Authors' accepted manuscript. Article to be published in *Journal of Lightwave Technology*, 2023
https://doi.org/10.1109/JLT.2023.3252441

Forecasting Lightpath Quality of Transmission and Implementing Uncertainty in the Forecast Models

Somaiieh Yousefi, Hussein Chouman, Petar Djukic, Firouzeh Golaghazadeh, Christine Tremblay, Christian Desrosiers

***Abstract*—The recent popularity of using deep learning models for the forecasting of time series calls for methods to not only predict the target but also measure the uncertainty of the prediction accurately. Working with time series requires reliable and stable forecasters. An essential component of the reliability of machine learning (ML) and deep learning (DL) models is the estimation of the uncertainty. In this work, we address building and characterizing time series forecasters, including N-Beats, Long Short-Term Memory (LSTM) and Multilayer Perceptron (MLP) against the Naive model, and define the confidence margins, and uncertainty for the selected model.**

All the implementations are conducted in Python programming language. Random sampling is performed to avoid overfitting. Our target field data is North American Service Provider data sets (NASP). Among the implemented models, the MLP model is selected to measure the uncertainty and confidence level, and the Monte Carlo dropout, which approximates Bayesian uncertainty, is applied during inference to render the implementation of uncertainty calculations. Quantile Regression is also implemented on the MLP algorithm as a baseline to predict the confidence intervals and to evaluate our strategy for estimating uncertainty. To establish reliable uncertainty estimation in time series predictions, we performed uncertainty calibration. Motivated by recent developments in Expected Uncertainty Calibration Error (UCE), we modified the uncertainty calculated by the probabilistic Bayesian estimations. Detailed experiments and architectures of the solution are presented.

***Index Terms*— Time series, Forecaster, N-Beats, LSTM, MLP, Hyperparameter, Uncertainty, Bayesian approximation, confidence intervals, quantile regression**

I. Introduction

THE increasing rate of data traffic due to the popularity of video on demand and cloud applications, as well as emerging 5G and internet of things (IoT) technologies, requires an efficient increase in the total capacity of optical networks at a minimal cost. This was the main motivation for the development and deployment of elastic optical networks (EON). The main component of EON that makes it flexible and efficient is a bandwidth variable transponder (BVT) which has the ability to dynamically tune its data rate and optical bandwidth through the adjustments of certain parameters, such as modulation format, forward error corrections (FEC) coding, and optical spectrum shaping, with respect to the quality of transmission (QoT) and the optical reach of the lightpath.

With this level of flexibility, the challenge becomes to guarantee the QoT of the lightpath in transparent networks where no optical-to-electrical-to-optical (OEO) conversion occurs in the middle nodes [1]. This becomes more complex with the number of factors that can affect the performance of the lightpaths, such as equipment degradation, fiber aging and power fluctuations together with the uncertainties of physical parameters used as the input for the QoT estimator and of its own uncertainties [2]. Thus, when a lightpath is planned, operators introduce a mandatory signal-to-noise ratio (SNR), or system margin, to ensure service remains unobstructed throughout the lifespan of the network. This margin hinders operator efforts to fully use the available network bandwidth.

To reduce the need for system margins, it is helpful to know the future QoT of the lightpath because this will enable operators to respond proactively to performance degradation of lightpaths in service. As such, research has recently been undertaken to forecast QoT of lightpaths, thanks to the availability of field data collected by coherent receivers which make it possible to leverage machine learning (ML) in QoT forecast. In this context, deep learning algorithms based on the feed-forward network, recurrent neural network (RNN), convolutional neural network (CNN), and residual neural network (ResNet), were investigated to minimize the QoT prediction error, for different horizon periods. The results obtained showed that ML is a promising application for predicting QoT with very low prediction percentage error [3].

On the other hand, for operators to fully rely on the forecasting model's prediction to lower system margins, the prediction error is not an appropriate metric and the level of certainty of the algorithms' prediction must be provided. Moreover, it is also essential for the forecasting algorithms to output the confidence of their predictions.

Among the deep Learning algorithms that have been applied to predict the QoT for the North American Service provider (NASP) data set, multilayer perceptron (MLP) model demonstrated the highest accuracy [3]. In this work, we propose a first approach for uncertainty estimation in time series predictions. Uncertainty analysis using the Monte Carlo dropout method is performed. The predicted confidence

Manuscript submitted on November 23, 2022. This work was supported in part by Mitacs under grant IT14046 and Ciena Corp.

S. Yousefi, H. Chouman, C. Tremblay and C. Desrosiers are from the École de technologie supérieure, Montréal, QC, Canada (e-mail: christine.tremblay@etsmtl.ca).

P. Djukic and F. Golaghazadeh are from Ciena Corp., Nepean, ON, Canada.

.

margins are evaluated using Quantile Regression and the uncertainty is assessed via UCE calculations.

The remaining sections are organized as follows: Related work on lightpath QoT forecasting is presented in Section II, followed by an overview of the methods for estimating uncertainty. Section III details the methodology and field dataset used for building the forecasting models and their implementation. Then, the uncertainty tool used is illustrated and the way it is implemented in the chosen forecasting models is explained. We also illustrate details about the quantile regression, the confidence margin baseline used in this work. Section IV shows the forecasting results obtained and the uncertainty margin. Finally, Section V concludes with a summary of this work and future research directions.

II. Related Work

A. Forecasting Lightpath QoT

Time series are sequences of data occurring over time and which can exhibit patterns such as seasonality and trends [4]. They have a wide range of applications from monitoring industrial processes to tracking business trends. Forecasting times series accurately can have a huge financial impact, in the millions of dollars for businesses [5]. Forecasting with statistical forecasting methods becomes more complex in areas that have temporal components like optical communication and networks [6][5]. Thus, in the last years, ML has been explored to help optical network operators predicting future traffic, equipment degradation, and the QoT of the lightpath [7].

Previous studies have investigated the prediction of QoT in complex optical networks using machine learning techniques [8]–[13]. Aladin et al. [2] employed support vector machine (SVM) and recurrent neural networks based on Long Short-Term Memory (LSTM), Encoder-Decoder LSTM and Gated Recurrent Unit (GRU) to estimate lightpath QoT of unestablished lightpaths with 13-months field data. Using the root-mean-square (RMSE) and R-square metrics for comparing models, GRU was shown to yield the best overall performance.

Allogba et al. [14] implemented SVM and NN models for the real-time estimation of lightpath QoT. Univariate and multivariate LSTM and GRU models were also compared for forecasting tasks. SVM showed a more reliable QoT estimation compared to the NN models, whereas a lower RMSE and absolute maximum error (AME) was observed for single-step univariate LSTM over the multi-step encoder-decoder LSTM and GRU. In [8], Ayassi et al. studied various ML models to estimate the QoT of lightpaths and assessed the feasibility of lightpath establishment in terms of the contributing parameter uncertainties. Ayoub et al [9] proposed an approach based on Exploiting Explainable Artificial Intelligence (XAI) to help understand the behavior of models for lightpath QoT estimation. A long short-term memory (LSTM) deep neural network (DNN) architecture was employed to forecast SNR for one lightpath over a 24-hour horizon based on 13-month historical field data collected in a production network [12]. This work was extended to include, in addition to LSTM, encoder-decoder LSTM and Gated Recurrent Unit (GRU) DNN architectures to forecast SNR over a 96-hour horizon based on field data [11][2]. A 1D Convolutional Neural Network (CNN) trained with historical field data has also been used to predict lightpath SNR over a 24-hour horizon [15]. In these works, the models were trained with field data sets of limited size. In the research conducted by Chouman et al. [3], a multilayer perceptron (MLP) DNN architecture was trained using field data from 52 lightpaths deployed in two optical networks, and its performance compared with that of the LSTM model and linear regression methods. DNN models have been shown to leverage the historical field performance metrics collected by the network control system for predicting lightpath performance.

In this work, we compare several forecasting methods: naive, multilayer perceptron (MLP) based on a feed-forward network, LSTM based on RNN networks and the N-Beats model based on ResNet [5]. The naive method predicts a future value in the time series based on the last time series value seen. The metrics used in our work, to quantify the performance of the prediction algorithms, were the root-mean-square error (RMSE), R-Squared, and training time. RMSE is calculated as the square root of the mean of the squared differences between the ground truth value and predictions. Thus, it is the standard deviation of the residual (prediction errors). RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. The R-Squared is a relative metric used to compare the model's performance with the baseline models that is trained on the same data. The model's training time is relevant in industrial applications due to computing costs. Those metrics are those typical used in ML to quantify algorithm performance. However, the learning algorithms need to also output their confidence in their prediction. Thus, in this work, we intend to investigate the model from a new perspective: the confidence of the model and its level of uncertainty.

B. Uncertainty

Despite numerous studies on ML and its fast deployment in measuring systems, there is a relatively smaller number of works on implementing uncertainty in models. The calculation of uncertainty is a necessity if ML is to be adopted in commercial services and products because the accuracy of the prediction is just as important [6], [16], [17][16]–[21].

Stigler [22] noted for the first time the transformation from the point estimation to the distribution estimation [23]. This can be done by forecasting a collection of points instead of one point for a particular quantity. This quantifying of the variances in the prediction is defined as probabilistic forecasting [24]. Two of the main approaches for distribution estimation are the conditional quantile regression and conditional expectile regression, both performing inference around quantile functions [23], [25]. To estimate the regression coefficients induced by the training data, quantile regression uses asymmetric piecewise scoring functions for the τ quantile [23], [25], [26]. Expectile regression is similar to quantile regression, the principal difference being that it is based on a quadratic scoring function [23], [27], [28]. Although these methods extend regression beyond the simple prediction of the mean, they can also lead to crossing quantile curves in case of small data set or with dense quantiles [29].

One approach that can be used to obtain probabilistic forecasting is to predict a collection of points using ensemble learning [24], [30]. Then, this ensemble of prediction models

can be statistically corrected using post-processing methods, such as non-homogeneous regression and ensemble model output statistics, as proposed by Gneiting et al. [31]. This method offers powerful techniques for statistical post-processing, including non-homogenous regression (NR), ensemble model output statistics (EMOS) [31] and Bayesian averaging (BMA) [32].

Another approach that is employed to obtain probabilistic forecasting is the Bayesian inference method. This method assigns distribution to parameters based on prior experience before data collection and applies Bayes' theorem to revise the distribution after obtaining data [10]. The drawback of the Bayesian approach is its complex and high cost computation [20]. Many approaches have been developed to solve the complexity of the Bayesian methods by proposing approximations such as the Markov chain Monte-Carlo methods, variational approximations, sequential Monte-Carlo and expectation-propagation [33], [34]. Still, this method suffers from high computational cost, which hinders its use in practical applications [20].

Variational inference applied to Bayesian neural networks showed little success [35]. Even sampling-based variational inference and stochastic variational inference [36], [37], [20], as in the approximation to the Bayesian approach, were limited in the means of application due to the computational costs. In practice, the sample-based variational method is equivalent to using Monte Carlo for sampling from the posterior distribution [35]. This procedure involves performing random moves in the weight space according to their probabilities. Stochastic variational inference is a scalable algorithm for the approximation of the posterior. The main idea is using stochastic optimization to optimize a variational objective [38]. The method was designed to build a classifier adapted to handle huge datasets; however, it is more complex to deploy for purposes other than classification.

Therefore, using some approximations to the Bayesian probabilistic approach could be helpful. Teye et al. [39] showed that Batch Normalization could be used as an approximation to the Bayesian model. This method can lead to variational output, by introducing randomness to the model as in Ioffe et al. [40], and results in uncertainty estimation.

Gal et al. [20] use a dropout method as an approximation for the Bayesian theory. In fact, dropout is used in ML as a regularization tool to prohibit over-fitting. However, they showed that applying dropout before every weight-layer in a network with any arbitrary depth and non-linearity is mathematically equivalent to a well-known probabilistic model known as the Gaussian Process (GP) [21]. Therefore, applying dropout in the inference time produces different outputs for the same input because, in each run, the input is passing through a slightly different network due to the dropped units. This is known as the Monte-Carlo dropout. Using this method can provide target variable distribution instead of point estimation.

In this study, we aim to estimate the uncertainty of prime ML models for QoT prediction. Our study focuses on well-known methods that are easy to implement and can be employed on any network. Among the Bayesian approximation approaches, the dropout-based method was selected for due to its simplicity and relatively low computational costs. As second method to assess uncertainFty, we investigated quantile regression which is also simple to implement but avoids the costly sampling process.

C. Deep Quantile Regression

Quantile regression is a statistical method to estimate and perform inference about conditional quantile functions [41]. Introduced by Koenker and Bassett [26], this method seeks to predict the conditional median of the target. A special case of quantile regression is the Least Absolute Deviation (LAD) [26] that fits the medians to the linear function of the covariates. An attractive property of LAD estimation is that the median offers a more robust measure than then mean.

In regression, the most commonly used loss function is the mean squared error (MSE). Therefore, when we predict using a neural net that minimizes this loss, we are predicting the mean value of the output which may have been noisy in the training set. In contrast, the quantile regression function is the weighted sum of absolute deviations, which is a robust measure of location. Therefore, the estimated vector of coefficients is not sensitive to the outliers of the dependent variables. Moreover, in the case of a non-normal error term, quantile regression could be more efficient than the least square error [41]. Deep quantile regression has been leveraged to calculate QoT uncertainty estimation over the unseen lightpaths [42]. Margin reduction and more accurate decisions for q-quantiles as lower estimate bounds were observed.

One strategy to tackle the problem of investigating the forecasting model's uncertainty is to combine deep learning and statistical tools such as quantile regression. The deep quantile regression method is based on minimizing quantile regression loss function in DNN learning models. This provides a quantitative assessment of the prediction methods for uncertainty estimation. Related works have obtained promising results using neural networks to approximate QoT and SNR uncertainties by minimizing the MSE function on the training set [13], [43].This paper aims to use deep quantile regression as the baseline for assessing the uncertainty of the forecasted data.

III. Methodology

This section details the knowledge base of this study as well as the training and testing of the lightpath QoT forecast models.

A. Data Preprocessing

The knowledge base used in this study is composed of the field bit error rate (BER) collected for 140 channels in the NASP production network and sampled at 15-minute intervals as part of the performance metrics (PMs) collection process. Each channel includes 32,000 samples. After windowing the time series from randomly selected channels, the resulting dataset includes 5120 samples. From the preprocessed data set, 70%, 15% and 15% of data are used for the train, validation, and test sets containing 3300, 910 and 910 data windows, respectively. To prevent overfitting of the models, historical sequences of data have been randomly picked and their corresponding target horizons were chosen as the data windows. The length of the history and target data are adjustable. Two days of history and one target in various time

intervals ranging from 1 h to 16 h have been considered.

B. *Forecasting Models*

In this work, we compare several forecasting methods: multilayer perceptron (MLP) based on the feed-forward network, LSTM based on RNNs and more recently the N-Beats model based on ResNet [5], using a naive method as baseline. The naive method predicts a future value in the time series using the value last seen in the time series.

MLP, LSTM and N-beats algorithms are based on a DNN, which is a part of ML methods based on artificial neural networks (ANNs), also called neural networks (NNs). DNN consists of an input layer of source nodes, one or more hidden layers of nodes and an output layer, which has one node for the value of the network output [44]: MLP is a simple class of feed-forward neural network, which maps an input (historical window) to an output (future target).

In a feed-forward network, there are no feedback connections in which outputs of the model are fed back into itself. When feed-forward neural networks are extended to include feedback connections, they are called RNNs. RNNs are a family of neural networks for processing sequential data. The LSTM is a popular type of RNN with an internal memory setup to allow long-term dependencies to affect the output [44], [45].

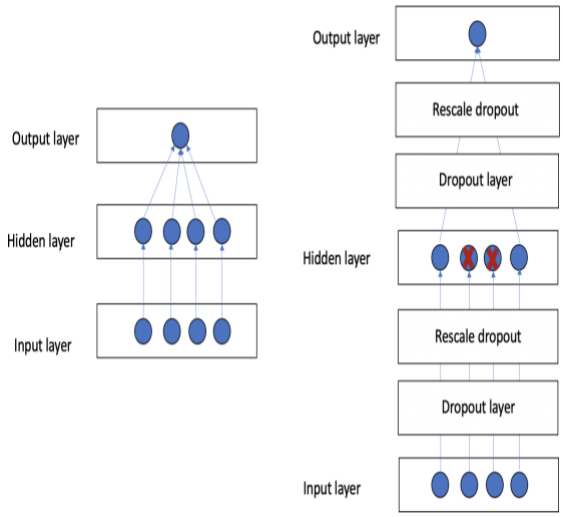
ResNet networks are called "residual" because at each stage of the network some output of the stage is subtracted from the input of the stage and the difference (the "residual") is forwarded to the next[5]. N-beats proposes a deep neural architecture based on backward and forward residual links and a very deep stack of fully connected layers [5].

Our deep learning models MLP LSTM and N-Beats are implemented with the Keras library. As shown in Fig. 1, the MLP model contains two dense layers of 16 neurons, with ReLU activation function, and a single neuron in the final layer with the linear activation function, which corresponds to 33 neurons in total. No dropout layer is employed for regularization. Selecting this structure for our MLP model is based on the experiments on the performance of the model after hyperparameter optimization.

The LSTM model is implemented with three LSTM layers of 256, 128, 64 and 32 neurons and an output dense layer of 1 neuron. The number of neurons is selected based on the validation set, kept the same for most of the experiments and adopted according to the hyperparameter tuning for other experiments. The number of neurons in the LSTM layers were selected based on the hyperparameter tuning.

Our N-Beats architecture is composed of three stacks [5]. Each stack contains 4 blocks. Each block includes 3 dense layers and 2 theta layers, each one corresponding to the forecast and backcast followed by the forecast and backcast layers, respectively and the residual layer at the end [5]. In the dense layers, the theta layers, and the backcast layers contain 64 neurons per layer. The forecast layers contain 1 neuron, which corresponds to 3,468 neurons in total.

The first objective of this research is to compare these models against our baseline naive model, and select the best performing forecaster, i.e., the one with the lowest RMSE to estimate its uncertainty using the Monte-Carlo dropout method at the inference phase.



(a) (b)

Fig. 1. The MLP models with and without dropout: a) the MLP model without dropout, b) the MLP model with dropout and with rescaling the dropout.

C. *Uncertainty Calculations*

There are three forms of uncertainty in regression problems [46]: model uncertainty or epistemic [47], model misspecification, and inherent noise. Model uncertainty refers to the blind spots near the model's parameters, and these can be decreased by increasing the number of training samples [47].

Inherent noise grasps the uncertainty in the process of data generation and is irreducible [46]. Maximizing the likelihood is how we learn the distribution parameters. The uncertainty is encoded by some of these parameters.

A well-known method to address estimating the model's uncertainty quantitively uses the variational inference Monte Carlo dropout [20] which employs dropout to sample from the posterior distribution [48].

In this section, we discuss the implementation of the dropout method in the inference phase as an approximation to the Bayesian approach and how the loss function is calculated.

1) *Uncertainty Implementation*

To estimate the uncertainty using Monte Carlo dropout, the dropout is applied during the inference phase. Our strategy is to pre-train a MLP model with the optimized set of hyperparameters and to use the weights for the prediction model with dropout layers. Running the model N number of times along with inference, generates distributions for the predictions.

Different MLP structures have been deployed to estimate the uncertainty. Specifically, we compared two models: a pre-trained MLP model with and without dropout while training, and prediction MLP model with dropout after the weight layers in the inference time have been compared. We implemented MLP models with 16, 128 and 1024 neurons in the dense layers to perform the experiments.

Applying dropout during inference causes differences in the output. Using the dropout for the regularization purposes causes the input elements to be randomly set to zero, which will decrease the output. Thus, to have a constant output, non-dropped elements are rescaled [49]. For instance, if elements of

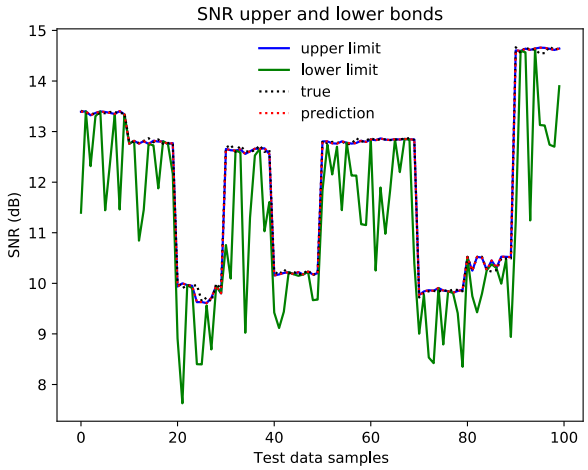


Fig. 2. Upper and lower bounds, true value, and median, using dropout method.

input x are dropped (set to zero) with a given probability rate, then (1-$rate$) will remain. The remaining elements will be scaled by $1/(1\text{-rate})$ [50][44]). This way, the output values will stay intact. Since we did not apply the dropout during training, so that TensorFlow (or any other framework used for the implementation) could rescale the output, we needed to do the rescaling explicitly. Hence, we added layers to multiply the output of each dropout later by $1/(1\text{-rate})$ so that we could have the dropout for the uncertainty approximation and, simultaneously, rescale outputs to the expected values as shown in Fig. 2. It is worth noting that failing to do rescaling causes a shift to appear in the PDF diagrams (this will be discussed in the results). The greater the dropout rate, the greater the shift is. Therefore, performing this rescaling is essential.

When we run the model at the test time N number of times ($N\rightarrow\infty$), according to the central limit theorem, the accumulation of a set of random variables is also random and it increasingly becomes Gaussian.

2) Applying Dropout with Inference

With dropout, the output of the network is multiplied by a Bernoulli distribution of random variables [17]. For each run, it randomly drops some units, which generates different networks and consequently yields a different output for each network. We consider the input data set $\{x_1, \dots, x_N\}$, the outputs $\{y_1, \dots, y_N\}$ and the prediction vector $\{\hat{y}_1, \dots, \hat{y}_N\}$, and the goal is to estimate function $y = f(x)$. Following the Bayesian approach, our task is to find the distribution of the posterior, having our data set over the space of function $p(f)$ [21].

$$p(f|X, Y) \propto p(Y|X, f)p(f) \tag{1}$$

Considering a NN in its simplest expression, that is, only one hidden layer [49], we denote the weight matrix between the first layer and the hidden layer as W_1, and the one connecting the hidden layer to the output layer as W_2 [49]. These are the linear transforms before applying the nonlinearity, which is activation function $\sigma(.)$, such as ReLU or hyperbolic tangent (TanH). The bias is added to shift the non-linearity and denoted by b. We assume that the output has the dimension of D while its inputs are the vectors with Q dimension, and we have K hidden layers.

Thus, we have two matrices: W_1 with dimension $Q \times K$ and W_2 with dimension $K \times D$ and b, which is a D dimensional vector. The output of a standard NN is formulated as [49]:

$$\hat{y} = \sigma(x\, W_1 + b)W_2 \tag{2}$$

During the optimization process, a regularization term could be added. L_2 regularization could be applied by a weight decay λ while minimizing the loss function (Eq. 1) [20]:

$$\mathcal{L}_{\text{dropout}} = \frac{1}{N}\sum_{i=1}^{N} E(y_i, \hat{y}_i) + \lambda \sum_{i=1}^{L} (\|W_i\|_2^2 + \|b_i\|_2^2) \tag{3}$$

in which $E(y_i, \hat{y}_i)$ is the Euclidean loss function of the output value and the prediction, and b_i is the bias term in each layer of the network. When the dropout is applied, two binary vectors of dimensions Q and K are taken into consideration, z_1 and z_2. The elements follow the Bernoulli distribution with parameters $p_i \in [0, 1]$, for $i = 1, 2$. The value of each binary variable is 1 with a probability p_i for layer i. If the value of the binary value is 0, the unit will be dropped [49].

With dropout, if we consider an input x, the $1\text{-}p_i$ portion of this input is set to zero. Considering the two Bernoulli distributions of z_1 and z_2, we can rewrite Eq. (2) as:

$$\hat{y} = \sigma(x\,(z_1 W_1) + b)(z_2 W_2) \tag{4}$$

The same procedure is repeated for a network with more layers. Here, by z_1 we mean the $\text{diag}(z_1)$. The loss function of Eq. (3) is now extended as Eq. (5).

$$\mathcal{L}_{\text{dropout}} = E + \lambda_1\|W_1\|_2^2 + \lambda_2\|W_2\|_2^2 + \lambda_3\|b\|_2^2 \tag{5}$$

D. Deep Quantile Regression Implementation

For a quantile regression, some information about linear regression is required. In a simple linear regression model, we have a function relating the independent variables x to the dependent variable y:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{np} \tag{7}$$

in which, p is the number of regressors and $i \in \{1, \dots, n\}$ defines the number of data points. One of the best error estimators for linear regression is MSE:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 \tag{8}$$

The quantile regression model equation for the τ-th quantile becomes:

$$Q_\tau(y_i) = \beta_0(\tau) + \beta_0(\tau)x_{i1} + \cdots + \beta_0(\tau)x_{ip} \tag{9}$$

where, again, p is the number of regressor variables and n, the number of data points. The β coefficients are dependent on the quantile. Finding the β coefficients at the specific quantile

is a similar process as for linear regression except in calculating the median absolute deviation (MAD):

$$\mathrm{MAD} = \frac{1}{n}\sum_{i=1}^{n} \rho_\tau(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})) \quad (10)$$

Here, we use the check function [27] which gives asymmetry to the error regarding the defined quantile. We can define ρ as:

$$\rho_\tau(u) = \begin{cases} \tau \max(u, 0), & u \geq 0 \\ (1-\tau)\max(-u, 0), & u < 0 \end{cases} \quad (11)$$

In this equation, u is the error, which implies that if the error is positive, the check function will return the error multiplied by the quantile (τ), and if the error is negative, it will return the absolute error multiplied by the quantile (1- τ).

To implement quantile regression, we use our MLP model, and compile the model using the loss function defined in Eq. 10. Thus, for the defined quantile value of interest, we can find the region where the mass of the data is located. For instance, we can consider 5% and 95% quantiles, define the distributional mass, and compare the results with the upper and lower bounds of what we calculated earlier using the dropout method as the approximation to the GP.

IV. Performance Results and Analysis

The RMSE results for all the algorithms for the NASP data set are shown over all the horizons (1 h, 2 h, 4 h, 8 h and 16 h) in Table 1. Remember that the time series in this data set do not exhibit any seasonality and trend.

TABLE 1
RMSE Metric vs. Forecast Horizon. NASP Data Set

RMSE	1 h	2 h	4 h	8 h	16 h
MLP	0.06	0.06	0.07	0.09	0.10
LSTM	1.86	1.90	1.89	1.90	1.99
N-BEATS	0.13	0.14	0.14	0.16	0.18
NAIVE	0.03	0.04	0.05	0.07	0.09

The RMSE increases for the longer horizons. N-Beats shows nearly 0.13 to 0.18 dB prediction error for channel SNR in the NASP data set. Its error increases gradually from 1-h to 16-h horizon. Naive exhibits the minimum RMSE, increasing from nearly 0.03 to 0.09 dB. The closest RMSE values to that of Naive belong to the MLP model, 0.06 dB. N-Beats represents a moderate raise in RMSE from 0.13 to 0.18 dB. The RMSE of LSTM shows a slight increase from 1.89 dB to 1.99 dB. Still, comparing the RMSE of all the algorithms, the prediction error of LSTM is the highest for the NASP data set.

According to the performances of the DL models presented in Table 1, the MLP model shows minimum RMSE. We chose to work with the NASP dataset with 1-h time interval time series as the data set for the experiments for implementing uncertainty. We used the same window format setting (192 SNR data as the history to predict a point data in the future) and

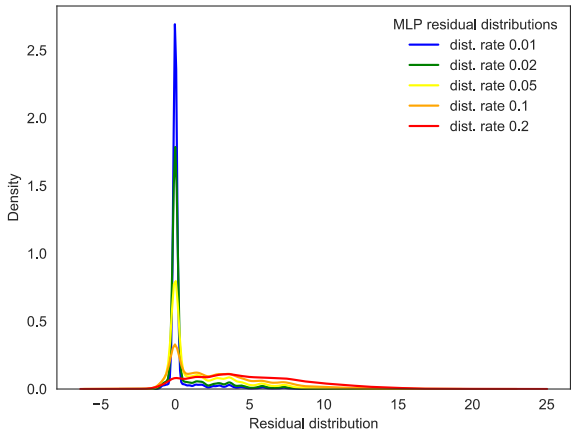


Fig. 3. PDF of the residuals per rate.

performed random sampling. We conducted a series of experiments with different dropout rates to define the best rate and the number of runs during the test time. With the MLP structure of 16 neurons, dropout rates 0.01, 0.02, 0.05, 0.1 and 0.2, and 100 number of variational inferences were considered for the test time. The variational inference of each prediction is a vector with Gaussian distribution due to the central limit theorem. To select the number of runs, we calculated the mean of the standard deviation for each prediction and the mean of RMSE of the test set for different dropout rates. The mean of the standard deviation of each distribution was closer to the mean RMSE of the whole test set in case of 0.01 dropout and 100 runs. Therefore, we selected 100 runs for inference.

Fig. 3 represents the Probability Distribution Function (PDF) of the residuals, calculated as the subtraction of the distributions of each prediction from their corresponding actual value. Residuals are defined as $residuals = Y_m - \hat{Y}_{m\times n}$, where Y_m is the vector of the actual values and $\hat{Y}_{m\times n}$ is the matrix of variational inference. A m × n matrix of residuals (m being the number of predictions, or the number of test instances) and n being the number of runs (100)) is generated. The observed positive tails for all PDFs imply that most variational inference values are less than the actual values

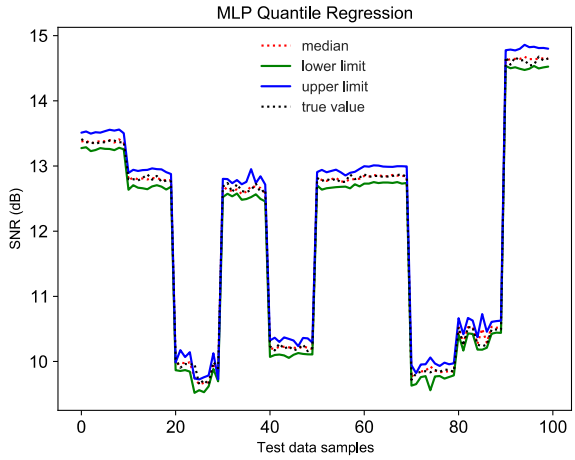


Fig. 4. The upper and lower bounds, the true value and the median, as obtained by using quantile regression.

(Fig. 2). Therefore, since almost all the variational inference values are less than the actual values, we only have positive residuals (Fig. 3).

To calculate the upper and lower bounds of the data with the dropout method, we used the quantile of the variational inference data. We selected the 5 and 95% quantiles for lower and upper bounds, respectively. Due to the very low residual (RMSE = 0.062 dB), the difference between the true value and the prediction is minimal. Moreover, as expected, the upper bound is very close to the actual value because, as shown in Fig. 3, the negative tail is very short. On the other hand, since the tails of the distributions are all in the positive region (Fig. 3), the difference between the lower bound and true value attains 4 dB.

To evaluate the upper and lower bounds, we used quantile regression. This method yielded the median value, and upper and lower limit of the 95% quantile (Fig. 4). In Fig. 4, the upper and lower bounds are at the same distance from the median (nearly 0.1 dB).

As can be observed from Fig. 3 and 4, the two lower bounds are different. The reason for this is that the dropout method generated the distribution of the values during inference in the credible region $(-\infty, y]$. Most of the data are centered around the most probable values (Fig. 4), and the frequency of lower values is very small. However, to capture the 90% (or 95%) quantile, we included all the infrequent data. This is why, Fig. 4 obtained from the approximation to the GP, the lower bound (green line) is considerably different from the prediction.

Moreover, earlier, we assumed that using the Gaussian process (approximation to the Bayesian probabilistic) would result in Gaussian distribution in the inference. However, in this case, true distributions were not completely Gaussian (Fig. 3). In fact, uncertainty calculated using the Bayesian approach tends to be uncalibrated [48], [51], [52], and further processes are required to achieve calibrated uncertainty measures. In this work, we chose to use Uncertainty Calibration Error (UCE), developed by [48].

Uncertainty Calibration

Calibration measures are performed to adjust the lower bound of the prediction in the Bayesian uncertainty method. The calibration measures are assessed using sets of visualizations and UCE calculations.

Uncertainty Calibration Error

The expected Uncertainty Calibration Error (UCE) is the extension of the term for classification. Following Guo et al. [53] and Laves et al. [48], the uncertainty of the DL models is divided into M bins of equal size. UCE can be calculated as the weighted average of the difference between the predictive error and uncertainty of the bin:

$$UCE = \sum_{m=1}^{M} \frac{|B_m|}{n} |err(B_m) - uncertainty(B_m)|$$

where n is the number of inputs and m the indices of the bins (B_m), for which the uncertainty falls into. The error per bin and the uncertainty per bin are defined as:

$$err(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \|y_i - \hat{y}_i\|^2$$

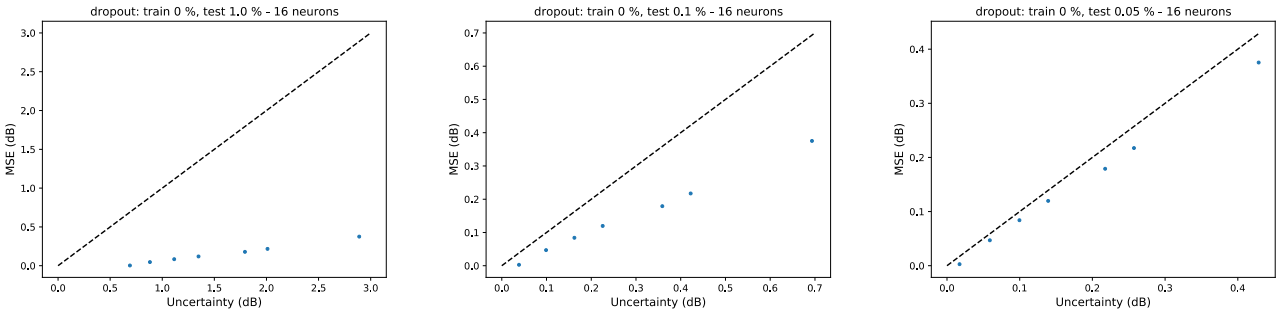
and

$$uncertainty(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} (\sigma_i^2)$$

where σ is the variance and the uncertainty is denoted with percentage ([48]).

Calibration is performed after training using the validation dataset. To perform the UCE calculations, we considered different scenarios: increasing the number of neurons in each layer, increasing the number of variational inferences, varying the dropout rates, and adding the dropout layers while training as well as the inference time. Previously, the PDF of residual distributions for 0.01 dropout and 100 variational inference (Fig. 3) showed asymmetry to the right. Therefore, to quantify miscalibration, we designed the experiments to establish the MLP forecaster architecture so that the UCE is at a minimum and as for the results, the uncertainty is calibrated. Several experiments were conducted, increasing the number of variational inferences from 100 to 500, varying the number of neurons (16, 128, 1024), and considering different dropout rates during training (0, 0.01, 0.1) and test (0.001, 0.005, 0.01, 0.1, 1). The resulting uncertainty vs. MSE curves are shown in Fig. 5 and Fig. 6 (without dropout during training), and Fig. 7 (with dropout during training). On these curves, the miscalibration is revealed by the degree of inequality to the identity function.

The experiments demonstrate that once the number of variational inferences increases to 500, when the dropout rate is 1%, for the basic MLP time series forecaster with 16 neurons in



(a) (b) (c)

Fig. 5. MLP with 16 neurons in each layer, 500 variational inferences during the test time, and no dropout while training, with dropout rates during testing time of: a) 1% (UCE 1.34); b) 0.1% (UCE = 0.08); c) 0.05% (UCE = 0.03). The best correlation between the MSE and the variance is at 0.05% dropout ratio.

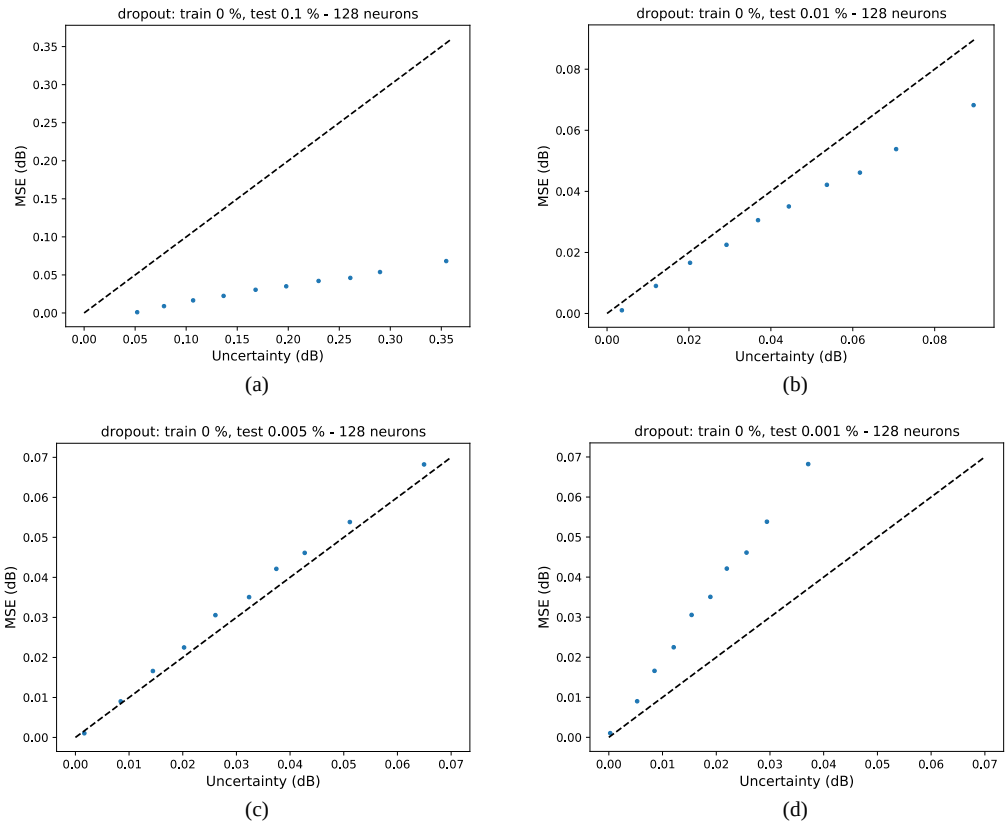


Fig. 6. MLP with 128 neurons in each layer, and no dropout during training with dropout rates of: a) 0.1% (UCE = 0.11); b) 0.01% (UCE = 0.003); c) 0.005% (UCE = 0.001); d) 0.001% (UCE = 0.002). 500 variational inferences.

each layer, the correlation between MSE and the variance of the distribution increases and the calibration error is 1.3% (Fig. 5).

When the dropout rate decreases to 0.1% and 0.05%, with the same number of variational inferences, the UCE drops (from 0.08 to 0.03). The best correlation between the variance and MSE is obtained at 0.05% dropout. Increasing the number of neurons to 128 for each layer with 500 variational inferences at the test time improves the UCE (Fig. 6) from 0.11 for 0.1% rate (Fig. 6.a) to 0.001 (Fig. 6.c). for 0.005% rate of dropout during the test time. Simultaneously, the correlation between the MSE and variance of the distributions increases (Fig. 6.a, 6.c), with the best correlation occurring at 0.01% dropout rate (Fig. 6.b).

However, decreasing the rate of test dropout to 0.001% does improve the correlation between MSE and distribution variance, nor does it improve the UCE (Fig. 6.d, UCE = 0.002). Increasing the number of neurons to 1024 does not bring any improvement neither.

The improvement of the uncertainty after calibration can be observed in the PDF of the residuals distributed around zero (Fig. 8). The skewness of the PDFs (Fig. 4) has been modified, the PDF is symmetrical and is concentrated around zero. The

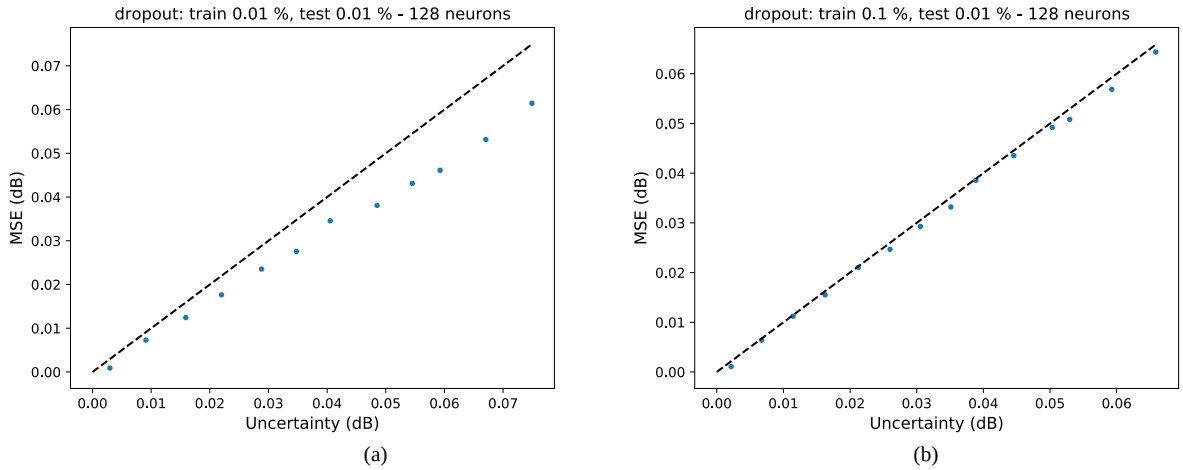


Fig. 7. MLP with 128 neurons and: a) 0.01% dropout rate during training and 0.01% dropout during testing (UCE = 0.002); b) 0.1% dropout during training and 0.01% dropout rate during testing (UCE = 0.0008).

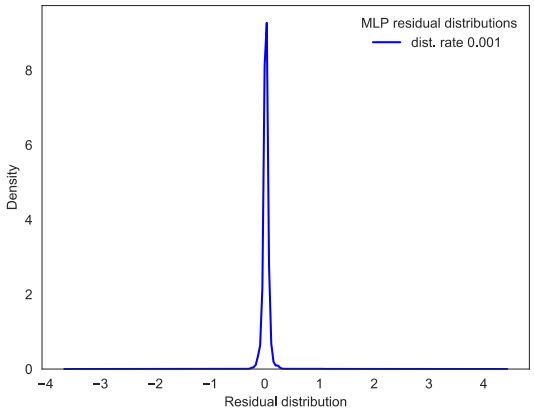


Fig. 8. PDF of MLP model with 128 neurons and no dropout during training.

upper and lower bounds of the predictions show less difference with the predictions (Fig. 9) as compared to those calculated with the Monte Carlo dropout and quantile regression (Fig. 2 and 4, respectively).

Comparing Fig. 7 with Fig. 6.b, the MLP forecaster with the same architecture but with dropout at the train time, Fig. 7.a and Fig. 7.b, show better UCE and correlation between the variance and the MSE of the distributions. Therefore, MLP time series forecaster with 0.1% train dropout and 0.01% test dropout, outperforms all the architectures considered in this study.

V. Conclusion

This paper successfully explored the application of DL forecasters for time series analysis in the field of optical signal transmission. Implementing N-Beats, whose code is not yet offered as a library in Python, and having it outperform other forecasters were some of the contributions of this article. For the target time series of this project, the performance of the naive model was better for the point prediction of the time intervals of choice. A naive model shows optimal performance when the behavior of the WDM channel is stable over time.

The uncertainty estimation was implemented using the Monte-Carlo dropout with the MLP model. The margins as the indicators of having variational inference, instead of point prediction, were also evaluated by the margins found with the Quantile Regression method. The results of the two approaches differed.

UCE calculations were performed to calibrate the uncertainty. We found the architecture of the forecaster directly affects the calibration and propose a MLP structure with minimum UCE and maximum correlation between variance and MSE of the variational inference distribution.

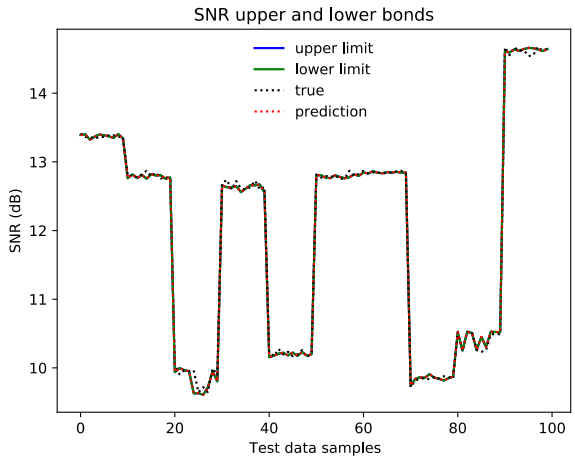


Fig. 9. Upper and lower bounds of time series forecast after UCE calculations

References

[1] H. Choi and S. S. Yang, "Network Survivability in Optical Networks with IP Prospective," *Encyclopedia of Internet Technologies and Applications*, 2008. https://www.igi-global.com/chapter/network-survivability-optical-networks-prospective/www.igi-global.com/chapter/network-survivability-optical-networks-prospective/16874 (accessed Jul. 21, 2021).

[2] S. Aladin, A. V. S. Tran, S. Allogba, and C. Tremblay, "Quality of Transmission Estimation and Short-Term Performance Forecast of Lightpaths," *Journal of Lightwave Technology*, vol. 38, no. 10, pp. 2807–2814, May 2020, doi: 10.1109/JLT.2020.2975179.

[3] H. Chouman, P. Djukic, C. Tremblay, and C. Desrosiers, "Forecasting Lightpath QoT with Deep Neural Networks," in *2021 Optical Fiber Communications Conference and Exhibition (OFC)*, Jun. 2021, pp. 1–3.

[4] *2.3 Time series patterns | Forecasting: Principles and Practice (2nd ed)*. Accessed: Aug. 01, 2022. [Online]. Available: https://otexts.com/fpp2/tspatterns.html

[5] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *arXiv:1905.10437 [cs, stat]*, Feb. 2020, Accessed: Jul. 12, 2020. [Online]. Available: http://arxiv.org/abs/1905.10437

[6] J. Brownlee, "What Is Time Series Forecasting?," *Machine Learning Mastery*, Dec. 01, 2016. https://machinelearningmastery.com/time-series-forecasting/ (accessed Aug. 02, 2021).

[7] Y. Ujjwal and J. Thangaraj, "Review and analysis of elastic optical network and sliceable bandwidth variable transponder architecture," *OE*, vol. 57, no. 11, p. 110802, Nov. 2018, doi: 10.1117/1.OE.57.11.110802.

[8] R. Ayassi, A. Triki, M. Laye, N. Crespi, R. Minerva, and C. Catanese, "An Overview on Machine Learning-Based Solutions to Improve Lightpath QoT Estimation," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, Bari, Italy, Jul. 2020, pp. 1–4. doi: 10.1109/ICTON51198.2020.9203755.

[9] O. Ayoub *et al.*, "Towards explainable artificial intelligence in optical networks: the use case of lightpath QoT estimation," *Journal of Optical Communications and Networking*, vol. 15, no. 1, pp. A26–A38, Jan. 2023, doi: 10.1364/JOCN.470812.

[10] G. Bergk, B. Shariati, P. Safari, and J. K. Fischer, "ML-assisted QoT estimation: a dataset collection and data visualization for dataset quality evaluation," *Journal of Optical Communications and Networking*, vol. 14, no. 3, pp. 43–55, Mar. 2022, doi: 10.1364/JOCN.442733.

[11] S. Aladin, S. Allogba, A. V. S. Tran, and C. Tremblay, "Recurrent Neural Networks for Short-Term Forecast of

Lightpath Performance," in *Optical Fiber Communication Conference (OFC) 2020 (2020), paper W2A.24*, Mar. 2020, p. W2A.24. doi: 10.1364/OFC.2020.W2A.24.

[12] C. Tremblay, S. Allogba, and S. Aladin, *Quality of transmission estimation and performance prediction of lightpaths using machine learning*. 2019, p. 23 (3 pp.). doi: 10.1049/cp.2019.0757.

[13] F. Usmani *et al.*, "Cross-feature trained machine learning models for QoT-estimation in optical networks," *OE*, vol. 60, no. 12, p. 125106, Dec. 2021, doi: 10.1117/1.OE.60.12.125106.

[14] S. Allogba, S. Aladin, and C. Tremblay, "Machine-Learning-Based Lightpath QoT Estimation and Forecasting," *Journal of Lightwave Technology*, vol. 40, no. 10, pp. 3115–3127, May 2022, doi: 10.1109/JLT.2022.3160379.

[15] A. Mezni, D. W. Charlton, C. Tremblay, and C. Desrosiers, "Deep Learning for Multi-Step Performance Prediction in Operational Optical Networks," in *Conference on Lasers and Electro-Optics (2020), paper STh4M.1*, May 2020, p. STh4M.1. doi: 10.1364/CLEO_SI.2020.STh4M.1.

[16] D. Levi, L. Gispan, N. Giladi, and E. Fetaya, "Evaluating and Calibrating Uncertainty Prediction in Regression Tasks," *arXiv:1905.11659 [cs, stat]*, Feb. 2020, Accessed: Jun. 10, 2021. [Online]. Available: http://arxiv.org/abs/1905.11659

[17] H. Al Osman and S. Shirmohammadi, "Machine Learning in Measurement Part 2: Uncertainty Quantification," *IEEE Instrumentation & Measurement Magazine*, vol. 24, pp. 23–27, May 2021, doi: 10.1109/MIM.2021.9436102.

[18] A. Brando, J. A. Rodríguez-Serrano, M. Ciprian, R. Maestre, and J. Vitrià, "Uncertainty Modelling in Deep Networks: Forecasting Short and Noisy Series," in *Machine Learning and Knowledge Discovery in Databases*, Cham, 2019, pp. 325–340. doi: 10.1007/978-3-030-10997-4_20.

[19] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," *arXiv:1505.05424 [cs, stat]*, May 2015, Accessed: Feb. 14, 2021. [Online]. Available: http://arxiv.org/abs/1505.05424

[20] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *arXiv:1506.02142 [cs, stat]*, Oct. 2016, Accessed: Sep. 08, 2020. [Online]. Available: http://arxiv.org/abs/1506.02142

[21] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds. Berlin, Heidelberg: Springer, 2004, pp. 63–71. doi: 10.1007/978-3-540-28650-9_4.

[22] S. M. Stigler, "The transition from point to distribution estimation," *Bulletin of the International Statistical Institute*, vol. 46, no. 2, 1975.

[23] A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic Forecasting of Sensory Data With Generative Adversarial Networks – ForGAN," *IEEE Access*, vol. 7, pp. 63868–63880, 2019, doi: 10.1109/ACCESS.2019.2915544.

[24] T. Gneiting and M. Katzfuss, "Probabilistic Forecasting," *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 125–151, 2014, doi: 10.1146/annurev-statistics-062713-085831.

[25] "Roger Koenker - Quantile Regression (2005) | PDF," *Scribd*. https://www.scribd.com/document/600212035/Roger-Koenker-Quantile-Regression-2005 (accessed Jan. 11, 2023).

[26] R. Koenker and G. Bassett, "Regression Quantiles," *Econometrica*, vol. 46, no. 1, pp. 33–50, 1978, doi: 10.2307/1913643.

[27] "Efron, B. (1991). Regression percentiles using asymmetric squared error loss. Vol.1, No.1." http://www3.stat.sinica.edu.tw/statistica/j1n1/j1n16/j1n16.htm (accessed Feb. 13, 2021).

[28] W. K. Newey and J. L. Powell, "Asymmetric Least Squares Estimation and Testing," *Econometrica*, vol. 55, no. 4, pp. 819–847, 1987, doi: 10.2307/1911031.

[29] T. Kneib, "Beyond mean regression," *Statistical Modelling*, Aug. 2013, doi: 10.1177/1471082X13494159.

[30] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," *arXiv:1612.01474 [cs, stat]*, Nov. 2017, Accessed: Dec. 02, 2021. [Online]. Available: http://arxiv.org/abs/1612.01474

[31] T. Gneiting and A. E. Raftery, "Weather Forecasting with Ensemble Methods," *Science*, vol. 310, no. 5746, pp. 248–249, Oct. 2005, doi: 10.1126/science.1115255.

[32] A. E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski, "Using Bayesian Model Averaging to Calibrate Forecast Ensembles," *Monthly Weather Review*, vol. 133, no. 5, pp. 1155–1174, May 2005, doi: 10.1175/MWR2906.1.

[33] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An Introduction to Variational Methods for Graphical Models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, Nov. 1999, doi: 10.1023/A:1007665907178.

[34] A. Doucet, N. de Freitas, and N. Gordon, "An Introduction to Sequential Monte Carlo Methods," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York, NY: Springer, 2001, pp. 3–14. doi: 10.1007/978-1-4757-3437-9_1.

[35] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory - COLT '93*, Santa Cruz, California, United States, 1993, pp. 5–13. doi: 10.1145/168304.168306.

[36] J. Paisley, D. Blei, and M. Jordan, "Variational Bayesian Inference with Stochastic Search," *arXiv:1206.6430 [cs, stat]*, Jun. 2012, Accessed: Oct. 28, 2021. [Online]. Available: http://arxiv.org/abs/1206.6430

[37] M. Titsias and M. Lázaro-Gredilla, "Doubly Stochastic Variational Bayes for non-Conjugate Inference," in *International Conference on Machine Learning*, Jun. 2014, pp. 1971–1979. Accessed: Oct. 28, 2021. [Online].

Available: https://proceedings.mlr.press/v32/titsias14.html

[38] M. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic Variational Inference." arXiv, Apr. 22, 2013. Accessed: Feb. 01, 2023. [Online]. Available: http://arxiv.org/abs/1206.7051

[39] M. Teye, H. Azizpour, and K. Smith, "Bayesian Uncertainty Estimation for Batch Normalized Deep Networks," in *Proceedings of the 35th International Conference on Machine Learning*, Jul. 2018, vol. 80, pp. 4907–4916. [Online]. Available: http://proceedings.mlr.press/v80/teye18a.html

[40] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Mar. 2015, Accessed: Jul. 12, 2021. [Online]. Available: http://arxiv.org/abs/1502.03167

[41] R. Koenker, "Quantile Regression: 40 Years On," *Annual Review of Economics*, vol. 9, no. 1, pp. 155–176, 2017, doi: 10.1146/annurev-economics-063016-103651.

[42] H. Maryam, T. Panayiotou, and G. Ellinas, "Learning quantile QoT models to address uncertainty over unseen lightpaths," *Computer Networks*, vol. 212, p. 108992, Jul. 2022, doi: 10.1016/j.comnet.2022.108992.

[43] E. Seve, J. Pesic, C. Delezoide, S. Bigo, and Y. Pointurier, "Learning process for reducing uncertainties on network parameters and design margins," *Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. A298–A306, Feb. 2018, doi: 10.1364/JOCN.10.00A298.

[44] R. Lambrugh, *Deep Learning With Python by Francois Chollet*. Accessed: Jul. 13, 2020. [Online]. Available: https://www.academia.edu/40318927/Deep_Learning_With_Python_by_Francois_Chollet

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[46] N. Chan, "Uncertainty estimation for Neural Network — Dropout as Bayesian Approximation," *Medium*, Feb. 02, 2019. https://towardsdatascience.com/uncertainty-estimation-for-neural-network-dropout-as-bayesian-approximation-7d30fc7bc1f2 (accessed Jun. 05, 2022).

[47] C. M. Bishop, "Mixture density networks," 1994. http://publications.aston.ac.uk/id/eprint/373/ (accessed Feb. 14, 2021).

[48] M.-H. Laves, S. Ihler, J. F. Fast, L. A. Kahrs, and T. Ortmaier, "Well-Calibrated Regression Uncertainty in Medical Imaging with Deep Learning," in *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, Sep. 2020, pp. 393–412. Accessed: Jun. 05, 2022. [Online]. Available: https://proceedings.mlr.press/v121/laves20a.html

[49] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Appendix," *arXiv:1506.02157 [stat]*, May 2016, Accessed: May 26, 2021. [Online]. Available: http://arxiv.org/abs/1506.02157

[50] "tf.nn.dropout | TensorFlow Core v2.5.0," *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/nn/dropout (accessed Jul. 09, 2021).

[51] M.-H. Laves, S. Ihler, and T. Ortmaier, "Uncertainty Quantification in Computer-Aided Diagnosis: Make Your Model say 'I don't know' for Ambiguous Cases," arXiv, arXiv:1908.00792, Aug. 2019. doi: 10.48550/arXiv.1908.00792.

[52] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate Uncertainties for Deep Learning Using Calibrated Regression," *arXiv:1807.00263 [cs, stat]*, Jun. 2018, Accessed: Feb. 14, 2021. [Online]. Available: http://arxiv.org/abs/1807.00263

[53] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *Proceedings of the 34th International Conference on Machine Learning*, Jul. 2017, pp. 1321–1330. Accessed: Jun. 05, 2022. [Online]. Available: https://proceedings.mlr.press/v70/guo17a.html