On Reporting Robust and Trustworthy Conclusions from Model Comparison Studies Involving Neural Networks and Randomness

Odd Erik Gundersen
Norwegian University of Science and Technology
Norway
Aneo AI Research
Norway
odderik@ntnu.no

Saeid Shamsaliei
Norwegian University of Science and Technology
Norway
Aneo AI Research
Norway
saeid.shamsaliei@ntnu.no

Håkon S. Kjærnli
Aneo AI Research
Norway
hakon.slatten.kjernli@aneo.com

Helge Langseth
Norwegian University of Science and Technology
Norway
Aneo AI Research
Norway
helge.langseth@ntnu.no

ABSTRACT

The performance of neural networks differ when the only difference is the seed initializing the pseudo-random number generator that generates random numbers for their training. In this paper we are concerned with how random initialization affect the conclusions that we draw from experiments with neural networks. We run a high number of repeated experiments using state of the art models for time-series prediction and image classification to investigate this statistical phenomenon. Our investigations show that erroneous conclusions can easily be drawn from such experiments. Based on these observations we propose several measures that will improve the robustness and trustworthiness of conclusions inferred from model comparison studies with small absolute effect sizes.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Neural networks, deep learning, reproducibility, random initialization, robust conclusions.

ACM Reference Format:
Odd Erik Gundersen, Saeid Shamsaliei, Håkon S. Kjærnli, and Helge Langseth. 2023. On Reporting Robust and Trustworthy Conclusions from Model Comparison Studies Involving Neural Networks and Randomness. In *2023 ACM Conference on Reproducibility and Replicability (ACM REP '23), June 27–29, 2023, Santa Cruz, CA, USA*. ACM, New York, NY, USA, 25 pages. https://doi.org/10.1145/3589806.3600044

ACM REP '23, June 27–29, 2023, Santa Cruz, CA, USA

ACM ISBN 979-8-4007-0176-4/23/06.
https://doi.org/10.1145/3589806.3600044

1 INTRODUCTION

Conclusions drawn from empirical studies in artificial intelligence can be affected by many different factors, see [13] for an overview. Empirical studies involving deep neural networks are especially exposed to factors related to randomness. *Algorithmic factors* are model design choices that introduce stochasticity by design while *implementation factors* are introduced by the software used to implement an algorithm and the hardware that the experiments run on [33, 46]. Different laboratories typically use different hardware and software stacks, which means that the implementation factors are varied in inter-laboratory studies where independent research re-execute code from the experiments originally conducted by someone else.

Scientific knowledge is gained through repeated observations or repeated experiments. Repetition is required to avoid drawing conclusions based on spurious results, as results of empirical studies vary for many different reasons. In many domains, the conditions of a study changes every time one repeats an experiment no matter how careful the procedures are carried out. Hence, proper methodology in domains seeking to obtain an accurate and precise estimation of a variable of interest does not consider conclusions from empirical studies to be robust unless variation is accounted for. For example, analytical chemistry distinguishes between inter- and intra-laboratory variation [29] while medicine discriminate between inter- and intra-observer variation [15]. Studies must account for variation not only between laboratories and observers but also within a laboratory and for the same observer in order for the conclusions drawn from these studies to be considered robust.

One source of variation in neural networks is the initialization of the weights controlling the signal strength between neurons before learning starts [11]. All other things equal, the errors on hold-out data will vary for neural networks trained from random initializations of their weights, see Figure 1. The cause of this difference in performance is stochastic gradient descent or one of its variants that is used to train the neural networks [4]. The loss functions that are optimized are typically non-convex over the networks'

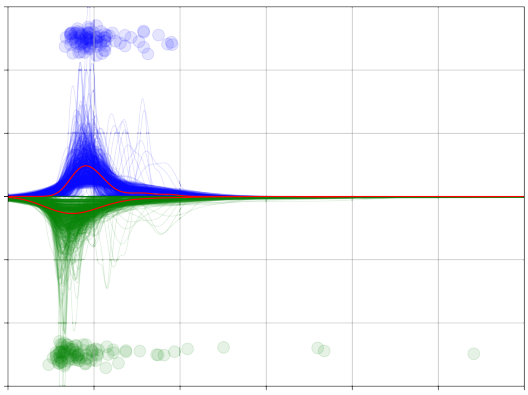


Figure 1: The performance of the two models N-BEAT (blue) and DeepAR (green) on the data-set exchange_rate measured using MAPE (x-axis). The merits of a model depends on the random initialization seed, and each blue (green) curve is a kernel density estimate for MAPE using N-BEAT (DeepAR) based on 5 random seeds; a total of 1000 curves are shown per model. Green lines are plotted on the reverse y-axis to improve visibility. The two red curves give KDEs based on 100 random seeds. The actual MAPE values for these 100 seeds for both models are given as semi-transparent circles for reference. Notice that the variability between the different KDE plots within one model (different curves of same color) is comparably larger than the variability between the models (the two red curves). This indicates that if one wants to draw conclusions regarding which model is better on this dataset, the variability due to random initializations will heavily influence the results, and the validity of the analysis may be jeopardized if care is not taken. Further details about this experiment are given in Section 3 and we discuss our findings in Section 4.

weights, and they are complex with multiple saddle points and (local) minima. Therefore, stochastic gradient descent may converge at different stationary points depending on the hyperparameters of the optimization function used and the exact initialization of the weights. Other sources of variation relying on the seed used to initialize the pseudo-random number generator are the initialization of the hyperparameter optimization algorithm, stochastic layers in the neural network, such as Dropout, data shuffling and batch ordering of data [13].

Random initialization has been exploited by, for example, Wenzel et al. [45] to increase the robustness and allow for uncertainty quantification in deep ensembles [25]. It has also been exploited in pruning, which is a technique for removing unnecessary weights from neural networks to identify subnetworks that reach test accuracy comparable to the original network in a similar number of iterations when trained in isolation [10].

Lately, however, several empirical studies on neural networks have shown how methodological weaknesses can lead to false conclusions being drawn because of variation not being properly accounted for. Reimers and Gurevych [38] analyzed LSTM-networks for sequence tagging and showed that initialization seeds could result in statistically significant differences ($p < 10^{-4}$) in state-of-the-art-systems. Melis et al. [27] investigated the evaluation of neural language models and found that standard LSTM models outperform more recent models under the same conditions when they control for hyperparameters and unlimited computing resources. Further investigations show that results vary significantly when they control for non-deterministic ordering of floating-point operations in linear algebra routines, different initialization seeds and the validation and tests being finite samples of an infinite population. Similarly, Lucic et al. [26] found that none of the algorithms proposed to improve the original generative adversarial network performed better when tested under the same conditions, taking hyper-parameter search and network initialization into account.

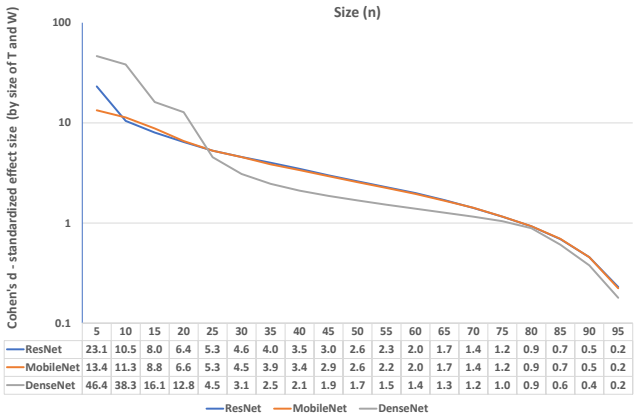
Henderson et al. [18] investigated challenges related to reproducing results in deep reinforcement learning. Both random initialization of the deep neural networks and randomness in the environment affected the results. Bouthillier et al. [6] showed that the variation in performance caused by the initialization of deep neural networks for image classification could lead to false conclusions unless properly controlled for in the experimental design. Zhuang et al. [46] analyzed both algoritmic and implementation factors and found that model performance on certain parts of the data distribution is far more sensitive to the introduction of randomness than the top-line metrics such as top-1 accuracy. They also found that the cost of ensuring determinism varies dramatically between neural network architectures and hardware types with overhead up to 741%. Bouthillier et al. [5] modelled the whole benchmarking process for neural networks and found noticeable variance in performance caused by data sampling, parameter initialization, and hyperparameter design. Gundersen et al. [14] found that executing the exact same experiment while varying the implementation factors could lead to the wrong conclusions at the 95% confidence level when classifying handwritten letters from the MNIST dataset using a simple convolutional neural network.

Hence, artificial intelligence, at the very least deep learning, does not go clear of the so-called *reproducibility crisis* [2, 12, 21] reported in other sciences such as psychology [28, 30] and medicine [36]. Given the prevalence of deep learning [3], this is a severe issue for research in artificial intelligence that is recognized by the community [12, 16, 35].

In this paper, we investigate how initializing pseudo-random number generators with different seeds affect the conclusions that can be drawn. We conduct experiments where we vary the initialization seed before conducting experiments on different state-of-the-art deep neural networks, and we make several observations. Based on these observations, we suggest remedies to reduce the chance of drawing erroneous conclusions from empirical studies involving deep neural networks.

2 ADVERSARIAL MODEL COMPARISON

As prior work, we investigate *model comparison*; it is a common methodology in research on artificial intelligence and used in most scientific articles on deep learning. In model comparison studies, the relative performance of different models (i.e. network architectures, such as DeepAR or DenseNet) are compared when executed



	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
ResNet	23.1	10.5	8.0	6.4	5.3	4.6	4.0	3.5	3.0	2.6	2.3	2.0	1.7	1.4	1.2	0.9	0.7	0.5	0.2
MobileNet	13.4	11.3	8.8	6.6	5.3	4.5	3.9	3.4	2.9	2.6	2.2	2.0	1.7	1.4	1.2	0.9	0.7	0.5	0.2
DenseNet	46.4	38.3	16.1	12.8	4.5	3.1	2.5	2.1	1.9	1.7	1.5	1.4	1.3	1.2	1.0	0.9	0.6	0.4	0.2

Figure 2: The standardized effect size (Cohen's d) tells how many standard deviations lie between the two means for the best performing seed replicates, $\mathcal{T}^n$, and the worst performing seed replicates, $\mathcal{W}^n$, for the image classification task. Note the log scale.

on the same tasks under the same conditions. When their absolute effect size measured using a performance metric is large, distinguishing between these models is not a problem. However, when the performance relies on non-determinism and the absolute effect size is small (models have similar performance), deciding which model performs better requires care.

As randomness is such an important tool when training neural networks, controlling for the variation it introduces becomes important to understand the results of experiments. When drawing a random value, a pseudorandom number generator (PRNG) is invoked. The numbers provided by the PRNG are not actually random. If a specific initialization seed is set, the PRNG will draw the same sequence of numbers every time, but if an initialization seed is not set, a default seed, such as current time, will be used and seemingly random numbers will be provided by the PRNG. Given the sensitivity of neural networks to randomness, keeping track of which initialization seeds are used in testing becomes important. Deciding to use stochasticity when training neural networks is a design choice, and therefore an algorithmic factor, while the selection of which exact seed to use is an implementation factor [13, 33, 46].

Prior work shows that a single initialization seed is not enough to draw robust conclusions and indicates that more seeds are needed. Pham et al. [33] suggests that the testing can stop when the probability of a model being better than another one is higher than 75%, meaning that one can end up with not true findings in 25% of tests. What would it take to give more confident and concrete advice? We seek to answer the following **research question:** *Given that two deep learning models being compared have the same performance, how many different initialization seeds must be tested to decide that we cannot distinguish between their performances?*

We formulate the research question in an adversarial manner. Let us consider a situation where we want to compare two models, A and B, that objectively have the same performance, but where an adversarial agent promotes model A when comparing it to model B. Whether the adversarial agent is *Mother Nature* that has setup such a situation by chance or it is designed by an actor with bad intentions is not relevant. The reason for formulating the problem in an adversarial manner is to facilitate an analysis of how we can increase the probability of reaching a true conclusion about the performance of deep neural networks in worst case scenarios.

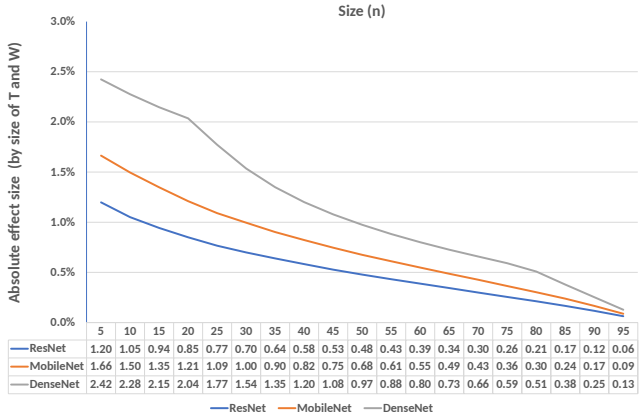
Initialization seed s is an element of a set of initialization seeds $\mathcal{S}$, that is $s \in \mathcal{S}$. All other things being equal, the performance, p, of a model, m, that is trained and tested on data, d, is a function of the initialization seed, s, that has been used to initialize the model: $p = f(m, d, s)$ for some deterministic, yet unknown, function f. Note that the dataset d is divided into training, validation and test sets, so that p is achieved on a hold-out set and not trained and tested on the same data. Given a set of initialization seeds $\mathcal{S}$ that have been used to initialize a model m that is subsequently trained and tested on dataset d, we achieve a set $\mathcal{P}$ of performances, that is $\mathcal{P} = \{f(m, d, s)\}_{s \in \mathcal{S}}$. According to the terminology of Bouthillier et al. [6] and Vaux et al. [43], a model m that is initialized using intialization seed s and trained and tested on a dataset d is a *seed replicate* while a model m initialized using the set of intialization seeds $\mathcal{S}$ and trained and tested on a dataset d is a *dataset replicate*.

For each dataset replicate, the seed replicates can be sorted on performance. The subset of n top performing seed replicates for model m is $\mathcal{T}_m^n$, while the subset of n worst performing seed replicates is $\mathcal{W}_m^n$. Now, the bad actor who wants to promote model A by comparing it against the baseline model B could compare $\mathcal{T}_A^n$ against $\mathcal{W}_B^n$. Reimers and Gurevych [38] found that for some models A and B, the performance in a sequence tagging task would differ significantly at confidence level $\alpha = 10^{-3}$ when comparing $\mathcal{T}_A^1$ with $\mathcal{W}_B^1$, that is, when comparing the best seed replicate against the worst seed replicate. In this way, the model that the bad actor wants to promote can be shown to perform better than the baseline model at a high confidence level. Given the high confidence level, the conclusion, although false, would appear to be strongly supported. It is important to note that they calculated the confidence level based on the model performance on each test sample in the test dataset. They did not, however, calculate the confidence level based on the variation in performance on the whole testset caused by varying the initialization seeds. The difference is important as it explains why performance has to be measured on more than one initialization seed. The question is how many initialization seeds are required to get a good representation of the actual performance.

3 EXPERIMENTS

As our objective is to analyze the variation of results given different initialization seeds, we train and test deep neural network models using 100 different seeds, $\mathcal{S} = \{0, 1, \ldots, 99\}$. Hyper-parameters and the training setups are identical for each seed replicate. All models are trained using the same software stack on the same hardware. Hence, the only variation that is introduced is the initialization seeds that are used to initialize the PRNGs. We conduct experiments for two different tasks: times-series forecasting and image classification. The code needed to reproduce our results is available in a public repository.[1]

[1] https://github.com/TronderEnergi/deepts



	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
ResNet	1.20	1.05	0.94	0.85	0.77	0.70	0.64	0.58	0.53	0.48	0.43	0.39	0.34	0.30	0.26	0.21	0.17	0.12	0.06
MobileNet	1.66	1.50	1.35	1.21	1.09	1.00	0.90	0.82	0.75	0.68	0.61	0.55	0.49	0.43	0.36	0.30	0.24	0.17	0.09
DenseNet	2.42	2.28	2.15	2.04	1.77	1.54	1.35	1.20	1.08	0.97	0.88	0.80	0.73	0.66	0.59	0.51	0.38	0.25	0.13

Figure 3: The absolute effect size is the magnitude of the difference between two groups. Here the two means are for the best performing seed replicates, $\mathcal{T}^n$, and the worst performing seed replicates, $\mathcal{W}^n$, for the image classification task.

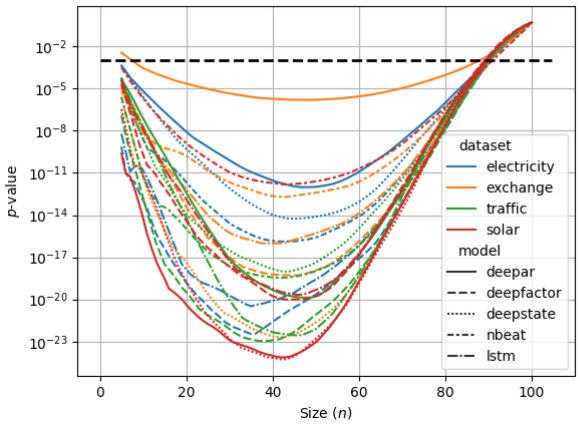


Figure 4: A statistical test comparing whether $\mathcal{T}^n$ and $\mathcal{W}^n$ are from the same distribution results in statistical significance for all models and all datasets even with n as large as $n \sim 80$. The reported p values are from a one-sided two-sample t-test.

3.1 Time-series Forecasting

We benchmark five deep learning time-series forecasting models on four datasets. The forecasting models are DeepAR [40], DeepFactor [44], DeepState [37], N-BEATS [31] and LSTM [19]. The first four models are implemented in the popular GluonTS[2] library [1], but as GluonTS did not provide a default implementation of a plain LSTM, we implemented one on top of it. As our goal was to investigate how the random seeds used for initialization affects the performance, and not necessarily to reproduce state of the art performance of the models, we did not spend computational resources on hyper-parameter search. Instead we relied on the default hyper-parameters defined by GluonTS whenever possible. All training was done using GPUs except for DeepState, which we were unable to run on our GPU setup. As a consequence, the computational burden of DeepState was decreased by the following changes to the hyperparameters: the length of the conditioning range was set to the prediction length and the cardinality of the model was set equal to the number of unique sequences in the data-set. The supporting material details the hyperparameters of each model.

The four datasets we examine are *electricity* [9], *exchange_rate* [24], *traffic_nips*, [8] and *solar_nips*[3], see the supporting material for a more detailed description of the datasets. MAPE (defined for a forecasting-period of T as $\frac{1}{T}\sum_{t=1}^{T}|\frac{y_t-\hat{y}_t}{y_t}|$, where y_t is the actual value and $\hat{y}_t$ is the predicted value at time t) was calculated for all test data. MAPE was chosen as it is normalized and allows for comparison between different datasets.

3.2 Image Classification

We benchmark three deep learning image classification models on one dataset. The image classification models we used were DenseNet [20], ResNet [17] and MobileNet [41], and the dataset was CIFAR10 [23]. We used the hyperparameters reported in the original papers for DenseNet and ResNet to ensure that we get models with good performance given limited computation resources. The hyperparameters for MobileNet were found experimentally and chosen so that the model size was similar to the other two. Each seed replicate was trained for 300 epochs and evaluated using standard accuracy.

[2]We used GLuonTS 0.5.0 for the experiments.
[3]https://www.nrel.gov/grid/solar-power-data.html

4 RESULTS AND DISCUSSION

Average performance: Table 1 and Table 2 show the average MAPE for the time-series forecasting task and the accuracy for the image classification task, respectively. As mentioned above, our goal is not to achieve state of the art performance for all the models, but to have a broad spectrum of results to analyze the effect of network initialization. As can be seen from Table 1, the performance on the time-series forecasting task vary quite a lot with some of the results such as electricity and traffic being reasonable given that we use a different performance metric and longer forecasting horizons than what is reported in the literature [32]. The results of DenseNet are comparable (slightly better) than those reported by Huang et al. [20].

Effect size: Figure 2 shows the standardized effect size[4] for the best performing seed replicates, $\mathcal{T}^n$, with the worst performing seed replicates, $\mathcal{W}^n$, for the image classification task. Note that we compare $\mathcal{T}^n$ and $\mathcal{W}^n$, both containing seed replicates from the same model. The accuracy estimated from $\mathcal{T}^n$ is optimistic, while the seed replicates $\mathcal{W}^n$ are pessimistic. *Cohen's criteria* for large, medium and small effect sizes are 0.8 or greater, 0.5 and 0.2. When both groups contain less than 80% of the seed replicas, the effect size is considered large, meaning that the performance difference

[4]Cohen's d or standardized effect size tells how many standard deviations lie between two means, expressed as $d = \frac{\overline{x}_1-\overline{x}_2}{\tau}$, where the $\overline{x}_j$ are the means and τ is the pooled standard deviation.

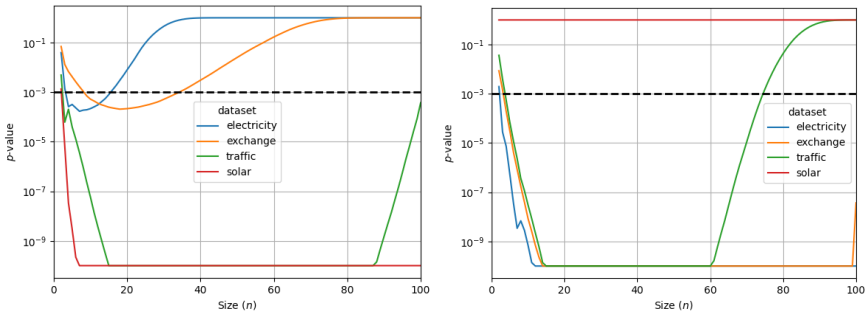


Figure 5: Best vs worst: Trying to prove that LSTMs perform better than DeepAR (left) or vice versa (right). For both experiments the n best seed replicates were from the preferred model were compared to the n worst for the baseline. p-values for the one-sided t-test are reported. By carefully selecting $\mathcal{S}$ we can conclude that LSTMs are better than DeepAR for all datasets ($p \leq 10^{-3}$), and also that DeepAR is better than LSTM for all datasets except solar. p-values smaller than 10^{-10} are set to 10^{-10} to make the figures clearer.

	electricity	exchange_rate	traffic_nips	solar_nips
LSTM	0.118 ± 0.011	0.030 ± 0.012	**0.152 ± 0.008**	**0.386 ± 0.016**
N-BEATS	0.129 ± 0.004	0.020 ± 0.005	0.248 ± 0.012	0.439 ± 0.003
DeepAR	**0.102 ± 0.005**	0.019 ± 0.014	0.156 ± 0.011	1.112 ± 0.084
DeepFactor	6.215 ± 1.290	10.06 ± 0.191	2.933 ± 0.582	3.589 ± 1.345
DeepState	0.621 ± 0.217	**0.013 ± 0.001**	1.023 ± 0.310	1.912 ± 0.627

Table 1: Average MAPE score ± one standard deviation for 100 seed replicates for 100 epochs (lowest error in bold).

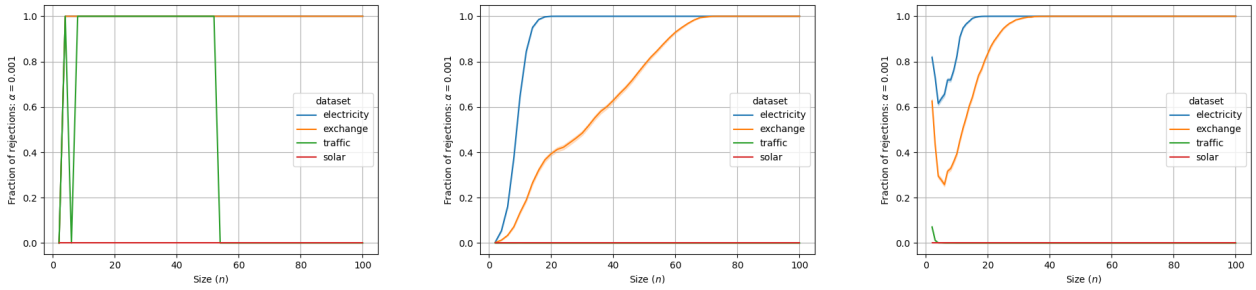


Figure 6: An adversarial agent trying to prove that DeepAR performs better than LSTM using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: The adversarial can still succeed by cherry-picking the seeds where the differences between the models are the largest. The x-axis gives the size of $\mathcal{S}$, while the y-axis shows whether a dataset can be extracted to make sure that it will show that the preferred model is indeed better. The results for electricity are hidden below those from exchange. Results obtained using a t-test. Middle: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the middle panel, but now using a Brunner-Munzel test.

between the models represented by $\mathcal{T}$ and $\mathcal{W}$ is large. The absolute effect size is the magnitude of the difference between two groups. Figure 3 shows the absolute effect size for the three image classification algorithms. The absolute effect size between $\mathcal{T}^5_{DenseNet}$ and $\mathcal{W}^5_{DenseNet}$ is 2.42% and it is considered extremely large by the standardized effect size. Additionally, the 99.9 confidence intervals do not overlap for any of the groups with less than 80 seed replicas, see Figure 10 in Appendix B. In an adversarial model comparison study where two groups of seed replicas with very good or bad

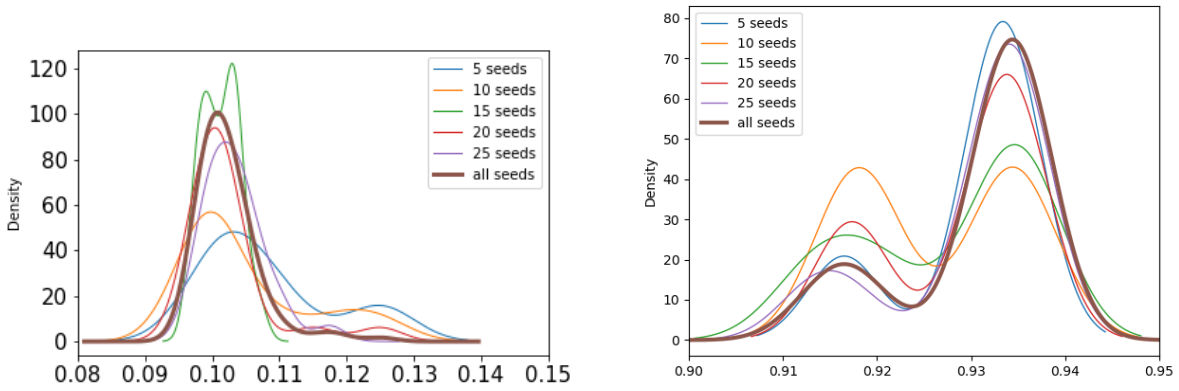


Figure 7: KDE plots: Comparison of KDE based on 5, 10, 15, 20 and 25 seeds sampled from a population of 100 seeds, together with all 100 seeds for MAPE of DeepAR on Electricity (left) and accuracy of Densnet on CIFAR 10 (right).

	CIFAR10
DenseNet	**0.931 ± 0.008**
ResNet	0.879 ± 0.003
MobileNet	0.851 ± 0.004

Table 2: Image classification accuracy ± one standard deviation for 100 seed replicates for 300 epochs of three DNNs with comparable number of hyperparameters.

seeds are compared – whether it is by chance or by bad actor design – the study can therefore conclude with high confidence that the performance of $\mathcal{T}^5_{DenseNet}$ is better than the performance of $\mathcal{W}^5_{DenseNet}$.

We can also see that the degree of variation does not depend on the performance of the model. DenseNet has by far the best performance, refer to Table 2, but it also varies the most. MobileNet has the worst average performance, but the absolute effect size varies more than for ResNet. The results for the time-series models are similar and can be found in the supplementary materials.

Selecting random seeds freely: Figure 4 shows the effect of the "best vs. worst" on the time-series forecasting task. See Figure 9 in the appendix for the same scenario on the image classification task. We compare the best n seed replicates with the worst n seed replicates out of 100; all from the same dataset using the same model. The x-axis gives n, the y-axis gives the p-value of the hypothesis test H_0: *The two sequences have the same mean* vs. H_1: *They do not.* We used a two-sided t-test, and the dashed line gives the line for $\alpha = 10^{-3}$. As can be seen, even with more than $n = 80$ of the 100 seeds represented in each group, the test will with high significance say the means are different. This establishes that seed variation is important, and that one by cherry-picking seeds can get very significant results of effects that are not really there. Hence, the search space is so difficult in all these runs that initialization will have a significant effect on performance.

Figure 5 shows the effect of the "Best vs. worst" setup when comparing the two models LSTM and DeepAR on the same dataset: We compare the best n examples out of 100 seed replicates from LSTM to the worst n seed replicates of DeepAR (left) or vice versa (right). The x-axis gives n, the number of seed selected in $\mathcal{S}$, y-axis gives the p-value of the hypothesis test H_0: *The two sequences have the same mean* vs. H_1: *The LSTM is better (left) / DeepAR is better (right).* In this scenario, the adversarial agent seeks to prove that one model is better than the other, and use a one-sided t-test to validate the difference. The line indicates $\alpha = 10^{-3}$. The results are as expected from Table 1: DeepAR is clearly better than LSTMs on electricity and exchange_rate, yet the adversarial will be able to conclude otherwise in some instances (for carefully selected $\mathcal{S}$ of specific values of n). The difference between the two is smaller for traffic, yet with LSTMs appearing slightly better. LSTMs are clearly better than DeepAR on solar, and the adversarial agent is unable to provide results showing otherwise.

Observation 1: *Selecting random seeds freely when comparing models can lead to untrustworthy results.*

Models compared using same set of seeds: From the results above it seems clear that one should not be allowed to cherry-pick the seeds separately for each model. A slightly more restricted approach is to say there should be the same set of seeds $\mathcal{S}$ used for both models that are to be compared. The left part of Figure 6 shows the results when we initially have a pool of 100 seeds out of which we have selected a subset of size n, and let this subset play the role of $\mathcal{S}$. As long as no constraints are put on $\mathcal{S}$, meaning that as long as the same set of seeds is used for both models we are free to choose exactly what seeds to include, an adversarial agent can then of course optimize which seeds to use. This is done simply by choosing those seeds where the two models' performances differ the most in the preferred direction. In Figure 6 (left) the aim is to show (using a one-sided t-test, rejection-level $\alpha = 10^{-3}$) that the mean performance of DeepAR is better than that of LSTM. The x-axis gives the size of $\mathcal{S}$, the y-axis show if the test is significant ($y = 1$) or not ($y = 0$). As can be seen, the adversarial agent can

easily promote DeepAR for the datasets exchange and electricity (the latter curve hidden below the former), and also for some n devise a test to indicate that DeepAR is indeed the better also on traffic.

> ***Observation 2:*** *Enforcing the same set of seeds to be used for all models when comparing them is not enough to ensure a fair comparison.*

Using a predefined set of seeds: Finally, one way to constrain the adversarial agent further is to not only force both models to be evaluated using the same set of seeds, but also to define what that set of seeds should be used, e.g., $\{0, 1, \ldots, n-1\}$. This is investigated in the middle panel of Figure 6. Here we choose a random seed set of size n (n given on the x-axis) and report on the y-axis the fraction of experiments out of 10.000 repetitions where DeepAR is found to be better than LSTM (again using a one-sided t-test with $\alpha = 10^{-3}$). This approach prevents the adversarial from promoting DeepAR as better on exchange. We can also see that we for instance choose $n = 20$, the probability that a fixed $\mathcal{S}$ will reveal that DeepAR is better on electricity is very close to 1.0 and about 0.4 for exchange. Results comparing other models are given in the appendix.

> ***Observation 3:*** *Prescribing exactly which initialization seeds to use when comparing models will increase the difficulty and cost for an adversarial agent to influence the results.*

Using the t-test: The analysis so far has been based on t-tests, meaning that we had to (implicitly) assume that each data sample is from a Gaussian distribution. However, this assumption is not met in our data. (Qualitative evidence for this claim based on KDE plots can be found in the appendix, see Figure 11-12). We examine this further in the right panel of Figure 6. The setup is identical to the middle panel of the same figure, but we have now exchanged the t-test for a Brunner-Munzel test [7]. The *test strength*, i.e., the ability to expose two models as providing different results when they in fact are, is particularly eye-catching: notice how the strength is improved by changing to a non-parametric test for both electricity and exchange-rate.

> ***Observation 4:*** *A non-parametric test like the Brunner-Munzel test should be used when comparing models, because the often-considered t-test is vulnerable to outliers.*

Number of seeds to use: In order to identify the number of seeds that must be tested to decide that we cannot distinguish between two models, we sample n seed replicates from the dataset replicates and test whether they come from distributions with the same mean using the Brunner-Munzel test. See Figure 8 for details. The figure shows that the fraction of rejection crosses the dashed line ($\alpha = 0.001$) for all time-series forecasting models when $n = 25$, and thus indicates the number of seeds that must be tested for the test to be trusted. Figure 13, found in the appendix, shows that $n = 10$ for $\alpha = 0.05$. The results from the image classification task supports this finding. These can be seen in Figure 14, also in the appendix. Qualitative support for this finding can be found in Figure 7. The figure shows kernel density estimation (KDE) plots over the performance when varying the number of seed replicates. The KDE plots were selected subjectively to show the diversity of the results. Both five and ten seeds can be very misleading while 25 gives a better approximation of the actual distribution. We mean that the additional plots provided in the appendix, Figures 11 and 12, clearly show how superior to KDE plots are to illustrate the variation in performance caused initialization seeds when compared to tables with confidence interval and error bars in bar charts.

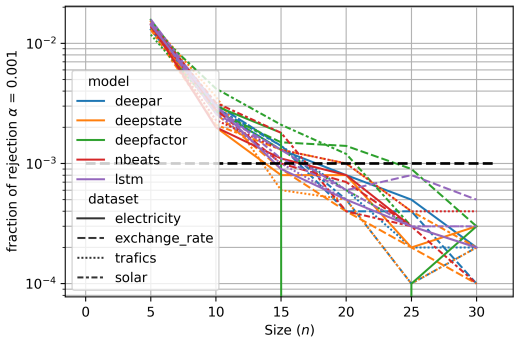


Figure 8: We repeatedly sample two independent sets of seed candidates of the same size learned from using the same model on the same dataset, then test if the two samples come from distributions with the same mean using a two-sided Brunner-Munzel test. The x-axis gives the number of seed candidates in each sample, the y-axis the fraction (out of 10.000 repetitions), where the two sets were found to be significantly different ($\alpha = 10^{-3}$); note the log-scale. The dashed line at $p = \alpha = 10^{-3}$ indicates the rejection level, and thus the fraction of rejections we theoretically expect.

> ***Observation 5:*** *Even when using a predefined set of seeds and a non-parametric hypothesis test like Brunner-Munzel ($\alpha = 10^{-3}$), more than $n = 25$ seeds are needed for the test to reliably conclude that two indistinguishable models are indeed performing on an equal footing.*

5 CONCLUSIONS AND RECOMMENDATIONS

Our experiments support the claims made in prior works that one must be careful when drawing conclusions based on model comparison experiments in deep learning.

As pointed out by Ioannidis [22], the smaller the effect sizes are, the less likely the research findings are to be correct. This holds true for model comparison studies as well. Our study shows that conclusions can easily be forced by a bad actor when the difference in performances are small. We show that the gap between the best and worst performing seed replicates can be more than 2% even for well-performing models and for five seed replicates. This tells us that we should not fully trust results when the effect sizes are small – unless variation caused by random initialization is given proper treatment.

Furthermore, our results clearly indicate that initialization seeds should not be selected freely in model comparison studies. This is also in line with the observations by Ioannidis [22], who states that the greater the flexibility in design and analytical modes (i.e. hyperparameter settings) are, the less likely are the research findings to be true. We found that it is not enough just to restrict the models to

be compared using the same set of seeds; the set of seeds should be pre-defined. This increases the effort required to force false results.

Finally, we showed how reporting sensitivity to variation could be improved by visualizing the variation in performance using KDE plots. We find that such plots provide more information of how sensitive a result is to variation. Both compared to only reporting variation by mean performance and variance numerically, which is most commonly the case for model comparison studies, but also when error bars are used, as suggested by Pineau [34].

Based on our observations, the following should be done to reach robust and trustworthy conclusions for model comparison studies involving neural networks and randomness:

(1) use a predefined set of seeds $\{0, 1, \ldots n-1\}$,
(2) test with a non-parametric hypothesis test like the Brunner-Munzel,
(3) use KDE plots to visualize the variation in performance caused by initialization seeds,
(4) evaluate more than $n = 25$ seeds for confirmation studies (testing with $\alpha = 10^{-3}$) when effect sizes are small (less than 2.5%).

Item number 4 above is the answer to our research question. However, following these recommendations will not solve all problems. For example, fixing the set of seeds will potentially bias the models in random ways, as the performance of a seed replicate is a function of the optimization algorithm, dataset, and model [10]. Also, there is a trade-off between computational resources and the reliability of results, as mentioned by Melis et al. [27]. To reduce the computational demand, using $n = 10$ and testing at $\alpha = .05$ will reduce the probability of false findings compared to using even smaller values of n, albeit reduce the strength of the claims when they cannot be made more robust. However, for many deep learning studies, it is practically impossible for most labs to run even $n = 10$ experiments only to increase reliability of the findings. Thus, we avoid making hard recommendations for the number of random seeds that should be used for studies. It is still important to properly understand how sensitive the performance of neural networks is to random initialization. Typically, in order for papers proposing new deep learning algorithms to be accepted at top AI conferences, clear wins when compared to state-of-the-art baselines are required [42]. As our investigation shows, statistically significant "wins" do not necessarily lead to true conclusions. Hence, we should avoid requiring clear wins for a paper to be publishable. New ideas should be accepted based on other properties as well, such as efficiency, explainability, robustness, reliability and so on. Also when clear wins are declared, whether randomness could affect the results should be discussed.

One could argue that focusing only on initialization seeds is a limitation of this work. The sources of variation are many, and their impact on performance is significant. It is no use in removing just one of them [46] and adding more sources of variation to an imperfect estimator approaches the ideal estimator faster and better [5]. However, we argue that a proper understanding of initialization seeds has clear advantages over the other implementation factors. This is because experiments in most academic settings are mostly conducted in the same laboratory using the same software stack and hardware platform and using one implementation of all baselines. Controlling for all implementation factors requires a huge effort, so it is practically impossible for most research groups. Evaluating intialization seeds requires less effort while providing insights into the sensitivity of the algorithms to variation. Hence, it is a reasonable proxy, at least for exploratory studies, which entails most empirical studies of deep learning. Confirmatory studies require more care, but fewer such studies will be conducted precisely because of the computational costs.

REFERENCES

[1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. 2020. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research* 21, 116 (2020), 1–6. http://jmlr.org/papers/v21/19-820.html

[2] Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. *Nature* 533, 7604 (may 2016), 452–454. https://doi.org/10.1038/533452a

[3] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. 2021. Deep learning for AI. *Commun. ACM* 64, 7 (2021), 58–65.

[4] Léon Bottou et al. 1998. Online learning and stochastic approximations. *On-line learning in neural networks* 17, 9 (1998), 142.

[5] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. 2021. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems* 3 (2021), 747–769.

[6] Xavier Bouthillier, César Laurent, and Pascal Vincent. 2019. Unreproducible Research is Reproducible. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 725–734.

[7] Edgar Brunner and Ullrich Munzel. 2000. The Nonparametric Behrens-Fisher Problem: Asymptotic Theory and a Small-Sample Approximation. *Biometrical Journal* 42, 1 (2000), 17–25.

[8] Marco Cuturi. 2011. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 929–936.

[9] Dheeru Dua and Casey Graff. 2019. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[10] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, 42 pages. https://openreview.net/forum?id=rJl-b3RcF7

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[12] Odd Erik Gundersen. 2020. The Reproducibility Crisis Is Real. *AI Magazine* 41, 3 (2020), 103–106.

[13] Odd Erik Gundersen, Kevin Coakley, Christine Kirkpatrick, and Yolanda Gil. 2023. Sources of Irreproducibility in Machine Learning: A Review. arXiv:2204.07610 [cs.LG]

[14] Odd Erik Gundersen, Saeid Shamsaliei, and Richard Juul Isdahl. 2022. Do machine learning platforms provide out-of-the-box reproducibility? *Future Generation Computer Systems* 126 (2022), 34–47.

[15] Michael Haber, Huiman X Barnhart, Jingli Song, and James Gruden. 2005. Observer variability: a new approach in evaluating interobserver agreement. *Journal of Data Science* 3, 1 (2005), 69–83. https://doi.org/10.6339/JDS.2005.03(1).181

[16] Benjamin Haibe-Kains, George Alexandru Adam, Ahmed Hosny, Farnoosh Khodakarami, Levi Waldron, Bo Wang, Chris McIntosh, Anna Goldenberg, Anshul Kundaje, Casey S Greene, et al. 2020. Transparency and reproducibility in artificial intelligence. *Nature* 586, 7829 (2020), E14–E16.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*. AAAI Press, Article 392, 8 pages.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[21] Matthew Hutson. 2018. Artificial intelligence faces reproducibility crisis. *Science* (2018).

[22] John PA Ioannidis. 2005. Why most published research findings are false. *PLoS medicine* 2, 8 (2005), e124.

[23] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report, CIFAR.

[24] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 95–104.

[25] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf

[26] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2018. Are GANs created equal? A large-scale study. In *Advances in neural information processing systems*. 700–709.

[27] Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. In *International Conference on Learning Representations*.

[28] Taciano L Milfont and Richard A Klein. 2018. Replication and reproducibility in cross-cultural psychology. *Journal of Cross-Cultural Psychology* 49, 5 (2018), 735–750.

[29] J.N. Miller and J.C. Miller. 2018. *Statistics and Chemometrics for Analytical Chemistry*. Pearson Education, London, England.

[30] Open Science Collaboration. 2015. Estimating the reproducibility of psychological science. *Science* 349, 6251 (2015), aac4716.

[31] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1ecqn4YwB

[32] Boris N. Oreshkin, Grzegorz Dudek, Paweł Pełka, and Ekaterina Turkina. 2021. N-BEATS neural network for mid-term electricity load forecasting. *Applied Energy* 293 (2021), 116918. https://doi.org/10.1016/j.apenergy.2021.116918

[33] Hung Viet Pham, Shangshu Qian, Jiannan Wang, Thibaud Lutellier, Jonathan Rosenthal, Lin Tan, Yaoliang Yu, and Nachiappan Nagappan. 2020. Problems and opportunities in training deep learning software systems: An analysis of variance. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 771–783.

[34] Joelle Pineau. 2020. The machine learning reproducibility checklist. *URL: https://www.cs.mcgill.ca/ ˜jpineau/ReproducibilityChecklist.pdf* (2020).

[35] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d,ÄôAlché Buc, Emily Fox, and Hugo Larochelle. 2021. Improving reproducibility in machine learning research: a report from the NeurIPS 2019 reproducibility program. *Journal of Machine Learning Research* 22 (2021).

[36] Florian Prinz, Thomas Schlange, and Khusru Asadullah. 2011. Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews Drug discovery* 10, 9 (2011), 712–712.

[37] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep State Space Models for Time Series Forecasting. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7785–7794. http://papers.nips.cc/paper/8004-deep-state-space-models-for-time-series-forecasting.pdf

[38] Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 338–348.

[39] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. 2019. High-dimensional multivariate forecasting with low-rank Gaussian copula processes. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 6827–6837.

[40] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[41] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[42] David Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. 2018. Winner's curse? On pace, progress, and empirical rigor. In *ICLR 2018 Workshop Track*.

[43] David L Vaux, Fiona Fidler, and Geoff Cumming. 2012. Replicates and repeats?what is the difference and is it significant? A brief discussion of statistics and experimental design. *EMBO reports* 13, 4 (2012), 291–296.

[44] Yuyang Wang, Alex Smola, Danielle C. Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. 2019. Deep Factors for Forecasting. *arXiv e-prints*, Article arXiv:1905.12417 (May 2019), arXiv:1905.12417 pages. arXiv:1905.12417 [stat.ML]

[45] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. 2020. Hyperparameter Ensembles for Robustness and Uncertainty Quantification. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 6514–6527. https://proceedings.neurips.cc/paper/2020/file/481fbfa59da2581098e841b7afc122f1-Paper.pdf

[46] Donglin Zhuang, Xingyao Zhang, Shuaiwen Song, and Sara Hooker. 2022. Randomness in neural network training: Characterizing the impact of tooling. *Proceedings of Machine Learning and Systems* 4 (2022), 316–336.

A APPENDIX A: EXPERIMENT SETUP

This section contains more details on models and datasets used in the forecasting and image classification tasks.

A.1 Forecasting models

The models were trained for 100 epochs on a GPU (except for DeepState which we had to train on the CPU).

DeepAR [40] is a probabilistic model utilizing a single autoregressive LSTM for all timeseries in a data-set. The output of the LSTM is fed through two different dense layers, outputting the mean and variance of a Gaussian distribution. When producing predictions, the history of the timeseries is fed into the model, a sample is then drawn from the produced distribution and then fed back into the model to produce the next prediction. DeepAR in GluonTS uses a student-T distribution instead of a Gaussian.

DeepFactor [44] consists of a deep global model and a local model. The global model encodes timeseries into K deepfactor global factors using an LSTM, that are then combined through a linear combination to produce the "fixed effect" of the model. The local model provides the "random effect" to the model and can be any classical probabilistic method for timeseries forecasting. The fixed and random effects are then combined, through a freely defined function, to parameterize the sampling distribution. In GluonTS this model is implemented with one global LSTM producing the mean of the distribution, while one local LSTM producing the variance. These two LSTM models have different architectures and receive different inputs. This is similar to the DF-RNN proposed in the paper.

DeepState [37] is a probabilistic model using an LSTM to produce time-varying parameters of a state-space model. More specifically, the LSTM receives the covariates of the timeseries as input and produces the state space parameters. Predictions are produced by using the posterior of the latent state, together with the transition equation and observation model of the state-space model. New parameters for the state-space model is computed by continuing to unroll the LSTM, which can then be used to sample a prediction for the next time step.

N-BEATS [31] is a pure deep learning model for time series forecasting producing point forecasts. The model consists of a number of stacks, and the final forecast of the model is produced by taking the sum of each stack forecast. Each stack is made up of several blocks which has two outputs: "back-cast" (the estimate of the input to the block) and "forecast". In each stack, the back-cast of block N will be subtracted from the input to block $N + 1$. This allows subsequent blocks to focus on the part of the signal that has not been learned by the previous blocks. The forecasts from a single stack is the sum of the stack's internal blocks. we used a single module of NBEAT and not the ensemble. The reason is that the ensemble implementation crashed.

LSTM [19] is a recurrent architecture. The LSTM has encoder-decoder architecture with 2 layers, 40 nodes and we set the dropout to 0.1. The input features are the same as the DeepAR model.

A.2 Image classification models

In order to select image classification models, we chose models with close to state-of-the-art performance along with having a small number of learnable parameters, so that we are able to train 100 seed replicates for 300 epochs.

ResNet [17] makes use of residual learning framework which allows to train considerably deeper networks. Authors show that this approach is also easier to optimize since it alleviates vanishing gradients. For our experiments, we used the hyperparameters mentioned in the paper for the variation ResNet110. This model has 1.7M learnable parameters

DenseNet [20] pushes the idea of short connections further and connects each layer to every other proceeding layer. Due to the dense connection, DenseNet requires less computation and fewer parameters to achieve high accuracy. We used the same hyperparameters as mentioned in the original paper and chose the model with growth rate of 12 and depth of 40 which has approximately 1 million learnable parameters.

MobileNetV2 [41] is a mobile architecture which is based on inverted residual structure and made use of lightweight depthwise convolutions in the intermediate expansion layers and removed non-linearities in the narrow layers. Due to small network size and low complexity cost, this network is suitable for mobile devices. We use an architecture similar to what is implemented in Tensorflow library which has 2.2M parameters.

A.3 Forecasting Data

For Electricity and Exchange Rate, we use the same pre-processing as in [24] while PEMS-SF Data Set and Solar Power were pre-processed as in [39]. These datasets corresponds to the electricity, exchange rate, traffic_nips and solar_nips data-sets provided as part of GluonTS.

Electricity [9] is the electricity consumption of 370 measuring points, sampled every 15 minutes.

Exchange Rate [24] is the daily exchange rate between USD and eight different currencies including Australia, British, Canada, Switzerland, China, Japan, New Zealand and Singapore from 1990 to 2016.

PEMS-SF [8] is the hourly occupancy rate, measured as a number between 0 and 1, of 963 car lanes in San Francisco bay area freeways. The measurements cover the period from Jan. 1st 2008 to Mar. 30th 2009 and are sampled every 10 minutes.

Solar Power[5] captues the solar power production of 137 plants in Alabama in 2016, sampled every 10 minutes and provided by the National Renewable Energy Laboratory.

A.4 Image Classification Data

CIFAR10 [23] is an image classification dataset with 10 classes and 6000 colored images per class. It has 50000 training and 10000 test images. This dataset is labeled subset of the 80 million tiny images. In our experiments, no data augmentation were done as we wanted to reduce the sources of variation.

A.5 Experiment Setup: Time-series Forecasting

Hardware

GPU: NVIDIA Tesla K80
CPU: Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz

[5] From: https://www.nrel.gov/grid/solar-power-data.html.

Cores: 6
RAM: 65 GiB

Software

OS: Ubuntu 18.04.4 LTS
Python: 3.7.7
Python packages:

- GluonTS 0.5.0
- Keras-Preprocessing 1.1.2
- matplotlib 3.2.2
- mxnet-cu101 1.6.0
- numpy 1.18.5
- pandas 1.0.5
- scipy 1.4.1
- tensorflow 2.3.0

A.6 Experiment Setup: Image Classification

Hardware

GPU: NVIDIA Tesla V100-PCIE-32GB
CPU: Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz
Cores: 8
RAM: 65 GiB

Software

OS: Ubuntu 18.04.5 LTS
Docker Image: ray-ml:1.3.0-gpu
Additional Python packages:

- tensorflow-datasets 4.4.0

A.7 Experiment Setup: Hyperparameters

DeepAR

Number of layers: 2
Number of cells: 40
Dropout: 0.1
Number of parallel sample: 100
Output distribution: StudentT

DeepState

Number of layers: 2
Number of cells: 40
Dropout: 0.1
Number of parallel sample: 100
Number of periods to train: 4

DeepFactor

Number of factors: 10
Number of layer global: 1
Number of layer local: 1
Number of hidden global: 50
Number of hidden local: 5
Number of parallel sample: 100
Embedding dimension: 10
Output distribution: StudentT

N-BEATS

Number of stacks: 30
Expansion coefficient lengths: 32
Stack type: Generic
Sharing weights between stacks: False
Number of block layers: 4
Number of blocks: 1
Width: 512

LSTM

Number of layers: 2
Number of cells: 40
Dropout: 0.1

DenseNet

Depth: 40
Number of dense blocks: 3
Growth rate: 12
Number of filters: 16
Weight decay: 1E-4
Dropout: 0.2
Batch size: 64
Learning rate:

- Epochs 0 to 150: 0.1
- Epochs 150 to 225: 0.01
- Epochs 225 to 300: 0.001

ResNet

Version: ResNet110
Dropout: 0
L2 regularization: 1E-6
Batch size: 128
Learning rate:

- Iterations 0 to 400: 0.01
- Iterations 400 to 32000: 0.1
- Iterations 32000 to 48000: 0.01
- From iterations 48000: 0.001

MobileNetV2

Alpha: 1
Depth multiplier: 1
Batch size: 128
Dropout: 0.25
Number of bottlenecks: 7
Momentum: 0.9
Learning rate:

- Epochs 0 to 5: 2E-2
- Epochs 5 to 50: 1E-2
- Epochs 50 to 100: 8E-3
- Epochs 100 to 150: 4E-3
- Epochs 150 to 200: 2E-3
- Epochs 200 to 300: 1E-3

B APPENDIX B: ADDITIONAL RESULTS

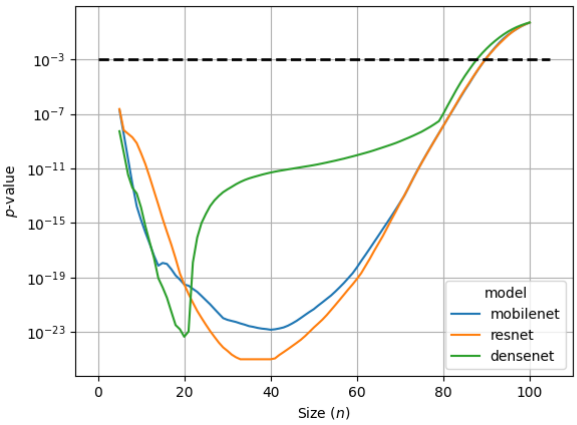


Figure 9: A statistical test comparing whether B^n and W^n are from the same distribution results in statistical significance for all models and all datasets even with n as large as $n \sim 80$. The reported p values are from a one-sided two-sample t-test.

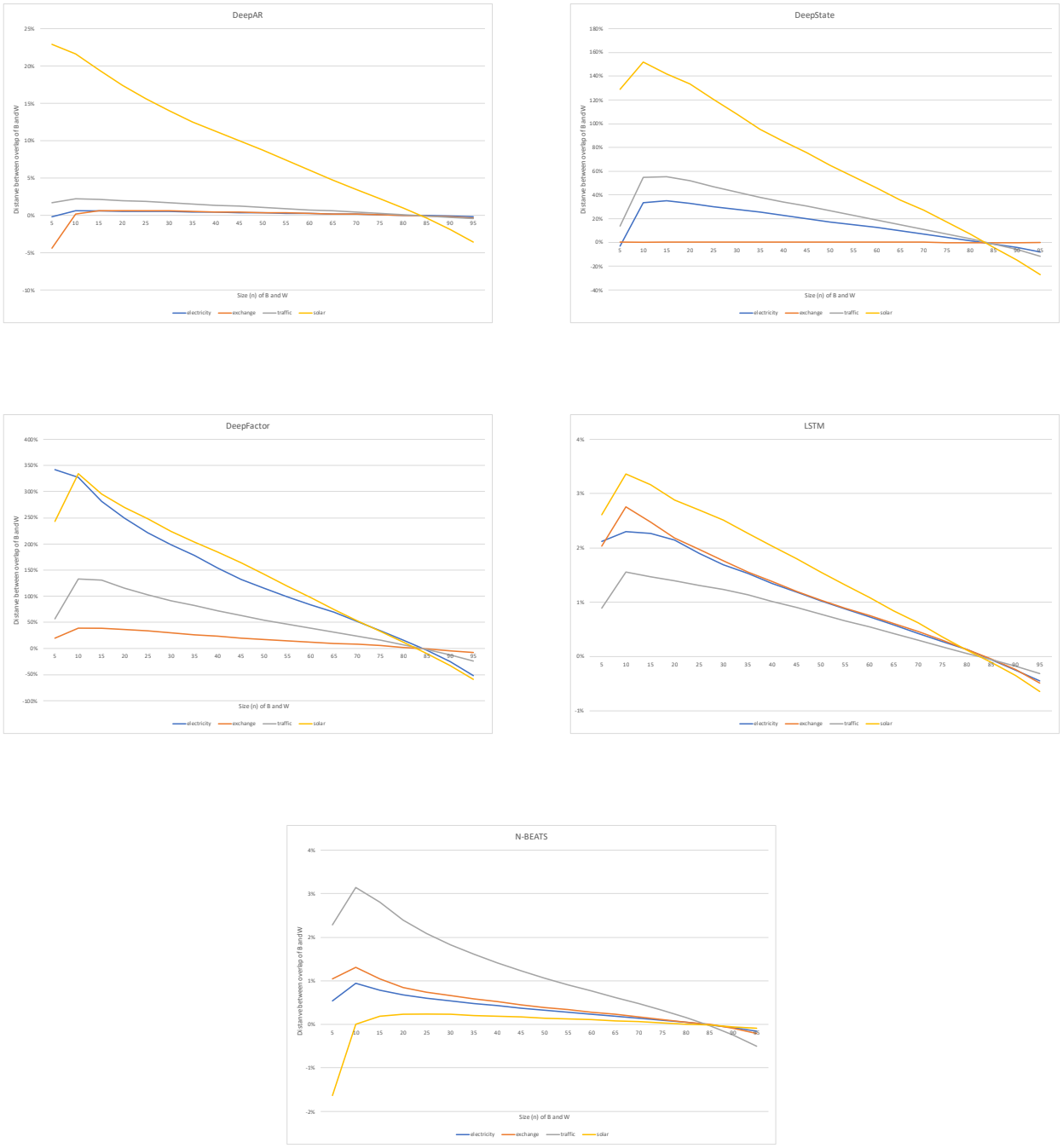


Figure 10: The gap between the confidence intervals (99.9% level) of B_n and W_n for all time-series forecasting models. This figure shows the gap, G^n, between the 99.9% confidence intervals (CI) of the true accuracy estimated from the best performing seed replicates, B^n, when compared to the worst performing seed replicates, W^n, for the image classification task. Note that we compare B^n and W^n who are seed replicates of the same model. In this way we compare models that we know have the same performance (as they are the same model).

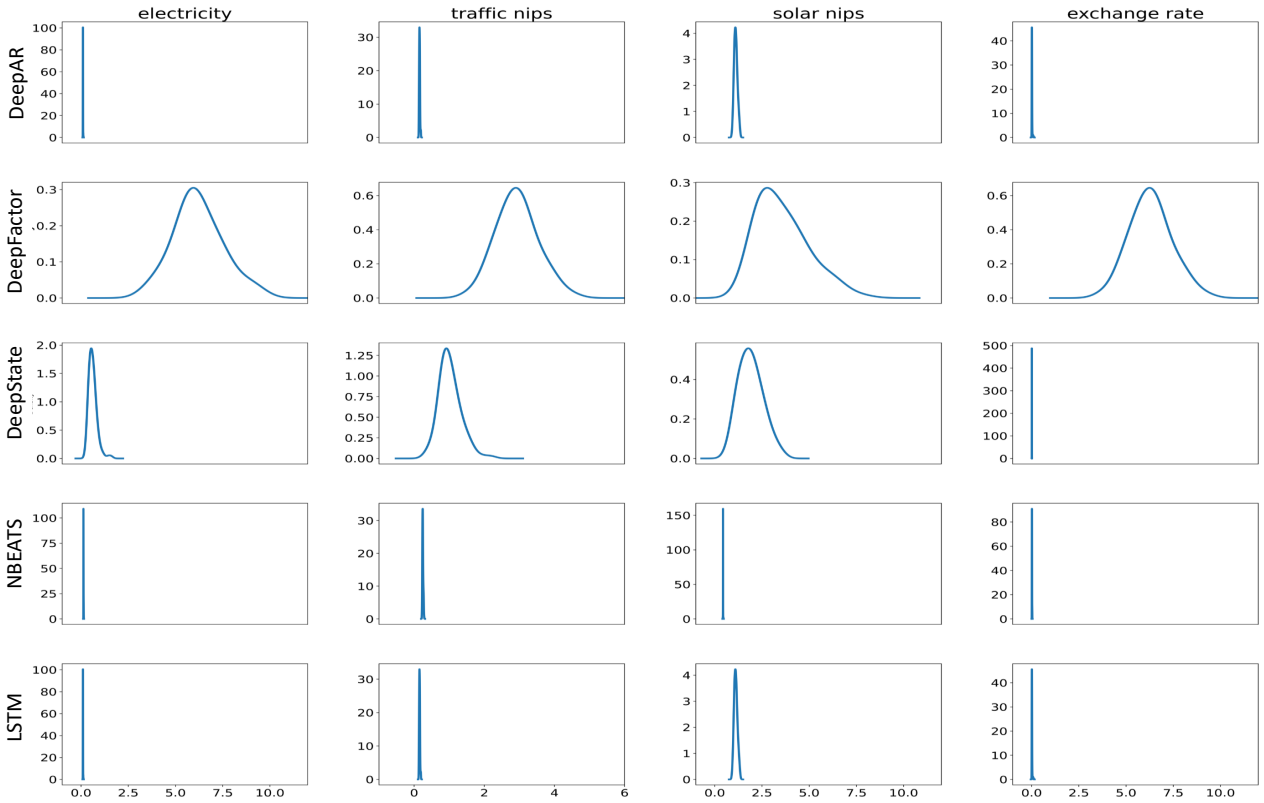


Figure 11: KDE plot of the MAPE scores for all 100 seed replicates shown for all five models and all four datasets.

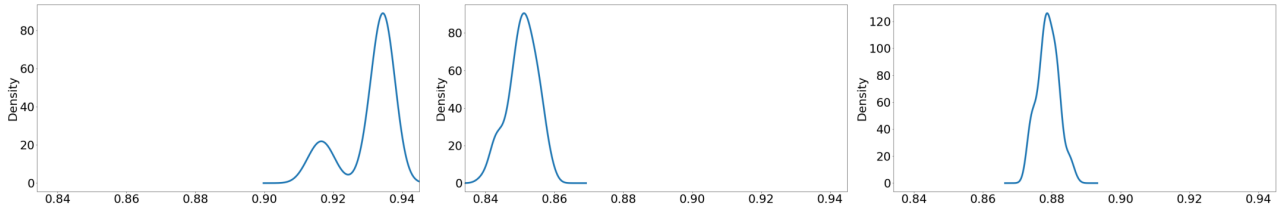


Figure 12: KDE plot of the accuracy of DenseNet, MobileNet and ResNet on image classification task on CIFAR10.

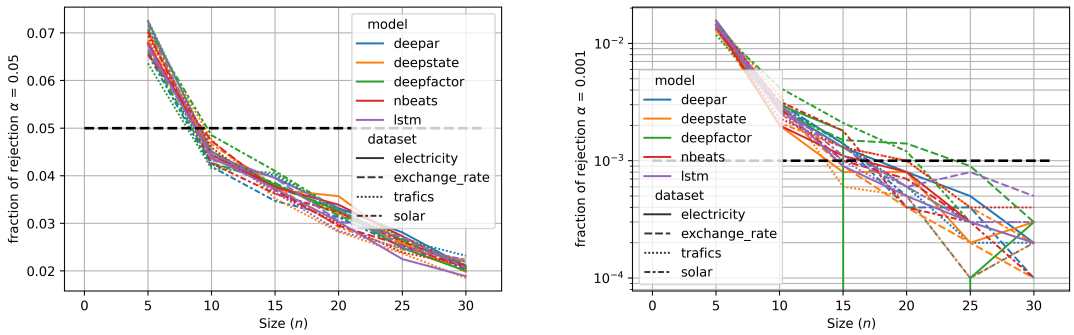


Figure 13: Time-series forecasting: We repeatedly sample two independent sets of seed candidates of the same size learned from using the same model on the same dataset, then test if the two samples come from distributions with the same mean using a two-sided Brunner-Munzel test. The x-axis gives the number of seed candidates in each sample, the y-axis the fraction (out of 10.000 repetitions), where the two sets were found to be significantly different. The dashed line indicates the rejection level, and thus the fraction of rejections we theoretically expect. Left: Rejection level $p = \alpha = 0.05$. Right: Rejection level $p = \alpha = 10^{-3}$. Note the log-scale.

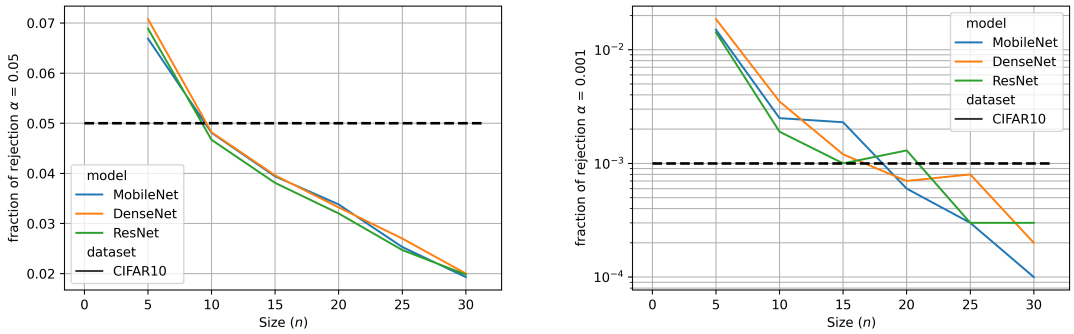


Figure 14: Image classification: We repeatedly sample two independent sets of seed candidates of the same size learned from using the same model on the same dataset, then test if the two samples come from distributions with the same mean using a two-sided Brunner-Munzel test. The x-axis gives the number of seed candidates in each sample, the y-axis the fraction (out of 10.000 repetitions), where the two sets were found to be significantly different. The dashed line indicates the rejection level, and thus the fraction of rejections we theoretically expect. Left: Rejection level $p = \alpha = 0.05$. Right: Rejection level $p = \alpha = 10^{-3}$. Note the log-scale.

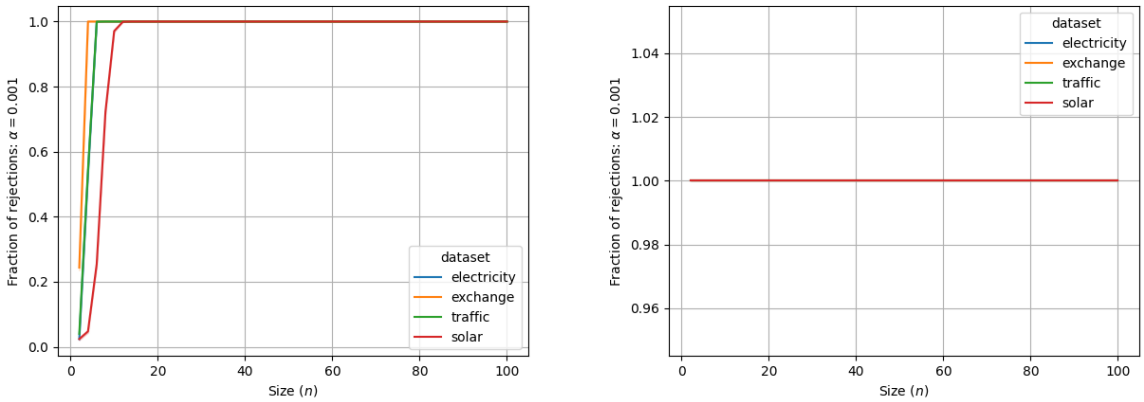


Figure 15: An adversarial trying to prove that DeepAR performs better than DeepFactor using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

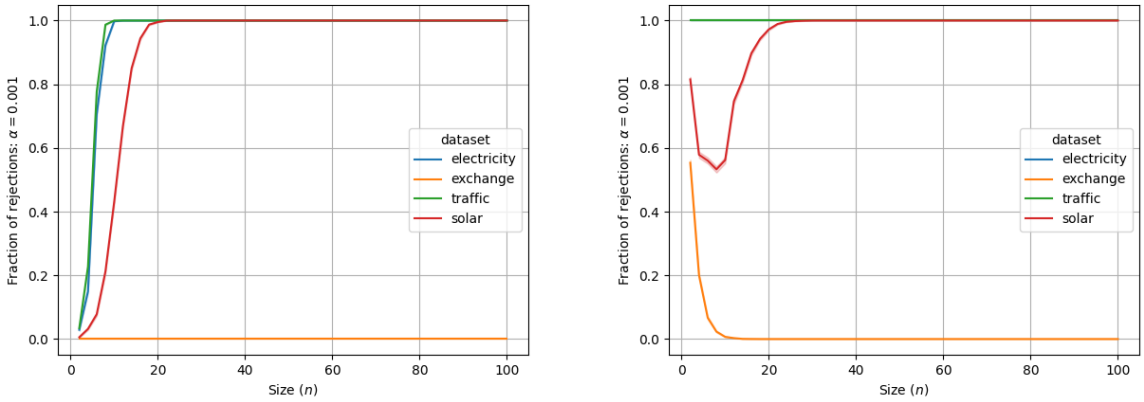


Figure 16: An adversarial trying to prove that DeepAR performs better than DeepState using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

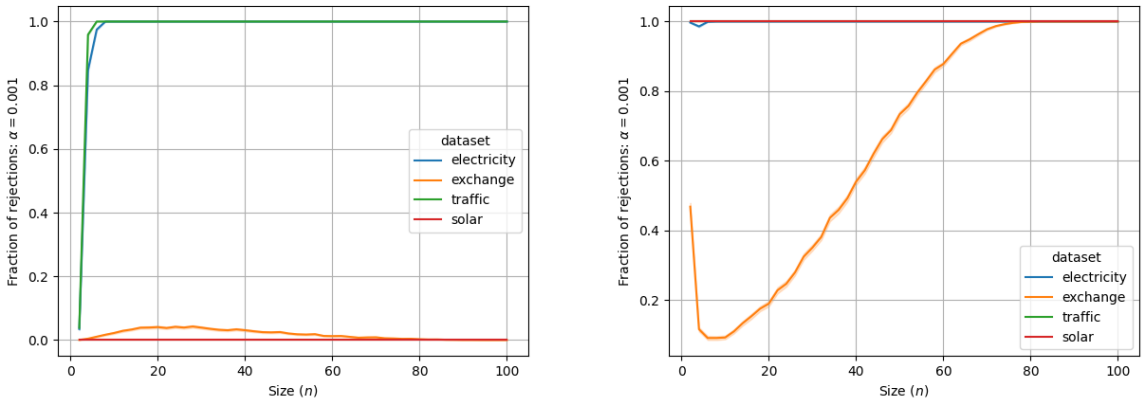


Figure 17: An adversarial trying to prove that DeepAR performs better than N-BEATS using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

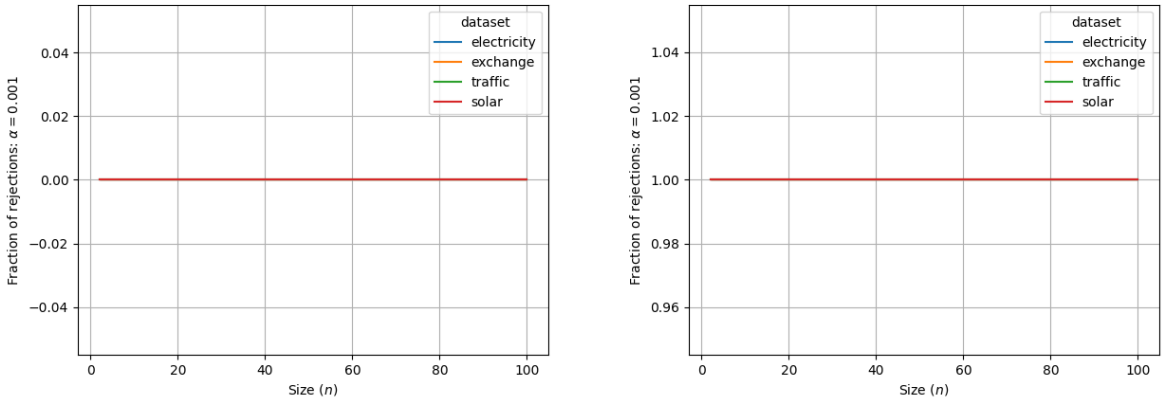


Figure 18: An adversarial trying to prove that DeepFactor performs better than DeepAR using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

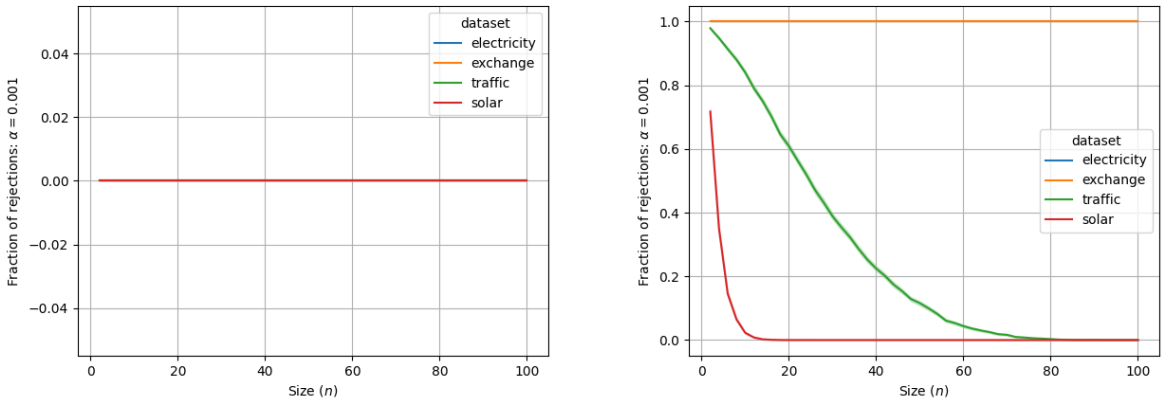


Figure 19: An adversarial trying to prove that DeepFactor performs better than DeepState using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

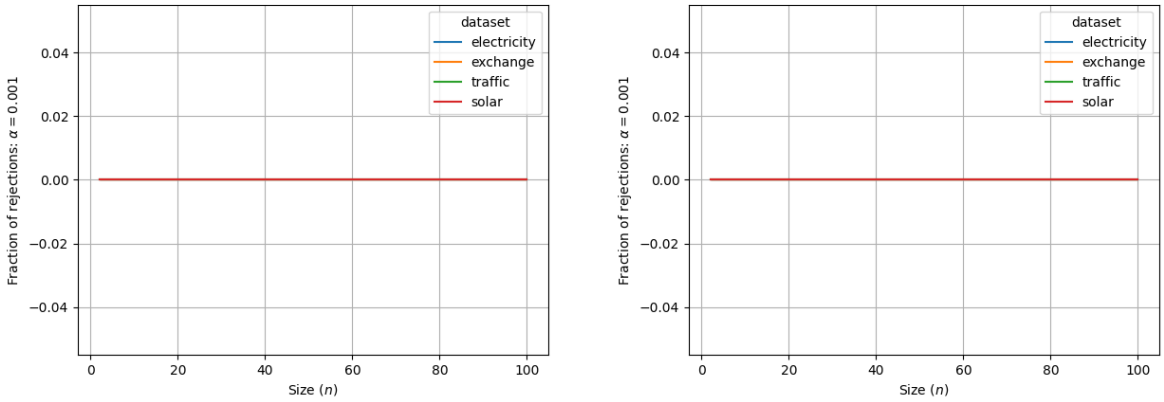


Figure 20: An adversarial trying to prove that DeepFactor performs better than LSTM using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

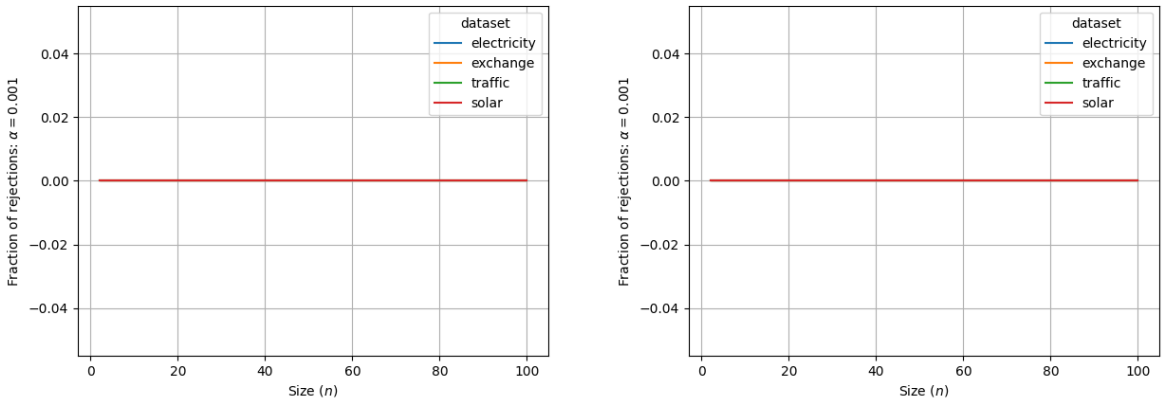


Figure 21: An adversarial trying to prove that DeepFactor performs better than N-BEATS using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

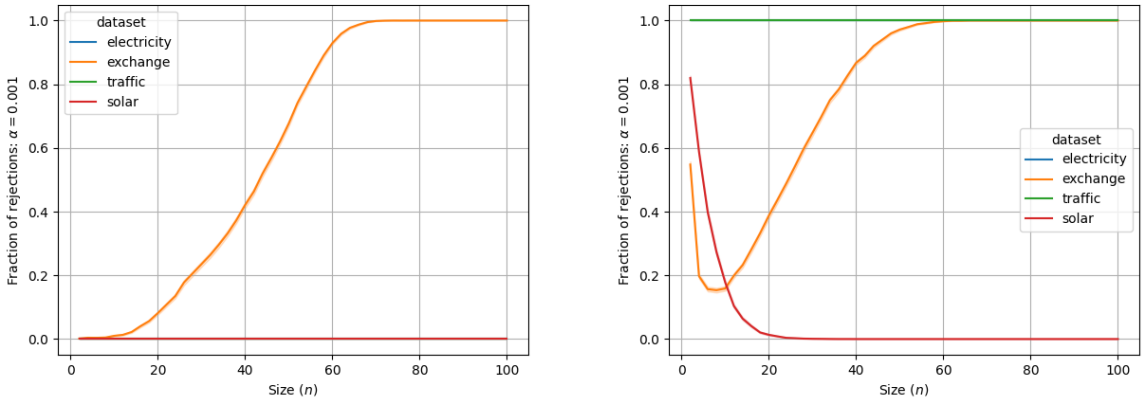


Figure 22: An adversarial trying to prove that DeepState performs better than DeepAR using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

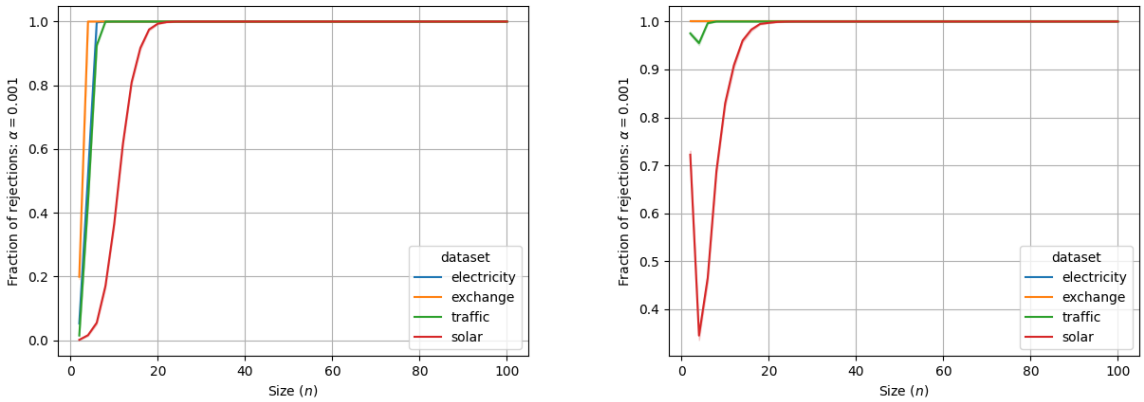


Figure 23: An adversarial trying to prove that DeepState performs better than DeepFactor using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

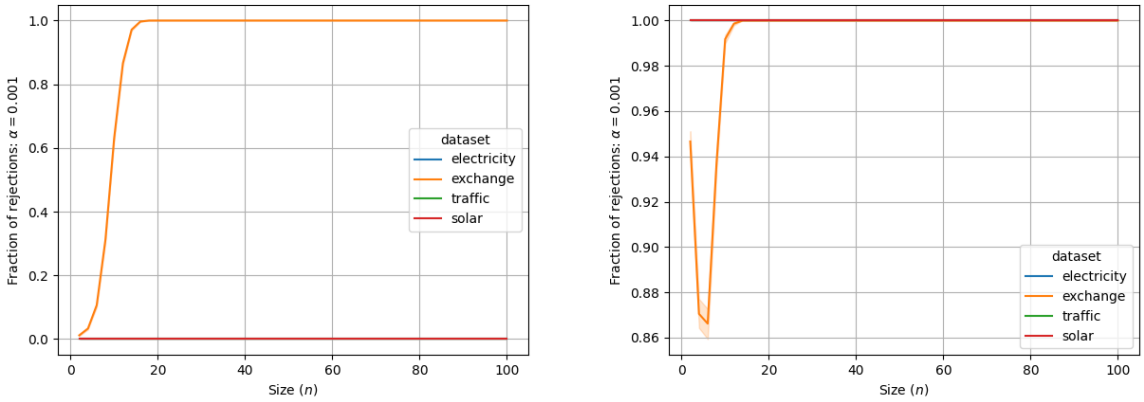


Figure 24: An adversarial trying to prove that DeepState performs better than LSTM using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

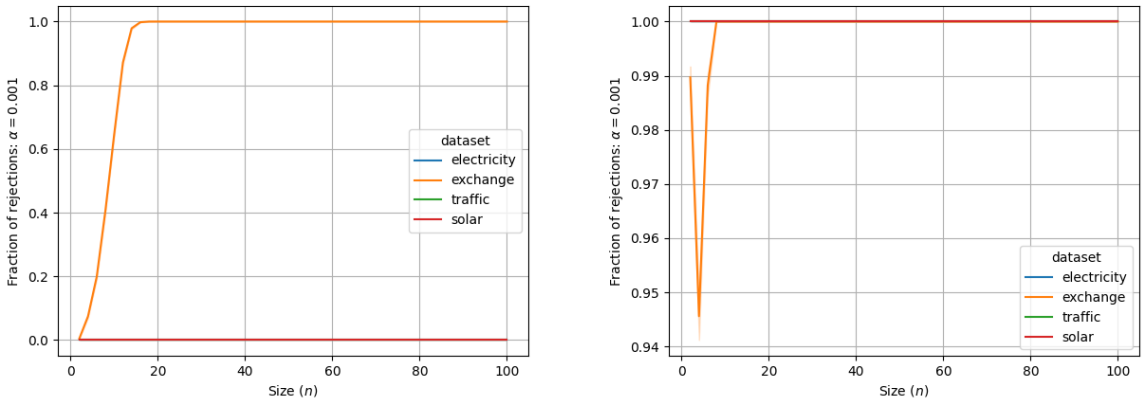


Figure 25: An adversarial trying to prove that DeepState performs better than N-BEATS using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

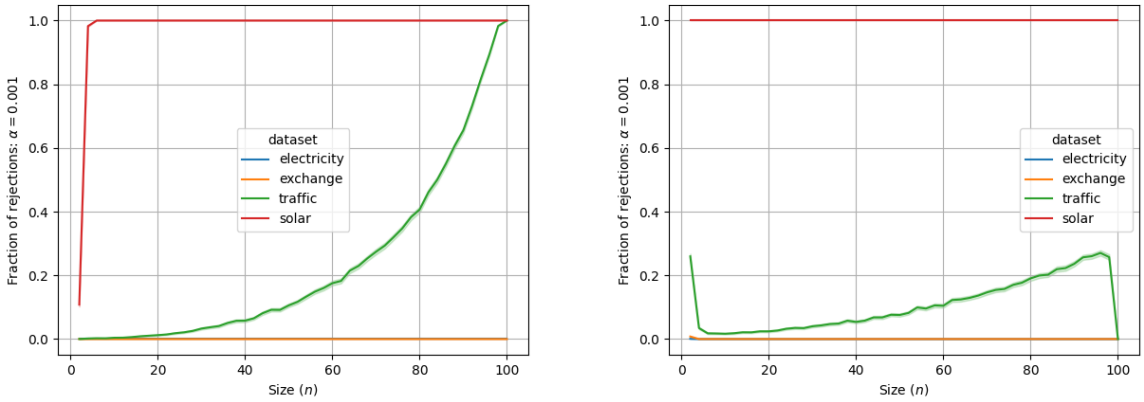


Figure 26: An adversarial trying to prove that LSTM performs better than DeepAR using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

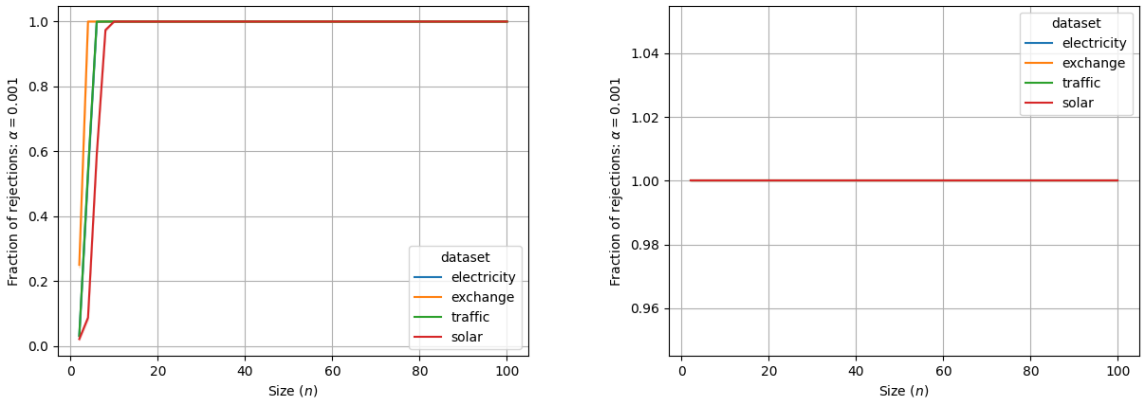


Figure 27: An adversarial trying to prove that LSTM performs better than DeepFactor using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

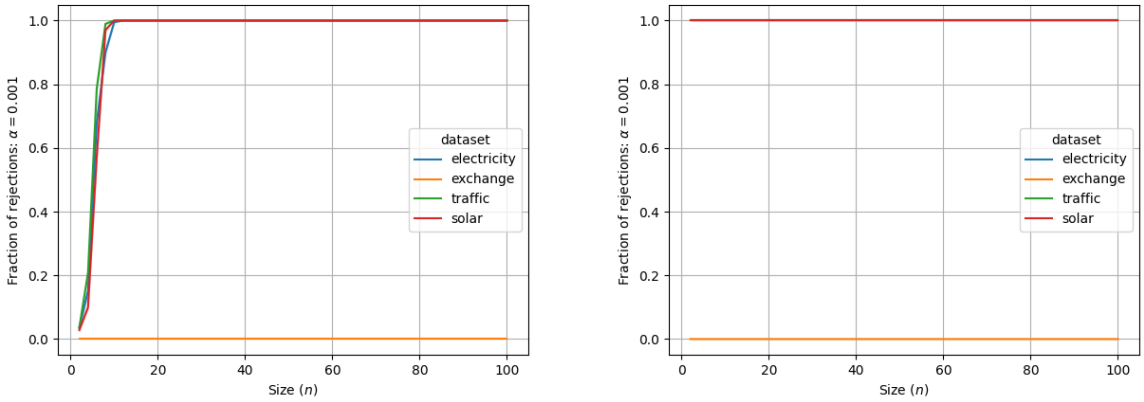


Figure 28: An adversarial trying to prove that LSTM performs better than DeepState using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

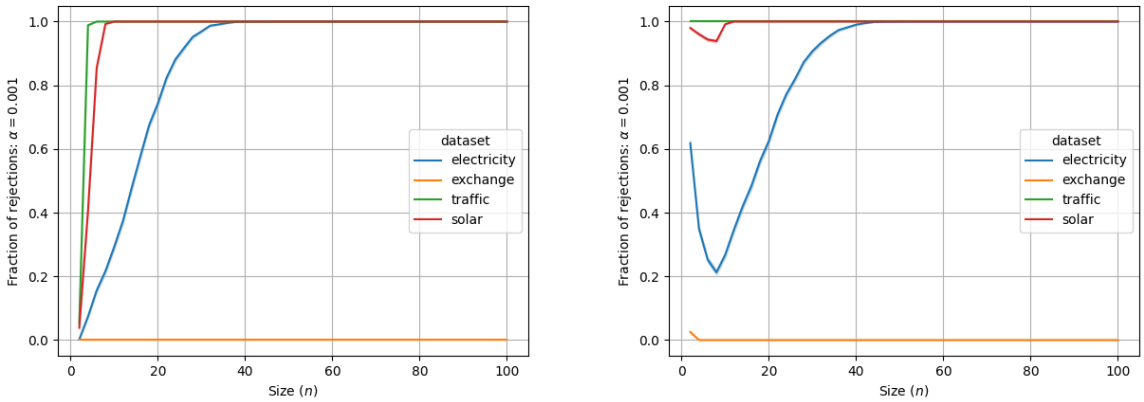


Figure 29: An adversarial trying to prove that LSTM performs better than N-BEATS using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

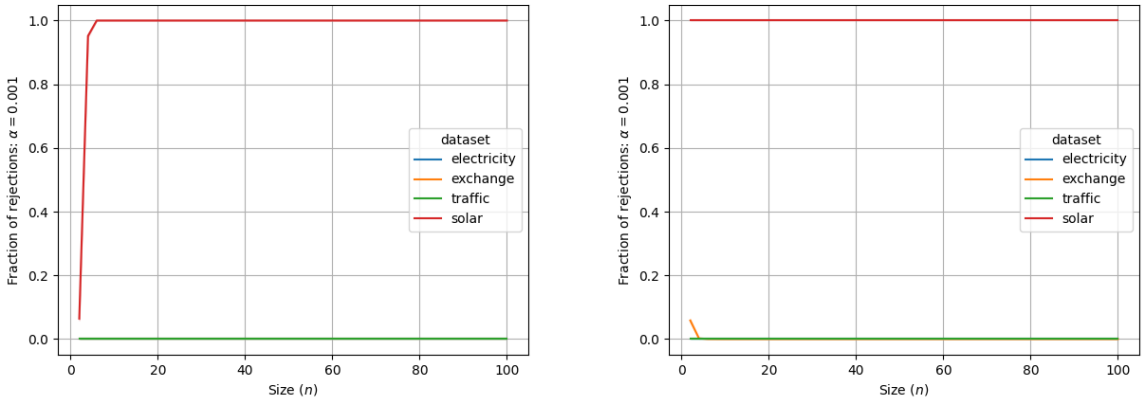


Figure 30: An adversarial trying to prove that N-BEATS performs better than DeepAR using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

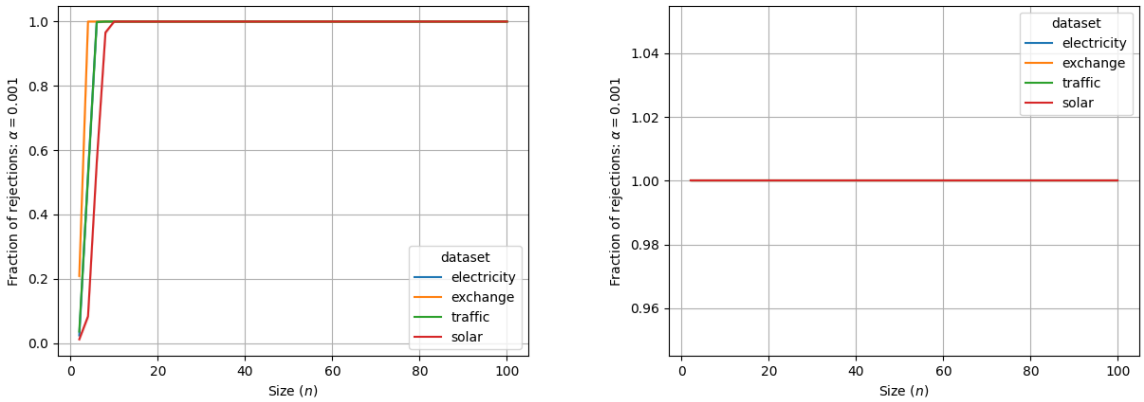


Figure 31: An adversarial trying to prove that N-BEATS performs better than DeepFactor using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

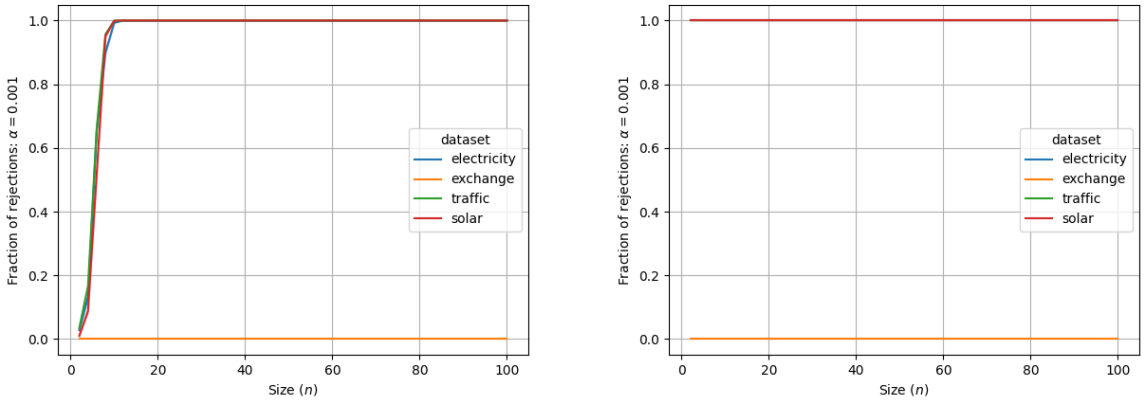


Figure 32: An adversarial trying to prove that N-BEATS performs better than DeepState using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.

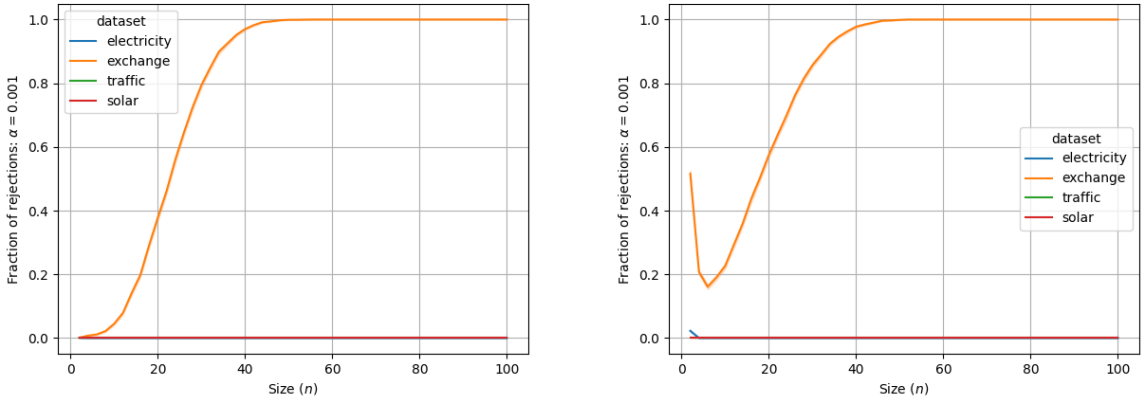


Figure 33: An adversarial trying to prove that N-BEATS performs better than LSTM using a one-sided test ($\alpha = 10^{-3}$). The same seeds are used for both the preferred model and the baseline. Left: Instead of opportunistically selecting $\mathcal{S}$, the set of seeds is a sample of size n sampled without replacement from the possible seeds $\{0, \ldots, 99\}$, and ensuring that the same set $\mathcal{S}$ was used for both models. The sampling of $\mathcal{S}$ was repeated 10.000 times, and the fraction of samples that lead to rejection is given on the y-axis. Results obtained using a t-test. Right: Exactly the same setup as in the left panel, but now using a Brunner-Munzel test.