2022 IEEE 15th International Conference on Cloud Computing (CLOUD) | 978-1-6654-8137-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CLOUD55607.2022.00018

Bayesian Uncertainty Modelling for Cloud Workload Prediction

Andrea Rossi
SFI CRT in Artificial Intelligence
University College Cork
Cork, Ireland
a.rossi@cs.ucc.ie

Andrea Visentin
School of Computer Science
University College Cork
Cork, Ireland
andrea.visentin@ucc.ie

Steven Prestwich
School of Computer Science
University College Cork
Cork, Ireland
s.prestwich@cs.ucc.ie

Kenneth N. Brown
School of Computer Science
University College Cork
Cork, Ireland
k.brown@cs.ucc.ie

***Abstract*—Providers of cloud computing systems need to manage resources carefully to meet the desired Quality of Service and reduce waste due to overallocation. An accurate prediction of future demand is crucial to allocate resources to service requests without excessive delays. Current state-of-the-art methods such as Long Short-Term Memory-based models make only point forecasts of demand without considering the uncertainty in their predictions. Forecasting a distribution would provide a more comprehensive picture and inform resource scheduler decisions. We investigate Bayesian Neural Networks and deep learning models to predict workload distribution and evaluate them on the time series forecasting of CPU and memory workload of 8 clusters on the Google Cloud data centre. Experiments show that the proposed models provide accurate demand prediction and better estimations of resource usage bounds, reducing overprediction and total predicted resources, while avoiding underprediction. These approaches have good runtime performance making them applicable for practitioners.**

***Index Terms*—Cloud Computing, Bayesian Neural Network, Workload Prediction, Deep Learning, Time Series Forecasting**

I. Introduction

The demand for cloud computing services is continuing to grow; it is estimated that over $482 billion will be spent in 2022, a rise from $313 billion in 2020 [1]. That demand has increased significantly during the COVID-19 pandemic [2], where work from home has been widely adopted. Ensuring that resources are available to meet demand is critical for a competitive service. Cloud computing providers aim to preconfigure the machines in advance in order to offer a high Quality of Service (QoS), including low latency, high availability, and high reliability [3]. If the demand can be predicted accurately, providers can expect higher QoS measures and better resource utilization by reducing preconfigured but idle machines [4]. However, cloud service demands are hard to predict, exhibiting characteristics such as diversity, large scale, burst and uncertainty [5]. An accurate model of the variance in the demand as time progresses would allow service providers to make a precise trade-off between QoS and resource cost and an overall reduction of energy and maintenance costs, with a possible improvement in environmental impact [6]. In this paper, we address the cloud service prediction problem and focus on modelling the uncertainty in the prediction.

There are many techniques used to forecast the demand based on time series analysis, including statistical [7, 8, 9], machine learning (ML) [10, 11, 12] and deep learning (DL) [13, 14, 15] approaches. However, most prior work in cloud computing has focused primarily on point estimate predictions, i.e. single values. Uncertainty aspects are taken into account [9, 16, 17], but to the best of our knowledge, no Bayesian Neural Network (BNN) models have been proposed that output the uncertainty around the prediction as well.

In this paper, we design DL models that accurately predict the future workload demand distribution in CPU and memory usage. We implement a Hybrid Bayesian Neural Network (HBNN) by adding a Bayesian layer into an Long Short-Term Memory-based (LSTM) architecture. Our approach is capable of modelling the epistemic uncertainty in the data through a Bayesian layer and the aleatoric uncertainty as it forecasts a probability distribution of the future demand employing the variational inference (VI) technique [18]. Moreover, we evaluate the performance of the Bayesian layer in terms of accuracy and runtime performance, comparing the model to its non-Bayesian version. The probability distribution prediction allows us to compute the confidence intervals. The confidence bounds can be used to predict an upper bound (UB) of the total resources needed to meet demand with a certain level of confidence corresponding to the desired QoS level. We evaluate our prediction method using 29-day load traces of 8 different Google Cloud clusters distributed around the world [19].

In particular, our contributions in this paper are as follows:

- We propose an HBNN model to forecast the distributional parameters of the future resource demand in order to model the uncertainty in the data through a Bayesian layer.
- We demonstrate the effect of the Bayesian layer in terms of accuracy and runtime performance efficiency.
- We implement the model and conduct an experimental comparison with predictions based on AutoRegressive Integrated Moving Average (ARIMA), Generalized autoregressive conditional heteroskedasticity (GARCH) and an LSTM model.
- We design experiments inspired by resource allocation, QoS goals and runtime efficiency for the deployment of the model in real-world applications. The experimental results show that the confidence-based predictions of our

2159-6190/22/$31.00 ©2022 IEEE
DOI 10.1109/CLOUD55607.2022.00018

model generate improved bounds on the resources needed compared to state-of-the-art DL methods based on point estimates and compared to heuristic approaches.
- We propose a new way of using historical data to automatically compute a UB from any point estimate in order to meet a certain QoS level.
- Our models compute a probability distribution of the demand from which we automatically obtain the UB of the prediction for any specified confidence level.

For the remainder of the paper, we interchangeably use the terms workload, future demand, and resource demand. The rest of the paper is organized as follows. Section II describes the related work in this field. In Section III we describe the baselines and the proposed models by highlighting the main differences. Section IV presents the evaluation results of our method based on Google's load traces data. Finally, Section V concludes the paper with a discussion on future work.

II. Related Work

Workload prediction has been widely investigated over the last two decades. Among the first approaches used for this task, there are statistical methods such as Moving Average (MA) and Auto-Regression (AR) [20]. In many cases, ML methods can provide a higher performance [14, 21]. Di *et al.* [10] designed a Bayesian model to predict the mean load of the Google data centre machines. They extracted nine candidate features that the model can use for characterizing trace fluctuation. Their model outperformed the state-of-the-art methods based on MA, AR or noise filtering. Hu *et al.* [11] designed a model which combines a support vector regression (SVR) algorithm with a Kalman smoother. They showed that their algorithm could meet QoS requirements by reducing the total allocated CPU.

DL models have been applied in many areas of cloud computing management, differing in terms of models, training approaches, datasets and prediction-based applications. Song *et al.* [14] applied an LSTM model to show the performance of the prediction of the host load compared to the standard statistical methods and ML approaches, reaching state-of-the-art results. Zhang *et al.* [22] used a Recurrent Neural Network (RNN) model to predict the workload of a cloud computing system. They compared their model to the ARIMA model, showing that the DL approach outperforms statistical methods due to the high variability of the dataset. Kumar *et al.* [23] trained a standard Neural Network (NN) to predict the resource demand. They trained the network using a differential evolution technique, outperforming the standard NN and focusing on the complexity of the training phase. Kumar *et al.* [24] used a NN for predicting the resource demand, but they trained the network using a black hole optimization algorithm, outperforming their previous work with differential evolution. Wang *et al.* [8] applied an LSTM model for predicting the CPU usage for autoscaling. The authors compared the performance of the LSTM model with a Holt-Winters exponential smoothing approach, showing the robustness of the LSTM model and the reduction of the over-provisioned CPU compared to traditional autoscaler. Zhang *et al.* [25] used a Deep Belief Network (DBN) to predict the CPU and memory demand in a cloud environment. Their model outperforms both the ARIMA predictor and the fractal modelling technique in terms of Mean Squared Error (MSE). Ensembles of models of different nature have also been developed. Liu *et al.* [26] proposed an ensemble of MA, NN, Multilinear Regression, and Weighted Average approaches, outperforming them in terms of MSE and Mean Absolute Error (MAE) when applied individually. Similarly, Herbst *et al.* [27] combined different statistical methods and used a decision tree algorithm to select the best approach to apply in the forecasting task according to a user-specified objective. This ensemble outperforms all the models applied individually in terms of Service Level Agreement (SLA) violation. Zhou *et al.* [28] proposed an ensemble model based on R-Transformer combined with an AR component. Before feeding the model, they decomposed the trace into multiple Intrinsic Mode Functions by applying the Variational Model Decomposition method. They compared this ensemble with ARIMA, Gated Recurrent Unit (GRU), LSTM encoder-decoder network (LSTM-ED) and LSTM-ED with attention, outperforming them.

All the research above focused on point estimate workload prediction without capturing the concept of the uncertainty around that prediction. There have been attempts of modelling uncertainty aspects of the workload prediction. Calheiros *et al.* [9] developed an ARIMA model to predict the future workload demand for web servers requests. They evaluate the model in terms of accuracy and resource utilization and consider QoS parameters such as response time and rejection rate. Resource allocation is based on the confidence bounds of the prediction of their model. Minarolli *et al.* [17] developed a probabilistic model of the prediction error and compared it to uncertainty-unaware architectures, leading to more stability and better QoS performance for migration applications. None of the aforementioned approaches exploits DL approaches to tackling uncertainty in the prediction. BNNs are able to capture the uncertainty and have been applied in many different scenarios, especially in meteorology and atmospheric phenomena prediction, but not in the context of cloud computing workload forecasting. Liu *et al.* [29] designed a variational Bayesian deep learning model for probabilistic spatiotemporal forecasting and the estimation of temporal and spatial uncertainty of the wind speed. Khan *et al.* [30] applied a BNN for daily river flow and reservoir inflow simulation. Their work is one of the first in which BNNs are applied to model the uncertainty using confidence bounds. Xie *et al.* [31] evaluated the application of a BNN model for predicting motor vehicle crashes and reducing the over-fitting of the data compared to standard NNs. Lampinen *et al.* [32] reviewed three different applications of BNN. The first case study applies a BNN as a regression task for predicting the quality of fresh concrete. In this case, the optimization algorithm does not learn the weights of the network because the dataset is too small due to data collection costs. The second case study includes the application of a BNN for the electrical impedance tomography problem where, from

surface measurements, the aim is to reconstruct the object. The BNN reaches promising results in this problem. The last case study uses a BNN for an image classification task. The aim is to correctly classify tree images based on their volume, showing that the BNN is competitive with state-of-the-art methods. However, none of the works mentioned above used the BNN to model uncertainty in workload prediction in cloud computing.

III. Workload Prediction Models

The resource manager stores the workload history and provides it to the predictive model to forecast the future workload. The resource manager uses the prediction for the configuration of the virtual machines (VMs). The workload prediction schema is depicted in Fig. 1. In this section, we describe all the time series forecasting models used herein.

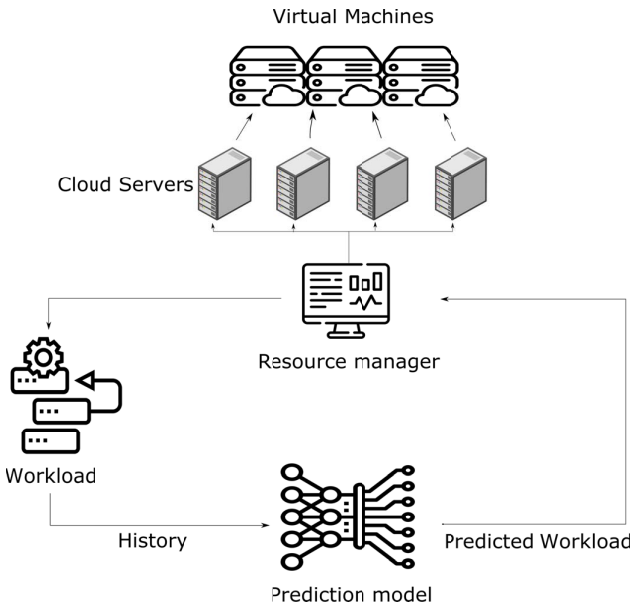

Fig. 1. Workload prediction schema in Cloud Computing

A. Baseline models

- ARIMA: is a generalization of ARMA. In an ARIMA model, there are three parameters that are used to help model the major aspects of a times series: seasonality, trend, and noise. It is a well-known baseline in workload prediction [9]. We found the best order of the model for each dataset via grid search by splitting the training set into two parts where the second, corresponding to the 20% of the data, is used as the validation set. We selected the best parameters according to the best Akaike Information Criterion (AIC). Because of the size of the datasets (and the 5-minutes granularity), ARIMA does not converge with all the data available. For this reason, a sliding window of the past 288 timestamps is used to retrain the model at each timestamp.
- GARCH: aims to model the volatility of a time series [33]. It is typically used for dealing with financial data. It has been used as a baseline for workload prediction in Cloud Computing in [21]. This model focuses on modelling the variance of the output, while for the mean component a simple least square method is applied. We found the best parameters similarly to ARIMA. We also experimentally found that keeping a sliding window of the past history instead of the whole dataset is faster and provides better results. We used a sliding window of 288 timestamps (1 day) and we retrained the model at each forecast step. The model's parameters are estimated via Maximum Likelihood Estimation (MLE) with a Normal distribution with zero mean and unit variance as prior distribution.
- LSTM: We use a standard LSTM-based architecture as a baseline model. Inspired by a simpler architecture presented in [13], we design the architecture by varying the numbers and the types of layers of the network. We experimentally found the best architecture structured as follows. For a given workload interval to be predicted, the network's input is a sequence of past workload values. From now on, we will refer to it as the LSTM model. We experimentally found that the best input size is 144, corresponding to the 12 hours average workload before the target 5-minute interval of our prediction. The input is fed to a 1-dimensional convolutional (1DConv) layer; a common approach proved to be effective in time series prediction [34]. An LSTM layer follows the 1DConv layer. LSTM is a recurrent network layer that is widely used in DL to deal with sequential and time-series data. This layer can avoid pathological data using the forget gate and input gate, achieving high accuracy in predicting [12]. Two dense layers follow the LSTM layer. The last one has one single neuron whose output corresponds to the prediction of the network (point estimate). The network is optimized by minimizing the MSE through the Adam algorithm [35].

B. HBNN model

BNNs are conventional NNs that have been extended with posterior inference to control over-fitting and model uncertainty [36]. While a single set of weights characterizes the conventional NNs, the BNNs at each weight has an associated probability distribution. The learning process of these distributions includes the assumption of a prior distribution of the weights with fixed parameters. In particular, we experimentally saw that a Normal distribution with mean equals 0 and variance equals 1 works in practice, as in the VI approach used in GARCH. Then, Bayes theorem is used for modelling the posterior distribution of the weights, which is the distribution we want to learn. We modelled the posterior distribution as a Multivariate Normal distribution where the learnable parameters are the means, variances and covariances. The Bayes theorem formulation includes the prior distribution and the likelihood function of the observations, i.e. the training

data. The critical challenge in this approach is the computation of the integrals of the posterior distribution that is numerically intractable because of the dimensionality of the latent space, which depends on the number of network parameters. For this reason, the Bayesian framework includes a method to approximately compute these integrals, including the VI approach [18]. VI is used to find a distribution of the weights close to the distribution of the input data by minimizing the Kullback-Leibler divergence [37]. We adopted a network structure similar to the LSTM baseline, differing by just the last two layers. The differences in terms of architecture are minimal to better appreciate the effect of the new components of the proposed model. We tried to limit the differences to fully appreciate their impact on the result. A Bayesian layer replaces the first dense layer, making the model an HBNN, as done in [38], and [39]. We add this layer in order to capture the epistemic uncertainty due to the small size of the datasets, i.e. the lack of knowledge on the model parameters that can be reduced if we have more data [40]. The Bayesian layer is followed by a dense layer with two neurons used to predict the mean μ and the variance σ^2 of a Normal distribution ($\mathcal{N}(\mu, \sigma^2)$) incorporated in a distribution layer used as the output layer. The output distribution captures the aleatory uncertainty due to the randomness of the workload demand. To compute the likelihood of having the real demand (targets) from the distribution produced by the model, we use the negative log-likelihood as a loss function since the output is a distribution and not a point estimate. Therefore, the output of this model is a Gaussian distribution with a predicted mean and variance. An illustration of the HBNN model architecture is depicted in Fig. 2.

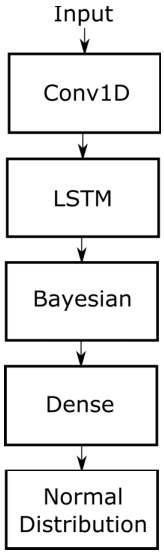

Fig. 2. HBNN model architecture

C. LSTMD model

The architecture of this model is very similar to the HBNN one, where a standard dense layer replaces the Bayesian layer, with two consecutive dense layers before the output distribution. Because it outputs a distribution, the negative log-likelihood is used as a loss function.

IV. Experiments

This section presents a set of experiments to show the potential benefits of considering the uncertainty in cloud workload prediction. First, we investigate if introducing the Bayesian layer and/or distribution as output affect the performances in point estimation in terms of MSE and MAE. The second part focuses on how including uncertainty in the prediction can provide more useful information. We use the two models to predict a UB of the upcoming demand that should hold for a given probability; this approach is equivalent to satisfying a certain QoS. Finally, we evaluate the runtime performance of the DL models.

A. Experimental Setup

The experiments have been conducted in Ubuntu 20.04 with a CPU Intel®Xeon®Gold 6240 at 2.60GHz and GPU NVIDIA Quadro RTX 8000 with 48 GB of memory. The experiments are run over Google Cloud Trace 2019 which includes the historical workload data of 8 clusters. The trace is used to train different models: ARIMA, GARCH and an LSTM-based model are used as baselines; HBNN and LSTMD are the proposed models. Since the machine pre-provisioning is not immediate, we need to give the resource manager enough time to exploit the forecast. According to [41], 10 minutes are enough for the majority of the applications, so we forecast the workload 10 minutes in the future with respect to the historical data as depicted in Fig. 3.

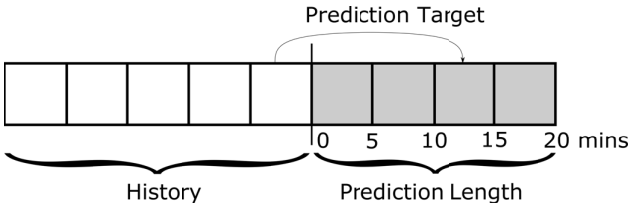

Fig. 3. Prediction target of our models. Each square represents a 5 min interval of the cloud workload.

The networks are trained by splitting the datasets into two parts. The first 80% of each dataset is used as the training set (corresponding to about 23 days). The remaining part is used as the test set (about 6 days). The networks are optimized and trained for each cluster. More details on the implementation of the predictive models and on the hyperparameters search can be found in the shared repository[1]. Given the best hyperparametrization, we train the DL models 10 times for each cluster and resource with different random seeds for

[1]https://github.com/andreareds/UncertaintyCloudWorkload

initialization and we select the best performing model in terms of MSE for both baselines and proposed models.

B. Dataset

The Google cluster trace 2019 records the total resource utilization of each task on eight different clusters (from A to H) geographically distributed around the world. The median number of machines per cluster is approximately 10000. The trace begins on Wednesday, May 1, 2019, and it records 29 days' resource usage of different jobs. Each job is divided into tasks, and each task is composed of a set of resource requirements. The resource requirements are measured in terms of CPUs and memory usage. We use this trace to predict demand and compare the various prediction methods.

For replicability, in this section, we briefly describe how we preprocessed the dataset. Missing records in the trace are ignored for simplicity. We processed it via Google BigQuery, a Big Data querying tool in Google Cloud Platform, allowing the trace to be processed more efficiently. The BigQuery platform compiles a time series dataset that details the average CPU and memory usage of machines for every 5-minute interval, similar to other works in cloud workload prediction [24, 26, 27]. Therefore, for the 29 days period, we obtain 8352 data points for each cluster. The dataset is derived from the Instance Usage table of Google's cluster dataset. To extract the average CPU and memory workloads for each time window of a given cluster, we consider that a task can run only partially on the 5-minutes window of interest. For this reason, we multiply the average CPU and memory workload by a weight corresponding to the run time interval within that window as done in [13]. As a result, a task with a one-minute run time in a five-minute window does not count as running for the entire period. Data is scaled in the range [-1, 1] because the scaling strategy can speed up the convergence when using the gradient descent algorithm [42].

C. Point Estimate Accuracy

This experiment compares the point prediction error of the three baselines and the two proposed models. We evaluate them in terms of MSE and MAE, widely used metrics in time-series analysis [43]. In the case of the baseline model, the computation of the errors is straightforward because it produces a single output at each prediction. The HBNN and LSTMD output the distribution parameters, i.e. the mean and the variance of a Normal distribution. This distribution can be reduced into a point estimate by considering the mean μ value and ignoring the variance [44]. MSE and MAE for the HBNN and LSTMD approaches are relative to the mean of the output (while the LSTM value is the best over its 10 runs).

Table I shows the results for CPU and memory usage. As excepted from the literature, the DL models outperform the statistical methods in terms of MSE and MAE. For the sake of brevity and to have more readable results, we will omit the statistical methods in the following experiments. Having a lower point prediction accuracy, they are consistently outperformed by the LSTM model. The results show that our models achieve slightly better performance for both CPU and memory demand compared to the LSTM. However, this difference is within the statistical error. According to the Diebold-Mariano Test [45] at a 95% confidence level, the three DL models have the same accuracy. Therefore, according to this experiment, it is hard to draw a clear conclusion on which of the DL approaches is better for point prediction.

TABLE I
MSE AND MAE COMPARISON

Model	CPU		Memory	
	MSE	**MAE**	**MSE**	**MAE**
LSTM	0.0172	0.0942	0.0174	0.0905
HBNN	0.0169	**0.0936**	**0.0163**	0.0908
LSTMD	**0.0167**	0.0937	**0.0163**	**0.0878**
ARIMA	0.0206	0.1031	0.0202	0.0982
GARCH	0.0857	0.2151	0.1025	0.23

D. Interval Estimate Evaluation

One of the benefits of workload demand forecasting is that it allows resource schedulers to make decisions in advance based on the prediction. In particular, we aim to predict an upper bound (UB) of the upcoming requests with a high probability in order to achieve a given QoS, dependent on the agreement with the customers [46]. QoS performance can include different parameters, such as latency between the arrival and serving time of the request. In the case of the HBNN and LSTMD, one of the advantages of predicting a distribution is that we can compute confidence bounds on the prediction. We compute the one-sided upper confidence bound with a certain confidence level, e.g. 95% of the prediction of resources units. In this way, the UB should be able to exceed the upcoming requests with a 95% probability.

The LSTM model does not capture the concept of uncertainty in its point estimate. To compute a UB of the upcoming demand prediction, we add an extra number of resources (e.g. 5% of the total available resources) to the point prediction of the model. This threshold approach has been previously used in the literature by [17], and [16]. Therefore, the point estimate plus the threshold represents the UB of the prediction for the LSTM model.

Defining N as the size of the test set, y_i as the target of the i-th instance of the test set and $\hat{y}_i^{UB}$ as the UB of the prediction $\hat{y}_i$ of the i-th instance of the test set, we compute the following statistics for the HBNN, the LSTMD and the LSTM models to compare them in terms of overprediction efficiency and QoS level:

- **Success Rate (SR)**: Percentage of corrected predicted values i.e. the portion of labels that are within the UB of the prediction, i.e.

$$\text{SR} = \frac{|A|}{N} \cdot 100 \quad \text{where} \quad A = \{\hat{y}_i^{UB} | \hat{y}_i^{UB} \geq y_i, \; i = 1, \ldots, N\} \tag{1}$$

- **Overprediction (OP)**: Sum of extra resources that are predicted at each timestamp for the requests that are below the UB of the prediction, i.e.

$$\text{OP} = \sum_{i}^{N} \left(\hat{y}_i^{UB} - y_i\right) \quad \text{s.t.} \quad \hat{y}_i^{UB} \geq y_i, \quad i = 1, \ldots, N \tag{2}$$

- **Underprediction (UP)**: Sum of unmatched demand for the requests that are above the UB of the prediction, i.e.

$$\text{UP} = \sum_{i}^{N} \left(y_i - \hat{y}_i^{UB}\right) \quad \text{s.t.} \quad \hat{y}_i^{UB} < y_i, \quad i = 1, \ldots, N \tag{3}$$

- **Total Predicted Resources (TPR)**: Sum of resources that are predicted, i.e.,

$$\text{TPR} = \sum_{i}^{N} \hat{y}_i^{UB} \tag{4}$$

These metrics aim to understand if, by modelling uncertainty, we can provide more useful information to the resource scheduler that manages a cloud cluster. We aim to predict a number of resources that, if available, should be able to satisfy all the upcoming requests with a given probability. In particular, *SR* corresponds to the percentage of satisfied requests, which is strictly related to the QoS level we want to achieve. However, we also want to reduce the costs of operating a cluster by avoiding overprediction. The *OP* measures the excess of resources predicted but not needed in order to serve the upcoming requests, while the *UP* measures the part of unmatched demand that would have to be processed in the next period. Finally, *TPR* measure the sum of the predicted bounds; this can be considered as an approximation of the cluster's variable operating cost since a higher prediction causes the scheduler to allocate more resources and so increases the electricity consumption.

1) Fixed Threshold: Tables II and III show the average results for all the clusters for CPU and memory requests for an HBNN with a confidence level of 95%, 97% and 99%. To compute the correct threshold for the LSTM, we varied it until reaching similar success rates (QoS measure) to each of the HBNN and LSTMD separately. We can see that with a similar *SR*, the *OP*, the *UP* and *TPR* are lower in the case of the HBNN for the CPU requests. This means that we can achieve similar QoS performances by predicting lower resource usage. Similarly, the LSTMD outperforms the LSTM in terms of memory and gets a lower *UP* for the memory. LSTMD appears to outperform HBNN on TPR and OP for the memory prediction, but it does so by undershooting on the success rate, with poorer performance on underprediction. Therefore, the HBNN model seem to be superior in terms of prediction quality. It benefits from predicting the distribution; the bounds (prediction's variance) are higher if the demand in the next interval is more uncertain and vice versa.

We then analyze how the uncertainty predicted by the Bayesian network varies across the clusters. Fig. 4 and 5 show the standard deviation and the actual resource demand for two clusters. For better visualization, we plot only a window of 24 hours. First, we can see that the predicted standard deviation strongly changes over time. In Fig. 4, the standard deviation and the resources seem to have an inverse relationship; the uncertainty is low when the utilization is high and vice versa. Other clusters display an opposite behaviour, where the higher demand is associated with higher uncertainty; this is particularly clear in the example in Figure 5. We believe that these are due to the different processes that originate the demand, e.g., users, companies, institutions, internal jobs. However, it is interesting to see that the network learns these patterns. A common feature is that the standard deviation tends to increase when there is a sharp demand change and tends to stabilize when the trend is steadier.

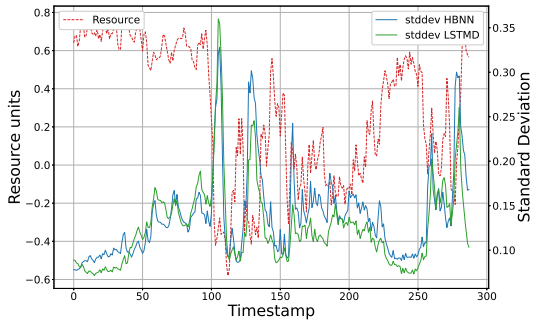

Fig. 4. Actual demand vs predicted standard deviation for a 24-hours window - Cluster A

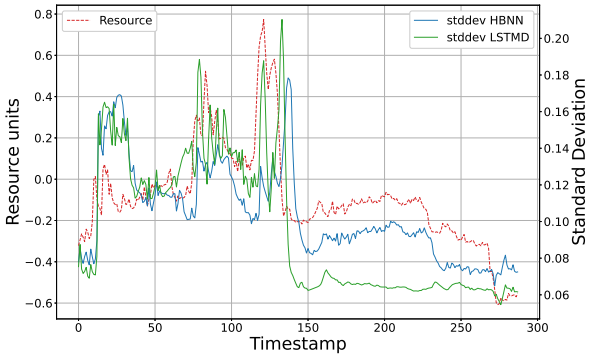

Fig. 5. Actual demand vs predicted standard deviation for a 24-hours window - Cluster H

In Fig. 6 we plot the average standard deviation of the clusters, which control the uncertainty of the prediction for both HBNN and LSTMD over different prediction lengths.

TABLE II
CPU USAGE STATISTICS OF THE HBNN, LSTMD AND LSTM. FOR A GIVEN CONFIDENCE LEVEL, THE BEST RESULT FROM THE 4 MODELS IS IN BOLD; OTHERWISE THE BEST OF A PAIRWISE COMPARISON IS IN ITALICS.

Model	LSTM	LSTMD	LSTM	HBNN	LSTM	LSTMD	LSTM	HBNN	LSTM	LSTMD	LSTM	HBNN
Confidence/ Threshold	19.6%	95%	20.0%	95%	22.4%	97%	23.4%	97%	28.8%	99%	31.5%	99%
SR	94.58	94.56	94.87	94.87	96.36	96.38	96.82	96.83	98.22	98.22	98.62	98.63
OP	*156.2*	158.4	159.6	**155.0**	*178.4*	180.2	185.4	**176.8**	228.1	*222.2*	249.4	**218.8**
UP	3.8	*3.2*	3.6	**2.9**	2.7	*2.2*	2.5	**1.9**	1.5	*1.1*	1.1	**0.9**
TPR	*1242.5*	1245.4	1246.1	**1242.3**	*1265.8*	1268.2	1273.0	**1265.0**	1316.7	*1311.2*	1338.4	**1308.0**

TABLE III
MEMORY USAGE STATISTICS OF THE HBNN, LSTMD AND LSTM. FOR A GIVEN CONFIDENCE LEVEL, THE BEST RESULT FROM THE 4 MODELS IS IN BOLD; OTHERWISE THE BEST OF A PAIRWISE COMPARISON IS IN ITALICS.

Model	LSTM	LSTMD	LSTM	HBNN	LSTM	LSTMD	LSTM	HBNN	LSTM	LSTMD	LSTM	HBNN
Confidence/ Threshold	18.0%	95%	20.2%	95%	20.4%	97%	23.8%	97%	26.2%	99%	30.4%	99%
SR	93.91	93.9	95.53	95.53	95.65	95.65	96.91	96.93	97.76	97.75	98.57	98.55
OP	149.5	**141.2**	*166.6*	175.6	168.1	**159.9**	*194.4*	199.3	212.7	**196.0**	246.2	*244.8*
UP	4.1	*3.4*	3.1	**2.4**	3.0	*2.4*	2.0	**1.6**	1.5	*1.3*	0.9	**0.8**
TPR	1003.0	**995.3**	*1021.0*	1030.7	1022.6	**1015.0**	*1049.9*	1055.2	1068.7	**1052.3**	1102.8	*1101.6*

As expected, we can see that the level of uncertainty grows proportionally with the prediction length.

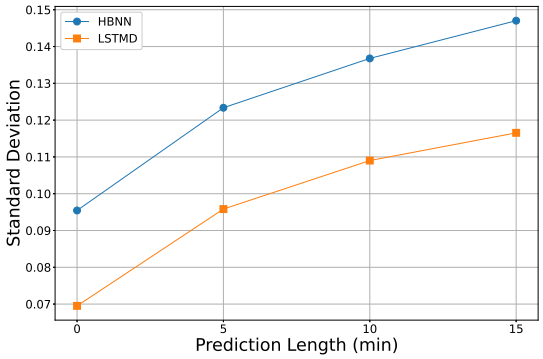

Fig. 6. Uncertainty over time

2) Learned Threshold: This experiment aims to see if it is possible to achieve the desired QoS level using the LSTM model by learning the threshold from the training set. Instead of computing the threshold for the LSTM model such that *SR* is closed to the results achieved by the HBNN model (like in the previous experiment), we calculate the optimal threshold over the training data.

With the LSTM model, we use as the threshold the minimum value such that the success rate is lower or equal to the target one in the training set. The goal is to model the uncertainty of future demand by considering the variability of the past demand. We consider this as the baseline approach to predict a bound of the forecast that satisfies a given QoS measure when dealing with a point workload predictor such as the ones available in the literature. We calculate two thresholds: a specific one for each cluster (C-LSTM) and an average among all the clusters (AVG-LSTM). We then compare its performances with the HBNN and LSTMD models. The better model should achieve a *SR* as close as possible to the desired target QoS while using fewer resources. If the success rate is too low, the demand has not been correctly forecasted too many times. If it is too high, it could have been possible to satisfy the desired QoS and fewer predicted resources, so fewer operative costs.

To assess the correctness of the desired prediction, we checked if the percentage of ground truth demand of the test set is in the confidence level of the HBNN predicted probability distribution. In Fig. 7 and 8, we plot the desired confidence level versus the percentage of time the upcoming request was inside the predicted probability. The points lie very close to the line $y = x$, meaning that the actual success rate is similar to the desired confidence level we would like to achieve. This means that the predicted probability is accurate.

The results are listed in Tables IV and V for the CPU and memory demand, respectively. The HBNN *SR* is the closest one to the desired value in four out of six experiments, except for the 99% confidence level for CPU and memory demands. This shows that the predicted probability distribution is really close to the realised workload. However, in those situations, the AVG-LSTM reaches an *SR* closer to the target, but with a not negligible increment of *TPR* (about 0.1% more accurate, but with 2.5% extra CPU units). Similarly, the C-LSTM is only 0.2% more accurate, but with 12.2% extra memory units.

A more straightforward way to appreciate a better control of the resources is shown in Fig. 9. The confidence level controls the predicted resources for the different models. We progressively increment the desired confidence level and compute the success rate and the number of predicted resources. If a model correctly predicts the future timestamps with more uncertainty,

TABLE IV
CPU USAGE STATISTICS OF THE HBNN, LSTMD AND LSTM WITH LEARNED STANDARD DEVIATION

Model	C-LSTM	AVG-LSTM	HBNN	LSTMD	C-LSTM	AVG-LSTM	HBNN	LSTMD	C-LSTM	AVG-LSTM	HBNN	LSTMD
Target QoS	95%				97%				99%			
SR	94.3	94.76	**94.87**	94.56	96.07	96.82	**96.83**	96.38	98.34	**98.71**	98.63	98.22
OP	**152.9**	158.5	155.0	158.4	177.5	184.8	**176.8**	180.2	245.4	252.8	**218.8**	222.2
UP	3.7	3.7	**2.9**	3.2	2.6	2.5	**1.9**	2.2	1.2	1.1	**0.9**	1.1
TPR	**1239.3**	1244.9	1242.3	1245.4	**1265.0**	1272.4	**1265.0**	1268.2	1334.4	1341.8	**1308.0**	1311.2

TABLE V
MEMORY USAGE STATISTICS OF THE HBNN, LSTMD AND LSTM WITH LEARNED STANDARD DEVIATION

Model	C-LSTM	AVG-LSTM	HBNN	LSTMD	C-LSTM	AVG-LSTM	HBNN	LSTMD	C-LSTM	AVG-LSTM	HBNN	LSTMD
Target QoS	95%				97%				99%			
SR	95.78	98.69	**95.53**	93.9	97.38	99.47	**96.93**	95.65	**99.23**	99.89	98.55	97.75
OP	258.8	259.7	175.6	**141.2**	310.8	311.9	199.3	**159.9**	398.4	403.5	244.8	**196.0**
UP	2.5	**0.8**	2.4	3.4	1.4	**0.3**	1.6	2.4	0.5	**0.0**	0.8	1.3
TPR	1113.8	1116.4	1030.7	**995.3**	1166.9	1169.1	1055.2	**1015.0**	1255.4	1261.0	1101.6	**1052.3**

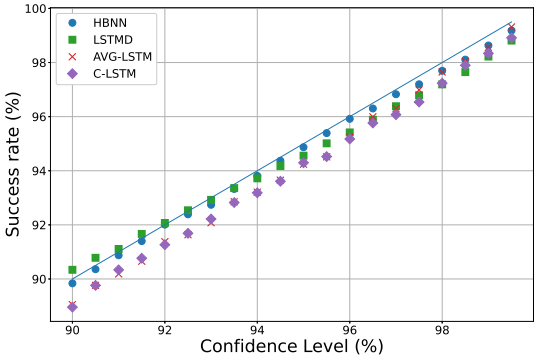

Fig. 7. Success rate vs confidence level for CPU demand

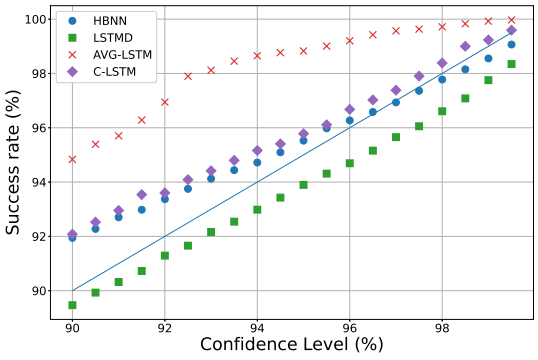

Fig. 8. Success rate vs confidence level for the memory demand

it should be able to achieve better success rates with fewer resources. The plot shows that the HBNN model is above the baselines, meaning that with the same number of *TPR* the HBNN is capable of satisfying a higher number of requests throughout the analysis, showing the potential benefits of the use of a Bayesian neural model instead of a standard one.

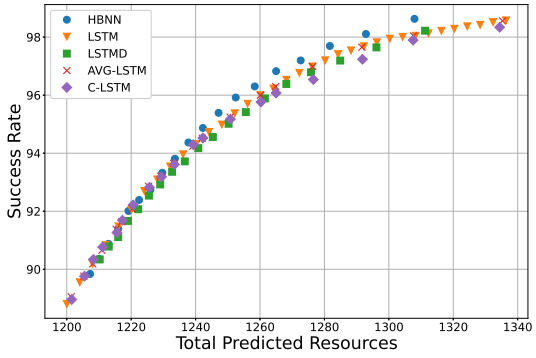

Fig. 9. Success rate vs total predicted resources

E. Runtime Performance Analysis

One of the most important aspects of the workload prediction is related to its applicability in real-world scenarios where the forecasting model is used to predict the future demand and the resource manager will use the forecasts to allocate the resources. We decide to measure the runtime performance of the DL models and compare them in terms of training time, fine-tuning time and inference time. The training phase refers to training the model from scratch, while the fine-tuning phase aims to refresh the weights of the networks based on newly

available data. We measure the training time for different sizes of the training set, corresponding to the 20%, 40%, 60% and 80% of the whole datasets, respectively. We measure the fine-tuning time where the fine-tuning using newly available data is performed every 6, 12, 18 and 24 timestamps, corresponding to a time period of 30, 60, 90 and 120 minutes, respectively. Finally, we measure the inference time i.e. the time necessary to compute a new forecast given a trained model and the history. Table VI summarises the time measurement with these configurations. We can see that the three models do not differ much for the fine-tuning and inference time, while for the training the LSTMD is the fastest model to be trained, followed by LSTM and HBNN. In particular, the time to train the HBNN increases proportionally to the training set size, while for LSTM and LSTMD the time is almost independent of the training size. All three models are suitable for real-world deployment, especially because the training phase is applied only when there are big changes in the workload and it can be done overnight when the workload is generally lower.

TABLE VI
RUNTIME PERFORMANCE ANALYSIS

		LSTM	HBNN	LSTMD
Training Time [s]	20%	76	57	**28**
	40%	66	116	**45**
	60%	73	156	**67**
	80%	71	204	**65**
Fine-Tuning Time [s]	6 (30 mins)	**3.3**	5.0	7.5
	12 (60 mins)	5.2	**4.3**	8.6
	18 (90 mins)	6.2	**4.3**	5.8
	24 (120 mins)	**4.5**	**4.5**	4.6
Inference Time [s]	1 sample	**0.0003**	0.0005	0.0005

V. CONCLUSION

Precise workload prediction is essential in cloud computing management. The resource scheduler performances are strongly affected by the quality of the forecasted demand. In this paper, we design DL models to capture the uncertainty of the data and predict the distribution of the future demand for the latest Google Cloud traces. We analysed its performances in an extensive computational study and compared them to widely used statistical methods and an LSTM-based model. The proposed models have results as good as the DL baseline in terms of the point estimate. We then tested the model to compute a UB of the upcoming demand that should hold a given probability; this information is helpful for a cloud scheduler to meet the desired QoS parameters. In this experiment, the advantage of outputting the uncertainty of the prediction becomes apparent. The HBNN committed fewer errors (leading to a better QoS) when predicting the same amount of resources needed. The difference is that it reduces the over-predicted resources by varying the standard deviation output to fit the uncertainty of the time series. We then extended the standard point prediction approach to satisfy a target QoS. We show that the results of the HBNN are closer to the desired value. Finally, we measure the runtime performance of the models for practical utilisation of the predictive models in real-world applications. The computational effort required by the models to predict the demand and update their weights is compatible with a practical deployment. This paper showed some potential benefits of including uncertainty and predicting a distribution compared to point estimate methods on cloud workload prediction with a trade-off analysis regarding the accuracy, resource prediction efficiency and runtime performance. We believe that modelling the uncertainty is more beneficial in scheduler decision-making than predicting just the mean workload and should become a standard approach in the future.

In future work, we will focus on improving the HBNN model and applying it to different cloud Computing datasets. Moreover, we would like to move to a multi-step ahead prediction to model the correlation between consecutive outputs of the network, investigating other DL architectures and Bayesian inference approaches. The final aim is to use the prediction result in resource allocation or scheduling problems such as pre-configuring the VMs of a cloud computing system in advance by predicting the amount of CPU and memory at a task level.

ACKNOWLEDGMENT

This publication has emanated from research supported in part by Science Foundation Ireland under Grant Nos. 18/CRT/6223, 16/RC/3918 and 12/RC/2289-P2. which are co-funded under the European Regional Development Fund; by TAILOR (GA No. 952215), a project funded by EU Horizon 2020 research and innovation programme, and by the Google Cloud Research Credits program with the award GCP203677602. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] B. Marr, *Tech Trends in Practice: The 25 Technologies that are Driving the 4th Industrial Revolution*. John Wiley & Sons, 2020.

[2] Z. R. Alashhab, M. Anbar, M. M. Singh, Y.-B. Leau, Z. A. Al-Sai, and S. A. Alhayja'a, "Impact of coronavirus pandemic crisis on technologies and cloud computing applications," *Journal of Electronic Science and Technology*, vol. 19, no. 1, p. 100059, 2021.

[3] M. Jelassi, C. Ghazel, and L. A. Saïdane, "A survey on quality of service in cloud computing," in *2017 3rd International Conference on Frontiers of Signal Processing (ICFSP)*. IEEE, 2017, pp. 63–67.

[4] S. Mireslami, L. Rakai, M. Wang, and B. H. Far, "Dynamic cloud resource allocation considering demand uncertainty," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 981–994, 2019.

[5] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the Fifteenth*

European Conference on Computer Systems, 2020, pp. 1–14.

[6] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Scheduling-based power capping in high performance computing systems," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 1–13, 2018.

[7] C. Vazquez, R. Krishnan, and E. John, "Time series forecasting of cloud data center workloads for dynamic resource provisioning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, pp. 87–110, 2015.

[8] T. Wang, S. Ferlin, and M. Chiesa, "Predicting CPU usage for proactive autoscaling," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, 2021, pp. 31–38.

[9] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE transactions on cloud computing*, vol. 3, no. 4, pp. 449–458, 2014.

[10] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a Google compute cloud with a Bayesian model," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–11.

[11] R. Hu, J. Jiang, G. Liu, and L. Wang, "Efficient resources provisioning based on load forecasting in cloud," *The Scientific World Journal*, vol. 2014, 2014.

[12] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *2020 29th international conference on computer communications and networks (ICCCN)*. IEEE, 2020, pp. 1–9.

[13] D. Janardhanan and E. Barrett, "CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 55–60.

[14] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6554–6568, 2018.

[15] M. Duggan, K. Mason, J. Duggan, E. Howley, and E. Barrett, "Predicting host CPU utilization in cloud computing using recurrent neural networks," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2017, pp. 67–72.

[16] D. Minarolli and B. Freisleben, "Cross-correlation prediction of resource demand for virtual machine resource allocation in clouds," in *2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks*. IEEE, 2014, pp. 119–124.

[17] D. Minarolli, A. Mazrekaj, and B. Freisleben, "Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1–18, 2017.

[18] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[19] J. Wilkes, "Google cluster-usage traces v3," Google Inc., Mountain View, CA, USA, Technical Report, Apr. 2020, posted at https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

[20] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[21] M. Barati and S. Sharifian, "A hybrid heuristic-based tuned support vector regression model for cloud load prediction," *The Journal of Supercomputing*, vol. 71, no. 11, pp. 4235–4259, 2015.

[22] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload prediction for cloud cluster using a recurrent neural network," in *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*. IEEE, 2016, pp. 104–109.

[23] J. Kumar and A. K. Singh, "Cloud resource demand prediction using differential evolution based learning," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*. IEEE, 2019, pp. 1–5.

[24] J. Kumar and A. Singh, "An efficient machine learning approach for virtual machine resource demand prediction," *International Journal of Advanced Science and Technology*, vol. 123, pp. 21–30, 2019.

[25] W. Zhang, P. Duan, L. T. Yang, F. Xia, Z. Li, Q. Lu, W. Gong, and S. Yang, "Resource requests prediction in the cloud computing environment with a deep belief network," *Software: Practice and Experience*, vol. 47, no. 3, pp. 473–488, 2017.

[26] B. Liu, Y. Lin, and Y. Chen, "Quantitative workload analysis and prediction using google cluster traces," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2016, pp. 935–940.

[27] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," *Concurrency and computation: practice and experience*, vol. 26, no. 12, pp. 2053–2078, 2014.

[28] S. Zhou, J. Li, K. Zhang, M. Wen, and Q. Guan, "An accurate ensemble forecasting approach for highly dynamic cloud workload with vmd and r-transformer," *IEEE Access*, vol. 8, pp. 115 992–116 003, 2020.

[29] Y. Liu, H. Qin, Z. Zhang, S. Pei, Z. Jiang, Z. Feng, and J. Zhou, "Probabilistic spatiotemporal wind speed forecasting based on a variational bayesian deep learning model," *Applied Energy*, vol. 260, p. 114259, 2020.

[30] M. S. Khan and P. Coulibaly, "Bayesian neural network for rainfall-runoff modeling," *Water Resources Research*, vol. 42, no. 7, 2006.

[31] Y. Xie, D. Lord, and Y. Zhang, "Predicting motor vehicle collisions using bayesian neural network models: An empirical analysis," *Accident Analysis & Prevention*, vol. 39, no. 5, pp. 922–933, 2007.

[32] J. Lampinen and A. Vehtari, "Bayesian approach for neural networks—review and case studies," *Neural networks*, vol. 14, no. 3, pp. 257–274, 2001.
[33] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of econometrics*, vol. 31, no. 3, pp. 307–327, 1986.
[34] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[36] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1613–1622.
[37] J. M. Joyce, "Kullback-leibler divergence." *International encyclopedia of statistical science*, vol. 720, p. 722, 2011.
[38] J. Zeng, A. Lesnikowski, and J. M. Alvarez, "The relevance of bayesian layer positioning to model uncertainty in deep bayesian active learning," *arXiv preprint arXiv:1811.12535*, 2018.
[39] N. Brosse, C. Riquelme, A. Martin, S. Gelly, and É. Moulines, "On last-layer algorithms for classification: Decoupling representation from uncertainty estimation," *arXiv preprint arXiv:2001.08049*, 2020.
[40] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.
[41] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *2010 11th IEEE/ACM International Conference on Grid Computing*. IEEE, 2010, pp. 41–48.
[42] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011, pp. 111–118.
[43] D. Koutsandreas, E. Spiliotis, F. Petropoulos, and V. Assimakopoulos, "On the selection of forecasting accuracy measures," *Journal of the Operational Research Society*, pp. 1–18, 2021.
[44] G. Shmueli, P. C. Bruce, I. Yahav, N. R. Patel, and K. C. Lichtendahl Jr, *Data mining for business analytics: concepts, techniques, and applications in R*. John Wiley & Sons, 2017.
[45] F. X. Diebold and R. S. Mariano, "Comparing predictive accuracy," *Journal of Business & economic statistics*, vol. 20, no. 1, pp. 134–144, 2002.
[46] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, no. 1, pp. 1–17, 2014.