UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Novin Shahroudi

Probabilistic Forecasting with Monte-Carlo Dropout in Neural Networks

Master's Thesis (30 ECTS)

Supervisor: Meelis Kull, PhD
Supervisor: Sebastian Haglund El Gaidi, MSc
Supervisor: Erik Ylipää, BSc

Tartu 2019

Probabilistic Forecasting with Monte-Carlo Dropout in Neural Networks

Abstract:
Integration of intelligent systems in our industries and society require more accurate and reliable algorithms. Recent attempts to model and explain uncertainty in deep learning has had many achievements, stepping forward toward making these models more reliable and respectively more practical. At the same time, probabilistic forecasting is a similar attempt to estimate the future value of a variable by a probabilistic expression rather a point estimate. The probabilistic forecasting is more useful as it incorporates uncertainty information on the forecast while the point forecast lacks this. To this day, there has not been much research or study on probabilistic forecasts with Bayesian deep learning, making it an interesting area for contribution. We considered this research to be more interesting when carried it out on a real-world problem and therefore, chose the wind power forecast beside an analytical study. We applied Monte-Carlo Dropout (MCDO), a variant of deep Bayesian network approximator, together with a network capable of estimating the variance. This predictive variance, produced by the model, is equivalent to a probabilistic forecast and further, we show how to obtain higher quality forecasts that are accurate and better-calibrated by scenario forecasting at test-time. We also assess whether the results of the MCDO implementation were consistent with other works. Our experiments show a successful use case of employing Bayesian deep learning and suggest that these methods are the way to go, with promises on improving the accuracy, scalability, and interpretability.

Keywords:
Probabilistic Forecasting, Bayesian Deep Learning, Wind Power Forecasting, Neural Network, Uncertainty

CERCS: P176 – Artificial intelligence

Tõenäosuslik prognoosimine Monte-Carlo väljajätumeetodil närvivõrkudes

Lühikokkuvõte:
Intelligentsete süsteemide integreerimine meie tööstusharudesse nõuab täpsemaid ja usaldusväärsemaid algoritme. Hiljutised katsed modelleerida ja selgitada sügavõppe ebakindlust on olnud edukad, astudes sammu edasi muutmaks need mudelid usaldusväärsemaks ja seega praktilisemaks. Samal ajal on tõenäosuslik prognoosimine sarnane katsele hinnata muutuja tulevast väärtust tõenäosusliku väljendusega mitte punkthinnanguna. Esimene on kasulikum, sest erinevalt punktihinnagust sisaldub selles ebakindluse informatsioon. Tänase päevani pole tehtud veel kuigi palju uuringuid tõenäosusliku prognoosimise kohta kasutades Bayesi süvaõpet, mistõttu on see huvitav valdkond mille uurimisse panustada. Meie arvates teeb uuringu reaalmaailma probleemiga sidumine selle huvitavamaks ja seetõttu valisime analüütilise uuringu kõrvale tuuleenergia prognoosimise. Me kohaldasime Monte Carlo väljajätumeetodi (MCDO) võrgustikku, mis on variant Bayesi süvavõrgu lähendajast, koos võrguga, mis on võimeline hindama dispersiooni. See prognoositav dispersioon, mis on toodetud mudeli abil, annab tõenäosusliku prognoosi ja lisaks näitame, kuidas saada kõrgema kvaliteediga prognoose, mis on täpsed ja paremini kalibreeritud testimisaegse stsenaariumite prognoosimisega. Samuti uurime, kas MCDO rakendamise tulemused on kooskõlas teiste töödega. Veelgi enam, meie lähenemisviis osutus võimeliseks looma kõrge kvaliteediga tõenäosuslike prognoose ning seda on üsna lihtne seadistada. Meie katsed näitavad edukat Bayesi süvaõppe rakendamist ning viitavad sellele, et need meetodid on lootustandvad, lubades parandada täpsust, skaleeritavust ja tõlgendatavust.

Võtmesõnad:
Tõenäoline prognoosimine, Bayesi süvaõpe, tuuleenergia prognoosimine, närvivõrk, ebakindlus

CERCS: P176 – Tehisintellekt

Acknowledgement

I would like to thank my thesis supervisors Prof. Dr. Meelis Kull of the *Insitute of Computer Science* at *University of Tartu*, Erik Ylipää of the *Swedish Institute of Computer Science* at *Research Institutes of Sweden (RISE)*, and Sebastian Haglund El Gaidi at *Greenlytics AB*. It would have not been possible for me to defend if it was not with the help and support of Dr. Meelis Kull. He steered me in the right direction whenever I needed it the most. I am also very grateful for the opportunity provided by the Greenlytics AB and RISE to conduct this thesis work in the area of wind power forecasting. Entrepreneurial attitude and environment that was prepared by Sebastian Haglund, added more motivation to my efforts. However in a short period of time, I learned a lot from Erik Ylipää and I appreciate his positive attitude, support, and involving me in the *Learning Machines Study Group*.

I would also like to thank my course mate Chia-Hsuan Chou, who had the patience to listen to my explanations and provided me a chance to develop some ideas by explaining these concepts. Also, thanks to Anett Hollas that helped me with my crippled Estonian to write the Estonian abstract of the thesis.

Finally, I must express my profound gratitude to my parents who were providing me the unconditional support and encouragement throughout all years of my studies and during researching and writing this thesis despite all the ups and downs. This accomplishment would have not been possible without them. Thank you

Novin Shahroudi
May, 2019

Contents

1 Introduction

It's all about "the future"...

Forecasting is all about the future, and whatever about the future comes with a degree of *uncertainty*. That is why a probabilistic expression of the future is better suited its remote and uncertain nature. Our confidence in a possible future would then be a function of its probability. Forecasting without sufficient consideration or ignorance about the underlying uncertainties which were buried in the observations and/or assumptions about the observations could lead to over/under confident results, which consequently appear like a mismatch of what is about to come true. That is because the future is not only derived from the past but also what could have happened in the past. Therefore, considerations regarding uncertainty is an integral and inseparable part of any forecast. Naturally with a higher degree of uncertainty, one could be less confident. All these matters, when put into action, i.e. in the context of decision making, where forecasts are being consumed to make decisions that shape the future. This being said, an underconfident forecast should be equally avoided as overconfident forecasts, as it translates to inconsistency in decision making that may even lead to more unpleasant consequences far worse than overconfidence.
Al Gore in his book *the future* [Gor13] writes:

> The cultural legacy that still influences the scientific method is reductionist – that is, by dividing and endlessly subdividing the objects of our research and analysis, we separate interconnected phenomena and processes to develop specialized expertise. But the focusing of attention on ever narrower slices of the whole often comes at the expense of attention to the whole, which can cause us to miss the significance of emergent phenomena that spring unpredictably from the interconnections and interactions among multiple processes and networks. That is one reason why linear projections of the future are so often wrong. [Gor13, p. xxi]

Thus, this thesis is an attempt to make a realization of a bigger whole by incorporating uncertainty information to make the projection(s) of the future less often wrong.

1.1 Uncertainty in machine learning and why should we care?

Machine Learning models and particularly *Deep Learning* are finding their way faster and on a broader scale than ever before in real-world use cases. They are going to be employed in more mission-critical tasks such as *self-driving cars* as well as tasks that may not require a safety measure but some measure of reliability such as in *stock market forecasting*. To this day, point estimation of the future has been the most common output

expected from these models. A single number is not informative enough, i.e. does not imply anything about the reliability of the estimation. Traditionally, assessing the *generalization* capability of a trained model has been one of the ways that its reliability could be put to a test. However, methods for assessing or improving generalization of the models such as *cross-validation*, *regularization techniques*, etc. are not sufficient to make the forecast itself more informative.

Moreover, there has been an ongoing trend and desirability to replace complex sub-intelligent systems with its end-to-end counterparts instead. This has been paced up and facilitated by deep learning models as they specifically provide a robust end-to-end scheme especially for cognitive tasks while outperforming their predecessors. However dealing with image, speech, text, and other complex formats of data especially in an end-to-end fashion require robust and capable models to deal with different aspects of the data and not only caring about the average cases where the model would only produce a point estimate of average cases. And more important of all is that the data is constantly changing in real-world scenarios, it has inherited noise, models have limitations, and many other factors contribute to not having a perfect forecast. The alternative is to account for all these by taking uncertainty into consideration and make the models uncertainty-aware, meaning that alongside the values of interest, the uncertainty should also be considered to be estimated. To put this preliminary into perspective, there are three main directions that are expected to achieve with this work by incorporating uncertainty:

- To improve the accuracy and robustness of the models
- To make models reliable for secondary systems e.g. control mechanisms
- To understand the models better by making them more expressive and interpretable

1.2 Renewable Energies and Electricity Power Forecast

There is no doubt that a measure of uncertainty could benefit every application domain that predictive models are being employed to. However, some domains depend on this information more than the other. Weather forecasting is among those areas that deal with a great amount of uncertainty as one tries to model a gigantic planetary system consisting of many different factors. [Gos96] It also has a long history and so a good case study for uncertainty. Likewise, forecasting electricity power generated from renewable energies heavily depends on environmental variables such as the weather. The recent challenge of integrating renewable energies in our energy sources brought research and engineering communities together as a collective effort towards climate crisis[1]. Data-intensiveness, dimensionality, complexity and other aspects of this challenge provide an opportunity

[1] `https://ec.europa.eu/clima/citizens/eu_en`

for machine learning practitioners as well as researchers to study, employ, and devise to methods to address this and its related challenges.

Over a decade or two, solar and wind power established better than the rest of their class and proved to be among the most promising renewable energy sources. The wind is much more ahead and also implemented very well in the Nordics. Over 25% of the electricity production in Sweden is provided by the wind power and that is planned to be doubled over a decade[2]. Similarly, Estonia has over 30% production of electricity from wind power [3]. Even in Iran with less coastline per its square meter area, it is estimated that wind can provide 50% of the total electricity consumption need. However, employing wind power and integrating it into the grid comes with its challenges [PT13]. Besides the engineering aspects of building and installing wind turbines, the intermittent nature of power generation from renewable sources such as wind makes it a constant challenge to balance the power grid. This makes accurate and reliable forecasts crucial for the operations and control mechanisms that sit on top of these forecasting models possible. Moreover, the economics of energy would heavily rely on these forecasts.
To this end, the contribution of this thesis work is as follows:

1. Study the relevance of uncertainty in deep learning with probabilistic forecasting
2. Employ Monte-Carlo Dropout technique
3. Empirical experiments and results on synthetic data
4. Empirical experiments and results on real data
5. Demonstrate a novelty to improve the results
6. Demonstrate usefulness of the developed models for wind power forecasting

1.3 Structure of the thesis

For the background an overview of time-series modeling and forecasting is given in Section 2, also an overview of Bayesian Deep Learning that is being used to model uncertainty and ultimately produce probabilistic forecasting is given in Section 3. Methods and models developed for this thesis are explained in Section 4. In Section 5, details of the experiments and the setup explained. Results and related discussions obtained for these experiments are demonstrated and discussed in sections 6 and 7, respectively. Finally, the conclusion of this work together with the possible future directions is addressed in Section 8.

[2] https://www.scb.se/en/finding-statistics/statistics-by-subject-area/energy/
[3] http://www.tuuleenergia.ee/en/windpower-101/statistics-of-estonia

2 Time-series Modeling and Forecast

Time-series is one of the very common forms of data. A time-series often contains real value numbers that appear one after another, constituting a series of numbers representing some phenomena over time. For that, each data point presents a form of dependency on other data points. Forecasting of a single time series which depends on the history of its own is referred to as univariate time series forecasting. Additional features contributing to univariate forecasting are referred to as exogenous variables whether it be another time series or any other data that could be used as an additional feature for the forecast. Also, forecasting of multiple time series is referred to as multivariate time series forecasting. Here, time series are treated as predictors for one another, while still other exogenous variables could also be employed. A multivariate time series forecasting is perhaps the most complex form of the forecast from the perspective of modeling. For instance, the most widely used assumption with tabular datasets, known as i.i.d.[4] could not be exercised here due to the dependency of each timestep with the previous timesteps. Moreover, the distribution of each data point is not identical to one another. That is why when modeling time-series each data point is being treated as a random variable. Figure 1 shows an observed time series on the top.

Due to these special characteristics, there have been specific tools and frameworks developed to understand, formulate and model time-series. In this section, we address the most important types of forecasts, modeling, evaluation, and formulations for time-series forecasting that provides a basis for later sections.

2.1 Patterns and influence of time

Time as an important property of time-series not only affects how we perceive time-series visually but also how we model, and forecast. Here we focus on the patterns emerging in time-series as a basis for the conventional statistical models in the following section.

Trend and *Seasonal* patterns are two fundamental patterns in time series. In terms of regression, the trend is the slope of the line that would fit the time series data. A time series may have a number of local trends and/or a global trend. In Figure 1 we can see an example of a global trend.

Seasonal patterns are those that are repeated in regular intervals over time and as the name implies their key characteristics is their regular occurrence whether it is hourly, daily, weekly, monthly, quarterly, yearly, or else. In Figure 1, seasonality of a signal is being demonstrated.

In a simple case, these patterns would be deterministic, which means one could perform forecasting, knowing the trend and seasonality components of the time-series. In practice however these patterns usually are not deterministic, however, it is possible to

[4] Independent and identically distributed random variables

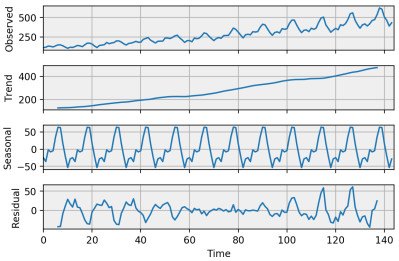


Figure 1. Timeseries additive decomposition into trend, seasonality and noise

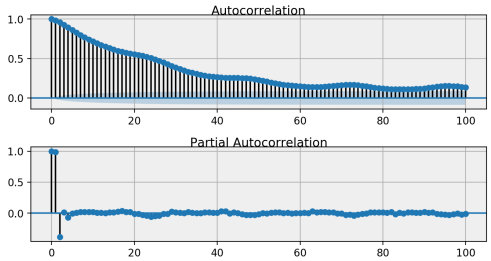


Figure 2. Auto-correlation and Partial auto-correlation plots

remove these components and try to model the residuals. And so by removing the trend and seasonality, one could focus on modeling the remainder. In such scenarios one could use decomposition techniques as demonstrated in Figure 1 to achieve this.

Another property of a time-series is its *stationarity*, meaning that whether the distribution of the time series changes over time or not. If a time series stationary it means that the data points in the series are identically distributed but they could still be dependent on one another (having historical dependency). Removing trend and seasonality could make a time series stationary or close to stationary. There are different tests such as *Dickey-Fuller* to check the stationarity of a series [HA14, ch. 8.1].

The historical dependency of the variables in time series, existence of trends and seasonality patterns are a basis for models that are designed to address all these aspects and properties of time series.

2.2 Statistical Models

In the previous section, important properties of the time series characterized. In the time-series forecasting, we usually have a history of data up to time (usually the present time) and we would like to forecast from the present into the future for a given number of steps. Each step in the forecast referred to as *lead time*, and the whole forecast steps referred to as *horizon*. Eq. (1) denotes the observed values of a time series from the past to the present and future.

$$y_{t-i}, y_t, y_{t+h} \qquad i, h \in \mathbb{N}_{>0} \tag{1}$$

It might also be desirable to refer to a beginning in the past which then denoted as in Eq. (2)

$$y_0, y_1, \ldots, y_t, y_{t+1}, \ldots, y_{t+h} \qquad h \in \mathbb{N}_{>0} \tag{2}$$

Statistical models are an attempt to model the behavior of a time-series over time by explaining recurring patterns and relationship between each variable at each time step.

Explaining different types of statistical models in depth is out of the scope of this thesis, however, we mention the most fundamental models that are related to concepts that are relevant to the thesis. As briefly mentioned earlier, decomposition of series could be used as either a pre-step to modeling or as a main approach to model and forecast time series. One can decompose time series into additive or multiplicative terms.

$$y_t = T_t + S_t + I_t \qquad \text{Additive decomposition}$$
$$y_t = T_t \times S_t \times I_t \qquad \text{Multiplicative decomposition}$$

Here y_t can be decomposed into Trend T, Seasonal S, and *Innovations* I components, where the modeling could be done on each of these components in order to achieve a forecast $\hat{y}$ for y_{t+h}. Taking a time-series as a random variable, if the future of this random variable could be explained by its past we can use regression over the past values of this random variable to explain its future. This is known as auto-regression over the lagged values of the random variable. An **Auto-regressive (AR)** model is essentially a linear combination of different steps from the past values of the variable. The number of parameters of AR is also referred to as the order of the model, that could be determined using Auto-correlation plots [HA14, ch. 2.8]. For the parameter estimation there different possibilities, one of which is the *maximum likelihood* [BJ76, ch.7]. Equation (3) demonstrates the AR model with a p weighted linear combination of p previous values of the variable Y with weights φ_i, an additive noise ε_t for each timestep t, and a constant c.

$$Y_t = c + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \varepsilon_t \tag{3}$$

If a time-series was not all explainable by an AR process then it means that the future values are not only explained by a linear combination of the past values, but there is a more complex relationship involved. Then after applying the AR model, there is still some gap to get closer to the observed values in the past. This remainder which is the difference of the observed values and the random variable referred to as innovations. If innovations is a constant then the constant term of the AR model can compensate for that, but if not then it is apparent that the AR model was incapable of modeling, i.e. produced biased results. Therefore, another additional term needed to model innovations. A **Moving Average (MA)** model can be used to fit the innovations produced as a result of the AR model and putting these two together constitutes the **ARMA** model. Equation (4) demonstrates the ARMA model that in addition to the AR terms it is a weighted linear combination of previous error terms ε_{t-i} with weights θ_i and constant c.

$$Y_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{4}$$

If a time-series exhibits some continuous increase or decrease, meaning that it has some trend, then it could be removed by differencing each time-step of the time-series from its previous time steps. Order of the differencing could be one or more meaning that one or more times may be required to remove the trend. This is modeled using an integrating ARMA referred to as **ARIMA**. It is called integrated since during the forecasting the differenced values should be integrated back in order to obtain the correct values for the forecast. ARIMA model could be extended into **Seasonal ARIMA (SARIMA)** in order to model the seasonal components the same way as ARMA but with a separate parameterization for the seasonal terms. Equation 5 is a general form of the SARIMA with B the backshift operator ($BY_t = Y_{t-1}$) and A_t corresponding to the innovations.

$$\phi(B)\Phi(B^s)Y_t = \theta(B)\Theta(B^s)A_t \tag{5}$$

where,

$$\phi(y) = 1 - \sum_{i=1}^{p} \phi_i y_i \qquad \Phi(y) = 1 - \sum_{i=1}^{P} \Phi_i y_i$$

$$\theta(y) = 1 - \sum_{i=1}^{q} \theta_i y_i \qquad \Theta(y) = 1 - \sum_{i=1}^{Q} \Theta_i y_i$$

Finding the right number of parameters for each of these models requires a systematic approach [HA14, ch. 8.6] and for that the *auto-correlation* and *partial auto-correlation* functions (ACF and PACF) are being used as depicted in Figure 2.

All of these models are defined for a uni-variate time-series forecasting. We can also use other time-series to enhance the forecasting by modeling the relationship and influence of a random variable not only on its own lagged values but also on other random variables lagged values. Methods such as SARIMAX and VAR [BJ76, ch.14] are models designed for such scenarios. These statistical approaches are all in the family of parametric methods, and for that, they have their own benefits and limitations. They provide a good foundation for understanding time series modeling that would benefit more complex and non-parametric models such as artificial neural networks that will be discussed in Sections 3 and 4.

2.3 Forecast Types and their Quality

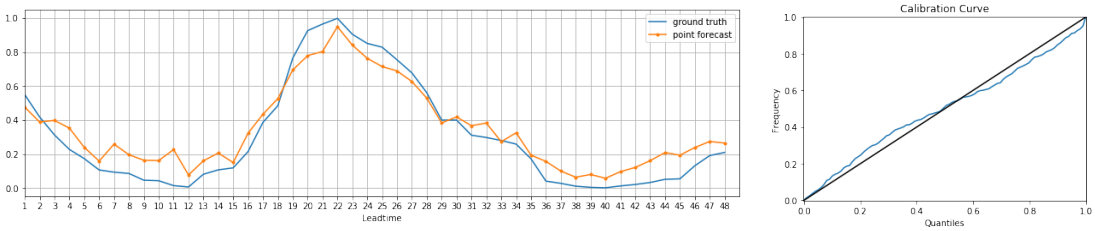
Different types of forecasts are employed by different use cases. Here an overview of possible forecast types and a brief discussion of quality measures for each type is given.

2.3.1 Quality and Calibration

One aspect of the quality is to know how reliable a forecast is. And reliability ties closely with the calibration of the produced forecasts. A well-calibrated model could be trusted with its outputs. In general, calibration refers to correcting a system so that it matches the underlying reality. Miscalibration is caused by a systematic error in our model. A highly accurate model is also calibrated but not vice-versa. A model could be calibrated but not calibrated at all. To demonstrate the concept of calibration take an example of a classifier to classify cats and dogs. The proportion of examples of cats and dogs in our data is equal, so there is a probability of 0.5 to have a cat and 0.5 to have a dog. If we have a random classifier, the probability of classifying an image as a cat is 0.5 and a dog is 0.5 which is matching the data distribution and hence calibrated. However, this classifier would only be accurate 50% of the times and it is not accurate at all. Later in Section 4 we show how the measure of uncertainty can help to provide better-calibrated models, matching the underlying reality that model would have missed otherwise.

2.3.2 Point forecast

Point Forecast as it implies, it is a point predicted for the future. In this scenario model as a function takes in point values and gives point values. Point forecasts are not interesting nor convenient, especially for decision making. The main problem with the point forecast is that there is no extra information besides the point estimate regarding the uncertainty of the forecast. The point could be far off the true value or very close to it. Given the fact that the model validated over many tests and produced very low errors over these tests, one may have high confidence in the model. However, there always a possibility for the model to observe novelty in its input. The fact that one cannot exhaust all the possible inputs for the all possible outputs is a good reason not to rely only on model performance on the validation set. As a conclusion, point forecasts are the least informative form of forecasts. Figure 3a is an example of point forecast.



(a) Point forecast

(b) Calibration Plot

Figure 3. A Point Forecast and its calibration plot

On the other hand, quality of the point forecasts could be measured by looking at

the distribution of the point forecasts also referred to as *predictive distribution (PD)* and comparing it with the empirical distribution of the data. This can be done by calculating the frequency of the ground truth observation falls into different quantiles of the PD starting from 0 to 1, also can be referred to as the least and most confidence levels, respectively. This will result in a monotonically increasing curve that in the ideal case it is expected to roughly follow the line $y = x$. Any deviation from the $y = x$ would indicate a miscalibration in the forecast. Figure 3b demonstrates a calibration/reliability plot for a set of point forecasts.

2.3.3 Quantile forecast

Despite the point forecast, a *Quantile Forecast* is more aware of the distribution of the data by giving a specific lower and upper bounds for the forecast. For example, the model would learn a specific quantile(s) of the input data and forecasts for that specific quantile(s). Still, it is not informative enough in the sense that as in point forecast, it does not provide a full picture of uncertainty, especially if the error in the parameter estimation of the model were not incorporated. Figure 4 is an example of quantile forecast. A reliable quantile forecast (calibrated) should include more true values, and so a quantile forecast would be better calibrated if the quantiles are tighter while inclusive of the true values.

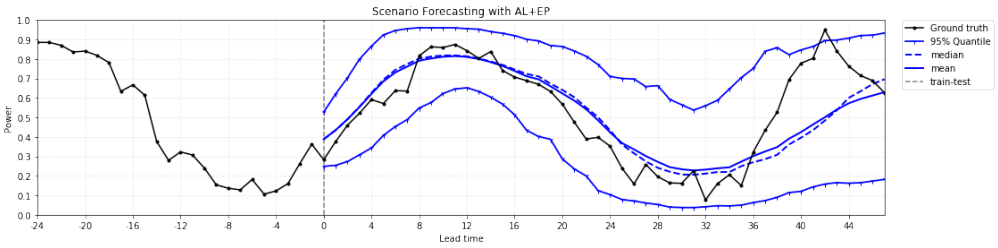


Figure 4. Quantile forecast

2.3.4 Probabilistic forecast

Probability or *Dense Forecast* is the most informative form of forecast. It can be thought of stacking different quantile forecasts together, which would construct a dense forecast. Being able to describe the distribution of each forecasted variable is the main purpose of the probabilistic forecast. The concentration of probability mass at one point would translate to a more confident forecast as in contrast to a less confident when this mass is spread. However, the confidence of a probabilistic forecast does not necessarily mean one has achieved an accurate or reliable forecast. Figure 5 depicts a probabilistic forecast. For the more reliable forecast the same condition as in quantile forecast should exist, and that inclusiveness of the model forecast of the true values.

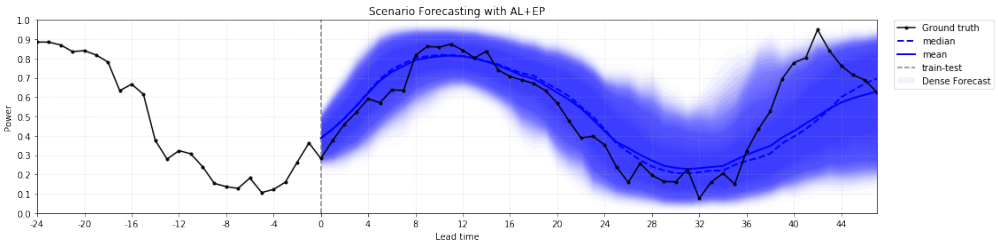


Figure 5. Probabilistic forecast

2.3.5 Scenario forecasts

Scenario Forecasting is another form of forecasting which is a more advanced case of how one could deal with uncertainty. Scenario forecast is essentially a way to extract information from the model about alternative, possible scenarios in the future. This is very intuitive and convenient when it is thought in the context of decision making as often users of the forecast would like to know uncertainty, as well as multi-modality as each mode of the forecast, may have its own interpretation i.e. translates into different decisions. Probabilistic forecasts not only provides more informative detail about the forecast but also provides a good ground for scenario forecasting. Since we are capable of explaining each forecast probabilistically then we can create different alternative futures by sampling from the distribution of the forecasts with different probability and policies over the scenarios. One of the contributions of this work explained in Section 4.5 was on scenario forecasting. Figure 6 depicts an example of this type of forecast.

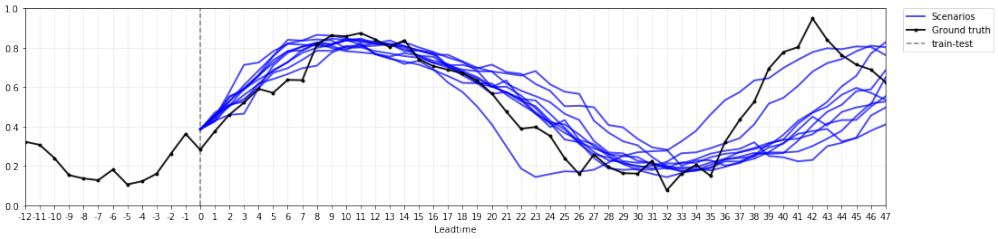


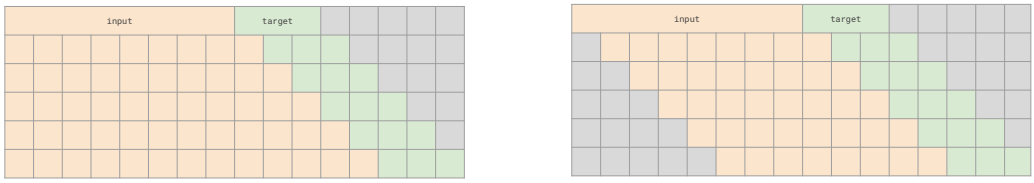
Figure 6. Scenario Forecast

Scenario forecasting in this work used to improve the quanlity of the probabilitic forecasting, and so evaluation of the scenarios is not part of the thesis.

2.4 Model Validation

Since there is a time dependency between variables of the time series it is not possible to shuffle or perform *k-fold cross validation* which is a common way to assess the ability of the model for generalizing. *Sliding Window* and *Expanding Windows* are a counterpart

for cross-validation on time-series [HA14, ch. 3.4]. Expanding the window is more desired when there is a limited number of samples available (short time series). Figure 7 depicts these two schemes.



(a) Expanding window (b) Sliding window

Figure 7. Time-series validation schemes

2.5 Baselines

Baselines are simple heuristics that model a simple relationship between the input and the target variables of a time series. They provide a reference to make sure the developed models are not performing worse than them. Depending on the difficulty of a time series forecasting task, a baseline may perform relatively better than a model. Here we introduce two baselines, one for the point forecasts and one for probabilistic forecasts. The goal of a model is to improve the forecasts over these baselines.

2.5.1 Naive

A conventional baseline for the time-series forecasting is a *persistence model*, also may be referred to as *naive*, where the last observed value is being used as the forecast for the whole horizon as shown in Eq. (6)

$$\hat{y}_{T+h|T} = y_T \tag{6}$$

Where $h \in \mathbb{N}_{>0}$ represents a lead time in the horizon, and T is time step in the observation that all lead time forecasts in the horizon were conditioned on.

2.5.2 Quantile Naive

To provide a baseline for the probabilistic forecast we introduce a variant of naive, and we name it as quantile naive[5]. In Quantile naive Instead of using the last observation as the forecast, it uses the quantiles of the last n observations is as the forecast. In this manner, quantile naive performs some form of probabilistic forecast or measurable as a probabilistic forecast.

[5]It seems that such baseline has not yet been documented in the literature.

$$
\begin{aligned}
F_Y(y) &= P(Y \leq y) \quad Y = \{y_{T-w}, y_{T-w+1}, \ldots, y_T\} \\
Q_Y(q) &= F_Y^{-1}(q) \\
\hat{y}_{T+h|Y} &= Q_Y(q)
\end{aligned} \tag{7}
$$

Equation (7) shows how quantile naive is being performed. $\hat{y}$ is the forecast for the horizon of size h which is calculated based on desired quantile q from the cumulative distribution F_Y obtained from the history Y of the last w observed values. For example, if $w = 4$ then $Y = \{y_{T-4}, y_{T-3}, y_{T-2}, y_{T-1}, y_T\}$ and the forecast for 90th percentile quantile with $h = 2$ would be $\hat{y}_{T+h|T} = Q_Y(0.9)$.

2.6 Evaluation Metrics

Here three evaluation metrics introduced, two for evaluation of a point forecast and one for a probabilistic forecast, which is also a generalization of the other two.

2.6.1 Mean Squared Error (MSE)

This metric is being used in this work mainly to be able to compare the results with [KG17]. Equation (8) shows how MSE can be calculated for each lead time h in the horizon, with $n(\mathbb{T})$ denoting the cardinality of the forecasts. MSE has the same measurement unit as of the squared of the series evaluated for. It is also equivalent to the variance of the model output if the model is unbiased.

$$
MSE(h) = \frac{1}{n(\mathbb{T})} \sum_{t \in \mathbb{T}} (\hat{y}_{t+h|t} - y_{t+h})^2 \tag{8}
$$

2.6.2 Continuous Ranked Probability Score (CRPS)

CRPS [MCM$^+$11] evaluates a probabilistic forecast represented by a predictive cumulative distribution function $F_t(y)$ against the true value or observed probability passed through a *heaviside* function $H(y, \hat{y})$ as shown in Eq. (9). Perhaps it would be easier to think of the CRPS as a *Brier score (BS)* shown in Eq. (10) which is essentialy an MSE. Both of the scores calculate a difference, where in BS is in discrete space where f_t can be thought of a specific quantile of F_t and o_t an indicator of whether the true value predicted correctly or not (for that specific quantile). The counterpart of o_t in CRPS is the heaviside function that $H(x) = 1$ for $x \geq 0$ and $H(x) = 0$ for $x \leq 0$. Another minor but obvious difference is that for the CRPS the BSs are integrated over all possible quantiles of the predictive distribution. If the intended forecast values were normalized, then the calculated score would be normalized and referred to as NCRPS.

$$NCRPS(y) = \frac{1}{n(\mathbb{T})} \sum_{t \in \mathbb{T}} \int_{-\infty}^{\infty} \left(F_{t+h|t}(y) - \mathbb{H}(y - y_{t+h}) \right)^2 \tag{9}$$

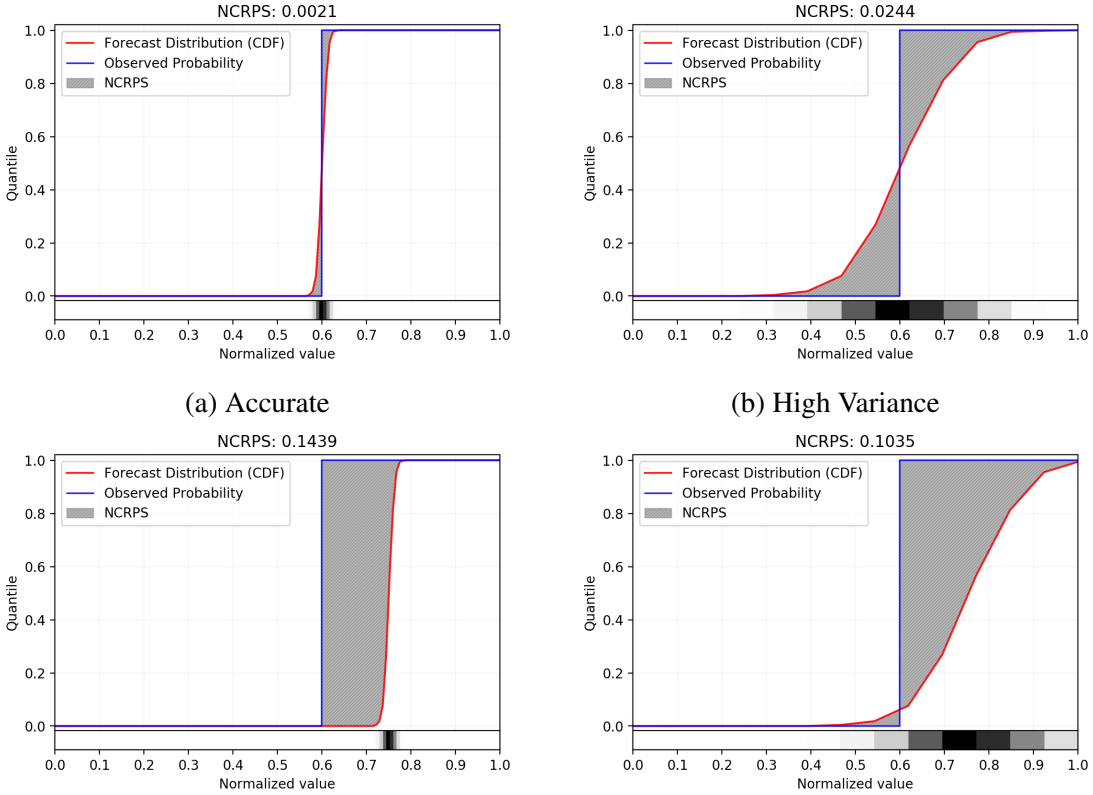
$$BS = \frac{1}{N} \sum_{t=1}^{N} (f_t - o_t)^2 \tag{10}$$

An important property of CRPS is that it measures the accuracy, as well as the sharpness of the predictions which means *calibratedness* of the forecasts is also reflected in the score. Besides, it is a *proper scoring rule*, and so the lower the better (taking minimum of 0), and with the same unit as the measured variables as it is a generalization of the MSE. Figure 8 illustrates four scenarios for the calculation of the NCRPS. To further convey the calculations four different scenarios were simulated for the forecast where the probabilistic forecast is accurate (not biased and low variance), not biased while high variance, biased while low variance, and finally biased while high variance. In all the cases the true value had set to $y = 0.6$. CDF of the forecast had altered with a $bias = 0.15$ and $\sigma = 0.1$. The barcode plot in the bottom of the plots shows the PDF of the forecast as it is visually appealing to see the bias and variance clearly.

2.6.3 Mean Absolute Error (MAE)

It is also a special case of the CRPS, where the forecast variable corresponds to a specific percentile (usually and also in this work $50th$ percentile) and so it is appropriate for evaluation of a point forecast.

$$MAE(h) = \frac{1}{n(\mathbb{T})} \sum_{t \in \mathbb{T}} \mid \hat{y}_{t+h|t} - y_{t+h} \mid \tag{11}$$



(a) Accurate

(b) High Variance

(c) Biased

(d) NCRPS calculation under different cases

Figure 8. Demonstration of NCRPS calculate under different scanarios

3 Uncertainty in Deep Learning

A *Bayesian framework* is a powerful tool for modeling and inference. The most notable aspects of Bayesian model is the power of modeling uncertainty which is an enabler to also improve performance of the model as well as generalization capability as shown by [KG17]. To begin with, an overview of the *non-Bayesian neural network* is given, then different types of uncertainty and how it is framed in a deep learning model is explained. Further, an overview of the Bayesian framework is presented by reviewing the required terminology, notations, *Bayes' theorem*, *Bayesian inference*, and *Bayesian prediction*. Then we look into how the Bayesian framework is applied on neural networks and explain a common approach that has become successful in deep learning and also being used in this thesis work.

3.1 Non-Bayesian Neural Network

An *Artifcial Neural Network (ANN)* is a general function approximator that initially inspired by the neuron cells in the brain. They first appeared as a *perceptron algorithm* [Ros58] and later many other additions to this family of methods emerged. They are composed of layers within which a number of nodes reside. These nodes also known as the weights of the network enable the network to take an input and transform it by passing it through each layer and weight of the network to the final layer and output of the network. This process called *forward pass* where network performs transformation with the current weights. During training time, the obtained outputs are compared with the target values and a *loss* measure calculated that provides a reference for optimizing the network parameters (weights). An optimization algorithm called *gradient descent* is often used to change the weights of the network with respect to each weight's contribution to the final output error/loss. This process is performed at the *back-propagation* stage. Each time the network is trained on all or roughly all of the training instances by going through the forward and backward pass. Every complete cycle of the forward and backward pass is then denoted as *epoch*. Every epoch learning may occur on a single instance or a batch of instances. Different loss measures could be chosen to perform optimization. For regression tasks, MSE loss is the most convenient one. This learning scheme is also referred to as *supervised learning* as targets (or labels) for matching the network output to, has a supervision role in the network's learning process.

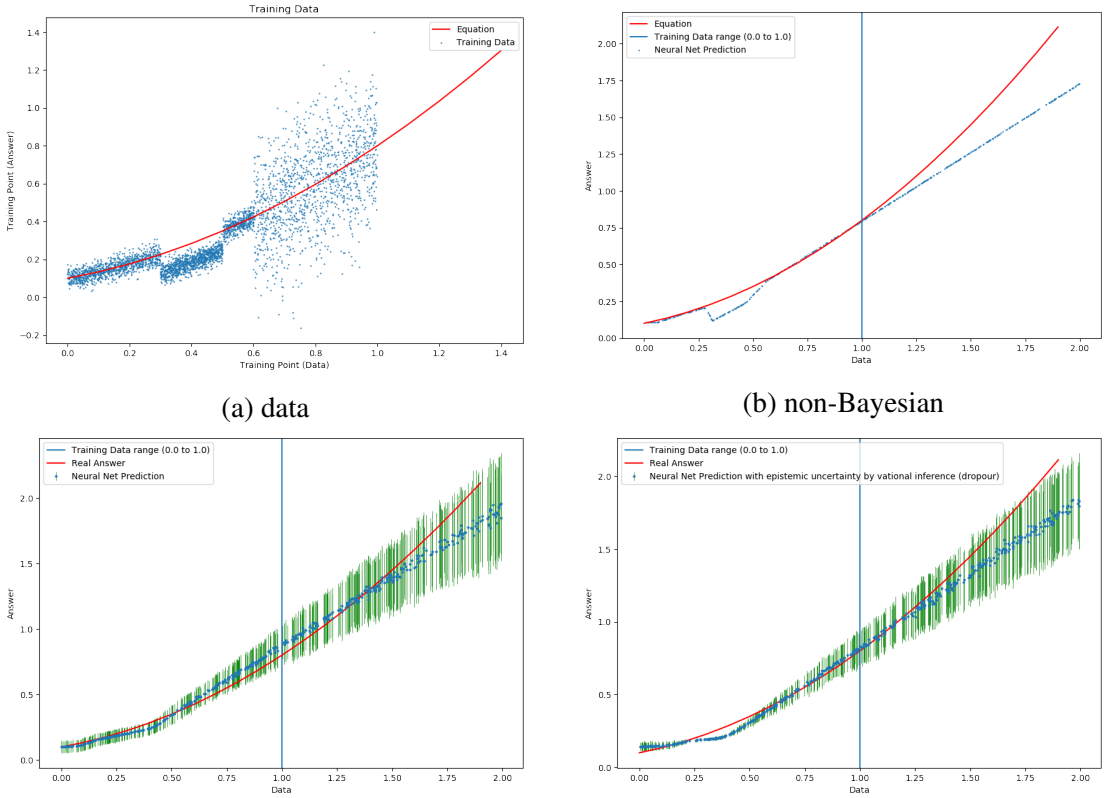
A *non-Bayesian Neural Network* equivalently a *Frequentist Neural Network* is a conventional form of neural networks where an optimum point estimate of the parameters is being estimated which means that by default is incapable of explaining the uncertainty and so the model would always underestimate the variance of the predictive distribution. That is why we already know that in theory, *Bayesian Neural Networks (BNNs)* are more capable both for interpretability as well as generalization compared to their non-Bayesian counterparts. Figure 9b demonstrates the output of a non-Bayesian network on a dataset that involves a considerable amount of uncertainty as depicted in Figure 9a. As a conclusion, every neural network model consists of following building blocks:

- training input and corresponding target values
- model (layers and nodes, also referred to as weights of the network)
- loss function
- optimizer

3.2 Types of Uncertainty

There are different taxonomies of uncertainty depending on the context. In the context of machine learning, we are mainly concerned about two main categories of uncertainty.

The root or main cause of uncertainty, however, is common for any context, and that is an error. And error can be caused by partial observability, stochasticity - both can be referred to as unknown or novelty in what it comes next - ignorance, indolence, or possibly some other form or combination of these. Here we address these two types of uncertainty that matter most to this thesis. Figure 9 adopted from astroNN[6] demonstrates how added noise to the data translates to aleatoric and epistemic uncertainty, as well as the output of a non-bayesian model to such noisy data. In Fig. 9a can be seen that there is a misalignment of some of the data points from the expected trajectory of the points. That could occur due to measurement error. Also, we can see that the distribution of the data changes. We can see how the non-Bayesian model in Fig. 9b failed to model and deal with these uncertainties. On the other hand, a model with the capability of capturing two different types of uncertainty is more capable of explaining these different situations in the data as can be seen in Fig. 9c and 9d.



(a) data

(b) non-Bayesian

(c) aleatoric

(d) epistemic

Figure 9. Comparison of different uncertainties

[6] https://github.com/henrysky/astroNN

3.2.1 Uncertainty in Observations

Uncertainty in the data, also referred to as **Aleatoric uncertainty**, is mainly caused by partial observability and stochasticity in the underlying process generating the data, and/or errors in measurements. The Aleatoric uncertainty is explained by variance of the observations which can be estimated the same way that the mean of a set of observations estimated. If the variance of the observations is the same for a given data/task it is referred to as a **homoscedastic uncertainty**, if it is changing across instances then it is referred to as **heteroscedastic uncertainty**. Figure 9c demonstrates the captured aleatoric uncertainty, as we can see there are regions with different changing variance in Figure 9a that corresponds to the generated data.

3.2.2 Uncertainty in Model(ing)

Uncertainty in the model also referred to as **Epistemic uncertainty**, is mainly caused by an error in the estimation of the model's parameters when modeling. To model any non-deterministic process, parameter estimation for the model is prone to errors. Also, model misspecification, e.g. using a wrong parametric family could lead to epistemic uncertainty. At the same time, this uncertainty is closely related to aleatoric uncertainty because the uncertainty in the data could reflect on epistemic uncertainty. For example, a model trained on a set of examples, when exposed to unseen instances are nothing like the previous examples, impose uncertainty on the model. To this end, the epistemic uncertainty is a function of error in parameter estimation and/or model misspecification as well as the aleatoric uncertainty. Figure 9d demonstrates the epistemic uncertainty. It is often expected to obtain larger uncertainty for future steps as the model is biased towards the history, and so the farther it goes out of the sample, the more it diverges from the reality.

3.3 Estimating Aleatoric Uncertainty

Since aleatoric uncertainty is a quantity to be measured from the data, it could be estimated by the network besides the mean value that is already being estimated, and so this can be done by adding another output to the network. This output would estimate the variance of the input data. Equation (12) corresponds to the MSE loss used in a regular regression task for optimizing a neural network. The net loss is the average of each outputs' loss as shown in Eq. (15). This loss comprised of two parts. The first part is the MSE as before but weighted by the inverse of the variance as in Eq. (13) and the second part is the logarithm of the variance as shown in Eq. (14).

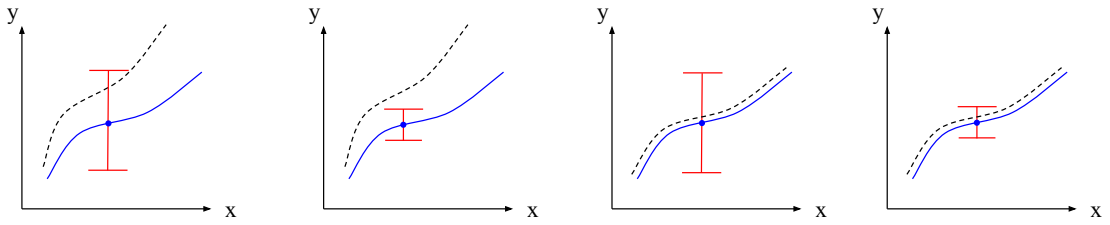
$$\mathcal{L}_{NN}(\theta) = \frac{1}{N}\sum_{i=1}^{N} || \mathbf{y}_i - f(\mathbf{x}_i) ||^2 \tag{12}$$

$$\mathcal{L}_1(\theta) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\sigma(\mathbf{x_i})^2} \mid\mid \mathbf{y}_i - f(\mathbf{x}_i) \mid\mid^2 \tag{13}$$

$$\mathcal{L}_2(\theta) = log\sigma(\mathbf{x_i})^2 \tag{14}$$

$$\mathcal{L}_{NN}(\theta) = \frac{\mathcal{L}_1 + \mathcal{L}_2}{2} \tag{15}$$

Using this loss function, the model will be able to learn the variance in the data and hence capturing the Aleatoric uncertainty. In this way, the network outputs a mean and variance that could be associated with a normal distribution over each prediction. Moreover, it is important to note that for learning the variance there is no need for more labels or target values. The same target values for the mean would suffice and helps the model to infer the logarithm of the variance.



(a) $\mathcal{L}_1 \leq \mathcal{L}_2$ (b) $\mathcal{L}_1 \geq \mathcal{L}_2$ (c) $\mathcal{L}_1 < \mathcal{L}_2$ (d) $\mathcal{L}_1 \ll \mathcal{L}_2$

Figure 10. Different cases for the loss - blue: prediction, dashed-black: real value, red-bar: variance

Figure 10 depicts different possible scenarios that this new loss in Eq. 15 may encounter:

(a) when MSE is large as well as the variance, $\mathcal{L}_1$ gets small because the variance is captured but then model gets penalized with $\mathcal{L}_2$ to get even lower variance if possible

(b) when MSE is large but the variance is not, $\mathcal{L}_1$ gets bigger because the difference in the MSE is not corresponding with the variance and model need to correct its bias first, $\mathcal{L}_2$ would not matter as much as $\mathcal{L}_1$ in this case but still model is being penalized to reduce the variance

(c) MSE is small but the variance is big, then $\mathcal{L}_2$ encourages the model to reduce the variance

(d) model would ideally get to this scenario from all the other step and $\mathcal{L}_2$ would still encourage the model to reduce the variance

As a conclusion, since $\mathcal{L}_1$ is the dominating term model needs to obtain a small $\mathcal{L}_1$ first, then $\mathcal{L}_2$ becomes important. It is also noteworthy that this loss function is equivalent to negative of the *maximum likelihood (ML)* function would obtain an ML solution by minimizing this loss [Bis06, p.27].

3.4 Estimating Epistemic Uncertainty

Bayesian Neural Networks use the Bayesian inference, and Bayesian prediction in order to explain a distribution possible values for the weights of the network as well as its output. These two concepts will be addressed first, and then Monte-Caro Dropout (MCDO) as the main method to practically obtain epistemic uncertainty from the model.

3.4.1 Bayesian Inference

One reason that Bayesian inference is very powerful goes to the fact of how the world and interaction with the world occurs. Any intelligent entity has an assumption about the world (a world model) which is updated continuously by observing the world and correcting and operating in the world as an intelligent entity that adapts. The Bayesian framework provides all the ingredients required for formulating, defining and employ at the same level as any intelligent being and so makes it broadly applicable across different disciplines and areas.

It all begins with the Bayes' theorem:

$$P(W \mid O) = \frac{P(O \mid W)}{P(O)} \cdot P(W) \tag{16}$$

Where W refers to the World-model, O to the observation we made. We begin with an initial world model $P(W)$ (prior probability), an estimate of the probability of our world-model to be true. And what we would like to do is to update this world model using the observation that has been made from the surrounding world and so to obtain $P(W \mid O)$ (posterior probability) of how our World Model looks like with the observation. This achieved by checking the compatibility of our World Model with the Observation given by the $P(O \mid W)$ (likelihood) which is a function of our observation with a given fixed World Model. This joined with the probability of this observation actually occurring $P(O)$ (marginal likelihood) because it is independent of our world model (marginalized).

Spherical model of the earth is a good example of updating a world model. Once it was thought that earth is flat ($P(W)$). Given the observation that traveling from one part of the earth to the other we observe different parts of the space then the chances that the

earth is flat and we have this observation ($P(W \mid O)$) should decrease because if earth was flat we would have not seen any new constellation by traveling from one part of the earth to the other as in a flat earth model all part of the sky should be visible so it must be some other shape that is compatible with this observation. Also the probability of we observing this phenomenon ($P(O)$) at least half of the time true because half of the time travel made across the equator and half of the time along the equator.

To achieve a better world model this process could be generalized and joined with other events and observations as well as this observation until obtaining a confident posterior.

3.4.2 Bayesian Neural Networks

Neural networks could be turned into Bayesian neural networks by learning the distribution of its parameters instead of a scalar value for each weight. This means that we put a prior on the weights of the network and we are interested in calculating the posterior given the data.

$$p(\theta \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \theta)p(\theta)}{p(\mathbf{X})}$$

Here θ represents parameters or weights of the network and $\mathbf{X}$ the data. $P(\theta)$ is the prior and we already know this (e.g. initialized by a Gaussian prior). $P(\mathbf{X} \mid \theta)$ is the likelihood of $\mathbf{X}$ under current weights of the network, which could be obtained simply by a forward pass. In the denominator, the $P(\mathbf{X})$ is the distribution of the observed values that is marginalized over the parameters and can be calculated by integrating over all the possible values of theta: $\int p(\mathbf{X} \mid \theta)p(\theta)d\theta$. This, however, is intractable for even small networks and so there exist methods for approximating this probability instead.

Given one could find optimal parameters of the network, then prediction for an unobserved instance can be done using posterior predictive distribution as follows:

$$p(\tilde{x} \mid \mathbf{X}, \theta) = \int p(\tilde{x} \mid \theta)p(\theta \mid \mathbf{X})d\theta$$

3.4.3 Monte-Carlo Dropout

As mentioned earlier, to alleviate the intractability of calculating the posterior because of the marginal likelihood a *variational Bayesian method* should be used instead that addresses the intractability by providing an approximate solution. Different approximation methods developed for deep learning to calculate the posterior [TAS18, Gra11]. Among all, Monte-Carlo Dropout [GG15] has been established since its introduction due to simplicity and availability. It uses the same dropout technique used for regularization of neural networks [SHK+14].

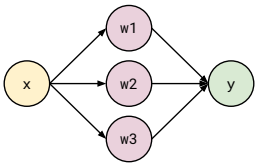


Figure 11. A simple one layer network with 3 nodes

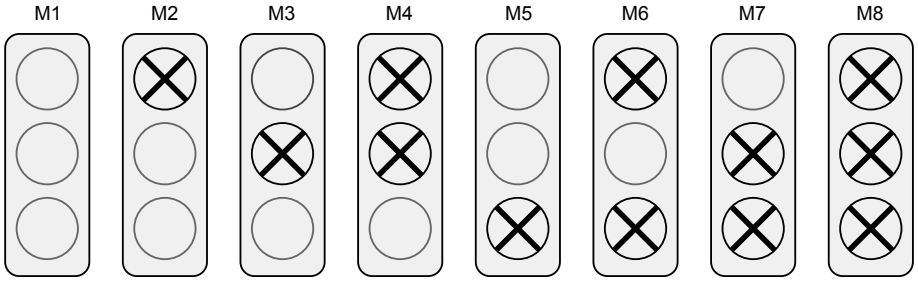


Figure 12. Possible Dropout Masks for a layer with 3 nodes, masks with cross indicate 0 and those with hallow circles indicate 1

What dropout does at training time is to randomly ignore some of the nodes at each epoch so that the learning takes place with the rest of the nodes. Nodes are being dropped randomly and alternatively across different epochs. In the short run this makes each part of the network trained independent of the other, in the long run however, it provides a level of redundancy as well as regularization effect. One of the implications of the redundancy is that the whole network could be thought of an ensemble of smaller sub-networks as a result of the dropout.

To better convey the concept, take a simple neural network with 3 nodes as in Figure 11. These three nodes could either be dropped or not dropped which makes up 2^3 possible states or *masks* to put on each node, depicted in Figure 12. Among all $m1$ and $m8$ are not desirable. If all nodes dropped ($m8$), no output will be obtained, therefore no learning can be achieved. If no nodes dropped ($m1$), there is no regularization effect in the network. To this end, in total $2^3 - 2 = 6$ different states are desirable for dropping these nodes. Each of these possible dropouts results in effectively a different network, depicted in Figure 13. And so for this simple example, learning these separate networks and combining their result would effectively be the same as performing the dropout for one network. Dropout technique makes it possible to keep the integrity of the network as a whole while performing some kind of an ensemble. Instead of having separate networks, a separate information flow constructed using this technique which its effect would be the same as if one would train separate models and combine them.

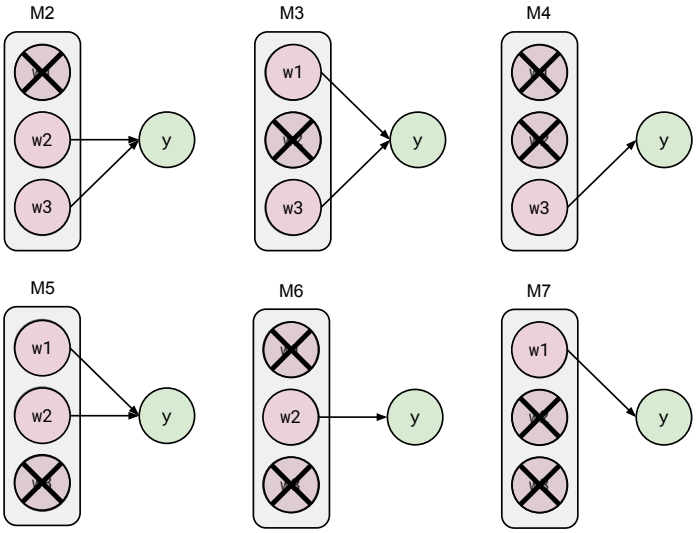


Figure 13. Possible networks after applying dropout mask

More formally, The simple network from Figure 11 can be written as in Eq. (17).

$$y = \mathbf{x}.\mathbf{w}' \tag{17}$$

Where the input x and output y are scalar values and $\mathbf{w} = (w_1, w_2, w_3)'$ a vector representing each node of the network and $\mathbf{x} = (x, x, x)$ is a broadcast version of the input with dimensions of 1×3 to match the first dimension of the $\mathbf{w}$. Dropout can be applied in a way shown in Eq. 18.

$$y = \mathbf{x}.M.\mathbf{w}' \tag{18}$$

Where M is a diagonal matrix of size 3×3 with its diagonal entries being the mask $\mathbf{m} = (m_1, m_2, m_3)$ with each of its entries being drawn from a Bernoulli distribution with probability of p. Probability p indicates the probability of an entry being 1 and so the probability of dropout or being 0 would be $1 - p$.

Dropout at the first layer would be equivalent to performing sampling from the input data, which is similar to the *bootstrap aggregating* in ensemble methods [Fla12, ch. 11.1]. As discussed intuitively, while training, the dropout acts as a regularizer by training portions of the network independent of the other at each time, which removes potential influences that may cause the network to get stuck in a local optimum and hence overfit . Moreover, another implication of such redundancy is that the dropout during test time would result in different possible outputs produced by different subnetworks which would imply the variance of the model, also known as epistemic uncertainty. Which is again, a similar concept in ensemble method for reducing the variance of prediction by using a number of unbiased models [Fla12, ch. 11.3].

4 Deep Probabilistic Forecasting

Deep Probabilistic Forecasting (DPF) as the name implies, performs probabilistic forecasting using deep learning methods. Although there have been many efforts in recent years with deep forecasting, it has been mostly limited to the estimation of quantiles. This work is among the first to perform probabilistic forecasting with Monte-Carlo Dropout Neural Network. In this section and the following the main contributions of this work is being explained which is mainly inspired by [KG17] to achieve probabilistic forecasting.

4.1 Framing the problem as supervised learning

Supervised learning is the common scheme that has dominated the deep learning methods for different tasks from analysis of spatial data such as images to temporal data such as time-series data. Time-series forecasting could be framed as supervised learning by placing a window of size n on the input sequence and a window of size m as the target of that input sequence. In time series forecasting usually, the target sequence is right after the input sequences. Hence, a sequence of size T produces $t-(n+m)+1$ instances when it is framed into a supervised learning problem, see Figure 14. It can be thought of as a window of size $n+m$ that slides on the time series, where the first n elements denoted as input X and the last m elements as the target y.

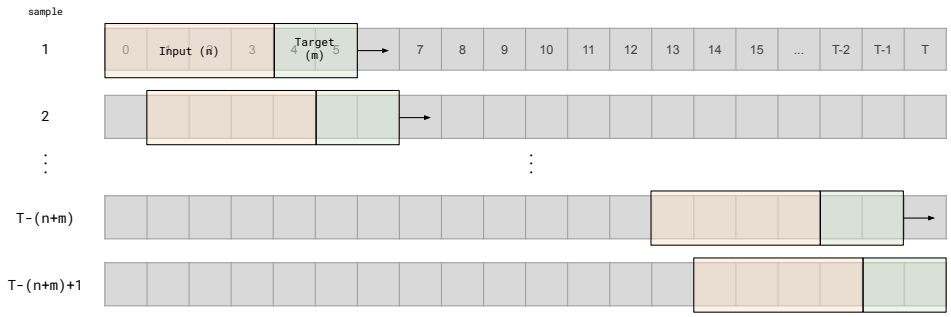


Figure 14. Creating train and target(s) with sliding window

And so the dimensionality of a time-series forecasting problem would be $(t-(n+m)+1, n)$ for the input and $(t-(n+m)+1, m)$ for the target vectors. If the forecasting problem is a multivariate time-series then the input dimension assuming that the length of all series is the same would be $(t-(n+m)+1, m, f)$, where f is the number of series. The output or target sequence would be $(t-m, m, i)$, where $i \in 1, \ldots, f$ depending on how many of these series need to be forecasted.

4.2 Models topology

In a sequence modeling problem, different numbers of input and numbers of output can be modeled. Figure 15 demonstrates possible mappings of multiple inputs to multiple outputs. In most of time-series forecasting problems a history of past and present is given and a model should forecast one step or a number of steps into the future, and so many-to-one or many-to-many mappings are the most common.

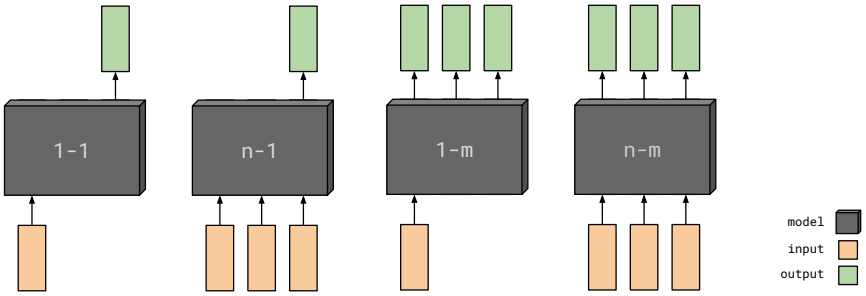


Figure 15. Possible input to output mappings in sequence modeling

The focus of this work is on the uncertainty aspects and so the simpler architecture with many-to-one output was used in the experiments.

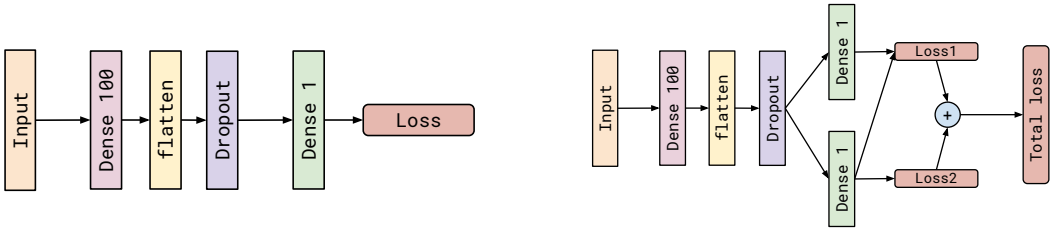
4.3 Neural Network Architectures

Two very well-known artificial neural network architectures were used. One was multilayer perceptron (MLP), the other was a *recurrent neural network (RNN)*. Recurrent neural networks are famous for sequence modeling, especially in the area of *natural language processing (NLP)*, and *machine translation* where sequences of characters or words are of particular interest. In time-series, the sequence modeling is on series of real number or signals of various forms. Besides these two architectures, *Convolutional Neural Networks (CNN)* may also be employed for sequence modeling. Since the main focus of the work was on uncertainty and probabilistic forecasting, we only focused on a simple version of MLP and RNN. The MLP architecture provides a good comparison with RNN to see if the complexity and computational cost of the RNN, which is specialized for sequence modeling, is worth it over the performance of MLP.

4.3.1 Multi-layer Perceptron

Multi-layer perceptron or fully-connected network consists of a number of layers, each consisting of units (also known as artificial neurons). These units are represented as

matrices of weights for each layer, stacked together to construct a deep architecture. This architecture is the most conventional, universal form of artificial neural networks that in theory are capable of approximating any function given enough data and complexity (layers and units per layer). In theory with this architecture, it is possible to learn any function of the input, picking any linear or non-linear combination of the input sequence. More complexity could be added to the network by adding more layers.



(a) 1-head output (b) 2-head output

Figure 16. Fully-connected Neural Network Architecture

Two variants of the architecture were employed. One with single output as shown in Fig. 16a and the other with two outputs as shown in Fig. 16b. The Latter estimates both the mean and variance of the data. The number of units and regularization determined based on the synthetic data with the objective of having the simplest possible model. Dropout was used as a regularizer to avoid overfitting in both models. In the second model, it was also used for the purpose of estimating the epistemic uncertainty.

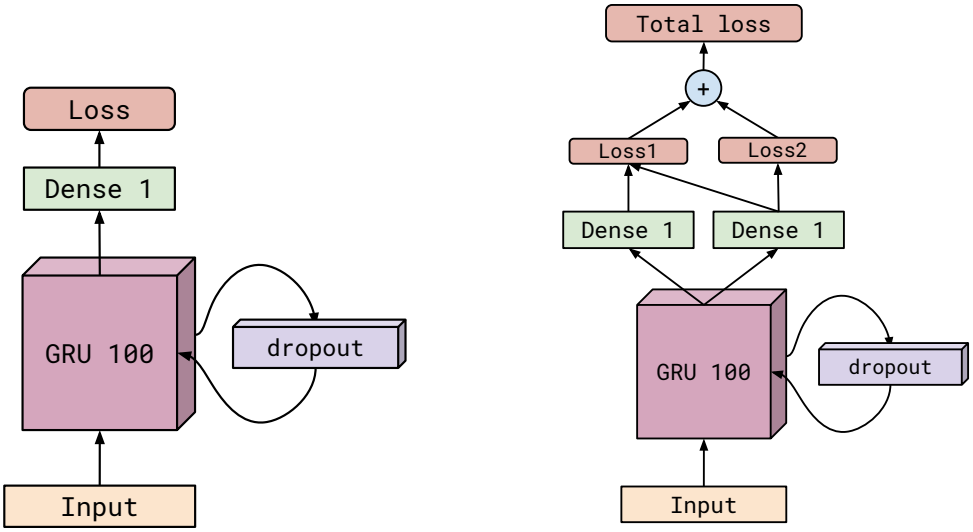
4.3.2 Recurrent Neural Network

Recurrent neural networks are ought to be good with memorizing sequences of data, for that the data flows not only through the depth of the network but also through lateral connections (referred to as recurrent connections) making the network capable of exercising a capacity for memorizing. Gated Recurrent Units (GRU) was employed that is a successor of Long-short Term Memory (LSTM) [CGCB14]. They are a simplified version and computationally more convenient. Recurrent Dropout [GG16] was also employed together with this model for both the purpose of regularization and MCDO approximation.

As in the case of MLP, two variants of RNN-GRU were used. One with a single output as shown in Fig. 17a and the other one with two outputs as shown in Fig. 17b.

4.4 Probabilistic Forecast

The 2-head models mentioned above were capable of predicting a Gaussian distribution as their outputs estimate the mean and variance of the forecast variable one step ahead.



(a) 1-head output

(b) 2-head output

Figure 17. Recurrent Neural Network Architecture

A preceding sequence to the target horizon for the forecasting is given to the model as a *warmup sequence*. This sequence is used to make a multistep ahead forecast, by appending the one step ahead forecasts to the input sequence and shift the warmup to the left such that a new warmup sequence with the last prediction at the end could be fed to the network for next step until the intended horizon steps are being covered. This multi-step forecasting scheme is known as *rolling forward forecasting* [HA14, ch. 3.4] depicted in Fig. 18.

This scheme, however, is biased because of the fact that the forecasts are being treated as new observations which are not true. Since network input accepts a vector of scalars and not distribution, the probabilistic forecast should either be sampled or the mean should be used to perform roll forwarding. This means that the there is systematic error/bias in the inputs that provided by the model forecast from previous steps and more importantly even if the bias is negligible the uncertainty involved with these forecasts are different from the uncertainty that the model is initially being trained on to capture. It includes both aleatoric and epistemic uncertainty, but the model was not trained to capture uncertainty about its own uncertainty as this scheme requires so. In other words, the many-to-one model requires rolling forward forecasting, which introduces an ignore by the model which consequently leads to biased results. In order to tackle this problem, we propose to perform a form of scenario forecasting.

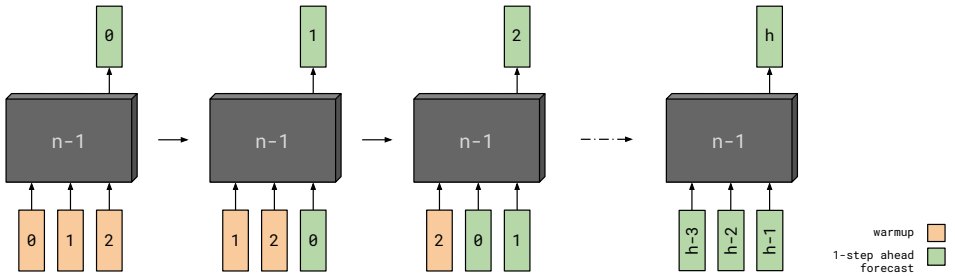
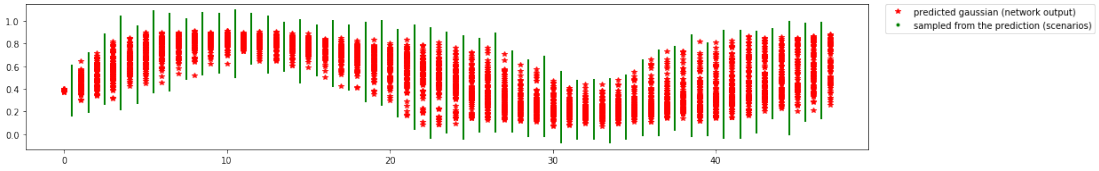


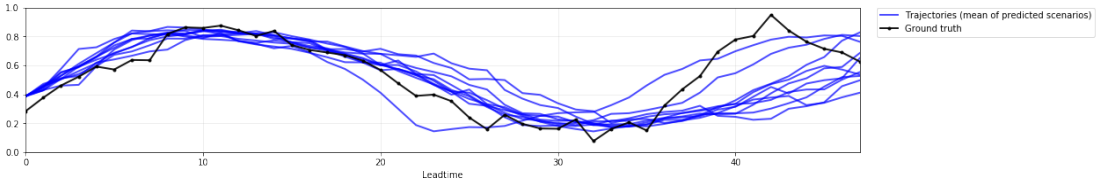
Figure 18. Roll-forward forecasting to achieve multi-step ahead forecast (from left to right forecast step 0 to h)

4.5 Probabilistic Scenario Forecasting

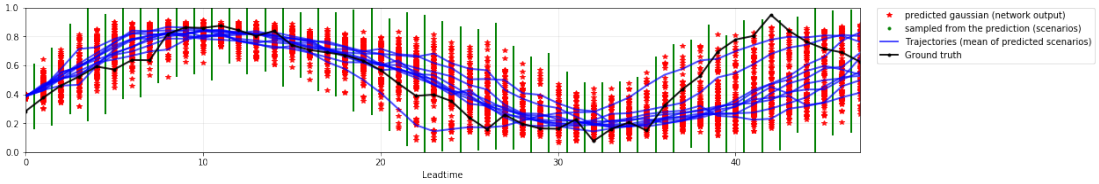
The purpose of scenario forecasting was two-fold. First, to tackle the problem of the model's ignorance. Second, to produce different forecast trajectories as another informative form of forecasting. Since the forecasts were probabilistic, one could sample from the forecast distribution and the sampled values could be used instead as the next value in the input sequence. Scenario forecasting itself was made possible thanks to MCDO variance estimation.



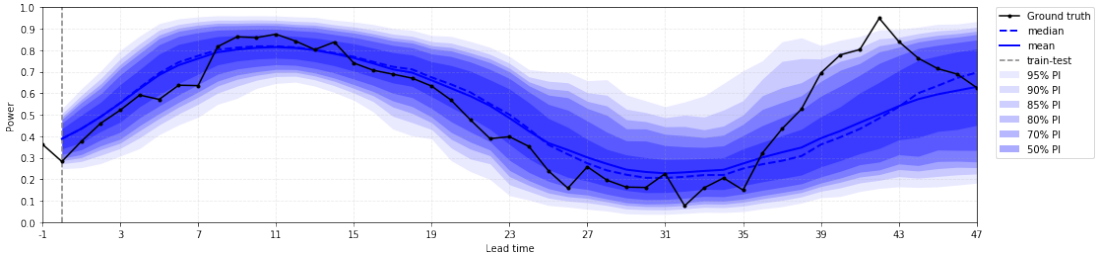
(a) pre-trajectory step



(b) generated trajectories



(c) overlay of a, and b



(d) Final Product

Figure 19. Scenario Forecasting Steps

Figure 19 demonstrates the construction of trajectories. Figure 19a corresponds to the generation of trajectories, where possible samples drew from the forecast distribution (colored by red) in order to perform scenario forecasting on these values (colored by blue). Figure 19b shows these generated trajectories. Each of these scenarios forecasts performed as part of one MCDO simulation and so were averaged over all to obtain the final result as shown in Fig. 19c. The final forecast would look like Fig. 19d.

As a conclusion, scenario forecasting was necessary for this work to achieve better-calibrated forecasts otherwise results were biased due to use of the many-to-one model for multistep ahead forecasting. It should also be emphasized that the multimodality that can be seen in the scenario forecasts were possible because of the MCDO.

5 Experiments

In this section, we explain the setup of the problem, a specific configuration of the models, datasets, baseline models, and evaluation metrics that were used for the experiments. The deep forecasting models that were explained in the earlier chapter were first built and tested on the synthetic data. Their performance was compared with the baselines so that to make sure the models were not performing worse than what could be achieved with simple heuristics. The performance and quality of the models were assessed using the evaluation metrics introduced in Section 2.

5.1 Synthetic Data

It was important to have a toy dataset that could help understand the behavior of the model and match it up with expected behavior. At the same time, since we are interested in the uncertainty and study of some particular aspects of the model, it is of greater importance to be able to diagnose the model while making sure that there is no unknown or unintended problem in the input of the models. To this end, we chose to build the models first based on some synthetic data. This not only provided a chance for building the models with an easier diagnosis but also to better understand and study the behavior of the model under different controlled input conditions.

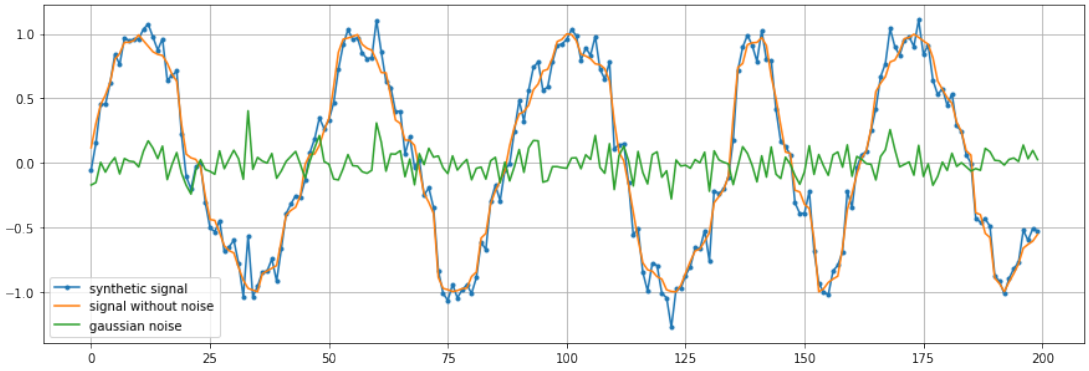


Figure 20. Generated sinusoidal signal with Gaussian noise sampled at irregular timesteps

A sinusoidal was chosen for the synthetic data as it is easy to implement, it is a relatively simple signal that is expected to be learned by the models relatively easily and with not too many samples. Also, adding uncertainty to the data could be done with an additive Gaussian noise as follows:

$$Y(t) = y(t) + \epsilon \tag{19}$$

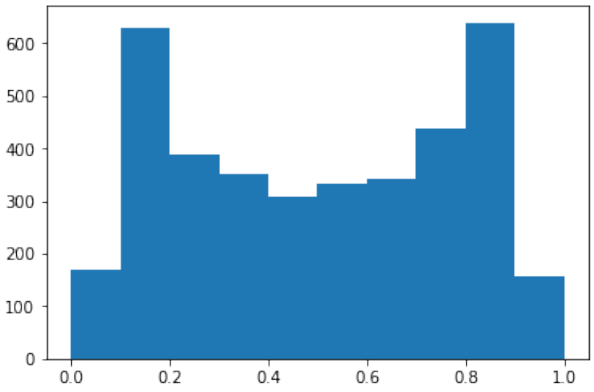


Figure 21. Distribution of the sinusoidal signal

$$
\begin{aligned}
y(t) &= Asin(\omega t + \phi) \\
\omega &= 2\pi f \\
\epsilon &\sim \mathcal{N}(\mu, \sigma^2)
\end{aligned}
$$

Frequency of the Sinusoid (f), parameters of the Gaussian noise(μ, σ^2), and amount of samples per period were the parameters that could be chosen arbitrarily. Furthermore, the sampling could be done in irregular time steps or regular. Irregular sampling was chosen as it creates another source of uncertainty and makes the series more challenging for the model to learn.

Figure 20 shows 200 samples of the synthetic dataset sampled in irregular time stops from a sinusoidal with a frequency of $f = 0.2$ with additive Gaussian noise with mean $\mu = 0$ and standard deviation of $\sigma = 0.1$. This data is then scaled to the range $[0, 1]$ to make it similar to the wind power data that will be explained in the next section. Figure 21 shows the histogram of the re-scaled sampled sinusoidal. This data is then split into train, development, and test sets for the fitting, model validation and final evaluation of the models respectively with the sizes of 2678, 1072, and 72. These splits were picked in the same order as of the original sequence of the data. The choice of size of the datasets was motivated by the data sizes available from *GEFCom'14* Wind dataset. Test dataset includes 72 data points and hence 72 lead time in the horizon to forecast, however, evaluations and demonstration were performed for 48 lead time instead. Irregular sampling performed by changing the sampling time with a random normal over the sampling timestep. A number of input sequences chose to be 24 as it gave the best result based on empirical results.

5.2 GEFCom'14: A Wind Power Forecasting Case Study

Global Energy Forecasting Competition 2014 [HPF+16] was chosen for the experiments on the real data. This competition had provided 4 different tracks including load, price, solar power, and wind power forecasting. Our experiments were focused on wind power data. History of wind power production from 10 different *wind farms* was provided by the competition organizers. Each wind farm consists of a number of wind turbines and the total power from each farm had to be forecasted. The data were categorized into 15 different tasks which were released by the organizers over a couple of months to the participants based on a schedule but that is not of any concern to our work. Our experiments are only based on TASK15 and first wind farm (wf1) which was the final task that participants were supposed to perform their best on. Besides generated power, for each wind farm, a dataset with wind speed information was provided for each wind farm. Wind speed contained information about the speed of the wind at two different heights above the ground next to the wind turbine. For each height, two measurements corresponding to *Zonal* and *Meridional* directions were given. These wind speed measurements could then be translated into angle θ and magnitude r if needed as depicted in Figure 22. Two different experiments conducted for this dataset, one without the wind speed information, and one with the wind speed, to see the influence. However, according to Eq. 20 it is expected that much better results be obtained as a result of employing the wind speed. The dates of the data for TASK15 begins from *2012-01-01 01:00:00* and ends on *2013-12-01 00:00:00*, roughly 1 year and 11 months. Frequency of the data is hourly. Also, dataset had some null values for the power forecast only in wf1, wf2, and wf3 which in total they summed to be 112 instances, which then filled with `ffill` null value[7].

We chose following dates for the train, development and test sets according to the [HG18]:

- train: '2012-01-01' to '2012-08-01
- development: '2012-08-01' to '2012-10-01
- test: '2013-10-01' to '2013-10-02' (48 hours)

$$P = \frac{1}{2} A \rho v^3 \tag{20}$$

Equation 20 shows the relation of wind power per unit of time, where A is the area of wind contact to the turbine's rotor, ρ is the density of the air, and v is the speed of the wind [GG05].

[7]Data exploration and cleaning adopted from `https://github.com/greenlytics/gefcom2014-wind`

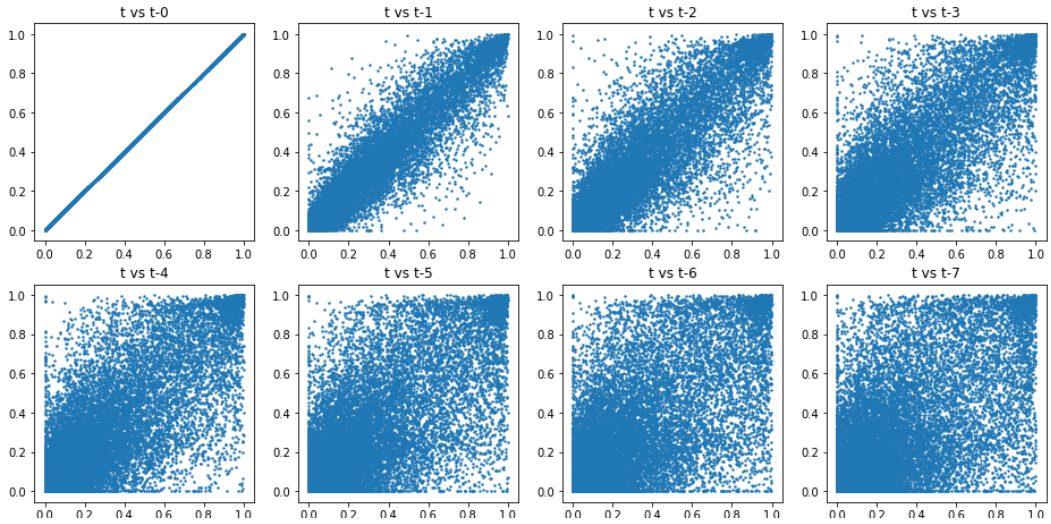
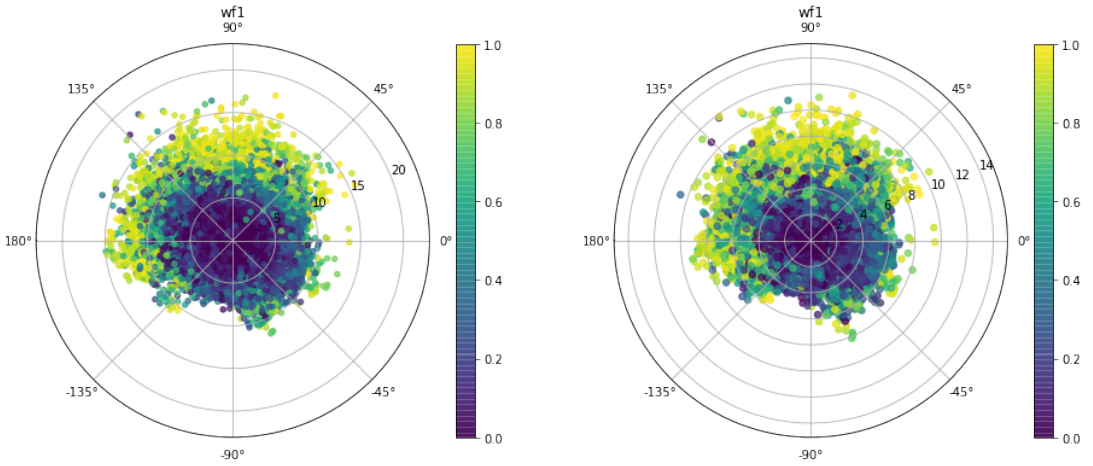


Figure 23. Autocorrelation of the wind power wf1 with lag 7



(a) Wind Speed Distribution at height 100 Meters above the ground

(b) Wind Speed Distribution 10 Meters above the ground

Figure 22. Wind speed distribution for wf1 with respect to the power, the higher the power the lighter the points become and vice versa.

The length of the input sequence for the warmup chose to be 4 (meaning 4 hours) as it was a right balance between performance and speed of training. As Fig. 23 also shows this choice was supported by the fact that the auto-correlation of power is high enough only for 4 lags.

5.3 Implementation Details

The `TimeSynth` was used with parameters of the sampler `stop_time`$= 500$, `num_points`$= 15000$, and `keep_percentage`$= 500$. The first parameter indicates the clock for sampling, the second is for the number of data points, and the last, for the number of samples to be kept (the less the sparser the samples from the original signal). And finally, the range of the values scaled between 0 and 1 as an arbitrary desired range for the data. Frequency of the sinusoidal set to $f = 0.2$ with additive Gaussian noise with mean $\mu = 0$ and standard deviation of $\sigma = 0.1$.

To pace up the learning, a Cyclical Learning Rate scheduler used with both of the models explained in Section 5. As demonstrated and discussed in [Smi15] this learning rate scheduler helps to traverse the saddle points in the weight space faster and so it paces up the learning process. Cyclical Learning rate Scheduler with parameters $gamma = 0.8$, and the lowest and highest rate of $1e - 5$ and $1 - e2$, respectively.

The number of units for both MLP and RNN models were set to 100. Dropout mask with rate 0.3 applied both for the regularization and for the epistemic uncertainty estimation. The batch size for all experiments set to 32. For the RNN, recurrent dropout was applied only. All models were run for 10 epochs with Adam optimizer and Cyclical Learning Rate Scheduler. Activation function for the first layer set to default that is *linear* for the *Dense layer* and *tanh* for the *RNN layer* with GRU cells. Activation of the first dense output that estimate the `mean` set to *hard sigmoid* that proved to be slightly better over *sigmoid*. Also, activation of the second dense output that estimates the logarithm of the variance set to linear. Input and target sequences were generated using Keras' `TimeSeriesGenerator` and a modified version of it for the model with two outputs.

The code for producing the results and materials presented in this thesis can be found in Github repository[8]. Here is an overview of the packages that were used in the implementation of this work:

- `Keras` version 2.2.4 was used for the implementation of neural networks [C+15]
- From `Scipy` version 1.2.1, `stats` was used for implementation of probabilistic forecasts [JOP+]
- `Numpy` version 1.13.3 was used for manipulation of the data arrays [Oli06]
- `Pandas` version 0.24.2 was used for reading the data from the `csv` files, manipulation time series and dataframes [McK10]
- `sklearn` version 0.20.3 was used for MSE and MAE measurements and time series splitting [PVG+11]
- `matplotlib` version 3.0.3 was used for the plots and visualizations [Hun07]
- An implementation of *Cyclical Learning Rate Scheduler* compatible with `Keras` was used [9]
- `TimeSynth`[10] was used to generate Synethtic dataset

[8] `https://github.com/novinsh/master_thesis`
[9] `https://github.com/bckenstler/CLR`
[10] `https://github.com/TimeSynth/TimeSynth`

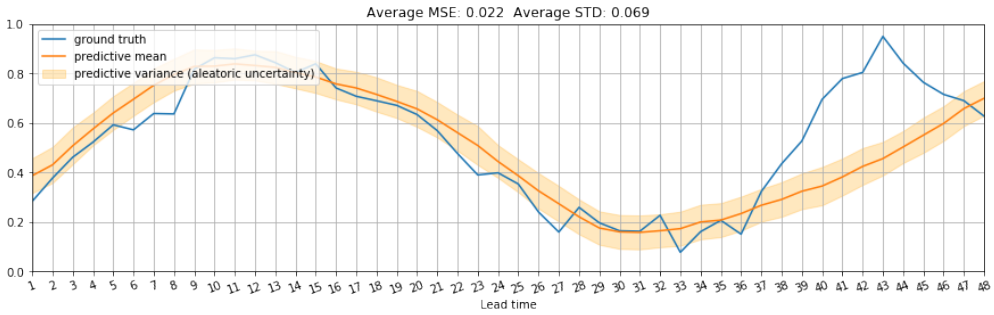
6 Results

All models were fitted to the training dataset and were validated over 20 different splits of the development. At all time, models were fed by input sequence of size 24 for the Synthetic data and of size 4 for the *GEFCom'14* data set. Input sequences for the purpose of validation were not revealed to the models during training time. Models had to forecast a horizon of size 48, or this amount of time-steps ahead of the input sequence. Two network architectures, MLP and RNN were tested and their results are being reported here. Moreover, 5 different variation of each model per each architecture studied and their results are being reported here. 4 of which correspond to the Bayesian models and 1 to a non-Bayesian model. For the baseline, two variations of Quantile Naive was used. One over all the dataset denoted as `QNaiveX` and one with only the warmup sequence denoted as `QNaiveW`. Beside the baselines, the Non-bayesian model denoted as `old` model used as another reference for comparing the point forecasts performance. The Bayesian models are denoted as `AL` for Aleatoric, the `EP` for Epistemic, the `ALEP` for Aleatoric+Epistemic, and `SFALEP` to for the Scenario Forecast on Aleatoric+Epistemic models. Figure 24 demonstrates how each of the Aleatoric 24a and Epistemic 24b uncertainties demonstrated. The final output of a complete model would add both of these uncertainties together as in `ALEP` or `SFALEP` case, that are being demonstrated as fan charts such as in Fig 31c and 31d.

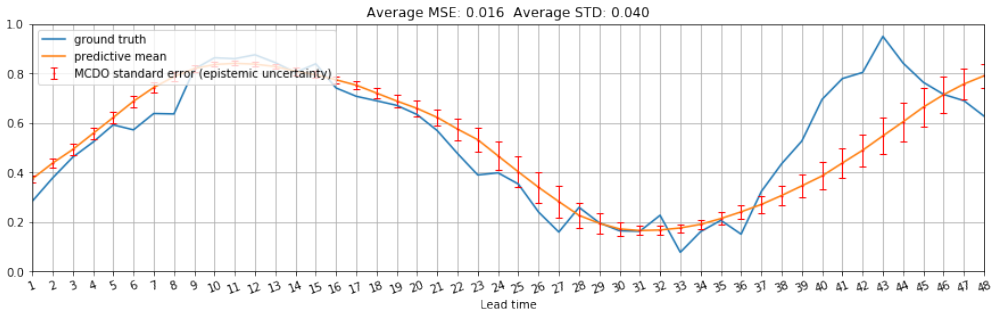
As for the evaluation metrics, CRPS was used primarily for the probabilistic forecasts. Beside CRPS, MAE and MSE were used as additional metrics for comparison with other works, as well as the old model. It is noteworthy that the median of the probabilistic forecasts was used to evaluate them across MSE and MAE.

6.1 Synthetic data

Figure 25a shows the results obtained from the RNN architecture. It can be seen that the `QNaiveX` achieves better MSE with much lower variance compared to the other two baselines. `no-noise` is the sampled signal without Gaussian noise. It was used as a lower bound that the models would ideally be able to get an error as low as this. On Synthetic dataset, Bayesian models were not able to perform much better and even in some cases even did slightly worse than the `old` model. The `SFALEP` model however usually reduced the variance if not performing better on the mean error compared to the `old` model. Figure 25b demonstrates the execution time of each model for the forecasting. This pattern can be seen in almost all other experiments as well, therefore we omit demonstrating the execution time for the rest of the experiment. The `SFALEP` higher execution time is due to the Monte-Carlo simulation, as well as the scenario forecasting. This can be reduced by running the simulations concurrently, yet the scenario forecasting execution is sequential and its execution time could not be reduced by running it in concurrently.



(a) Aleatoric Uncertainty (AL model)



(b) Epistermic Uncertainty (EP model)

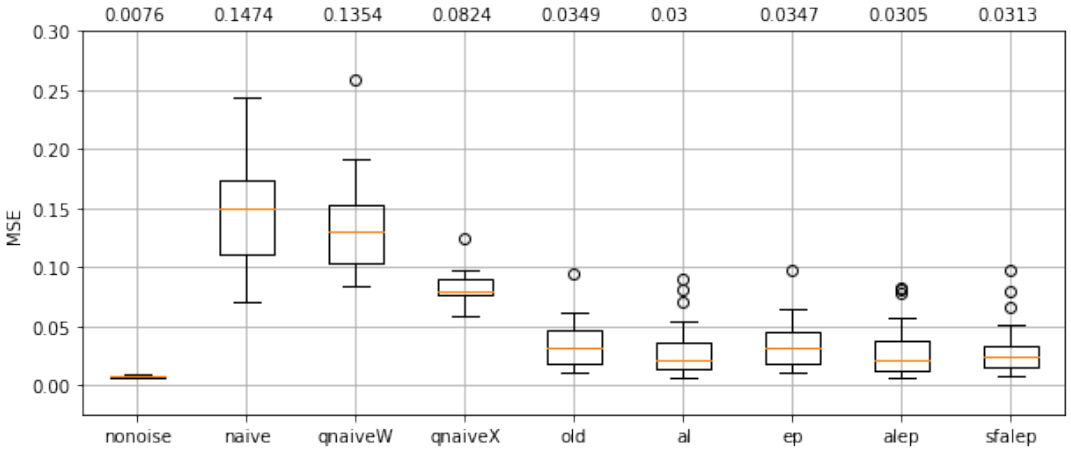
Figure 24. Demonstration of Aleatoric and Epistemic Uncertainty

Table 1 shows a numerical view on the performance of each model across different metrics and architectures. Figure 26 illustrates the same with one additional metric, the MAE. Based on these results for the synthetic dataset RNN did not improve results much over the MLP.

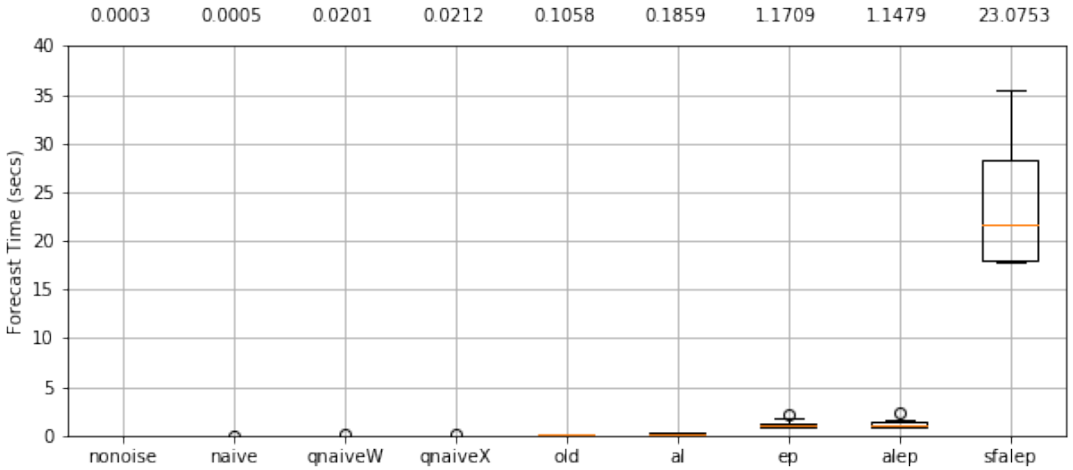
Metrics	CRPS		MSE	
Architectures	MLP	GRU	MLP	GRU
no-noise	N/A	N/A	0.007	0.007
old	N/A	N/A	0.033 ± 0.020	0.035 ± 0.021
QNaiveX	0.163 ± 0.055	0.163 ± 0.055	0.082 ± 0.014	0.082 ± 0.014
QNaiveW	0.210 ± 0.129	0.210 ± 0.129	0.135 ± 0.042	0.135 ± 0.042
AL	0.116 ± 0.100	0.103 ± 0.105	0.033 ± 0.019	0.030 ± 0.024
EP	0.131 ± 0.106	0.132 ± 0.111	0.033 ± 0.019	0.035 ± 0.022
ALEP	0.112 ± 0.095	0.103 ± 0.104	0.033 ± 0.019	0.030 ± 0.024
SFALEP	0.107 ± 0.086	$\mathbf{0.098 \pm 0.085}$	0.034 ± 0.019	$\mathbf{0.031 \pm 0.024}$

Table 1. Results obtained on the Synthetic Data

In Figure 26 the error bars are the standard deviation of the metrics and the bars

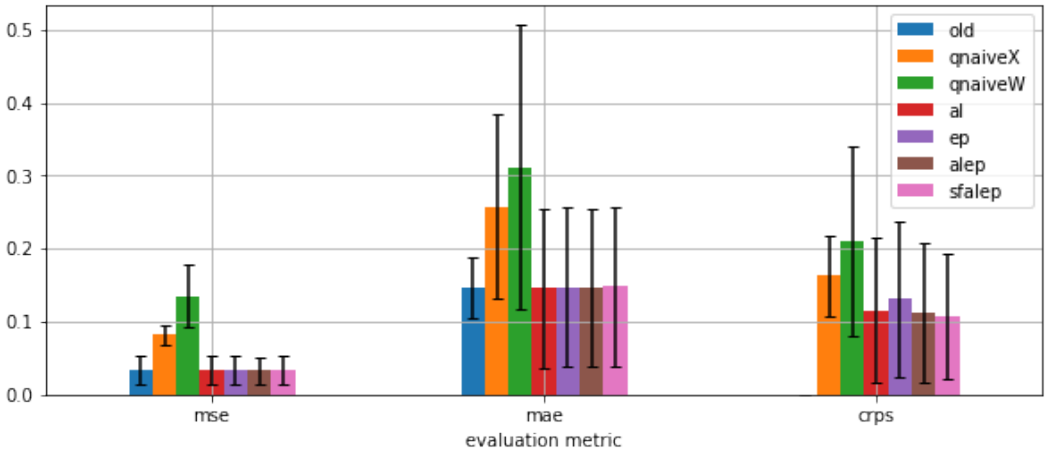


(a) Comparison of models MSE on development set (numbers on the top, average of MSEs for each model)

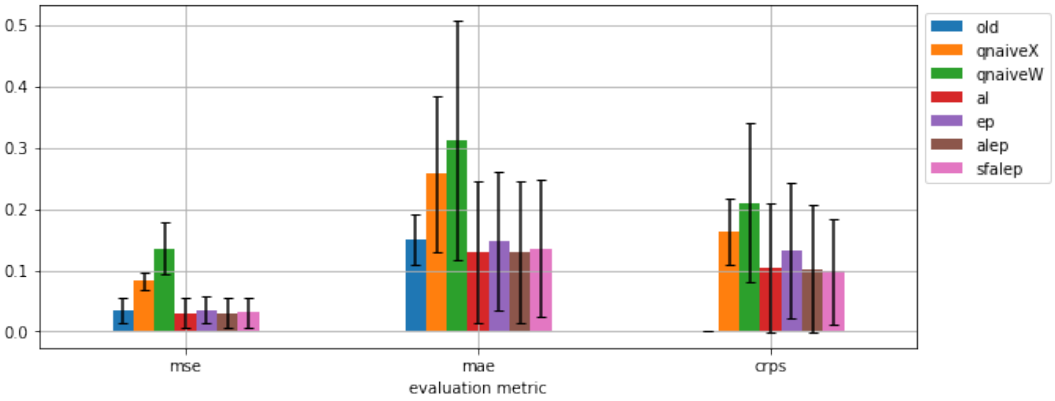


(b) Comparison of models run-time during forecasting on the development set (numbers on the top, average of runtime for each model)

are the mean value for that metric. These results obtained from running the models on different splits of the development set. As can be seen in this figure as well as the table, there is no much difference between the performance of the MLP and the RNN architectures, yet RNN is slightly better.



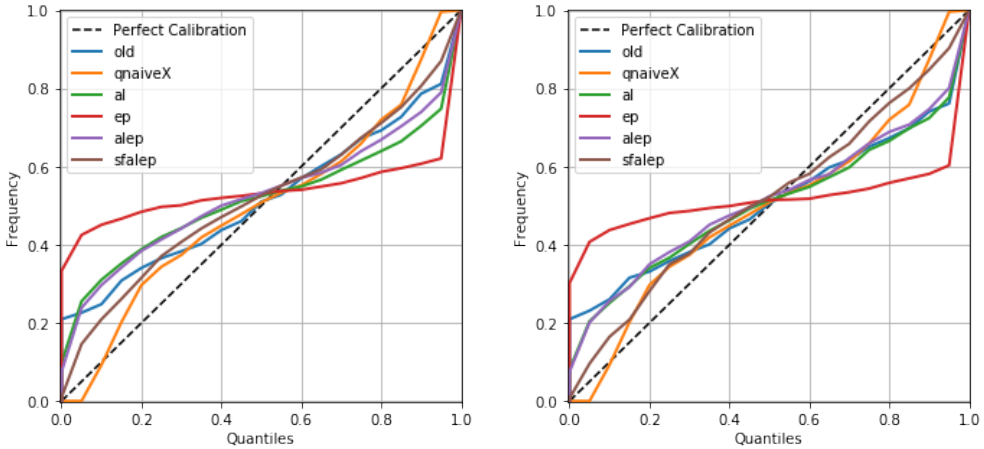
(a) MLP architecture



(b) RNN architecture

Figure 26. Performance on Synthetic data across different metrics and models

Figure 27 shows the reliability/calibration plots for each of the models. These plots produced by taking all forecasted variables under the same posterior predictive distribution. EP was the most miscalibrated model for both cases. SFALEP was better calibrated compared to all other methods at the RNN case, and so RNN should be preferred over the MLP architecture, although the performance gain was not significant.

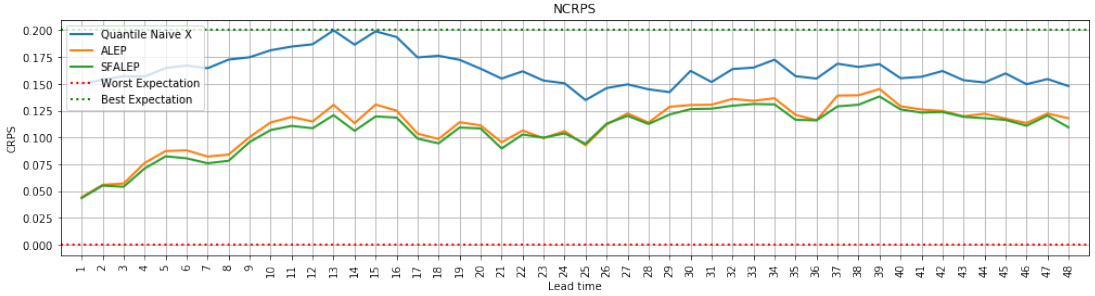


(a) MLP (b) RNN

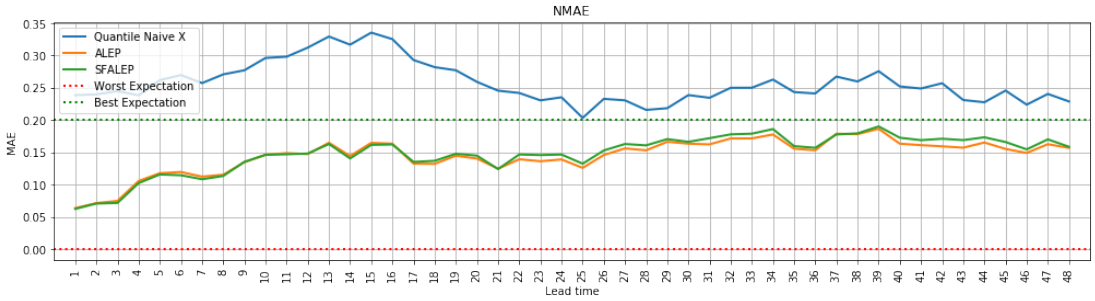
Figure 27. Calibration of models on Synthetic data

6.1.1 Stepwise Evaluation and Improvement

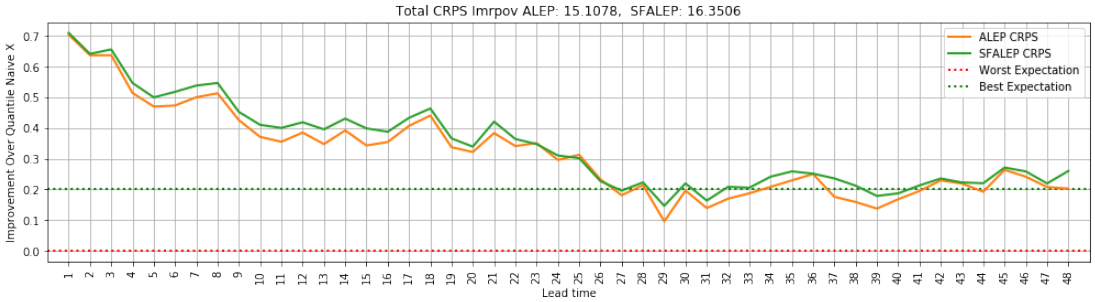
Performance of the forecasts for each lead time averaged over 20 splits of the development set and demonstrated in Figures 28 and 29 for MLP and RNN cases, respectively. In each of these figures, the top two plots show the NCRPS and NMAE with the worst and best expectation which were set as an ideal case based on the results achieved by [HG18]. Therefore, the expectation is to have a model that does not do worse than the red dotted line and ideally perform with zero error (the green dotted line). The same applied for the improvement plots, the two bottom plots. Going below the red line or $y = 0$ for the improvement plots not only means that the forecast was not an improvement over the `QNaiveX` but also performed worse. As it can be observed in these two figures, none of the models performed worse than expected and almost for all the horizon it is catching up with the best performance expected. However, the caveat is that this expectation was not initially set for the Synthetic data and due to the simplicity of this datasets nothing much strong could be said until looking at the results on the real data. SFALEP shows better performance in later lead times in the horizon.



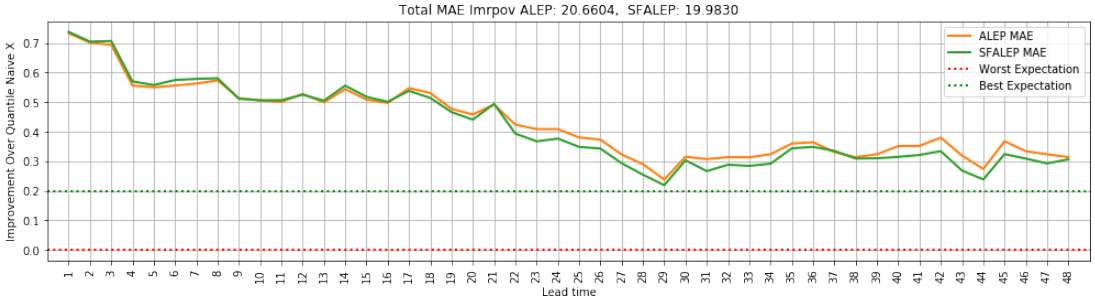
(a) NCRPS accross the forecasted horizon



(b) NMAE accross the forecasted horizon

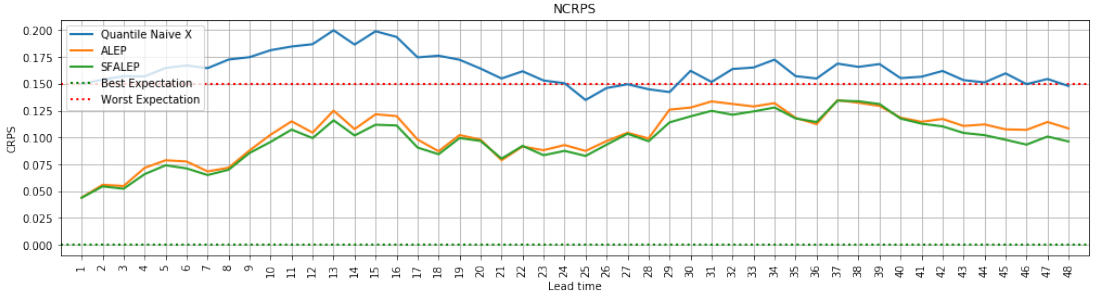


(c) NCRPS improvement over `QNaiveX`

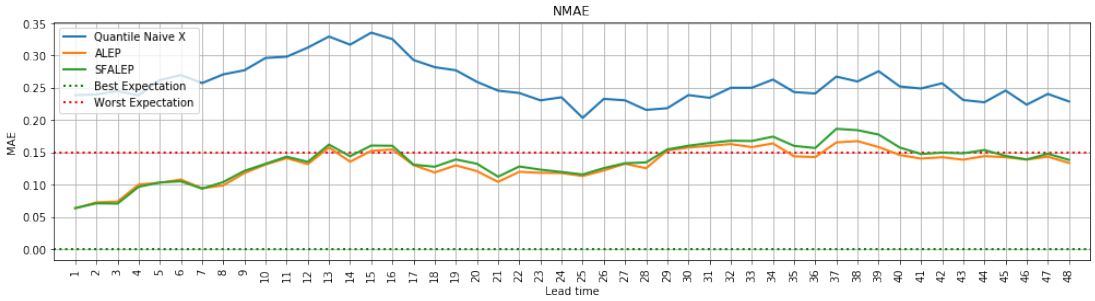


(d) NMAE improvement over `QNaiveX`

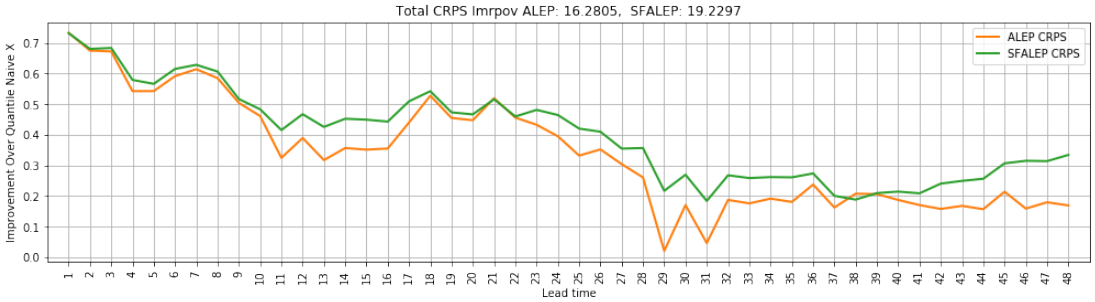
Figure 28. Stepwise Evaluation plots for MLP Forecasts



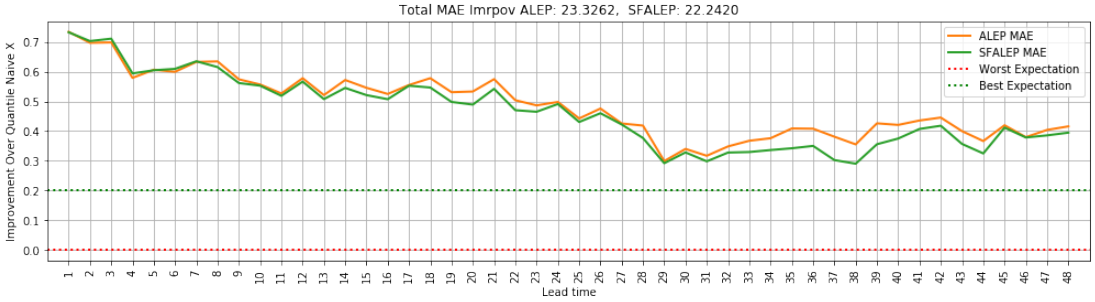
(a) NCRPS accross the forecasted horizon



(b) NMAE accross the forecasted horizon



(c) NCRPS improvement over `QNaiveX`

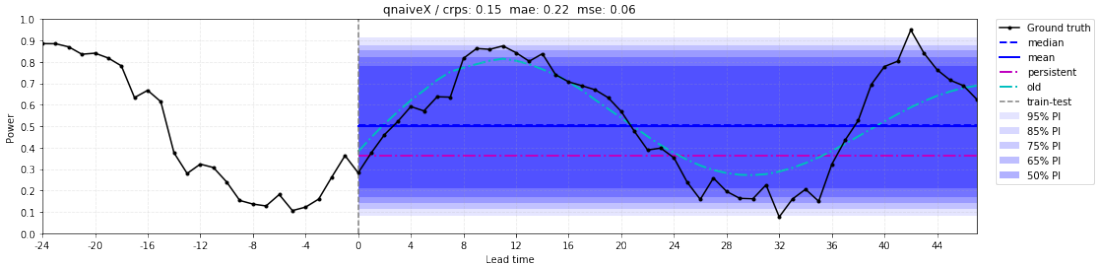


(d) NMAE improvement over `QNaiveX`

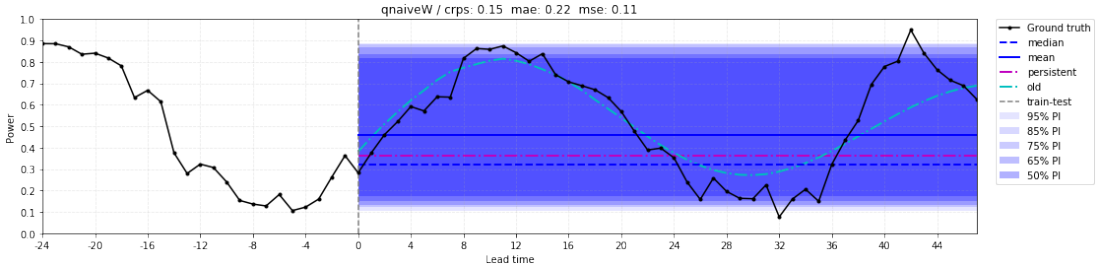
Figure 29. Stepwise Evaluation plots for RNN Forecasts

6.1.2 Fancharts for Probabilistic Forecasts on Test set

Probabilistic forecasts of the models on the Synthetic test dataset visualized using fan-charts. Each fan corresponding to a quantile of the predictive distribution for that lead time. Persistence or the naive (not the quantile naive) also plotted, together with the `old` (or non-Bayesian) model forecast. Figure 30 shows the QNaive forecast, Figure 31 and 32 show the MLP and RNN forecasts, respectively.

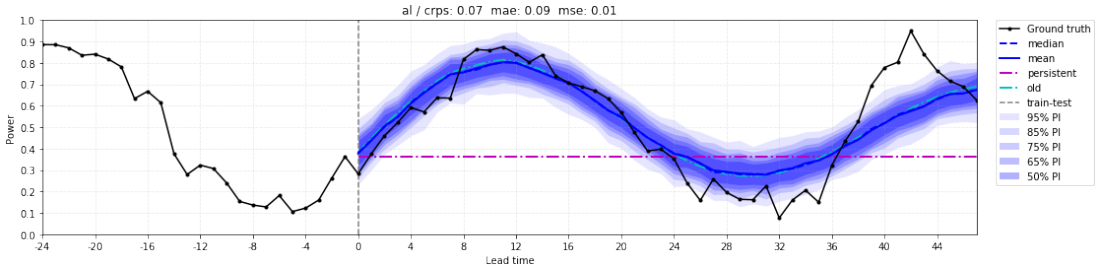


(a) `QNaiveX` forecast on test set

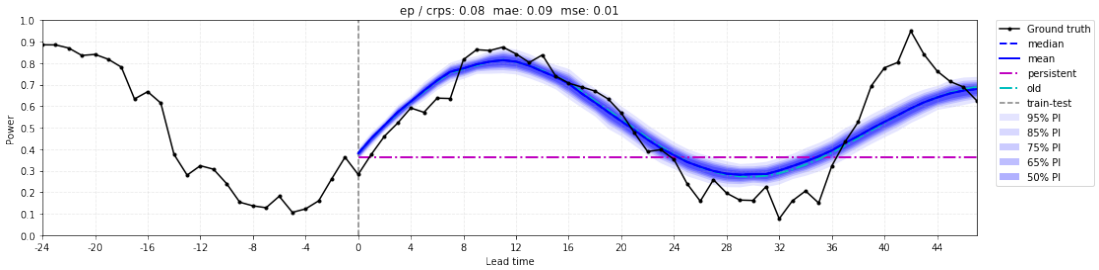


(b) `QNaiveW` forecast on test set

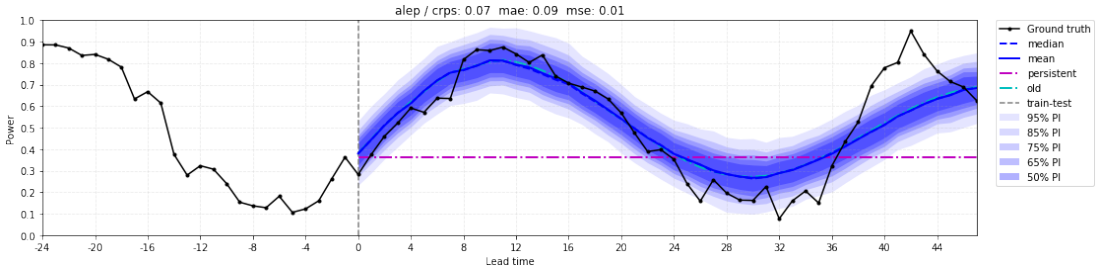
Figure 30. Forecast plots for QNaive on Synthetic data



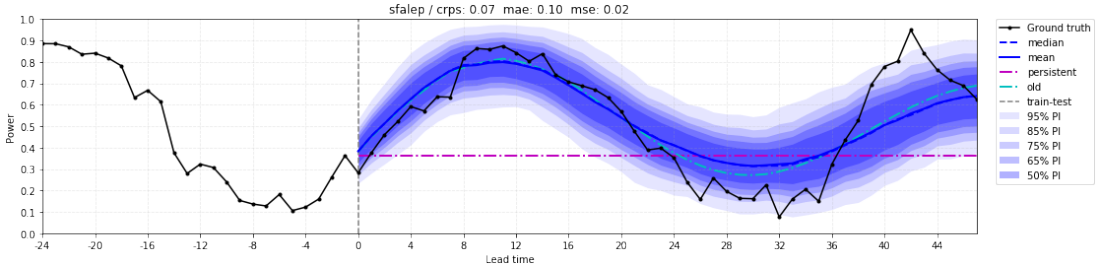
(a) Model Aleatoric (AL) forecast on test set



(b) Model Epistemic (EP) forecast on test set

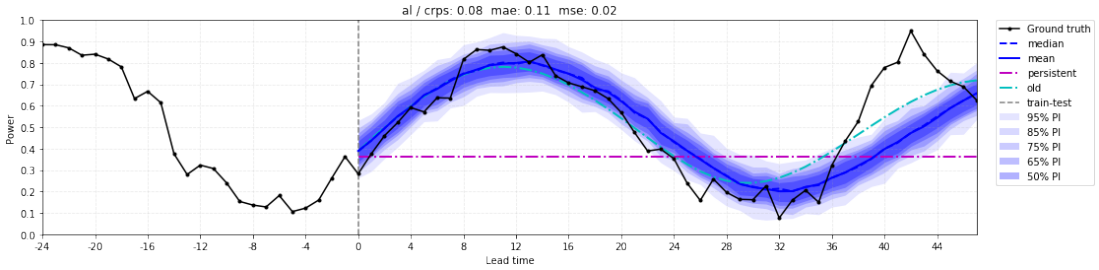


(c) Model AL+EP forecast on test set

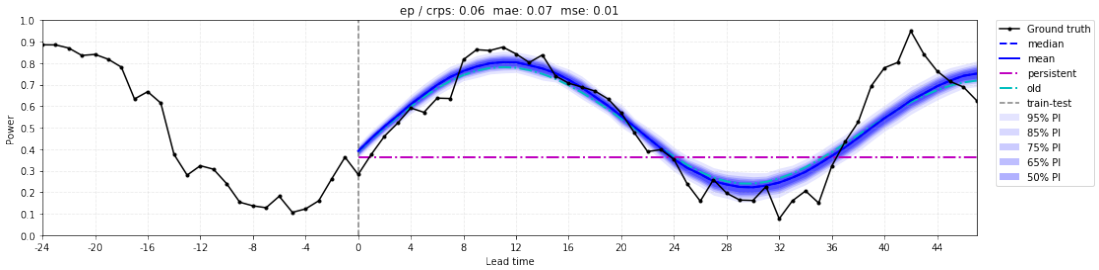


(d) Model AL+EP with Scenario Forecasting on test set

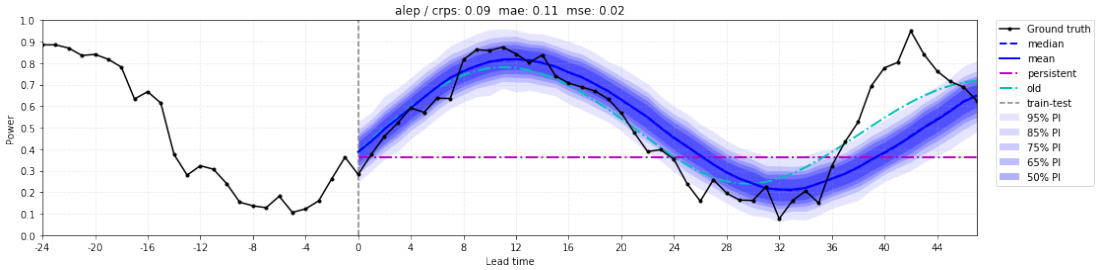
Figure 31. Forecast plots for MLP on Synthetic dataset



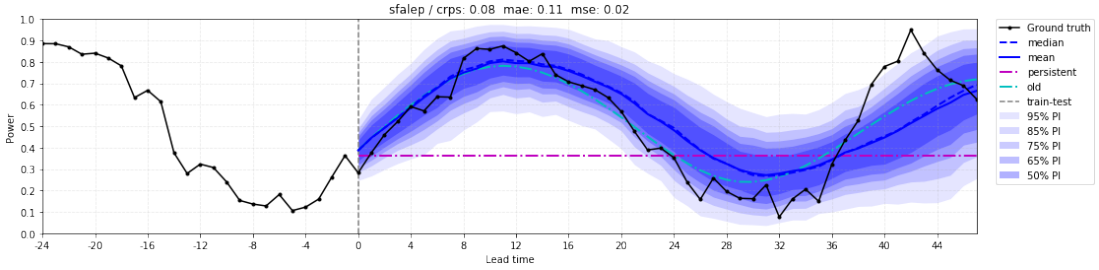
(a) Model Aleatoric (AL) forecast on test set



(b) Model Epistemic (EP) forecast on test set



(c) Model AL+EP forecast on test set



(d) Model AL+EP with Scenario Forecasting on test set

Figure 32. Forecast plots for RNN on Synthetic dataset

6.2 GEFCom Data

Results obtained from applying sample architectures and experiments on the GEFCom dataset are being demonstrated in this section. These results also include a multivariate case where it is reported separately. Table 2 and Table 3 give an overview over the NCRPS and MSE results obtained from this dataset, respectively.

NCRPS	Univariate		Multivariate	
Architectures	MLP	GRU	MLP	GRU
old	N/A	N/A	N/A	N/A
QNaiveX	0.211 ± 0.103	0.211 ± 0.103	0.209 ± 0.092	0.209 ± 0.092
QNaiveW	0.266 ± 0.246	0.266 ± 0.246	0.266 ± 0.256	0.266 ± 0.246
AL	0.215 ± 0.157	0.218 ± 0.155	0.167 ± 0.121	0.169 ± 0.150
EP	0.247 ± 0.197	0.263 ± 0.186	0.196 ± 0.154	0.157 ± 0.141
ALEP	0.208 ± 0.147	0.217 ± 0.150	$\mathbf{0.166 \pm 0.097}$	0.159 ± 0.137
SFALEP	$\mathbf{0.203 \pm 0.149}$	$\mathbf{0.183 \pm 0.097}$	0.173 ± 0.104	$\mathbf{0.144 \pm 0.099}$

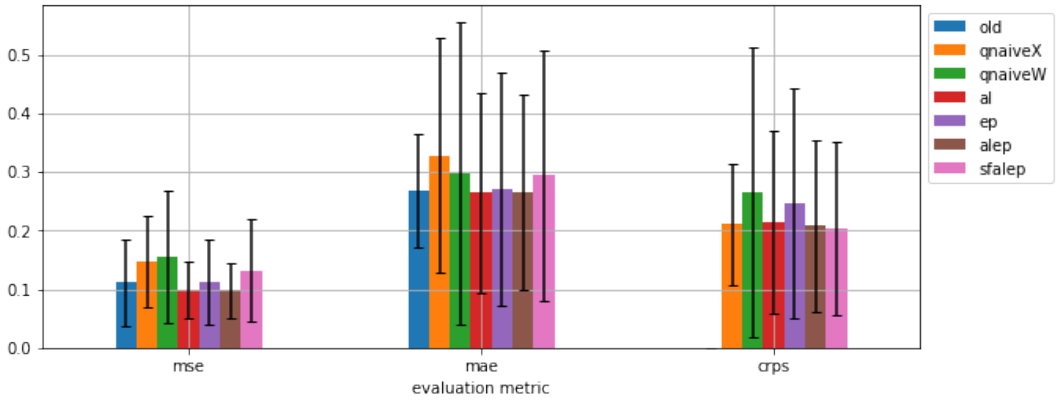
Table 2. NCRSP results obtained on the GEFCom'14 Dataset

MSE	Univariate		Multivariate	
Architectures	MLP	GRU	MLP	GRU
old	$\mathbf{0.112 \pm 0.073}$	0.107 ± 0.063	$\mathbf{0.084 \pm 0.062}$	$\mathbf{0.050 \pm 0.035}$
QNaiveX	0.147 ± 0.079	0.147 ± 0.079	0.136 ± 0.062	0.136 ± 0.062
QNaiveW	0.156 ± 0.114	0.156 ± 0.114	0.156 ± 0.114	0.156 ± 0.114
AL	0.099 ± 0.048	0.103 ± 0.051	0.080 ± 0.059	0.075 ± 0.048
EP	0.113 ± 0.073	0.107 ± 0.064	$\mathbf{0.080 \pm 0.058}$	$\mathbf{0.050 \pm 0.036}$
ALEP	$\mathbf{0.098 \pm 0.047}$	$\mathbf{0.102 \pm 0.046}$	0.082 ± 0.046	0.068 ± 0.042
SFALEP	0.132 ± 0.087	$\mathbf{0.103 \pm 0.037}$	0.090 ± 0.055	0.063 ± 0.044

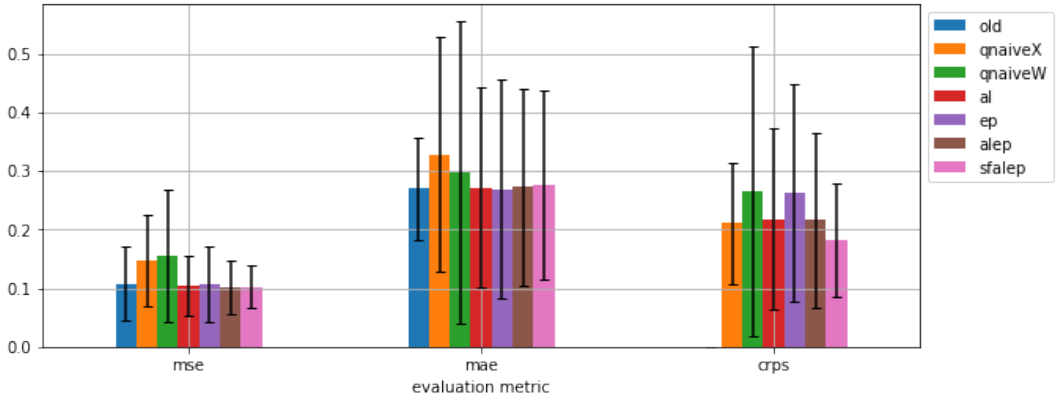
Table 3. MSE results obtained on the GEFCom'14 Dataset

6.2.1 Univariate

In Figure 33 the error bars are the standard deviation of the metrics and the bars are the mean value for that metric. These results obtained from running the models on different splits of the development set. The performance on RNN architecture is obviously better than the MLP, especially, with respects to the variance of the errors.



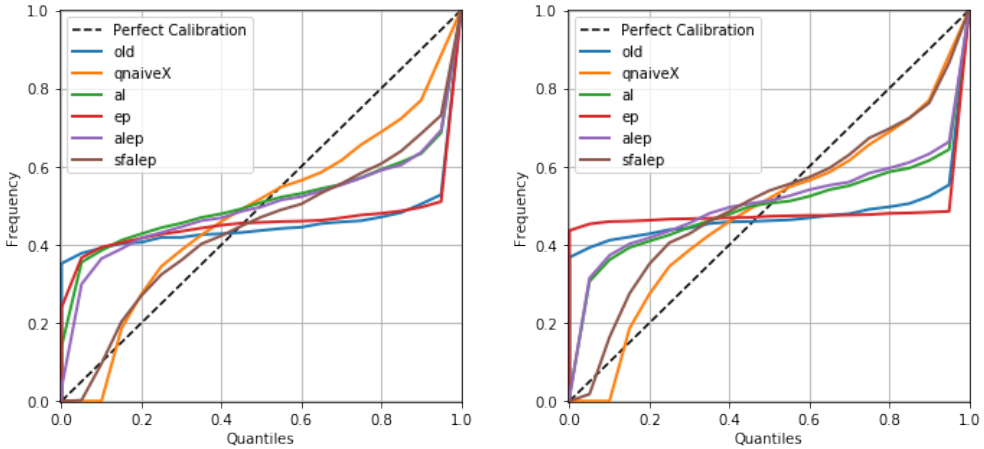
(a) MLP architecture



(b) RNN architecture

Figure 33. Performance on GEFCom data across different metrics and models

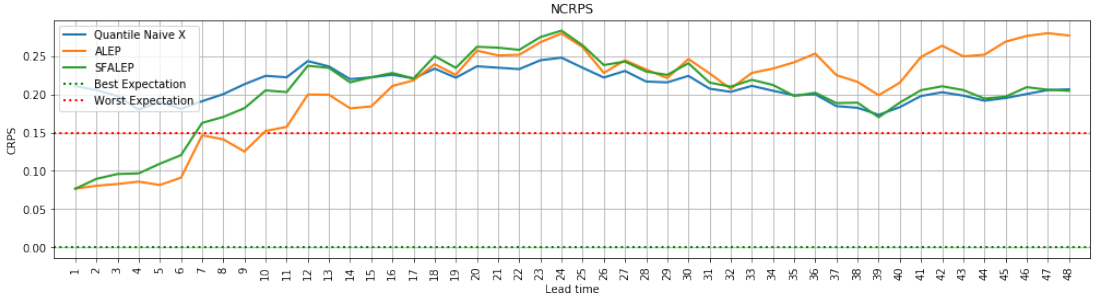
Figure 34 shows the reliability/calibration plots for each of the models. These plots produced by taking all forecast variables under the same posterior predictive distribution. `EP` was the most miscalibrated model for both cases. `SFALEP` was better calibrated from MLP to RNN case, and overall it was the second best-calibrated model after `QNaiveX`. RNN should also be preferred over the MLP architecture both for the sake of NCRPS performance as well as calibration of the model.



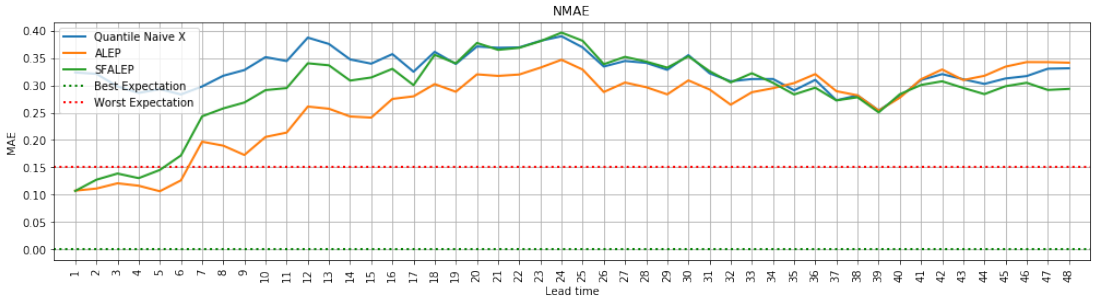
(a) MLP (b) RNN

Figure 34. Calibration of models on GEFCom data

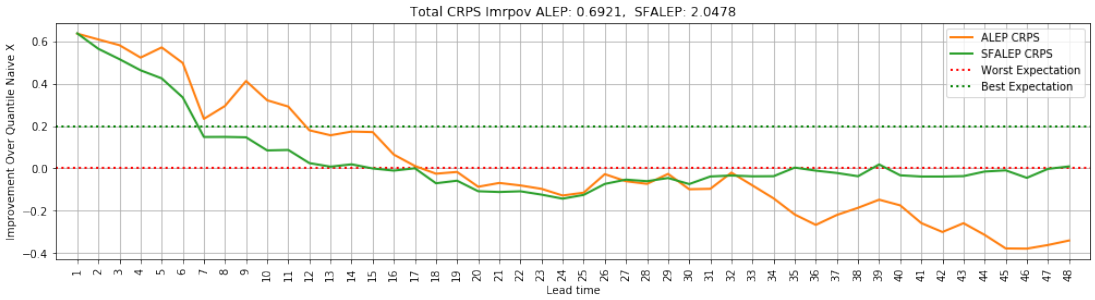
Stepwise Evaluation and Improvement Performance of the forecasts for each lead time averaged over 20 splits of the development set and demonstrated in Figures 35 and 36 for MLP and RNN cases, respectively. In each of these figures, the top two plots show the NCRPS and NMAE with the worst and best expectation which were set as an ideal case based on the results achieved by [HG18]. Therefore, the expectation is to have a model that does not do worse than the red dotted line and ideally perform with zero error (the green dotted line). The same applied for the improvement plots, the two bottom plots. Going below the red line or $y = 0$ for the improvement plots not only means that the forecast was not an improvement over the `QNaiveX` but also performed worse. Both models were within the expected region for 7 hours ahead but after that performing worse than expected, and even after 12 hours the improvement became either very subtle or negative. The gap between the `SFALEP` and `ALEP` is more considerable for the NCRPS. And that proves the effectiveness of the approach taken in `SFALEP`. This gap is apparent for `MSE` however.



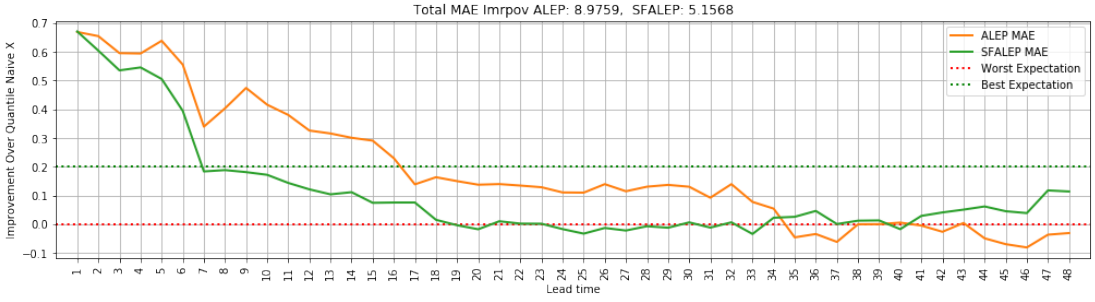
(a) NCRPS accross the forecasted horizon



(b) NMAE accross the forecasted horizon

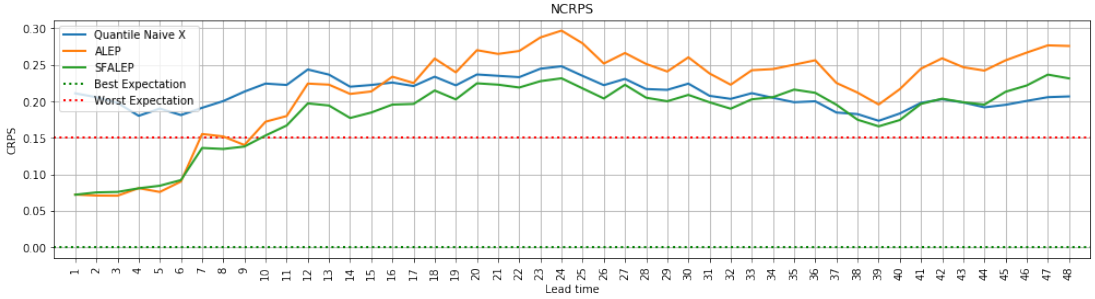


(c) NCRPS improvement over `QNaiveX`

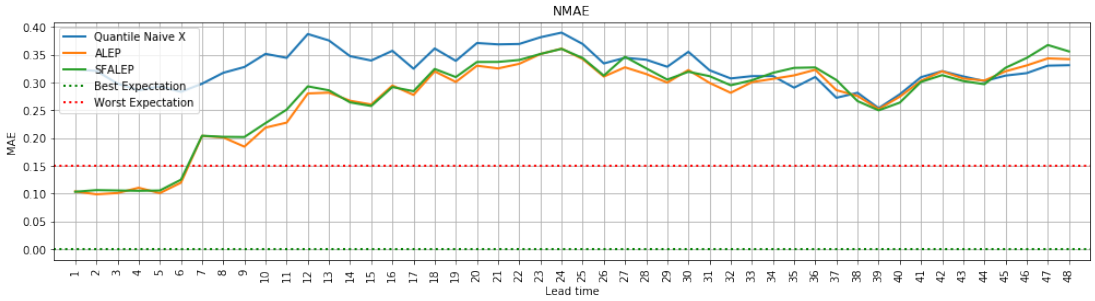


(d) NMAE improvement over `QNaiveX`

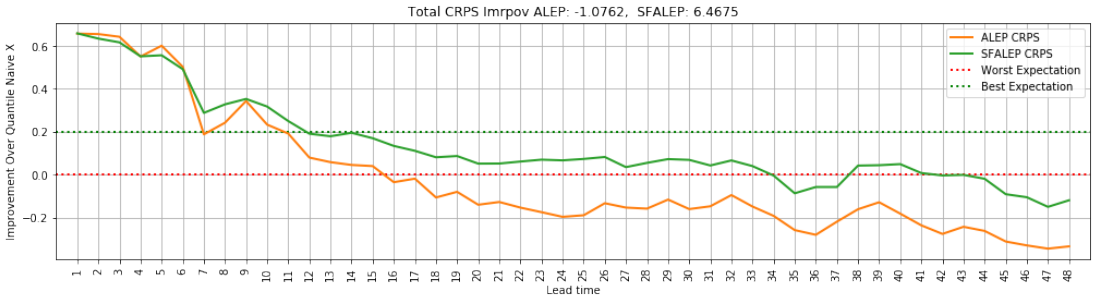
Figure 35. Stepwise Evaluation plots for MLP Forecasts



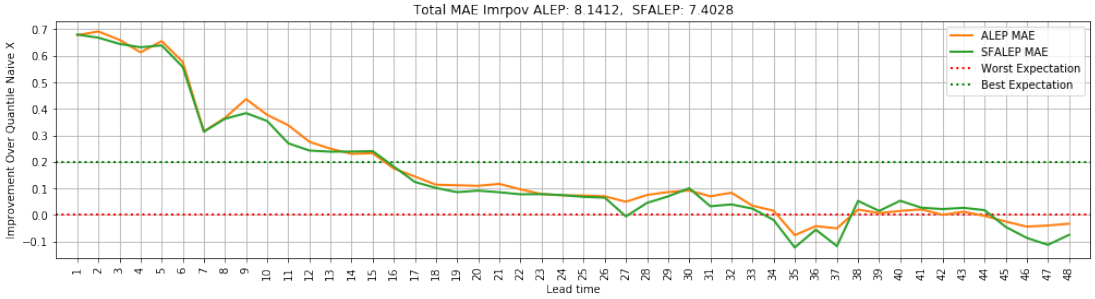
(a) NCRPS accross the forecasted horizon



(b) NMAE accross the forecasted horizon



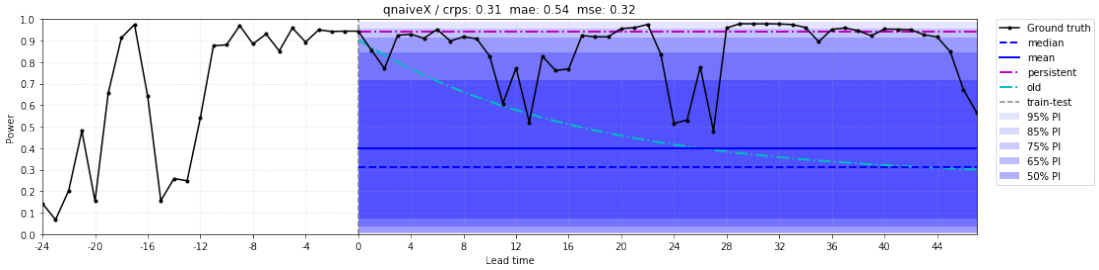
(c) NCRPS improvement over `QNaiveX`



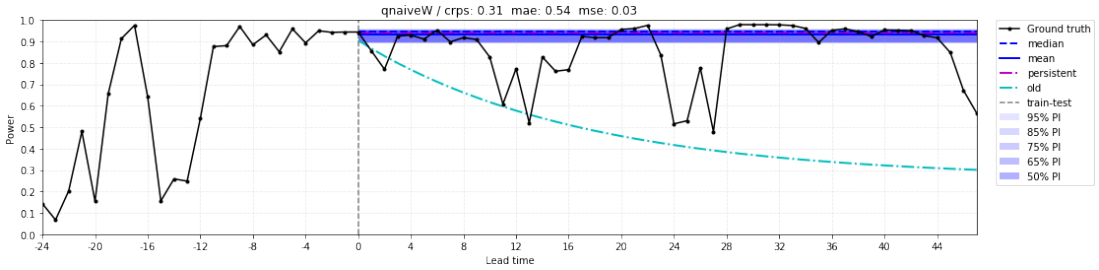
(d) NMAE improvement over `QNaiveX`

Figure 36. Stepwise Evaluation plots for RNN Forecasts

Fancharts for Probabilistic Forecasts on Test set Probabilistic forecasts of the models on the univariate GEFCom test set visualized using fan-charts. Each fan corresponding to a quantile of the predictive distribution for that lead time. Persistence or the naive (not the quantile naive) also plotted, together with the `old` (or non-Bayesian) model forecast. Figure 37b shows the QNaive forecast, Figure 38 and 39 show the MLP and RNN forecasts, respectively.

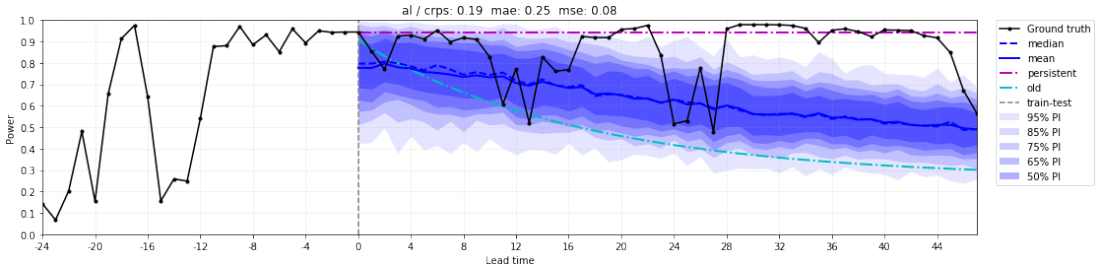


(a) `QNaiveX` forecast on test set

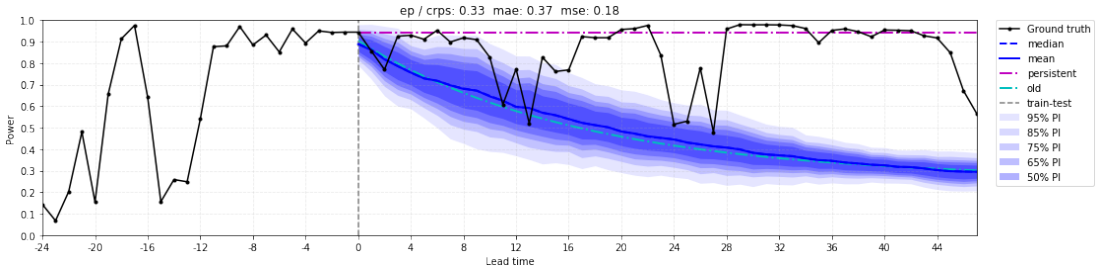


(b) `QNaiveW` forecast on test set

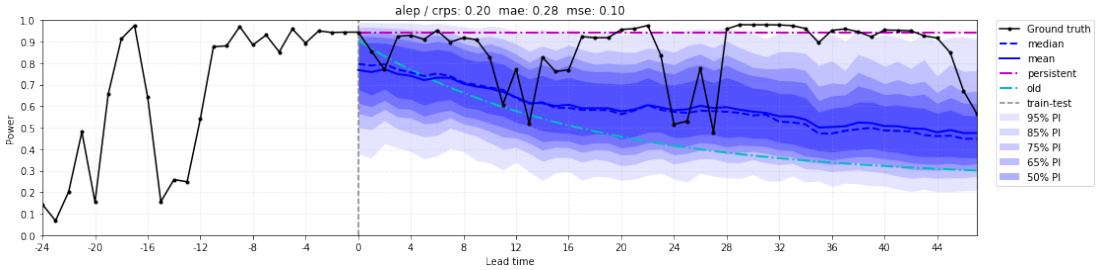
Figure 37. Forecast plots for QNaive on GEFCom dataset



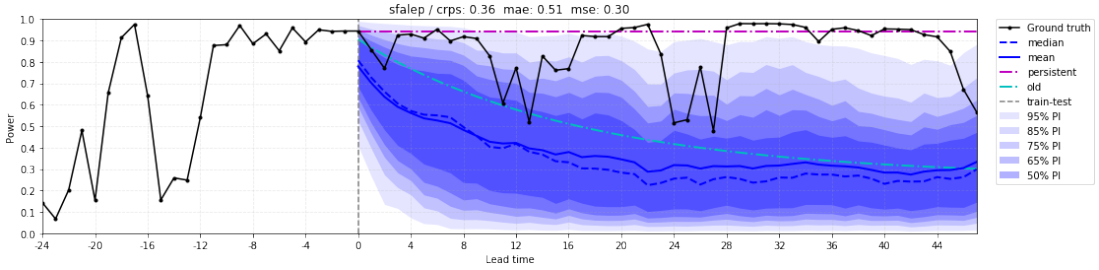
(a) Model Aleatoric (AL) forecast on test set



(b) Model Epistemic (EP) forecast on test set

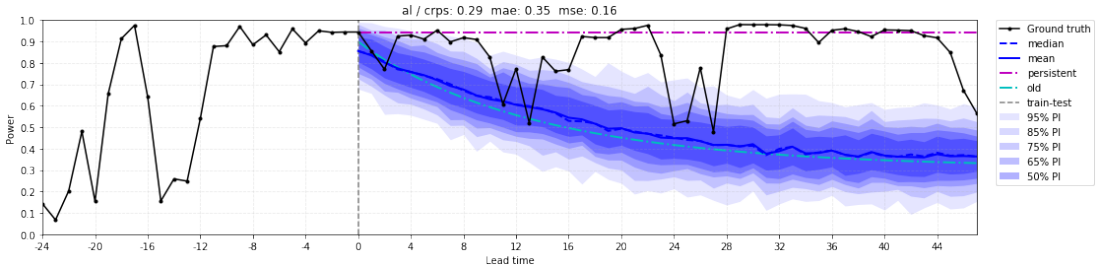


(c) Model AL+EP forecast on test set

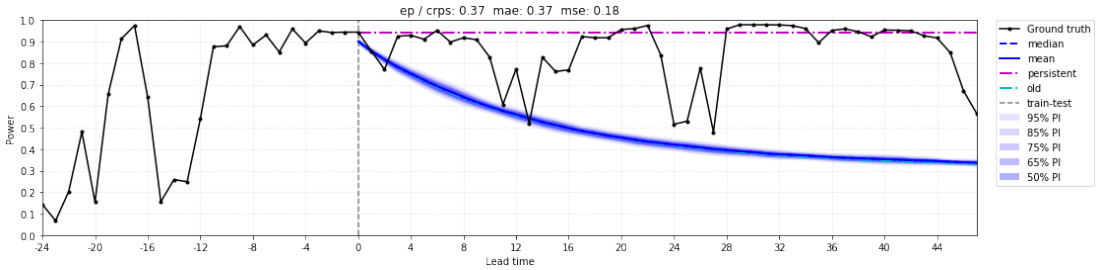


(d) Model AL+EP with Scenario Forecasting on test set

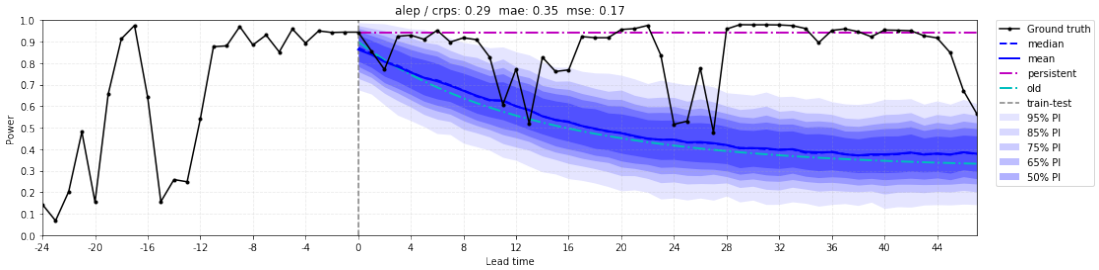
Figure 38. Forecast plots for MLP



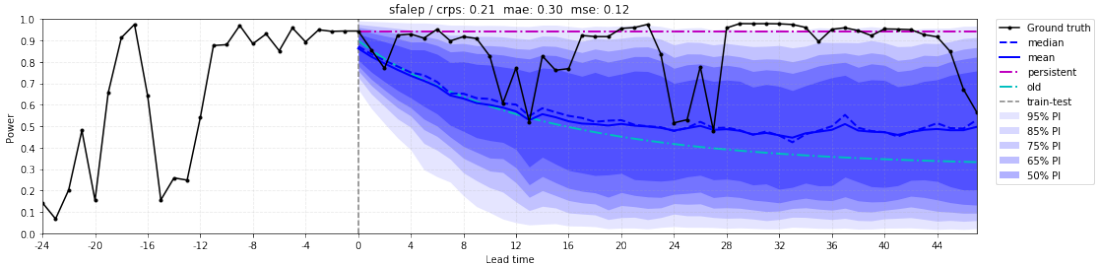
(a) Model Aleatoric (AL) forecast on test set



(b) Model Epistemic (EP) forecast on test set



(c) Model AL+EP forecast on test set

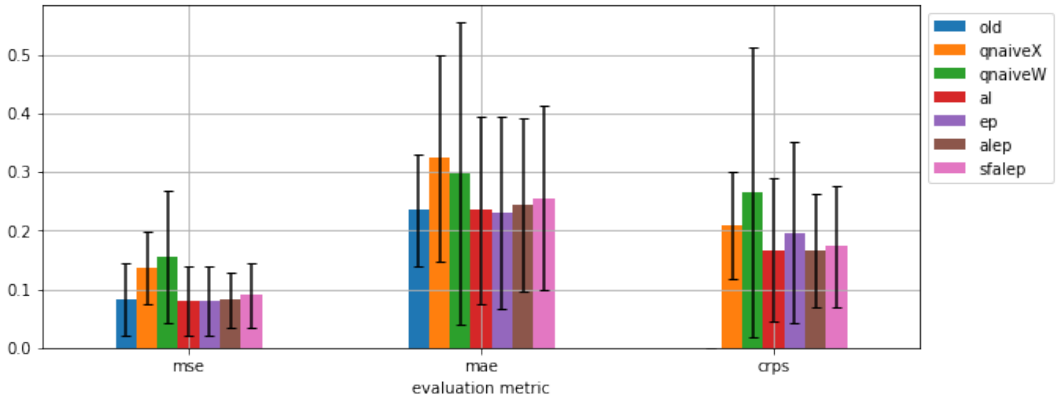


(d) Model AL+EP with Scenario Forecasting on test set

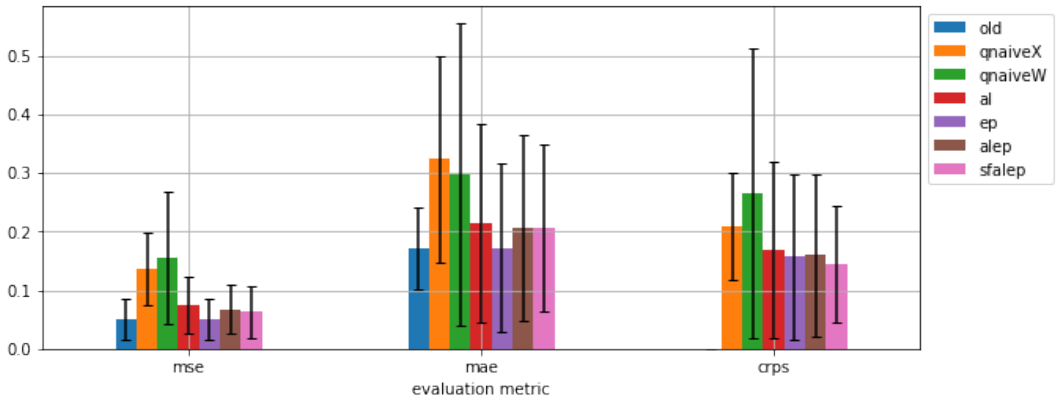
Figure 39. Forecast plots for RNN

6.2.2 Multivariate

In the multivariate case other than the power time-series, 4 wind speed variables were used. Results reported here correspond to the models that were trained on all these 5 variables in order to forecast the wind power time series. In Fig. 40 the error bars are the standard deviation of the metrics and the bars are the mean value for that metric. These results obtained from running the models on different splits of the development set. The performance on RNN architecture is obviously better than the MLP, especially, with respects to the variance of the errors. Also, the multivariate case improved results from the univariate.



(a) MLP architecture

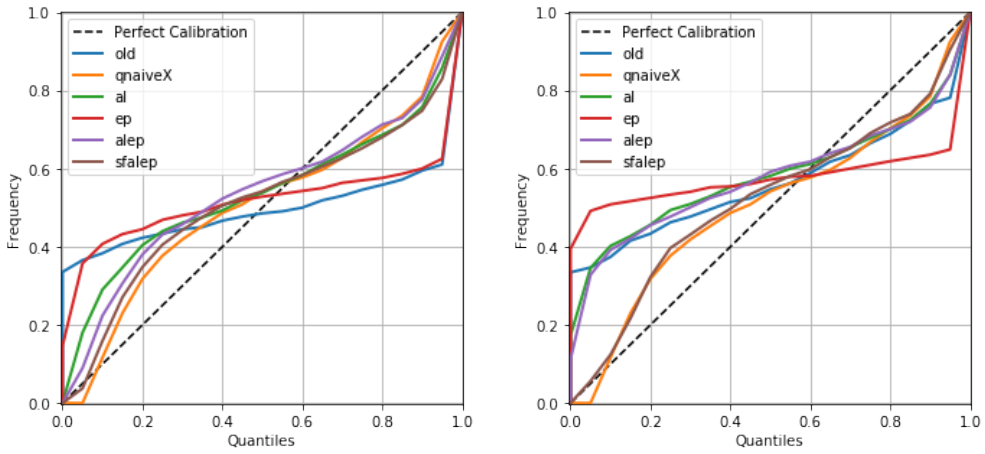


(b) RNN architecture

Figure 40. Performance on GEFCom data with wind data across different metrics and models

Figure 41 shows the reliability/calibration plots for each of the models. These plots produced by taking all forecast variables under the same posterior predictive distribution.

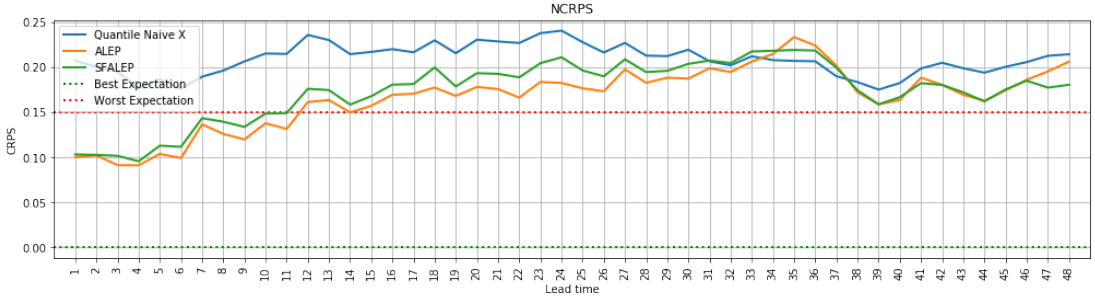
EP was the most miscalibrated model for both cases. SFALEP was better calibrated from MLP to RNN case, and overall it was the second best-calibrated model after QNaiveX. Other than SFALEP the rest of the models were better calibrated in the MLP case. For the SFALEP, RNN should also be preferred over the MLP architecture both for the sake of NCRPS performance as well as calibration of the model.



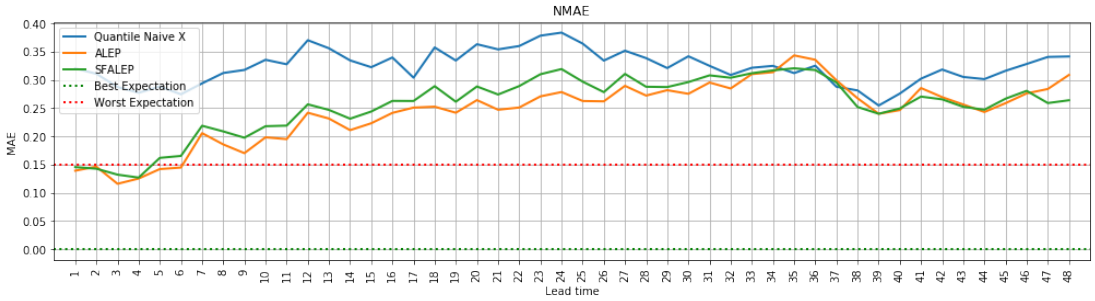
(a) MLP (b) RNN

Figure 41. Calibration of models on multivariate GEFCom data

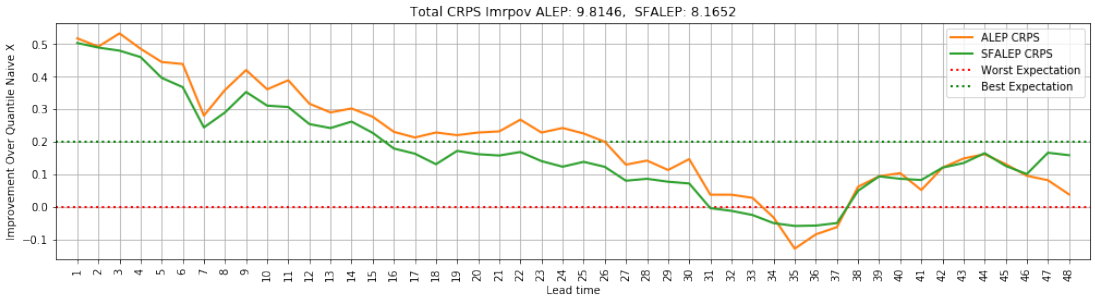
Stepwise Evaluation and Improvement Performance of the forecasts for each lead time averaged over 20 splits of the development set and demonstrated in Figures 42 and 43 for MLP and RNN cases, respectively. In each of these figures, the top two plots show the NCRPS and NMAE with the worst and best expectation which were set as an ideal case based on the results achieved by [HG18]. Therefore, it was expected to obtain a model that does not do worse than the red dotted line and ideally perform with zero error (the green dotted line). The same applied for the improvement plots, the two bottom plots. Going below the red line or $y = 0$ for the improvement plots not only means that the forecast was not an improvement over the QNaiveX but also performed worse. Compared to the univariate case results had considerable improvement. Now the performance is within the expected region for both of the methods and models for the most part of the horizon. For the MLP architecture, the NCRPS is within the expected region for almost 14 hours ahead, which is twice the univariate case, and for the RNN case, it is almost 21 hours, which is three times better than its same counterpart. Also, for the most part of the horizon, the improvements were positive. At the same time it seems at around 33 hours into the forecast the performance drops dramatically as low as the QNaiveX for both of the architectures. Moreover, SFALEP proved to be effective once more in this case.



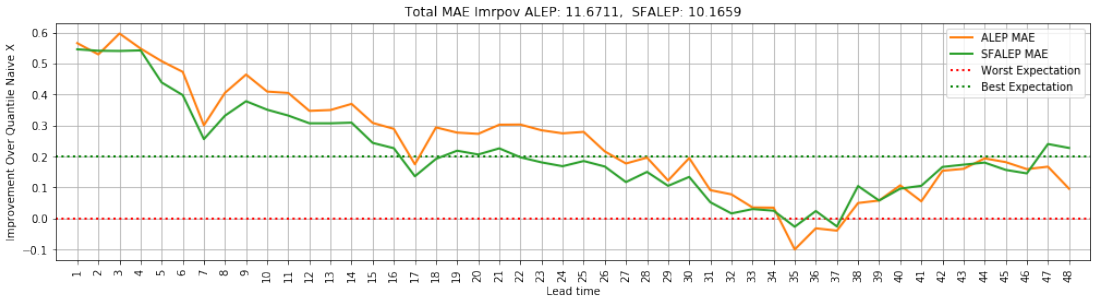
(a) NCRPS accross the forecasted horizon



(b) NMAE accross the forecasted horizon

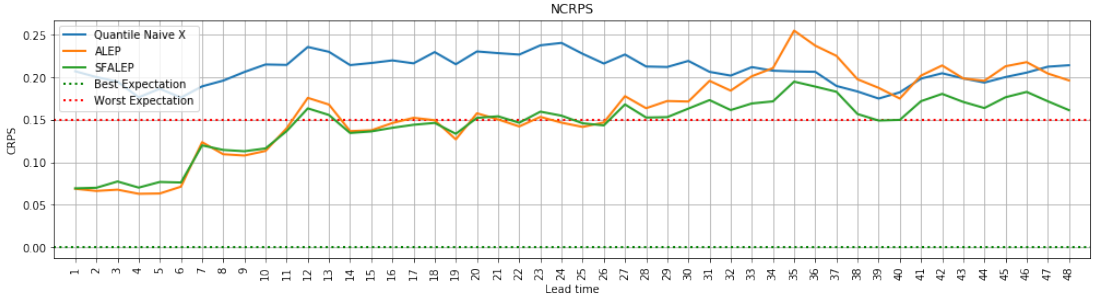


(c) NCRPS improvement over `QNaiveX`

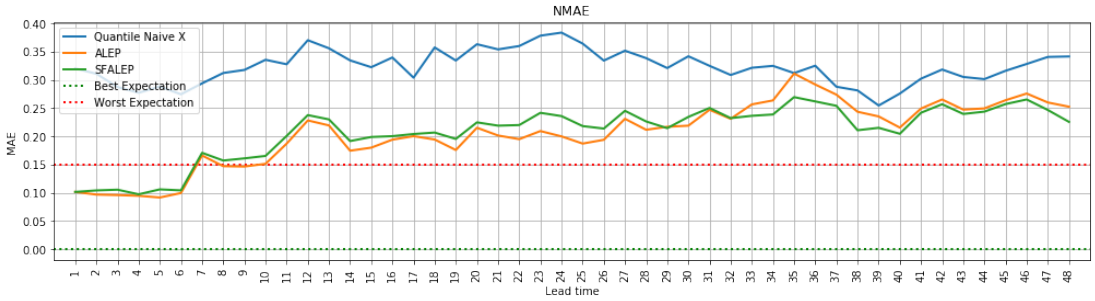


(d) NMAE improvement over `QNaiveX`

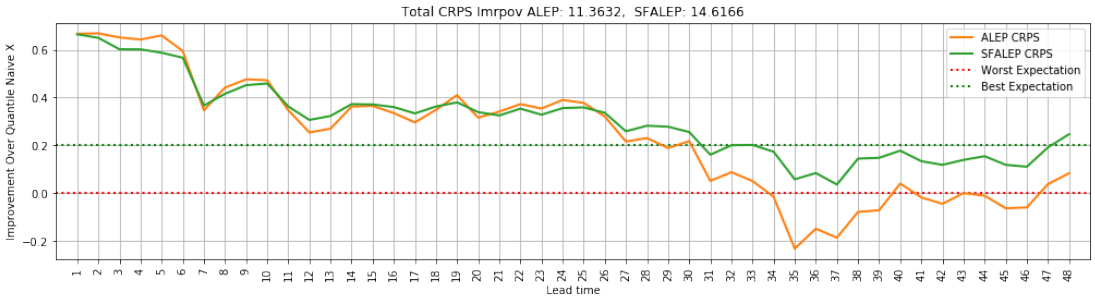
Figure 42. Stepwise Evaluation plots for MLP Forecasts



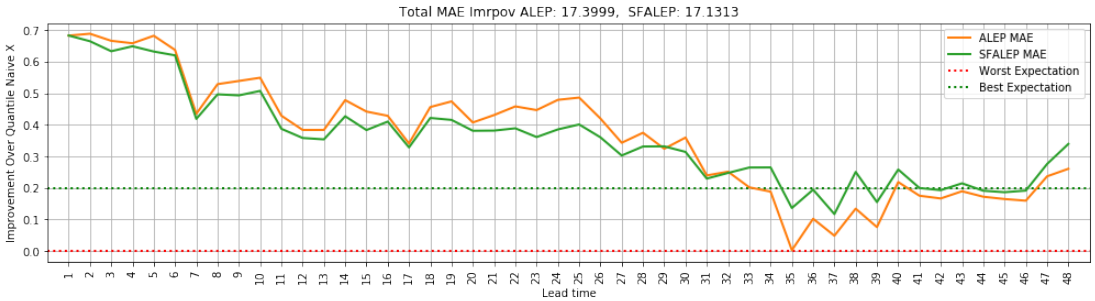
(a) NCRPS accross the forecasted horizon



(b) NMAE accross the forecasted horizon



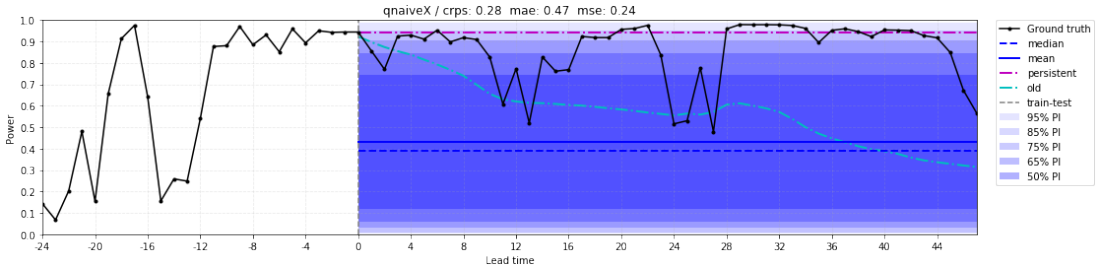
(c) NCRPS improvement over `QNaiveX`



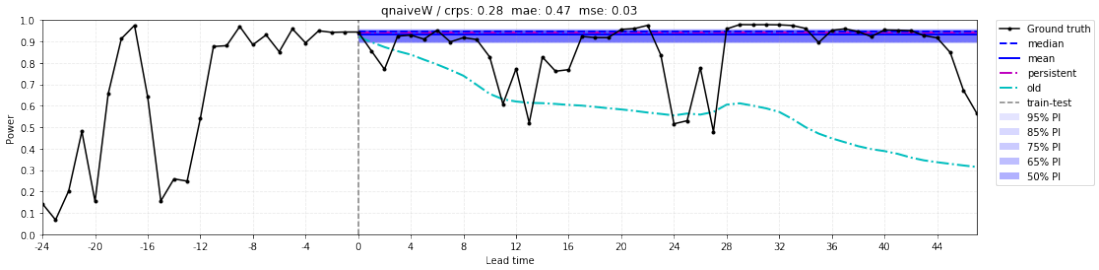
(d) NMAE improvement over `QNaiveX`

Figure 43. Stepwise Evaluation plots for RNN Forecasts

Fancharts for Probabilistic Forecasts on Test set Probabilistic forecasts of the models on the multivariate GEFCom test set visualized using fan-charts. Each fan corresponding to a quantile of the predictive distribution for that lead time. Persistence or the naive (not the quantile naive) also plotted, together with the `old` (or non-Bayesian) model forecast. Figure 37b shows the QNaive forecast which is the same as the univariate case as the additional predictors are not counted in the quantile naive forecast. Figure 45 and 46 show the MLP and RNN forecasts, respectively.

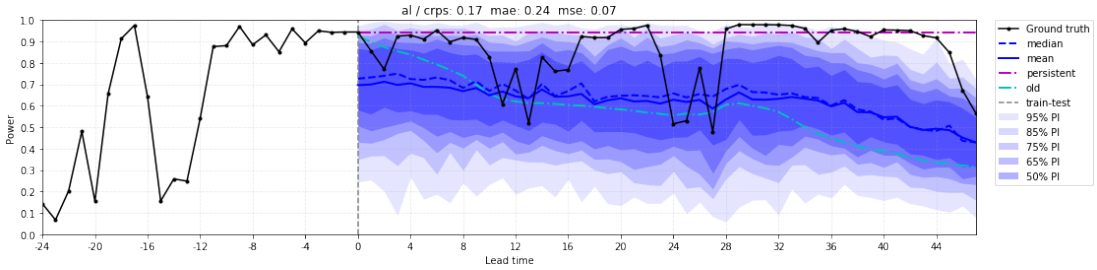


(a) `QNaiveX` forecast on test set

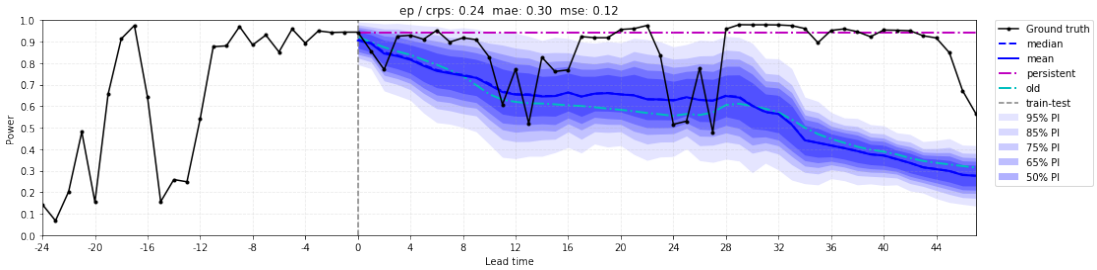


(b) `QNaiveW` forecast on test set

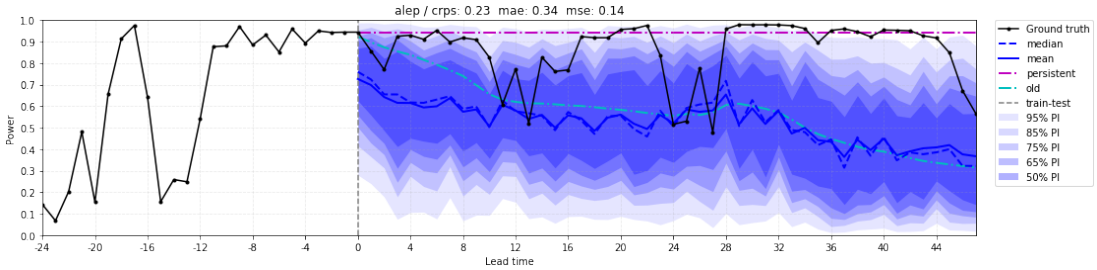
Figure 44. Forecast plots for QNaive on Multivariate GEFCom



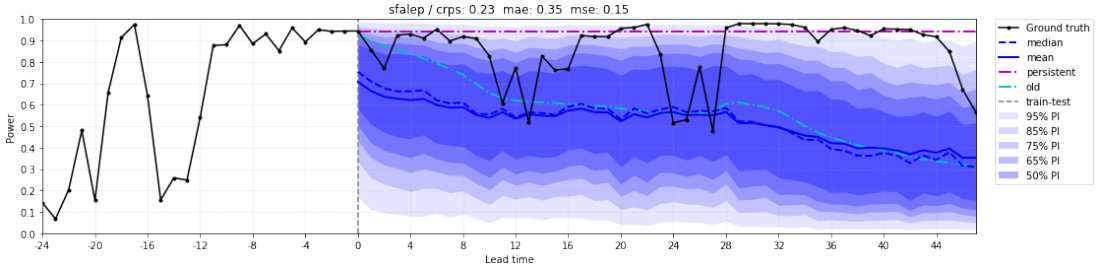
(a) Model Aleatoric (AL) forecast on test set



(b) Model Epistemic (EP) forecast on test set

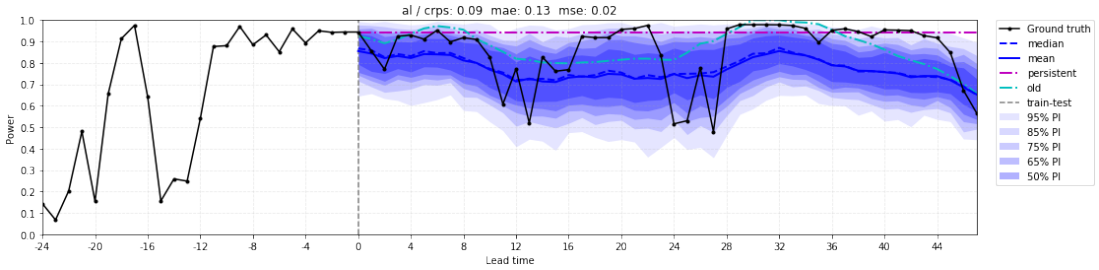


(c) Model AL+EP forecast on test set

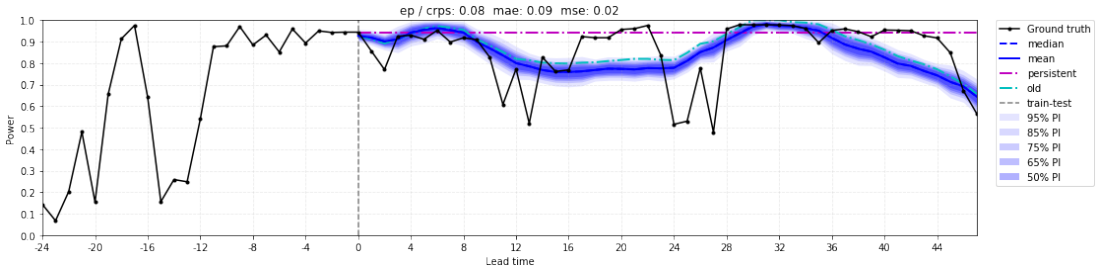


(d) Model AL+EP with Scenario Forecasting on test set

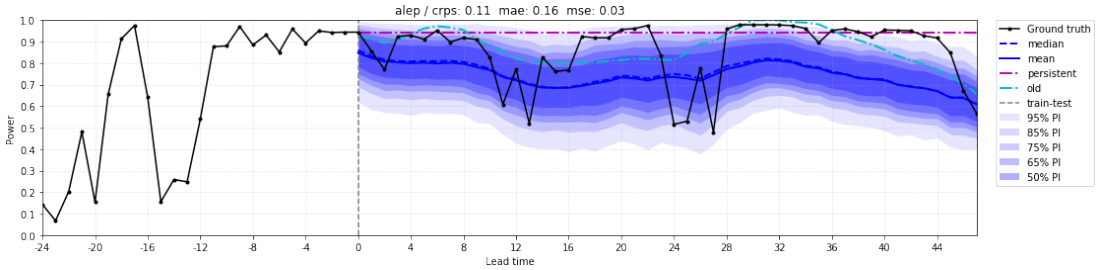
Figure 45. Forecast plots for MLP



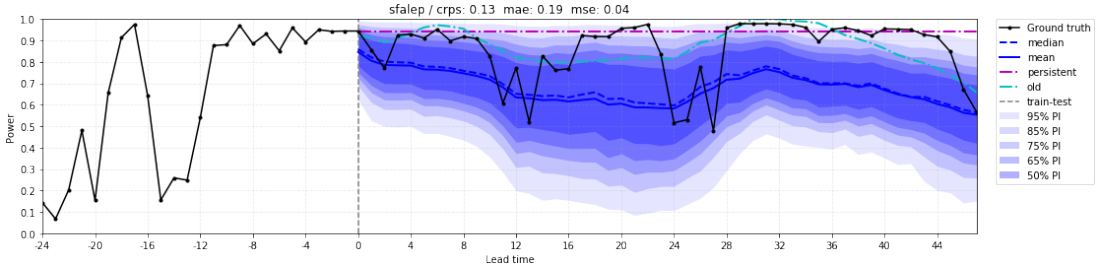
(a) Model Aleatoric (AL) forecast on test set



(b) Model Epistemic (EP) forecast on test set



(c) Model AL+EP forecast on test set



(d) Model AL+EP with Scenario Forecasting on test set

Figure 46. Forecast plots for RNN

7 Discussions

Study of the MSEs in Table 1 obtained from the AL, EP, and the ALEP models were slightly different from observations made in [KG17]. For example, in our experiments, MSE obtained by EP was slightly worse than AL, also its miscalibration was more severe. Direct comparison of the results in this work and the original paper is not possible since the problem at hand, network architecture, and the data are totally different. But the general claim that aleatoric model outperforms the old model can also be observed and other results in Table 3 support that as well.

To obtain multi-step ahead forecast in a many-to-one architecture, one should use the forecast $\hat{y}_{T+1}$ and concatenate it to the input sequence, then rolling the input sequence window forward, in order to make a forecast for next time step $\hat{y}_{T+2}$. This process is repeated as long as needed to obtain a forecast for the whole horizon. The problem with this scheme, however, is that the model treats the prediction in the previous step as a new observation which is not what it is. It matters because the error in these so-called new observations is no same as those in the original data, and so the uncertainty involved are different require the model to take that into account, otherwise the model would diverge from true values. In other sense, the model may get biased toward its own forecasts and the accumulated error in the forecasts lead to poorer and poorer performance in further steps ahead. The SFALEP could compensate for this, however, it seems that the model is still biased but not because of this ignorance, or at least not mainly because of it. This approach improved the quality of the probabilistic forecast by producing better-calibrated forecasts. It also improved the performance except for the MLP in Synthetic data and multivariate RNN in real data.

The Quantile Naive baselines proposed, provided fairly good baselines in the Sinusoidal synthetic dataset. They were also a good baseline for assessing the calibration of the models. These baselines could also give a better understanding of the data when quantiles of the recent history are being compared with all the history. For example, the mean and median of the QNaiveX implies the skewness in the dataset distribution, or when compared with QNaiveW in recent history (or a specific range of data). **Some**

important observations:

- In the synthetic results, the performance of the models were decreasing over time although the added noise was homoscedastic (Gaussian noise with constant standard deviation across the experiment). However, due to the fact that the signals from the underlying model were sampled irregularly, that had introduced some uncertainty for the model as well and therefore causing the model to perform worse over time.

- Lag 4 seemed to be just the right number of input sequence/history for the real

data and that is explainable by the ACF plot where shows the high correlation for the lags 1 to 4 of the series.

- Quantile Naive proved to be useful for: 1) probabilistic forecast, 2) diagnosis tool
- Quantile Naive over all the training data `QNaiveX` has a good calibration which makes sense. By definition, calibration means how closely the predictive distribution was to the empirical distribution of the observations and so `QNaive` is basically using different quantiles of the whole dataset in order to make forecasts for the horizon which makes it be in general fairly calibrated
- On the other empirical distribution of the observations in `QNaiveW` was limited to the recent observation and so were the results biased
- Results obtained on the Synthetic data were comparable in the GEFCom dataset with real data and proved the effectiveness of the Synthetic data
- Incorporating additional variables in the GEFCom dataset improved the results compared to the univariate case
- The improved model `SFALEP` improved the `NCRPS` for the farther lead time more than the `ALEP` which shows its importance and advantage to use this model for longer horizons.
- All models performed worse than expected at the extremes of the data (0 and 1). That is perhaps because the activation function used for the final layer mean value was hard sigmoid and that makes the learning to stall for very big or negative values, moreover values of the extremes are much rare near to 1 which makes the learning at those values even harder, but model has a less hard time for lower values around 0 because there were more examples of that in the data. Further investigation is needed to handle these cases either by changing the loss function or different outputs for separate modeling of these rare cases.
- hard sigmoid gave better results compared to sigmoid
- Scenario Forecasting in `SFALEP` was mainly devised to compensate for the ignorance of the model regarding feeding the prediction values as new observations in the roll forward forecasting. However, the benefit of `SFALEP` is two-fold. Now the model is also capable to generate scenarios or trajectories.
- Wind speed plays an important role in the wind power forecasting as one would expect because of the dynamics of the problem
- In the calibration plots, we could observe that the Model `EP` had the worst calibration. The reason yet to be investigated.

8 Conclusion

We used neural network models to estimate the variance of the data (aleatoric uncertainty) as well as the epistemic uncertainty using Monte-Carlo Dropout technique. These two uncertainties reflected in the predictive variance of the model which with the predicted mean helped to construct a probabilistic forecast. To this end, our models were capable of performing probabilistic forecasting with the use of MCDO approximation for Bayesian Neural Networks. Moreover, we enhanced the results by scenario forecasting to compensate for the ignorance of the many-to-one model. This approach helped to improve the accuracy of the model and make the probabilities more calibrated in general. Our devised approach made an improvement over all the baselines and models.

Synthetic data-set played an important role as a pre-step to the real data-set to make sure that the developed model is functional and then applying it on real data. Moreover, the results obtained from the synthetic data were consistent on the real data as well which shows that the synthesized data was a good indicator of the models' performance, at least on the wind power forecasting problem.

After obtaining promising results on the synthetic dataset, the same models with the same configuration were applied on the real data-set that obtained nearly as good results as in the synthetic case. Multivariate case of the wind power forecast was more successful than the uni-variate case as was expected based on the literature in the wind power generation.

In the end, our experiments to produce probabilistic forecasting were successful and we showed how one could use the idea of Monte-Carlo dropout as well as scenario forecasting in the many-to-one model to improve the result to obtain more accurate and calibrated probabilities. Two different paths could be drawn for future works. One that aims for the improvement of the current line of work and the other introduces new paths for performing deep probabilistic forecasting.

Possible improvements over the current methods:

- Using another measure of loss
- Try to improve the model at the extreme values
- Run the Monte-Carlo simulation concurrently for faster forecasting execution, especially for the `SFALEP`
- Handle ignorance of the model in another way, instead of using the scenario forecasting e.g. make the model aware of its own uncertainties, compensating for the bias by learning the innovations (similar to the moving average method)

- Investigate other possibilities other than the normal distribution as the output of the network
- Further studies on other Synthesized datasets, and effects of different network design choices on the calibration
- Evaluating scenario forecasts
- Deeper architectures
- Feature engineering for better results on the real dataset
- Comparing the results with more competitive baselines such as Bayesian Decision Trees [Lak16]

New directions for the future work:

- Explore other approaches such as [TLP18, PSP19] and more recent variational methods that approximate the Bayesian networks such as [TAS18]
- Other architectures such as convolution neural networks [vdODZ+16] and encoder-decoder architecture [VSP+17]
- What would be the challenges or quality of uncertainty estimation in different architectures
- How uncertainty estimation get better or worse if regression treated as a classification task by discretizing the output

As a conclusion, the main goal of the thesis fulfilled which was to apply to use a Bayesian Neural Network using MCDO in order to capture uncertainty and obtain probabilistic forecast. The main motivation to use a many-to-one architecture was its simplicity and possibility to focus on the uncertainty aspects. Moreover, it turned out to be useful to produce scenario forecasts with this type of architecture and possible to improve the calibration of the forecasts. We hope that these experiments together with the results and discussions that were obtained from this work would inspire the future works and contribute to the forecasting and deep learning communities.

References

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[BJ76] George.E.P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[C+15] François Chollet et al. Keras. `https://keras.io`, 2015.

[CGCB14] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[Fla12] Peter Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York, NY, USA, 2012.

[GG05] Kira Grogg and Kira Grogg. Harvesting the wind: The physics of wind turbines, 2005.

[GG15] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ICML*, 48, 2015.

[GG16] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA, 2016. Curran Associates Inc.

[Gor13] A. Gore. *The Future*. Ebury Publishing, 2013.

[Gos96] B. N. Goswami. The challenge of weather prediction. *Resonance*, 1(11):8–17, Nov 1996.

[Gra11] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

[HA14] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.

[HG18] Sebastian Haglund and El Gaidi. Bounded Probabilistic Wind Power Forecasting using Mixture Density Recurrent Neural Network. In *Workshop on the Meaning of 42*, 2018.

[HPF+16] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J. Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32(3):896 – 913, 2016.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[JOP+] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed <today>].

[KG17] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *CoRR*, abs/1703.0, 2017.

[Lak16] Balaji Lakshminarayanan. *Decision Trees and Forests: A Probabilistic Perspective*. PhD thesis, University College London, Sainsbury Wellcome Centre, 25 Howland St, London, United Kingdom, 2016. Gatsby Computational Neuroscience Unit.

[McK10] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[MCM+11] Juan.M Morales, Antonio Conejo, Henrik Madsen, Pierre Pinson, and Marco Zugno. *Integrating Renewables in Electricity Markets. International Series in Operations Research & Management Science*. Springer, 2011.

[Oli06] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[PSP19] Konstantin Posch, Jan Steinbrener, and Jürgen Pilz. Variational Inference to Measure Model Uncertainty in Deep Neural Networks. pages 1–13, 2019.

[PT13] N. Phuangpornpitak and S. Tia. Opportunities and challenges of integrating renewable energy in smart grid system. *Energy Procedia*, 34:282 – 290, 2013. 10th Eco-Energy and Materials Science and Engineering Symposium.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[Smi15] Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015.

[TAS18] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. *ICML*, 2018.

[TLP18] Natasa Tagasovska and David Lopez-Paz. Frequentist uncertainty estimates for deep learning. 2018.

[vdODZ+16] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Novin Shahroudi**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Probabilistic Forecasting with Monte-Carlo Dropout in Neural Networks,
   *(*title of thesis)

   supervised by Meelis Kull, Erik Ylipää, and Sebastian Haglund El Gaidi.
   *(*supervisors' name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Novin Shahroudi
16/05/2019