Adaptive Sampling for Probabilistic Forecasting under Distribution Shift

Luca Masserano[*]
Department of Statistics
Carnegie Mellon University
lmassera@andrew.cmu.edu

Syama Sundar Rangapuram
AWS AI Labs
rangapur@amazon.com

Shubham Kapoor
AWS AI Labs
kapooshu@amazon.com

Rajbir Singh Nirwan[*]
D2S Inc.
rajbir.nirwan@gmail.com

Youngsuk Park
AWS AI Labs
pyoungsu@amazon.com

Michael Bohlke-Schneider
AWS AI Labs
bohlkem@amazon.com

Abstract

The world is not static: This causes real-world time series to change over time through external, and potentially disruptive, events such as macroeconomic cycles or the COVID-19 pandemic. We present an adaptive sampling strategy that selects the part of the time series history that is relevant for forecasting. We achieve this by learning a discrete distribution over relevant time steps by Bayesian optimization. We instantiate this idea with a two-step method that is pre-trained with uniform sampling and then training a lightweight adaptive architecture with adaptive sampling. We show with synthetic and real-world experiments that this method adapts to distribution shift and significantly reduces the forecasting error of the base model for three out of five datasets.

1 Introduction

Time series forecasting uses historical data to forecast the evolution of a time series to assist decision making in several domains, for example retail [24, 5, 32], electric load planning [30, 13], cloud computing management [25, 26], or labor planning [3]. Most time series forecasting algorithms make the assumption that the data distribution does not change over time. However, this is violated in practice where real-world time series data is affected by drifts such as evolving customer demand or disruptive events (like the COVID-19 pandemic). These distribution shifts will inevitably be part of the time series history and therefore forecasting algorithms need to be able to account for them.

Distribution shifts can occur with different (possibly reoccurring) patterns, but likely result in higher test error because the train and test distribution differs. Thus, only a subset of the time series history might come from the same distribution as the test set. Driven by this observation, we propose an *adaptive sampling mechanism* that explicitly feeds the model with inputs from the history that are only relevant for forecasting in the current (most recent) distribution. Instead of sampling uniformly over the history, our method learns a discrete distribution over past time steps and explicitly allows continuous adaptation, so that the model can react and adapt to distribution shifts.

Our contribution is an adaptive sampling mechanism that learns the sampling distribution over time steps via gradient-free Bayesian optimization. We propose a two-step framework where the forecasting model is pre-trained with uniform sampling and then a lightweight architecture is trained with adaptive sampling. We evaluate the properties and performance of this approach with experiments on synthetic and real-world data. This paper is organized as follows: Section 2 introduces the reader

Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

[*]Work done while at AWS AI Labs

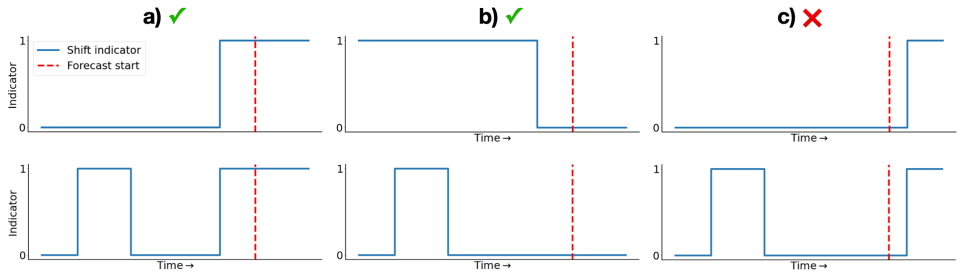

Figure 1: **Data distribution shift that our model can address.** Each frame shows an indicator function plotted over time with respect to an assumed "standard" distribution (a value of 1 indicates a distribution shift). The red dashed line separates the past (i.e., the training set) from the future (i.e. test set). **a)** New or reoccurring shift: the distribution over time steps should focus on the shifted regions. **b)** This case is complementary to the previous one: here the distribution over time steps should exclude the shifted regions. **c)** The shift occurs in the forecasting window but not in the most recent region before the forecast start date: our model needs to wait until part of the shift is in the training set in order to be able to adapt to it.

to the background of this paper and the types of data distribution shifts for forecasting that we address. and Section 3 introduces our method and we present experiments in Section 4. Section 5 concludes the paper. Note that the discussion of the related work can be found in Section A.1 in the Appendix.

2 Background

Notation We denote the value of time series i at time t by $z_{i,t}$, where $z_{i,t} \in \mathbb{R}$, and corresponding time-varying covariates as $\mathbf{x}_{i,t} \in \mathbb{R}^d$. The past of each time series (i.e., training set) is denoted as $z_{i,1:T} \coloneqq (z_{i,1}, \dots, z_{i,T})$, and the future (i.e., forecasting window or test set) as $z_{i,T+1:T+\tau} \coloneqq (z_{i,T+1}, \dots, z_{i,T+\tau})$. The number of time series in the dataset is denoted by N.

Our main assumption in this work is that we are able to observe at least some of the forthcoming shift in order to be able to react and adapt to it:

Assumption 1 *Let c be a constant number of (past) time steps set a priori. We assume that there is no (marginal) distribution shift between the most recent training window of c steps and the forecasting (test) window. More precisely, let $z_{i,t}$ be the given time series at time t and let $\bar{z}_{i,t}$ be the time series at time t obtained after removing trend, seasonalities and other effects of available features, then*

$$p(\bar{z}_{i,T-c}) = \cdots = p(\bar{z}_{i,T}) = p(\bar{z}_{i,T+1}) = \cdots = p(\bar{z}_{i,T+\tau}), \quad \forall i = 1, \dots, N. \tag{1}$$

This assumption implies that the data with $t \in [T-c, T]$ is the closest in time to the forecasting window, therefore it is reasonable to regard it as the most similar to the future. In practice, we take c to be equal to the sum of context and prediction lengths, so that we can use $\{z_{i,T-c:T}\}_{i=1}^{N}$ as a validation set to learn and adapt to any distribution shift. In general, we consider the following two scenarios illustrated in Figure 1): **(a)** A new or reoccurring shift in the past and no shift between the recent training and test window; and **(b)** a shift that occurred in the past but reverted back to the "standard" regime. Note that we do not address the case where the distribution shift occurs in the test window only, which would be more of a fundamental generalization problem (case **(c)** in Figure 1). See Section A.2 in the Appendix for a detailed definition of the cases that we consider.

3 Our Method

Training of deep learning models in time series settings is usually done by sampling fixed-size windows *uniformly at random* over the available time series history (i.e., the past). Since distribution shifts happen with different (possibly reoccurring) patterns over time, uniform sampling over the training set would likely feed the algorithm with irrelevant or even misleading data to forecast in the current distribution regime. Therefore we propose to adaptively generate these training windows depending on the regime of the validation set (and by Assumption 1, the test set). The key element of our approach is automatically learning which regions to focus on to accurately forecast the test set.

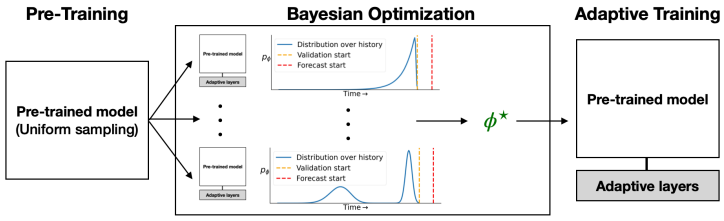

Figure 2: **Schematic description of the two-step procedure. i)** Pre-train the full model with uniform sampling; **ii.a)** use Bayesian optimization to find the optimal $\phi^\star$; **ii.b)** use $\phi^\star$ to train the lightweight architecture with adaptive sampling.

To achieve this, we introduce a discrete distribution $p_{\phi}(\cdot)$ over time steps $1, \ldots, T$ that determines the probability of sampling a training window starting at a time point t. In practice, $p_{\phi}(\cdot)$ can be any discrete distribution that is well defined over integers, such as Geometric, Negative Binomial, Poisson, or mixtures of these distributions, parameterized by ϕ. This distribution is responsible for sampling time windows that are used to train our forecasting model (see Figure A.1 in the Appendix for an illustration). We learn ϕ by minimizing validation loss of the model that is trained using examples generated according to $p_{\phi}(\cdot)$. Ideally, the learned distribution parameters $\phi^\star$ will avoid sampling distribution shifts in history that do not correspond to the current regime. This distribution replaces uniform sampling over the history, which cannot distinguish between distribution shifts and normal regions. However, our model *should* fall back to uniform sampling if this results in lowest validation error. This suggests that no distribution shift is present in the validation window and consequently in the forecast window (by Assumption 1).

Adaptive training of a lightweight architecture In practice, we would expect to frequently re-estimate $\phi^\star$ to adapt to the current distribution. To address this, we propose a two-step procedure with a lightweight adaptive architecture that is fast to train and can be frequently re-trained. In this sense, this approach is explicitly considering a notion of time-varying parameters. **i)** The model is pre-trained using uniform sampling over the whole training history. Intuitively, this step aims at learning common features that are useful everywhere in time. **ii)** After freezing the weights of the pre-trained model, an adaptive lightweight architecture is attached and trained using the adaptive sampling mechanism, where the training windows are generated according to $p_{\phi^*}(\cdot)$. The goal of this step is to learn time-specific features that are relevant to forecast in the current distribution. The lightweight architecture can be a short sequence of fully connected layers, or it can even be a subset of the pre-trained model.

Learn ϕ via Bayesian Optimization We now describe how we estimate the parameters ϕ of the distribution $p_{\phi}(\cdot)$. Let us denote the parameters of the pre-trained model as $\mathbf{w}_{\text{pre}}$ and those of the lightweight architecture as $\mathbf{w}_{\text{ada}}$. Moreover, let $f_{\mathbf{w}}(\{z_{i,1:T}\})$ denote the forecasts $\{\hat{z}_{i,T+1:T+\tau}\}$ of the model whose weights are given by $\mathbf{w}$. Given input time series $\{z_{i,t}\}, i = 1, \ldots, N$ and $t = 1, \ldots, T$, we set aside the last τ observations $\{z_{i,T-\tau+1:T}\}, \forall i$, for estimating ϕ. We learn ϕ by minimizing loss on this validation channel of the model trained according to $p_{\phi}(\cdot)$. More precisely, we have

$$\phi^* = \operatorname*{argmin}_{\phi} \ell(\phi),$$

where $\ell(\phi)$ is the validation loss of the model that is trained using adaptive sampling $p_{\phi}(\cdot)$,

$$\ell(\phi) = \texttt{loss}\Big(f_{\mathbf{w}_{\text{pre}} \cup \mathbf{w}^*_{\text{ada}}(\phi)}\big(\{z_{i,1:T-\tau}\}\big), \{z_{i,T-\tau+1:T}\}\Big).$$

Here $\mathbf{w}^*_{\text{ada}}(\phi)$ are the weights obtained after fine-tuning the pre-trained model on the input data $\{z_{i,1:T-\tau}\}$ where the training examples are generated according to the distribution $p_{\phi}(\cdot)$; note the dependence of $\mathbf{w}^*_{\text{ada}}$ on ϕ. A crucial observation is that one can only evaluate $\ell(\phi)$ and cannot obtain its gradients with respect to ϕ; hence we cannot train $(\mathbf{w}_{\text{ada}}, \phi)$ jointly end-to-end. We propose to treat ϕ as hyper-parameters and use Bayesian optimization [9] to find ϕ^*. Assumption 1 plays a key role in this step because the most recent region in the training set is used as validation channel to choose $\phi^\star$ during Bayesian optimization. The overall approach is depicted in Figure 2 and the implementation details with `DeepAR` [28] as the base model are given in Algorithm 1. In our experiments we assume $\mathbf{w}_{\text{ada}} \subset \mathbf{w}_{\text{pre}}$ and hence the algorithm is presented for this case.

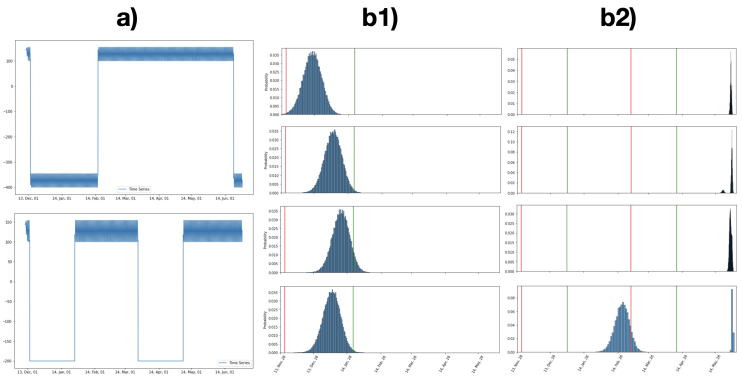

Figure 3: **Experiments with synthetic data.** **a**) Synthetic data with injected shift (top) and noise (bottom). **b1)** (top case in **a**): Adaptive sampling focuses on the shifted region in the past (red and green bars denote the shifted region) that is relevant for the current regime. **b2** (bottom case in **a**): Adaptive sampling avoids the injected noise shifts.

Algorithm 1 Adaptive DeepAR

procedure ADA-DEEPAR($\{z_{i,t}\},\ i \in [1\ ..\ N],\ t \in [1\ ..\ T]$)
  $Z_{\text{train}} = \{z_{i,t}\}, \forall i,\ \forall t$
  $\mathbf{w}_{\text{pre}} \leftarrow$ DeepAR(Z_{train}) ▷ Standard training with uniform train sampler
  $Z_{\text{ada\_train}} = \{z_{i,t}\},\ \forall i,\ t \in [1\ ..\ T-\tau]$
  $Z_{\text{validation\_labels}} = \{z_{i,t}\},\ \forall i,\ t \in [T-\tau+1\ ..\ T]$
  $\boldsymbol{\phi} = \boldsymbol{\phi}_0$ ▷ Random initialization
  $\mathbf{w} = \mathbf{w}_{\text{pre}}$
  Split $\mathbf{w}$ into $\mathbf{w}_{\text{ada}}$ and $\mathbf{w}_{\text{frozen}}$
  while CONVERGENCE **do** ▷ Solve $\min_{\boldsymbol{\phi}} \ell(\boldsymbol{\phi})$
    $\mathbf{w}_{\text{ada}} \leftarrow$ DeepAR($Z_{\text{ada\_train}}$, init $= \mathbf{w}$, sampler $= p_{\boldsymbol{\phi}}(\cdot)$, fixed $= \mathbf{w}_{\text{frozen}}$) ▷ Train with adaptive train sampler by fixing weights $\mathbf{w}_{\text{frozen}}$
    $\mathbf{w} = \mathbf{w}_{\text{ada}} \cup \mathbf{w}_{\text{frozen}}$
    $\hat{Z}_{\text{validation\_preds}} = \text{forecast}(Z_{\text{ada\_train}}, \text{model} = \mathbf{w})$
    $\ell(\boldsymbol{\phi}) = \text{loss}(\hat{Z}_{\text{validation\_preds}}, Z_{\text{validation\_labels}})$
    update $\boldsymbol{\phi}$ based on the new evaluation $\ell(\boldsymbol{\phi})$ ▷ Bayesian Optimization
  end while
  ▷ $p_{\boldsymbol{\phi}}(\cdot)$ likely has more mass on indices t that help predict $Z_{\text{validation\_labels}}$ accurately
  return $\mathbf{w}$ ▷ Model is ready to predict $\{z_{i,t}\},\ \forall i,\ t \in [T+1\ ..\ T+\tau]$
end procedure

4 Experiments

The instantiation of our model in this manuscript is called Ada-DeepAR and uses DeepAR [28] as a base model with a two-layer LSTM (other base models could be used). The adaptive part of our model is the second LSTM layer. Additionally, we perform Bayesian optimization using MOBSTER, which combines asynchronous successive halving and asynchronous Hyperband with gaussian-process based Bayesian optimization [17]. We use the implementation available in Syne Tune [29]. For any time series i with training history T and forecast horizon τ, we use $z_{i,T-\tau:T}$ as a validation channel for Bayesian optimization. We consider a geometric and a mixture of two negative binomial distributions as our sampling distributions $p_{\boldsymbol{\phi}}(\cdot)$. Training details are given in Section A.4.

Experiments on synthetic data: To demonstrate the applicability of our method, we first evaluate it on two synthetic experiments that mimic cases **(a)** and **(b)** of Figure 1. We generated an artificial sinusoidal dataset and then injected shifts or noise in different regions. **Focus on shifted regions distant in time:** In this example we injected a shift in a distant portion of the training dataset and also in the validation and test sets (see the top of panel **a)** in Figure 3). We show that the sampling distribution over time steps can focus on specific regions that are relevant to the current domain, regardless of when they occurred in the available history (see **b1)** in Figure 3). **Exclude noisy regions:** In this example, we injected noise (a negative constant) in two different regions of the training dataset. Our adaptive sampling mechanism avoids these noisy regions from the history (**b2)** in Figure 3). Interestingly, this sometimes corresponds to sampling only in the most recent window, while in other cases the distribution exploits its bimodality to focus on two distant windows.

Dataset	Ada-DeepAR (ours)	Ada-DeepAR-uniform (ablation)	RevIn	DeepAR	TFT
Covid-deaths	$0.074^{*}\pm0.013$	0.088 ± 0.011	0.13 ± 0.006	0.103 ± 0.053	$\mathbf{0.053}\pm0.061$
Electricity	$\mathbf{0.048}^{*\dagger}\pm0.002$	$0.054^{\dagger}\pm0.005$	0.063 ± 0.005	0.062 ± 0.011	0.082 ± 0.000
Traffic	0.172 ± 0.002	0.171 ± 0.004	0.172 ± 0.003	0.176 ± 0.004	0.306 ± 0.050
Solar	$\mathbf{0.370}^{*\dagger}\pm0.003$	$0.376^{\dagger}\pm0.003$	0.376 ± 0.004	0.388 ± 0.007	0.562 ± 0.015
Taxi	$0.286^{\dagger}\pm0.003$	$\mathbf{0.278}^{*\dagger}\pm0.001$	0.288 ± 0.004	0.279 ± 0.004	0.509 ± 0.001
Mean nCRPS	0.190	0.193	0.202	0.206	0.302
Mean Rank	1.8	1.8	3.4	3.4	4.2

Table 1: The nCRPS loss (lower is better) for all datasets and models. We average over ten runs and report the mean and standard deviation. We took the results for TFT from TSBench [4], which only uses two runs. Boldface numbers indicate that the differences in error between the best and the second-best method are statistically significant (paired t-test p-value < 0.05). * indicates a statistically significant result when comparing Ada-DeepAR and Ada-DeepAR-uniform. † indicates a statistically significant result when comparing Ada-DeepAR or Ada-DeepAR-uniform to its base model DeepAR.

Experiments on real-world datasets: Finally, we show preliminary results on five real-world datasets (see Section A.3 for details). We compare the following models: our two-step training procedure with adaptive sampling (Ada-DeepAR), reversible instance normalization (RevIn) [16], and the attention-based Temporal Fusion Transformer TFT [20]. We also consider an ablation variant of our method, Ada-DeepAR-uniform, which uses the same architecture and two-step procedure as Ada-DeepAR, but performs adaptive training (i.e., fine-tuning of the adaptive weights $\mathbf{w}_{\text{ada}}$) using uniform sampling instead of $p_{\boldsymbol{\phi}}(\cdot)$. This variant is included to highlight the effectiveness of learning the sampling distribution $p_{\boldsymbol{\phi}}$ by isolating the gains obtained by our model due to the fine-tuning step. For Ada-DeepAR, Ada-DeepAR-uniform and RevIn we use DeepAR[28] as the base model. Ada-DeepAR-uniform is different from DeepAR because of the fine-tuning step where adaptive weights are again trained one more time; this might help correcting the overfitting of the DeepAR model (and reduce test error) because of the lower capacity of adaptive weights and/or help escape local minima (and reduce training loss). We evaluate these probabilistic methods using the continuous-ranked probability score (CRPS) [11, 23]. Table 1 summarizes the results. Averaged over all datasets, Ada-DeepAR improves over the base model DeepAR by 8.4% (relative improvement). For three out of five datasets, the improvements over the base model are statistically significant (paired Student's t-test p-value < 0.05, see Table 1). Ada-DeepAR selected the mixture of two negative binomial distributions for all datasets except Covid-deaths (for which the geometric distribution was selected). RevIn also improves over DeepAR, but with a much smaller margin. TFT performs best on Covid-deaths, but overall worse than the other methods on these datasets. Interestingly, Ada-DeepAR-uniform also improves the error by 6.7% over its base model. In particular, Ada-DeepAR-uniform is the best model on the Traffic and Taxi datasets. Our hypothesis is that adaption to distribution shift is not necessary, and perhaps even counterproductive, for these datasets. This is also suggested by the performance of RevIn, which performs similar to Ada-DeepAR on these datasets. Note that a uniform distribution could be seen as one of the possible distributions $p_{\boldsymbol{\phi}}(\cdot)$ that our adaptive architecture should consider and if no other distribution reduces the validation error, the uniform distribution *should* be selected. The performance differences between Ada-DeepAR and Ada-DeepAR-uniform are statistically significant for all datasets except Traffic.

5 Conclusion

We introduced an adaptive sampling mechanism over the training history, whose parameters are learned using Bayesian optimization. Our experiments show that our method can improve the forecasting error over its base model (in this work, the improvements over the base model where statistically significant for three out of five datasets). While we find Bayesian optimization to be a powerful technique to find good adaptive sampling parameters, this approach is computationally expensive (because many parallel trials need to be evaluated) and could degrade as the number of parameters increases. Re-framing our approach as an input-selection layer that uses the re-parametrization trick for sampling could allow for cheaper end-to-end learning of the adaptive model parameters and $p_{\boldsymbol{\phi}}(\cdot)$.

References

[1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Sundar Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic and neural time series modeling in python. *J. Mach. Learn. Res.*, 21(116):1–6, 2020.

[2] Sercan O Arik, Nathanael C Yoder, and Tomas Pfister. Self-adaptive forecasting for improved deep learning on non-stationary time-series. *arXiv preprint arXiv:2202.02403*, 2022.

[3] Michael Bohlke-Schneider, Shubham Kapoor, and Tim Januschowski. Resilient neural forecasting systems. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, DEEM'20, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380232. doi: 10.1145/3399579.3399869. URL https://doi.org/10.1145/3399579.3399869.

[4] Oliver Borchert, David Salinas, Valentin Flunkert, Tim Januschowski, and Stephan Günnemann. Multi-objective model selection for time series forecasting, 2022. URL https://arxiv.org/abs/2202.08485.

[5] J. D. Croston. Forecasting and stock control for intermittent demands. *Journal of the Operational Research Society*, 23(3):289–303, Sep 1972. ISSN 1476-9360. doi: 10.1057/jors.1972.50. URL https://doi.org/10.1057/jors.1972.50.

[6] Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5):533–534, May 2020. ISSN 1473-3099. doi: 10.1016/S1473-3099(20)30120-1. URL https://doi.org/10.1016/S1473-3099(20)30120-1. Publisher: Elsevier.

[7] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 402–411, 2021.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[9] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[10] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1): 1–28, 1985.

[11] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

[12] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[13] Tao Hong, Jingrui Xie, and Jonathan Black. Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1389 – 1399, 2019.

[14] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper/2006/file/a2186aa7c086b46ad4e8bf81e2a3a19b-Paper.pdf.

[15] Rolf Isermann. *Adaptive Control Systems (A Short Review)*, pages 127–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991. ISBN 978-3-642-86420-9. doi: 10.1007/978-3-642-86420-9_12. URL https://doi.org/10.1007/978-3-642-86420-9_12.

[16] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

[17] Aaron Klein, Louis C. Tiao, Thibaut Lienart, Cedric Archambeau, and Matthias Seeger. Model-based asynchronous hyperparameter and neural architecture search, 2020. URL `https://arxiv.org/abs/2003.10865`.

[18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015, 2017. URL `http://arxiv.org/abs/1703.07015`.

[19] Fengpei Li, Henry Lam, and Siddharth Prusty. Robust importance weighting for covariate shift. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 352–362. PMLR, 26–28 Aug 2020. URL `https://proceedings.mlr.press/v108/li20b.html`.

[20] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4): 1748–1764, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL `https://www.sciencedirect.com/science/article/pii/S0169207021000637`.

[21] Linbo Liu, Youngsuk Park, Trong Nghia Hoang, Hilaf Hasson, and Jun Huan. Towards robust multivariate time-series forecasting: Adversarial attacks and defense mechanisms. *arXiv preprint arXiv:2207.09572*, 2022.

[22] Yucheng Lu, Youngsuk Park, Lifan Chen, Yuyang Wang, Christopher De Sa, and Dean Foster. Variance reduced training with stratified sampling for forecasting models. In *International Conference on Machine Learning*, pages 7145–7155. PMLR, 2021.

[23] James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976. ISSN 00251909, 15265501. URL `http://www.jstor.org/stable/2629907`.

[24] Srayanta Mukherjee, Devashish Shankar, Atin Ghosh, Nilam Tathawadekar, Pramod Kompalli, Sunita Sarawagi, and Krishnendu Chaudhury. ARMDN: Associative and recurrent mixture density networks for retail demand forecasting. *arXiv preprint arXiv:1803.03800*, 2018.

[25] Youngsuk Park, Kanak Mahadik, Ryan A Rossi, Gang Wu, and Handong Zhao. Linear quadratic regulator for resource-efficient cloud services. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 488–489, 2019.

[26] Youngsuk Park, Ryan Rossi, Zheng Wen, Gang Wu, and Handong Zhao. Structured policy iteration for linear quadratic regulator. In *International Conference on Machine Learning*, pages 7521–7531. PMLR, 2020.

[27] Dominique Picard. Testing and estimating change-points in time series. *Advances in Applied Probability*, 17(4):841–867, 1985. doi: 10.2307/1427090.

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

[29] David Salinas, Matthias Seeger, Aaron Klein, Valerio Perrone, Martin Wistuba, and Cedric Archambeau. Syne tune: A library for large scale hyperparameter tuning and reproducible research. In *First Conference on Automated Machine Learning (Main Track)*, 2022.

[30] Harshit Saxena, Omar Aponte, and Katie T. McConky. A hybrid machine learning model for forecasting a billing period's peak electric load days. *International Journal of Forecasting*, 35 (4):1288 – 1303, 2019. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.03.025. URL `http://www.sciencedirect.com/science/article/pii/S016920701930144X`.

[31] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper.pdf.

[32] Elham Taghizadeh. Utilizing artificial neural networks to predict demand for weather-sensitive products at retail stores. *arXiv preprint arXiv:1711.08325*, 2017.

[33] NYC Taxi and Limousine Commission. TLC trip record data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, 2015.

[34] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dj Dvijotham, and Ali Taylan Cemgil. A fine-grained analysis on distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Dl4LetuLdyK.

[35] TaeHo Yoon, Youngsuk Park, Ernest K Ryu, and Yuyang Wang. Robust probabilistic time series forecasting. In *International Conference on Artificial Intelligence and Statistics*, pages 1336–1358. PMLR, 2022.

A Appendix

A.1 Related Work

The problem of learning under distribution shift for time series has been extensively studied for many years. For example, change point detection analyzes when the statistical distribution of a time series changes [27]. Another related field is adaptive control, where the system parameters are continuously adjusted to account for discrepancies of expected and observed data [15]. We focus our review on recent methods that address distribution shifts when forecasting with neural networks. Kim et al. [16] employs a trainable instance-wise normalization-and-denormalization layer to address non-stationary mean and variance in time series forecasting. Arik et al. [2] proposes Self-Adaptive Forecasting to adapt the forecasting model using test-time training by "backcasting" and using the backcast errors to signal a potential distribution shift to adjust the weights of the model before inference. AdaRNN [7] splits the time series history into dissimilar segments and learns importance weights to combine the RNN hidden states over these segments. Another notion of distribution shift, adversarial attacks, has been recently considered to build more robust forecasting [35, 21].

Another class of approaches to time series forecasting use specific weighting schemes to weight the time series history to make predictions, for example exponential smoothing, the Non-Parametric Time Series (NPTS) Forecaster [10, 1], or stratified sampling [22]. In contrast to these models, our approach adaptively learns the weighting scheme over the history. Thus, our work can be seen as a specific instantiaten of weighted re-sampling (or importance sampling) for time-series forecasting. Importance sampling weights the log-likelihood in supervised learning tasks by importance weights $w(x) = p_{test}(x)/p_{train}(x)$ where p_{test} and p_{train} are the respective densities from which the train/test samples are drawn [31, 14, 34, 19]. Here, we propose to avoid estimating p_{train} and p_{test} and instead tune the parameters of a parametric re-sampling function using Bayesian optimization.

A.2 Scenarios of distribution shift occurrence

Assumption 1 allows us to provide a precise breakdown of the specific cases of distribution shift that our model can handle. This is important because it determines the exact applicability of the work described in this paper. As Figure 1 shows, we identify three domains:

New or reoccurring shift: When a distribution shift is new or reoccurring in the past and there is no shift between $t \in [T - c, T]$ and $t \in [T + 1, T + \tau]$. In this case the adaptive sampling mechanism is expected to focus only on the shifted regions that are relevant for the current regime.

Reverted shift: When a shift occurred in the past but the distribution eventually reverted back to the "standard" regime. This case is complementary to the previous one: here the adaptive sampling mechanism provides a way to exclude the shifted regions. One can see the shifts in this case as regions with noisy or corrupted data: excluding them during training would guarantee robustness against noise.

Future shift: When the shift occurs in the forecasting window but not in the most recent region before the forecast start date. Our adaptive sampling mechanism cannot work if we have no information regarding the distribution shift happening in the test set (no free lunch).[2]

A.3 Datasets

We use datasets from the Monash Time Series Forecasting Repository [12] and GluonTS [1] for our experiments. We use the same train/test splits as described in TSBench [4]. See Table A.1 for dataset statistics. The frequencies of our used datasets are 30 minutes (30 MIN), hourly (H), and daily (D).

In particular, we use the following datasets:

`Covid-deaths`: Daily COVID-19 deaths of different countries between January 22 and August 21 2020 [6].

[2]Note that in our setting we consider the forecasting window to be a mere continuation of the training set, which implies that the context window used at test time was part of training inputs. In this sense, our procedure cannot handle this case if and only if the shift happens solely in the forecasting range.

	Freq.	Horizon	Number of Series	Avg. Length	Number of observations
`Covid-deaths`	D	30	227	182	41,314
`Electricity`	H	24	321	21,044	6,755,124
`Traffic`	H	48	862	17,496	15,081,552
`Solar`	H	24	137	7,009	960,233
`Taxi`	30 MIN	24	1,214	1,488	1,806,432

Table A.1: Dataset statistics for the used datasets, taken from Borchert et al. [4].

`Electricity`: Hourly household consumption data between January 2012 and June 2014 [8].

`Traffic`: Hourly occupancy rates of freeways in the San Francisco Bay area between 2015 and 2017 [8].

`Solar`: Hourly power output of 137 photovoltaic power stations in the US [18].

`Taxi`: Number of taxi rides in different locations in New York sampled at 30 minute windows [33].

A.4 Training Details

The parameters of the distribution over history are optimized via Bayesian Optimization [9] using the recent scalable implementation provided in `Syne-Tune` [29]. We leveraged an AWS EC2 instance with ≈ 42 cores and set the number of random initializations to 44 to allow for sufficient exploration of the parameter space. That is, the optimizer starts with 44 trials, each with a different parameter configuration, and sequentially stops some of them or instantiate new ones given the information from the Gaussian Process posterior. The optimizer stopping criterion was chosen to be the maximum number of trials taken to completion, which was set to 200. We used the CRPS used to decide which trials were stopped or continued was the mean quantile loss over the validation set.

The lightweight architecture is supposed to be simple enough to allow frequent re-training through adaptive sampling, so that the forecasting model can quickly adapt to new distributions. Clearly, different architectures will work better with different datasets. In all our experiments with `DeepAR` [28], we found that using the default two-layer LSTM with 40 hidden units for the pre-trained model, and freezing the first layer only for adaptive training yielded the best results. To clarify, this means that $\mathbf{w}_{\text{ada}} \subset \mathbf{w}_{\text{pre}}$. Both pre-training and adaptive training were done over 100 epochs with a small dropout probability equal to 0.1.

For `TFT`, we selected the hyperparameter setting from the TSBench [4] that resulted in lowest CRPS loss on the validation set for each dataset. The hyperparameters tuned in TSBench are the context length, the number of hidden states, and the number of heads.

For the `DeepAR` and `TFT` models, we use the implementation available in GluonTS [1].

A.5 Additional Figures

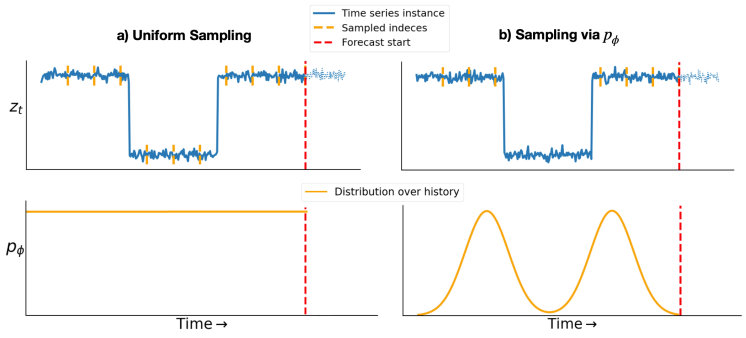

Figure A.1: **Adaptive sampling uses p_ϕ to sample training windows to come from distributions that are similar to the forecasting window.** Both panels show a time series (blue) at the top, with windows (orange) sampled during training, and the distribution over time steps at the bottom. **a)** Uniform sampling. **b)** Adaptive sampling.