counting the amount of occurences of a (V, I) pair in a given bin.

The following tests are performed using a leave-one-house-out cross-validation. This means per fold a house is left out for the test set, and the remaining houses are split into train and validation sets.

4.2. Training of the Neural Network

The neural network architecture of De Baets et al. [2] is modified to compare Stochastic Gradient Descent (SGD) and HMC. The basic structure is as follows: it takes as input a 50×50 VI trajectory, normalized to contain values between 0 and 1. Next, a convolutional layer with 20 filters of size 5, a pooling layer, another convolutional layer with 20 filters of size 5, another pooling layer, a layer which flattens the resulting features and an output layer with K nodes are added, where K denotes the amount of classes available in a given training set. The original model used a larger amount of filters and a hidden layer before the final layer, but due to computational restrictions a smaller version of this model was constructed.

First, the baseline neural network is trained with the Adam optimizer [20], a modified SGD which generally yields better results without heavy fine-tuning.

Next, the SGHMC method is sampled 250 times, while throwing away the first 50 samples (burn-in period) and only retaining every second model afterwards (thinning out). This results in 100 remaining parametrizations.

4.3. Improvement in accuracy

First, an improvement in accuracy is noted. In the leave-one-house-out setup, the weighted F_1-score of the network trained with SGD is 0.713, while that of the network trained with SGHMC is **0.730**. This is expected, as BNNs are naturally regularizing [21], with SGHMC having a connection with SGD with momentum, a common regularization method [18]. Conceptually, the Bayesian methods converge to the posterior probability distribution $p(\boldsymbol{\theta}|\mathcal{D})$, while a regular deterministic model would only be one sample out of these possible parameters $\boldsymbol{\theta}$. As a result, the BNN stores more information, and can be seen as an ensemble method, which often performs better than an analogous non-ensemble method [22].

4.4. Aleatoric uncertainty

Fig. 1 shows an example of aleatoric uncertainty arising for a given input. This sample is chosen as the one where the aleatoric uncertainty is dominant and takes on the highest value. This data point belongs to the Hairdryer class. The two classes that are most prominently visible in the output distribution are Hairdryer and Heater. Intuitively, this aligns with the expected resistive behaviour, as both devices contain similar electric components such as a heating element. The mean VI trajectories of the two most probable classes also imply there is a similarity between the VI trajectories visually. As expected, high aleatoric uncertainty arises when there are ambiguities in the data that are inherently difficult to discriminate.

4.5. Epistemic uncertainty

Fig. 2 shows an example of epistemic uncertainty arising for a given input. This sample is chosen as the one where the epistemic uncertainty is dominant and takes on the highest value. This data point belongs to the Fridge class. Here a clear confusion is seen between multiple classes, with Air Conditioner and Microwave being the most prominent. The distribution of classes this time contains many outliers, and no clear majority can be found between the different parametrizations of the model. This implies the input (Fridge VI trajectory) is not properly represented in the training data, and therefore requires more examples to reduce the error.

In Figs. 3 and 4 in the work by De Baets et al. [2], it turns out that the fridge is one of the appliances with lowest F1-measure and is most often confused with other appliances. This appliance type is also one of the least occurring in the specific fold highlighted in Fig. 2, only accounting for 53 of 1348 samples, or 4.38% of the training data.

5. Conclusion

The potential of uncertainty quantification and Bayesian Neural Networks is showcased in NILM by providing possible root cause identification of misclassifications to the end-user in the forms of epistemic, aleatoric and predictive uncertainty. The capability of uncertainty quantification for detecting potentially misclassified predictions is demonstrated by making use of Stochastic Gradient Hamiltonian Monte Carlo. This Bayesian Neural Network technique improves the weighted F_1-score of the appliance identification task to 0.730, compared to 0.713 with the same baseline model. This indicates a better generalization and avoidance of overfitting.

CRediT authorship contribution statement

Lorin Werthen-Brabants: Conceptualization, Methodology, Software, Validation, Formal-analysis, Investigation, Writing-original-draft, Visualization. **Tom Dhaene:** Writing-review-editing, Supervision. **Dirk Deschrijver:** Conceptualization, Writing-review-editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Medico, L. De Baets, J. Gao, S. Giri, E. Kara, T. Dhaene, C. Develder, M. Bergés, D. Deschrijver, A voltage and current measurement dataset for plug load appliance identification in households, Sci. Data 7 (1) (2020) 49, https://doi.org/10.1038/s41597-020-0389-7.
[2] L. De Baets, J. Ruyssinck, C. Develder, T. Dhaene, D. Deschrijver, Appliance classification using VI trajectories and convolutional neural networks, Energy Build. 158 (2018) 32–36, https://doi.org/10.1016/j.enbuild.2017.09.087.
[3] S. Giri, M. Bergés, An energy estimation framework for event-based methods in Non-Intrusive Load Monitoring, Energy Convers. Manage. 90 (2015) 488–498, https://doi.org/10.1016/j.enconman.2014.11.047.
[4] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, R. Steinmetz, On the accuracy of appliance identification based on distributed load metering data, in: 2012 Sustainable Internet and ICT for Sustainability (SustainIT), 2012, pp. 1–9.
[5] A. Ruano, A. Hernandez, J. Ureña, M. Ruano, J. Garcia, NILM Techniques for Intelligent Home Energy Management and Ambient Assisted Living: A Review, Energies 12 (11) (2019) 2203, https://doi.org/10.3390/en12112203.
[6] C. Zhang, M. Zhong, Z. Wang, N. Goddard, C. Sutton, Sequence-to-point learning with neural networks for non-intrusive load monitoring, 32nd AAAI Conference on Artificial Intelligence, AAAI 2018 (2018) 2604–2611.
[7] A.P. Medeiros, L.N. Canha, D.P. Bertineti, R. de Azevedo, Event Classification in Non-Intrusive Load Monitoring Using Convolutional Neural Network, in: 2019 IEEE PES Innovative Smart Grid Technologies Conference - Latin America (ISGT Latin America), IEEE, Gramado, Brazil, 2019, pp. 1–6. doi:10.1109/ISGT-LA.2019.8895291.
[8] M. Khodayar, J. Wang, Z. Wang, Energy Disaggregation via Deep Temporal Dictionary Learning, IEEE Trans. Neural Netw. Learn. Syst. 31 (5) (2020) 1696–1709, https://doi.org/10.1109/TNNLS.2019.2921952.
[9] J. Jiang, Q. Kong, M.D. Plumbley, N. Gilbert, M. Hoogendoorn, D.M. Roijers, Deep Learning-Based Energy Disaggregation and On/Off Detection of Household

Appliances, ACM Trans. Knowl. Discov. Data 15 (3) (2021) 50:1–50:21. doi:10.1145/3441300.
[10] L. Mauch, B. Yang, A novel DNN-HMM-based approach for extracting single loads from aggregate power signals, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 2384–2388. doi:10.1109/ICASSP.2016.7472104.
[11] T. Hassan, F. Javed, N. Arshad, An Empirical Investigation of V-I Trajectory Based Load Signatures for Non-Intrusive Load Monitoring, IEEE Trans. Smart Grid 5 (2) (2014) 870–878, https://doi.org/10.1109/TSG.2013.2271282.
[12] E. Hüllermeier, W. Waegeman, Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods, Mach. Learn. 110 (3) (2021) 457–506, https://doi.org/10.1007/s10994-021-05946-3.
[13] S. Duane, A. Kennedy, B.J. Pendleton, D. Roweth, Hybrid Monte Carlo, Phys. Lett. B 195 (2) (1987) 216–222, https://doi.org/10.1016/0370-2693(87)91197-X.
[14] M.D. Hoffman, A. Gelman, et al., The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo, J. Mach. Learn. Res. 15 (1) (2014) 1593–1623.
[15] F.K. Gustafsson, M. Danelljan, T.B. Schon, Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 318–319. arXiv:1906.01620.
[16] S. Chib, E. Greenberg, Understanding the Metropolis-Hastings Algorithm, Am. Stat. 49 (4) (1995) 327–335.
[17] M. Betancourt, A Conceptual Introduction to Hamiltonian Monte Carlo, arXiv:1701.02434 [stat] arXiv:1701.02434.
[18] T. Chen, E. Fox, C. Guestrin, Stochastic Gradient Hamiltonian Monte Carlo, in: Proceedings of the 31st International Conference on Machine Learning, PMLR, 2014, pp. 1683–1691.
[19] J. Gao, S. Giri, E.C. Kara, M. Bergés, PLAID: A public dataset of high-resolution electrical appliance measurements for load identification research, in: BuildSys 2014 - Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings, ACM Press New York, New York, USA, 2014, pp. 198–199, https://doi.org/10.1145/2674061.2675032.
[20] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015.
[21] A. Kendall, Y. Gal, What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?, arXiv:1703.04977 [cs] arXiv:1703.04977.
[22] T.G. Dietterich, Ensemble methods in machine learning, in: International Workshop on Multiple Classifier Systems, Springer, 2000, pp. 1–15.