An Empirical Study of Uncertainty Estimation Techniques for Detecting Drift in Data Streams

Anton Winter
TU Darmstadt, Germany
anton.winter@stud.tu-darmstadt.de

Nicolas Jourdan
TU Darmstadt, Germany
n.jourdan@ptw.tu-darmstadt.de

Tristan Wirth
TU Darmstadt, Germany
tristan.wirth@gris.tu-darmstadt.de

Volker Knauthe
TU Darmstadt, Germany
volker.knauthe@gris.tu-darmstadt.de

Arjan Kuijper
TU Darmstadt, Germany
arjan.kuijper@igd.fraunhofer.de

Abstract

In safety-critical domains such as autonomous driving and medical diagnosis, the reliability of machine learning models is crucial. One significant challenge to reliability is concept drift, which can cause model deterioration over time. Traditionally, drift detectors rely on true labels, which are often scarce and costly. This study conducts a comprehensive empirical evaluation of using uncertainty values as substitutes for error rates in detecting drifts, aiming to alleviate the reliance on labeled post-deployment data. We examine five uncertainty estimation methods in conjunction with the ADWIN detector across seven real-world datasets. Our results reveal that while the SWAG method exhibits superior calibration, the overall accuracy in detecting drifts is not notably impacted by the choice of uncertainty estimation method, with even the most basic method demonstrating competitive performance. These findings offer valuable insights into the practical applicability of uncertainty-based drift detection in real-world, safety-critical applications.

1 Introduction and Motivation

In high-stakes scenarios, such as industrial or medical applications, ensuring the reliability of machine learning model predictions is paramount. These domains often present dynamic and uncertain environments, necessitating adaptive machine learning solutions with minimal operational overhead. A prevalent issue impacting prediction reliability is *concept drift*, where a data distribution changes over time [14]. It can be denoted as $P_{\text{train}}(X, Y) \neq P_{\text{online, t}}(X, Y)$, representing disparities in data distributions during initial training and online operation. This is common in various domains, where alterations in conditions lead to non-stationary data streams. If overlooked, concept drift can degrade model performance across applications. Therefore, adaptive strategies like periodic model updates or retrainings, especially upon drift detection, can be applied to maintain model reliability in evolving operational landscapes. However, most conventional drift detection algorithms, e.g. [7, 1], rely on error rates that demand access to scarce and costly true labels. An alternative is given by a class of drift detectors that work in an unsupervised way, utilizing a model's prediction confidence / uncertainty as a proxy for the error rate, such as Confidence Distribution Batch Detection (CDBD) [11] and Margin Density Drift Detection (MD3) [17]. More recently, Uncertainty Drift Detection (UDD) was proposed by Baier et al. [2], which utilizes neural network uncertainty estimates

Workshop on Distribution Shifts, 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

from Monte Carlo Dropout (MCD) sampling [6] as input for the Adaptive Windowing (ADWIN) detection algorithm [3]. As MCD is only one possibility of extracting uncertainty estimates from neural networks, we investigate the influence of the choice of uncertainty estimation method on the performance of the overall drift detection capability. In prior work, Ovadia et al. [14] analyzed uncertainty estimation methods under dataset shift but only for synthetic drifts. While Baier et al. [2] consider real-world datasets, they limit their experiments to a single uncertainty estimation method. Thus, our core contribution is an empiric comparison of four state-of-the-art neural network uncertainty estimation methods, as well as a baseline method, for classification tasks in combination with the ADWIN detector to identify drifts in real-world data streams. These uncertainty estimators are evaluated using seven commonly used real-world datsets.

2 Methodology and Experiments

To compare the uncertainty estimation methods introduced in the following, we conduct two experiments for each method and dataset. Both start by training the method with the initial five percent of the whole data stream. The first experiment serves as a baseline and thus, the remaining data is tested without analyzing uncertainty estimates or triggering retrainings. In the main experiment however, batches of the stream are evaluated and uncertainty estimates are used as a proxy for the error rate of the ADWIN detector. Once a drift is detected, a retraining is triggered with the initial five percent plus the most recent samples equivalent to one percent of the stream size. Thereby, models may adapt to new concepts while retaining sufficient generalization. Every experiment is repeated five times with different random seeds and results are averaged to allow for a fair comparison. Figure 1 illustrates the process of the main experiment.

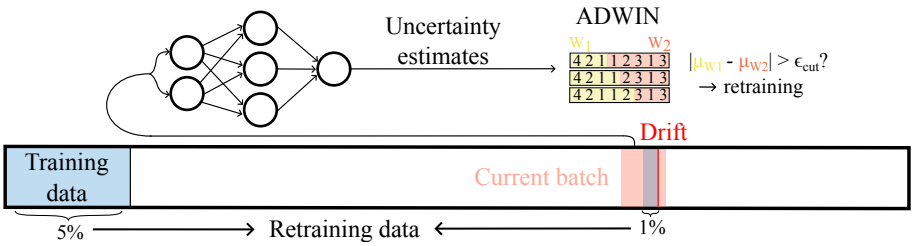

Figure 1: Approach to uncertainty drift detection.

2.1 Uncertainty Estimation Methods

To quantify model uncertainty, Bayesian neural networks, which learn a posterior distribution over model parameters, can be employed [14]. This distribution enables the application of Bayesian model averaging (BMA) during inference. Therefore, multiple weights w_i are drawn to gather a distribution of predictions $p_i(y|w_i, x)$, given input features x and target labels y. The final prediction $\hat{p}(y|x)$ is then given as the average

$$\hat{p}(y|x) = \frac{1}{P} \sum_{i=1}^{P} p_i(y|w_i, x). \tag{1}$$

For regression tasks, the uncertainty is the standard deviation of said distribution. While there are several uncertainty-related metrics for classification tasks, only Shannon's entropy H does not require ground-truth labels. Given the final prediction $\hat{p}(y|x)$ with K classes, it is computed as

$$H[\hat{p}(y|x)] = -\sum_{k=1}^{K} \hat{p}(y = k|x) \cdot \log_2 \hat{p}(y = k|x). \tag{2}$$

Although bayesian methods were previously considered state-of-the-art, they are computationally intractable for modern neural networks with millions of parameters [12]. Therefore, alternatives

have been developed, of which we analyzed the following in our experiments. To get an uncertainty estimate, Shannon's entropy H is applied to the final prediction of each method.

Basic Neural Network. Given the focus on classification tasks, a distribution of predictions is not necessarily required. Hence, the simplest method is to use a single prediction from an unmodified neural network. The motivation for this is to have a baseline for the more sophisticated methods.

Monte Carlo Dropout (MCD). Rather than drawing multiple weights from a posterior distribution as in BMA, a random dropout filter is applied to the neurons for several forward passes. These estimates are then averaged to get a final prediction. This allows for estimating the uncertainty in the model parameters based on the variability of the predictions across different dropout masks [2, 6].

Ensemble. A distribution of predictions can also be won by training multiple neural networks. Different seeds of members introduce randomness due to their influence on the initial weights as well as the shuffling of data during training. As Lakshminarayanan et al. [10] have shown, few members, i.e. 5, can be sufficient for good uncertainty estimates.

Stochastic Weight Averaging Gaussian (SWAG). Based on Stochastic Weight Averaging (SWA), a generalization technique in deep networks, Maddox et al. [12] propose a method to approximate a posterior distribution over neural network weights. Therefore, a Gaussian is fit utilizing the SWA solution as the first moment and a low rank plus diagonal covariance also inferred from stochastic gradient descent iterates. Given this posterior distribution, BMA is applied to get a final prediction.

Activation Shaping (ASH). The ASH method can be considered a more advanced version of the basic neural network, as it also works on single predictions. Djurisic et al. [5] introduced it as an out-of-distribution (OOD) detection method that reaches state-of-the-art performance. Assuming over-parameterized feature representations in modern neural networks, the hypothesis is that pruning a larger percentage of activations in a late layer helps with tasks such as OOD detection.

The hyperparameters of the introduced methods as well as the model architectures can be found in Appendix A.1. Furtheremore, we include details of the tuning process in Appendix A.3.

2.2 Drift Detector

Concept drift detectors, such as Drift Detection Method [7], Page Hinkley Test [15], and ADWIN [3], are typically error rate-based, necessitating access to costly true labels [8]. In contrast, data distribution-based detectors exclusively analyze input features, often using distance metrics like the Kolmogorov-Smirnov test [16] to identify changes in feature distribution. Regardless of the detection method employed, distinguishing between noise and genuine concept drift poses a significant challenge [19], requiring a balance between swift adaptation to changes and resilience to noise. ADWIN offers performance guarantees for false positives and false negatives, making it an attractive choice. Furthermore, it is able to work with any real-valued input instead of beeing limited to an error rate between 0-1. As introduced by Bifet et al. [3], ADWIN utilizes sliding windows of variable size. While no drift is present, new samples are added to a window W. After each sample, the algorithm attempts to find two sub-windows W_0 and W_1 that contain distinct averages. Once this happens a drift is assumed and the older sub-window is discarded. The variability of heterogeneous real-world data streams can be addressed by the sensitivity parameter $\delta \in (0, 1)$. The configuration for our experiments can be found in Appendix A.1.

2.3 Datasets

For our studies, we use seven real-world classification datasets from the USP Data Stream Repository [18]. They encompass abrupt, incremental and reocurring drifts, along with combinations thereof. In the **Gas** sensor dataset chemical sensor data is analyzed to identify one of six gases. The **Electricity** dataset focuses on predicting market price changes driven by supply and demand. For the **Rialto** dataset, segments of images from a timelapse with changing weather conditions shall be classified. Lastly, optical sensors are used to analyze moving patterns of flying insect species while drift is artificially introduced to generate the **InsAbr**, **InsInc**, **InsIncAbr** and **InsIncReo** datasets.

2.4 Metrics and Results

For evaluation, we focus on the following two metrics to capture the quality of the uncertainty estimates as well as the drift detection performance: **Expected Calibration Error (ECE)** $\downarrow$ [13] measures the average deviation between prediction confidence and accuracy. As the name suggests, it quantifies how well a model is calibrated. We expect that calibration correlates positively with drift detection capability. **Matthew's Correlation Coefficient (MCC)** $\uparrow$ is able to handle class imbalances which generally makes it a good metric for classification tasks [4]. We employ the MCC to measure the overall prediction performance of the models, averaged over the complete experiment runs. We expect that poor drift detection performance will lead to unsuitable retraining points, in turn producing low MCC scores and vice versa.

The results of our experiments can be found in Table 1. Analyzing the MCC values shows that the SWAG method offers the most balanced performance across all datasets. However, the gap in performance to the other methods is minimal. In fact, all methods perform fairly similarly. Surprisingly, even the basic method without any modifications keeps up with the others. Greater differences can be identified when analyzing the ECE as depicted in Figure 2. Here, the SWAG method offers significantly better calibrated predictions in nearly all datasets. The only exception is the InsIncAbr dataset, where all methods achieve a proficient calibration. All other methods appear to be similarly worse calibrated compared to SWAG for the remaining datasets. Despite that, this does not directly translate to a better drift detection performance, as shown by the MCC values.
Meanwhile, the total execution time fluctuates notably depending on the method selected, as presented in the last row of Table 1. As the basic and ASH method are based on a single sample, they serve as a lower bound in this regard. While MCD and the SWAG method both increase the inference runtime due to the sampling process, adaptations in the training process of the SWAG method incurr additional overhead. Although the execution time of the ensemble could be reduced by parallelizing the training and inference process of individual ensemble members, this would require additional computational resources. Hence, we choose not to, resulting in the highest execution time by far.
Appendix A.2 includes further details of the main experiment as well as an additional experiment to validate the retraining positions found by the uncertainty-based detector. Furthermore, it contains the standard deviations of our experiments.

Table 1: Performance comparison of uncertainty estimation methods for drift detection. Table cells contain the average MCC ($\uparrow$) values and (number of retrainings) for the naive baselines without retrainings (upper) and retrainings triggered by ADWIN when using the respective uncertainty estimation method (lower), respectively. Bold numbers indicate the best performance. To given an impression of the computational cost, the last row contains the total execution times for each method.

	Basic	MCD	Ensemble	SWAG	ASH
Gas	0.273 (0) 0.455 (36)	0.256 (0) 0.46 (55)	0.245 (0) **0.492** (50)	**0.299** (0) 0.46 (52)	0.275 (0) 0.459 (35)
Electricity	0.178 (0) 0.424 (11)	**0.198** (0) 0.421 (10)	0.183 (0) 0.405 (10)	0.191 (0) 0.419 (7)	0.175 (0) **0.438** (10)
Rialto	0.532 (0) 0.537 (43)	**0.534** (0) **0.553** (48)	0.505 (0) 0.527 (45)	0.52 (0) 0.54 (52)	0.525 (0) 0.539 (43)
InsAbr	0.471 (0) **0.519** (9)	0.472 (0) 0.509 (8)	0.461 (0) 0.503 (8)	**0.48** (0) 0.514 (6)	0.474 (0) 0.508 (7)
InsInc	0.087 (0) 0.241 (3)	**0.1** (0) 0.238 (3)	0.081 (0) 0.241 (3)	**0.1** (0) **0.301** (4)	0.085 (0) 0.231 (3)
InsIncAbr	0.304 (0) 0.53 (24)	0.307 (0) 0.525 (26)	0.308 (0) 0.518 (23)	0.299 (0) 0.445 (25)	**0.316** (0) **0.531** (25)
InsIncReo	0.141 (0) 0.253 (18)	0.133 (0) 0.247 (20)	**0.172** (0) 0.236 (18)	0.16 (0) **0.302** (21)	0.133 (0) 0.243 (20)
Total exec. time	6821s	7339s	15653s	9036s	6890s

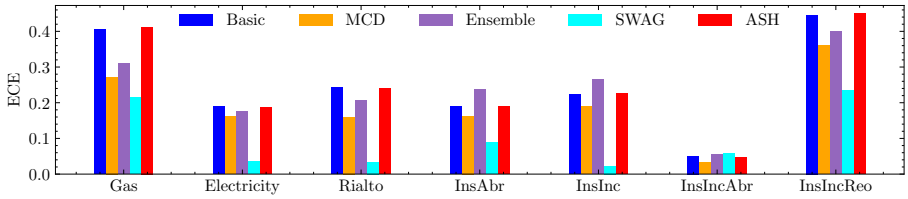

Figure 2: Calibration of the employed uncertainty estimation methods measured by ECE (↓) across the seven datasets.

3 Conclusion

In this work, we implemented five uncertainty estimation methods for classification tasks and evaluated them in experiments including seven real-world datasets. Our goal was to compare the utility of their uncertainty estimates for unsupervised concept drift detection by using them as a proxy for the error-rate in combination with the ADWIN detector. Thereby, drift points in data streams shall be identified to trigger retrainings at the appropriate time and ultimately prevent model decay. Interestingly, even our baseline method, relying solely on the entropy calculated from the softmax scores, performed competitively with more sophisticated state-of-the-art methods. Moreover, all methods performed fairly similar in terms of overall classification performance as measured by the MCC metric. While the SWAG method achieved the most balanced MCC values, differences were only marginal. However, this was not the case when analyzing the ECE. Here the SWAG method offers significantly better calibrated predictions than all other methods. Regardless, these did not translate to better results for the drift detection. Thus, the assumption can be made, that the choice of method does not have a noteworthy influence on the performance of uncertainty-based concept drift detection for real-world applications.
To confirm the previous assumption, future work may include testing further real-world datasets, including regression problems. For those, the basic neural network and the ASH method are no longer applicable. Instead, the effect of the ASH method in combination with the remaining approaches could be studied.

References

[1] Manuel Baena-Garcıa, José del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86. Citeseer, 2006.

[2] Lucas Baier, Tim Schlör, Jakob Schöffer, and Niklas Kühl. Detecting concept drift with neural network model uncertainty. *arXiv preprint arXiv:2107.01873*, 2021.

[3] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[4] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21:1–13, 2020.

[5] Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. *arXiv preprint arXiv:2209.09858*, 2022.

[6] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[7] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, pages 286–295. Springer, 2004.

[8] Paulo M Gonçalves Jr, Silas GT de Carvalho Santos, Roberto SM Barros, and Davi CL Vieira. A comparative study on concept drift detectors. *Expert Systems with Applications*, 41(18):8144–8156, 2014.

[9] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017.

[10] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[11] Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. Drift detection using uncertainty distribution divergence. *Evolving Systems*, 4:13–25, 2013.

[12] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[13] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2901–2907. AAAI Press, 2015.

[14] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[15] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[16] Christoph Raab, Moritz Heusinger, and Frank-Michael Schleif. Reactive soft prototype computing for concept drift streams. *Neurocomputing*, 416:340–351, 2020.

[17] Tegjyot Singh Sethi and Mehmed Kantardzic. Don't pay for validation: Detecting drifts from unlabeled data using margin density. *Procedia Computer Science*, 53:103–112, 2015.

[18] Vinicius MA Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34:1805–1858, 2020.

[19] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

A Appendix

A.1 Reproducibility

To make our experiments reproducible, Table 2 gives an overview of the neural network architecture used for each dataset. Hidden layers use Rectified Linear Unit activations, while softmax is applied in the final layer. The ADAM optimizer is used with binary or categorical cross-entropy loss, depending on the number of classes. For **MCD**, 100 forward passes are carried out. The **Ensemble** consists of three members. Bayesian model averaging is conducted with 100 samples from the posterior approximation of the **SWAG** method. Details on these choices are discussed in A.3. Furthermore, the estimated covariance matrix utilized in the approach has a rank of 25 and is updated each epoch, starting at the first iteration. For the **ASH** method, the version termed ASH-p was chosen, where unpruned activations are not modified at all. Pruning is applied in the penultimate hidden layer (i.e. third last overall layer) with a pruning percentage of 60%. Lastly, Table 3 indicates the sensitivity values δ for the ADWIN detector.

Table 2: Overview of model architectures.

Name	No. Layers	Neurons per layer	Dropout rate	Epochs
Gas	5	128, 64, 32, 16, 8	0.2	100
Electricity	3	32, 16, 8	0.1	400
Rialto	4	512, 512, 256, 32	0.2	200
InsAbr	5	128, 64, 32, 16, 8	0.1	200
InsInc	5	128, 64, 32, 16, 8	0.1	100
InsIncAbr	3	32, 16, 8	0.1	50
InsIncReo	3	128, 64, 32	0.1	400

Table 3: Sensitivity values for ADWIN detector.

Gas	Electricity	Rialto	InsAbr	InsInc	InsIncAbr	InsIncReo
0.1	1e-15	1e-20	0.002	0.002	0.1	0.1

A.2 Additional Results and Experiments

We generated reliability diagrams [9] in addition to Table 1 and Figure 2 of the main experiment. These diagrams illustrate the quantification of the ECE. Hence, buckets of confidence values are compared to their average accuracy. Furthermore, the gaps to a perfect calibration are visualized. Plots can be found in Figures 3 - 5. Consistent to Figure 2, they show that the SWAG method offers the best calibration.

To validate the retraining positions found by the uncertainty based drift detection, we also conducted an experiment with equally and randomly distributed retraining positions. We compared these against the SWAG-based drift detection. Hence, the same amount of retrainings was triggered as found by the SWAG approach for each dataset (see Table 4). While the retraining positions found by SWAGs uncertainty values yield significantly better predictions for *Gas*, *Electricity*, and *InsAbr*, the opposite is the case for *InsIncAbr* and *InsIncReo*. Here, the equally distributed approach for retrainings offers noticeably better results. For *Rialto* and *InsInc* there are only slight differences between all three methods. Nevertheless, the detection based approach still offers the best overall performance.

As we repeated all of these experiments five times with different random seeds, we also include the standard deviations in Tables 5 - 7.

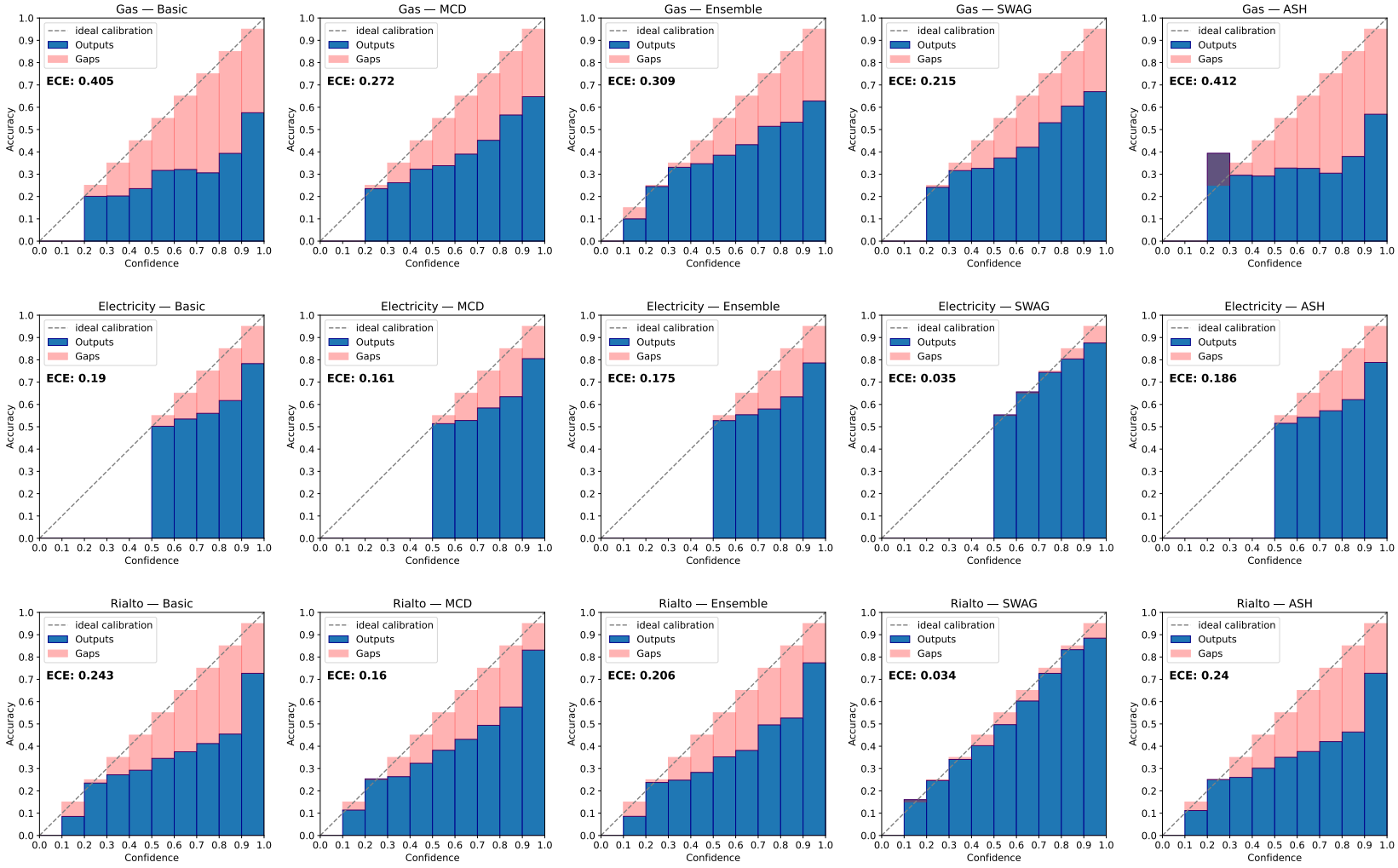

Figure 3: Reliability diagrams of main experiment (1/3)

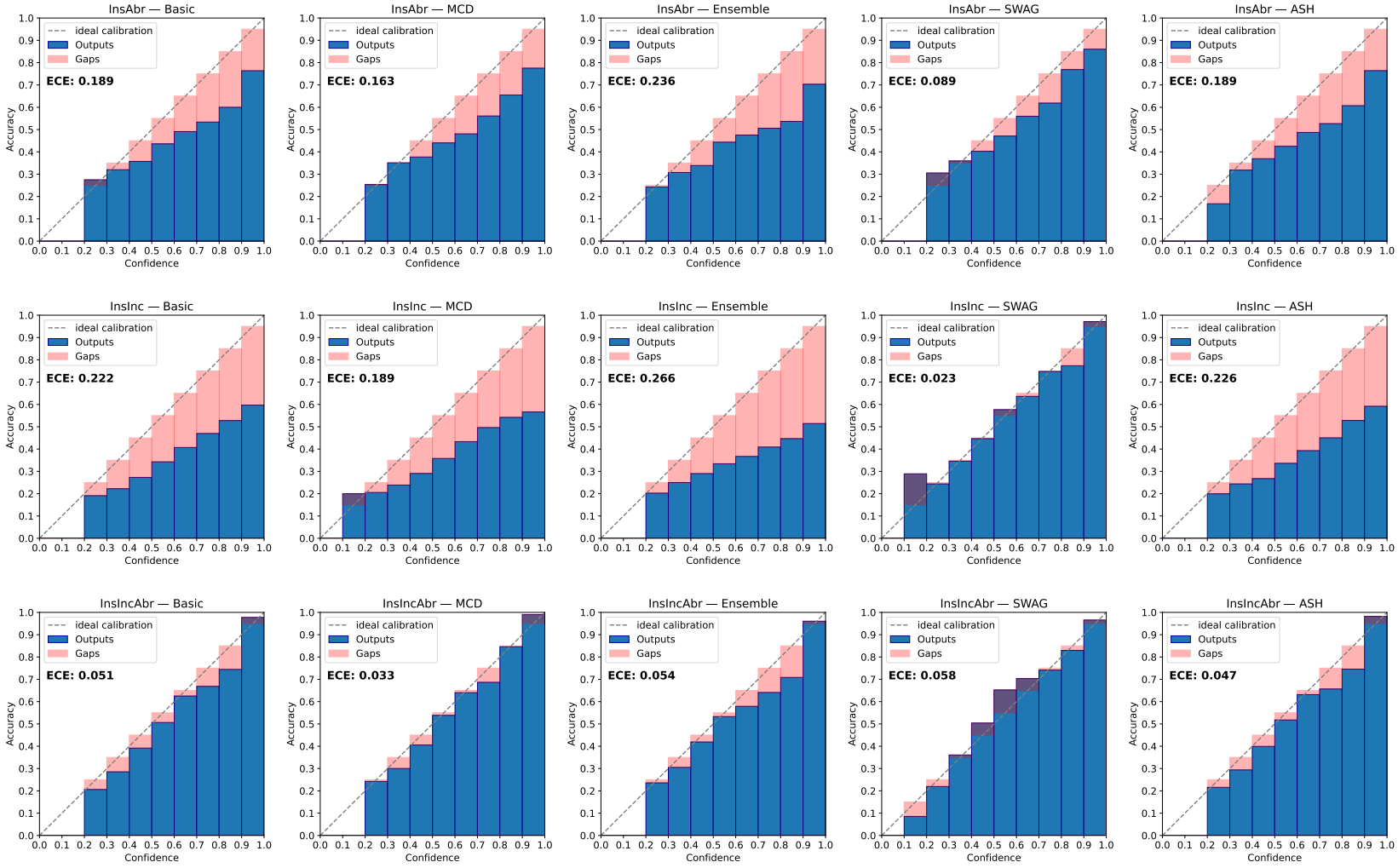

Figure 4: Reliability diagrams of main experiment (2/3)

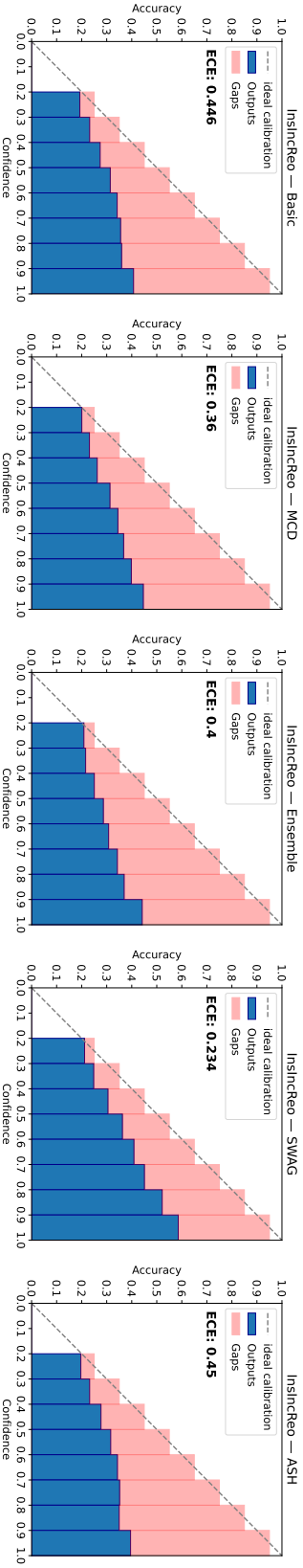

Figure 5: Reliability diagrams of main experiment (3/3)

Table 4: Retraining position validation.

	SWAG	Equal dist.	Random dist.
Gas	**0.46** (52)	0.387 (52)	0.38 (52)
Electricity	**0.419** (7)	0.346 (7)	0.351 (7)
Rialto	0.54 (52)	0.555 (52)	**0.557** (52)
InsAbr	**0.514** (6)	0.459 (6)	0.484 (6)
InsInc	**0.301** (4)	**0.301** (4)	0.293 (4)
InsIncAbr	0.445 (25)	**0.483** (25)	443 (25)
InsIncReo	0.302 (21)	**0.344** (21)	0.32 (21)

Table 5: Standard deviations of baseline experiment without retrainings.

	Basic	MCD	Ensemble	SWAG	ASH
Gas	0.0249	0.0437	0.0119	0.0274	0.0404
Electricity	0.016	0.0093	0.0125	0.0235	0.014
Rialto	0.0102	0.0118	0.0054	0.0035	0.0014
InsAbr	0.0066	0.0017	0.0017	0.0014	0.0022
InsInc	0.0099	0.0101	0.0101	0.0096	0.0075
InsIncAbr	0.0067	0.0035	0.0042	0.0086	0.005
InsIncReo	0.0073	0.0109	0.004	0.0033	0.0059

Table 6: Standard deviations of main experiment with ADWIN detection.

	Basic	MCD	Ensemble	SWAG	ASH
Gas	0.0331	0.0438	0.0366	0.0203	0.0347
Electricity	0.0199	0.0368	0.0356	0.0187	0.0136
Rialto	0.0034	0.0028	0.0030	0.0036	0.0026
InsAbr	0.0060	0.0082	0.0107	0.0251	0.0101
InsInc	0.0156	0.0125	0.0138	0.0162	0.0158
InsIncAbr	0.0021	0.0050	0.0096	0.0203	0.0078
InsIncReo	0.0088	0.0072	0.0111	0.0113	0.0095

Table 7: Standard deviations of retraining position validation experiment.

	SWAG	Equal dist.	Random dist.
Gas	0.0203	0.0094	0.0305
Electricity	0.0187	0.0208	0.0398
Rialto	0.0036	0.0013	0.0047
InsAbr	0.0251	0.0039	0.0247
InsInc	0.0172	0.0021	0.0316
InsIncAbr	0.0203	0.0081	0.0469
InsIncReo	0.0113	0.0036	0.0156

A.3 Hyperparameter Tuning

To tune the hyperparameters, the main experiment was run for several configurations with the same seed. Tables in the following show the MCC based on all predictions and the number of retrainings in parentheses.

For **MCD** the only additional hyperparameter is the number of stochastic forward passes T. As Table 8 reveals, we tested $T = 25,\ 50,\ 75$ and 100. Although $T = 25$ had the best performance in the majority of our experiments, this is not really representative for the overall performance. In fact, discrepancies are rather slight in datasets where $T = 25$ yields the best performance, while it is significantly outperformed in other datasets. We found that $T = 100$ offers the most balanced performance across all datasets. The additional computational cost is also negligible as $T = 100$ triggers the least amout of retrainings and thus incurrs the lowest execution time for all experiments combined.

Table 8: MCC values of MCD with 25, 50, 75, and 100 forwards passes.

	$T = 25$	$T = 50$	$T = 75$	$T = 100$
Gas	0.418 (49)	0.443 (48)	0.41 (44)	**0.451** (46)
Electricity	0.365 (8)	0.386 (10)	0.363 (11)	**0.415** (8)
Rialto	0.554 (59)	**0.56** (61)	0.554 (59)	0.553 (59)
InsAbr	0.509 (9)	0.491 (6)	**0.521** (10)	0.481 (5)
InsInc	**0.218** (2)	0.216 (1)	0.217 (1)	0.216 (1)
InsIncAbr	**0.54** (25)	0.538 (23)	0.538 (23)	0.538 (22)
InsIncReo	**0.249** (21)	0.24 (19)	0.24 (20)	0.235 (18)
Total exec. time	4712s	4712s	4875s	**4638**s

Similar to MCD there is only one hyperparameter for the **Ensemble**. Namely, the number of members M which was set to three, five, and seven during our tests, as shown in Table 9. Here we found very slight differences overall. Thus, we choose the version with the least computational cost, which is $M = 3$.

Table 9: MCC values of an ensemble of 3, 5, and 7 members.

	$M = 3$	$M = 5$	$M = 7$
Gas	0.479 (48)	**0.494** (51)	0.479 (52)
Electricity	0.422 (10)	0.407 (9)	**0.426** (10)
Rialto	**0.529** (46)	**0.529** (48)	0.526 (51)
InsAbr	0.474 (4)	**0.505** (8)	0.494 (8)
InsInc	**0.259** (3)	0.194 (1)	0.255 (2)
InsIncAbr	**0.53** (24)	0.514 (26)	0.508 (22)
InsIncReo	0.231 (21)	**0.255** (22)	0.25 (18)
Total exec. time	**12912**s	19511s	31092s

Other than the previous methods, **SWAG** comes with several hyperparameters. First, the influence of the number of weight samples S drawn from the approximated distribution was tested. Therefore, the rank K was set to 25, and weights were updated every epoch starting at the first iteration. As shown by Table 10, $S = 100$ offers the most balanced performance. The higher execution time compared to $S = 50$ and $S = 75$ is the result of more retrainings in datasets such as InsAbr and InsIncReo. Consequently, there is a noticeable performance gap in said datasets which mitigates the slower execution. Onwards, the effect of the rank K was studied with a fixed S. As seen in Table 11, the initial rank of $K = 25$ slightly outperformed the other settings. Starting at later epochs and

reducing the update frequency for the SWAG method showed no mentionable improvements neither in performance, nor in execution time. Thus the final configuration was $S = 100$ and $K = 25$ with updates in every epoch beginning at the start of training.

Table 10: MCC values of SWAG with 25, 50, 75, and 100 weight samples.

	$S = 25$	$S = 50$	$S = 75$	$S = 100$
Gas	0.436 (49)	0.433 (57)	**0.456** (55)	0.455 (53)
Electricity	0.414 (10)	**0.435** (11)	0.343 (13)	0.396 (10)
Rialto	0.544 (53)	**0.546** (54)	0.543 (49)	0.541 (53)
InsAbr	**0.543** (9)	0.503 (6)	0.517 (7)	0.542 (8)
InsInc	0.283 (3)	0.29 (4)	**0.304** (4)	0.296 (3)
InsIncAbr	0.51 (25)	0.487 (23)	**0.528** (23)	0.504 (22)
InsIncReo	0.332 (31)	0.282 (16)	0.311 (20)	**0.335** (28)
Total exec. time	5694s	5018s	**4913**s	5514s

Table 11: MCC values of SWAG with $S = 100$ and $K = 10,\ 25$, and 40.

	$K = 10$	$K = 25$	$K = 40$
Gas	0.443 (54)	**0.455** (53)	0.435 (55)
Electricity	**0.412** (10)	0.396 (10)	0.392 (13)
Rialto	0.543 (54)	0.541 (53)	**0.548** (50)
InsAbr	0.521 (7)	**0.542** (8)	0.54 (8)
InsInc	0.255 (3)	0.296 (3)	**0.318** (4)
InsIncAbr	0.487 (26)	0.504 (22)	**0.514** (21)
InsIncReo	0.316 (28)	**0.335** (28)	0.289 (22)
Total exec. time	5603s	5514s	**5472**s

All three **ASH** versions introduced by Djurisic et al. [5] were tested with pruning percentages between 60% and 90% in the penultimate layer. As Table 12 reveals, the best results performance was achieved by the ASH-p version with a rather low pruning percentage of 60%. This is surprising, as ASH-p was the worst method in tests from Djurisic et al. where it served as a baseline. Furthermore, experiments have shown that higher pruning percentages were hurting performance. Lastly, the placement of the pruning layer was tested for the previous best configuration. While differences were slight, the best performance was reached when pruning in the penultimate hidden layer (i.e. third last overall layer) as seen in Table 13.

Table 12: MCC values of ASH-p (top), ASH-b (middle) and, ASH-s (bottom) for pruning percentages 60%, 70%, 80% and 90%.

	60%	70%	80%	90%
Gas	**0.443** (42) 0.407 (38) 0.397 (28)	**0.443** (42) 0.407 (38) 0.397 (28)	0.398 (47) 0.357 (39) 0.396 (26)	0.293 (60) 0.388 (48) 0.347 (25)
Electricity	**0.475** (12) 0.347 (8) 0.318 (4)	0.395 (10) 0.464 (10) 0.339 (3)	0.408 (6) 0.444 (10) 0.338 (3)	0.422 (13) 0.412 (12) 0.398 (4)
Rialto	0.539 (42) 0.551 (38) 0.569 (35)	0.537 (41) 0.56 (36) 0.561 (35)	0.526 (45) 0.564 (39) **0.575** (35)	0.447 (43) 0.442 (41) 0.45 (39)
InsAbr	**0.496** (7) 0.471 (4) 0.491 (6)	0.474 (10) 0.386 (9) 0.405 (7)	0.435 (7) 0.426 (5) 0.435 (5)	0.322 (8) 0.301 (5) 0.342 (9)
InsInc	0.217 (1) 0.237 (2) 0.23 (2)	**0.252** (5) 0.155 (3) 0.196 (3)	0.179 (2) 0.162 (3) 0.196 (5)	0.172 (3) 0.153 (2) 0.223 (3)
InsIncAbr	0.502 (24) 0.473 (25) **0.526** (17)	0.502 (24) 0.473 (25) **0.526** (17)	0.477 (24) 0.435 (19) 0.453 (21)	0.336 (22) 0.424 (20) 0.419 (24)
InsIncReo	0.232 (17) 0.202 (12) 0.214 (9)	**0.254** (17) 0.196 (13) 0.171 (6)	0.208 (17) 0.142 (8) 0.223 (13)	0.171 (21) 0.168 (21) 0.116 (17)
Total exec. time	3570s 3034s 2804s	3551s 3122s **2715**s	3718s 3026s 3240s	3657s 3483s 3297s

Table 13: MCC values of ASH-p with a pruning percentage of 60% at outputlayer - 1, -2, and -3.

	$L-1$	$L-2$	$L-3$
Gas	**0.443** (42)	0.441 (36)	0.399 (39)
Electricity	**0.475** (12)	0.423 (13)	0.453 (13)
Rialto	0.539 (42)	**0.545** (43)	**0.545** (43)
InsAbr	**0.496** (7)	0.494 (6)	**0.496** (8)
InsInc	0.217 (1)	**0.237** (3)	0.234 (3)
InsIncAbr	0.502 (24)	**0.526** (21)	0.524 (24)
InsIncReo	0.232 (17)	**0.259** (21)	0.245 (20)
Total exec. time	**3570**s	3611s	3646s