Cluster Computing (2025)28:258
https://doi.org/10.1007/s10586-024-04933-2

Forecasting workload in cloud computing: towards uncertainty-aware predictions and transfer learning

Andrea Rossi[1,2] · Andrea Visentin[1,2,3] · Diego Carraro[2,3] · Steven Prestwich[1,2,3] · Kenneth N. Brown[1,2,3]

Received: 18 August 2024 / Revised: 12 November 2024 / Accepted: 17 November 2024

Abstract
Accurately forecasting workload demand in cloud computing environments is essential for optimizing resource allocation, minimizing costs, and ensuring reliable service quality. As cloud computing scales to meet the needs of diverse applications - from AI and machine learning to data-intensive analytics - predictive models play a critical role in dynamically managing multiple resources. Traditional models provide limited guidance for decision-makers since they are typically univariate models, ignoring the prediction of the interplay between multiple resources, and do not account for the uncertainty of their predictions, preventing resource management from acting promptly according to such uncertainty to ensure specific target service level requirements. To address these limitations, we introduce univariate and bivariate Bayesian deep learning models that predict future workload demand of one and multiple resources respectively, while quantifying the uncertainty of their predictions. In particular, our approach leverages Hybrid Bayesian Neural Networks and probabilistic Long Short-Term Memory models, enhanced with architecture modifications to handle complex, multivariate cloud workload patterns. Moreover, we investigate fine-tuning-based transfer learning methods to enhance their adaptability in real-world cloud scenarios where new data centres with different workload characteristics operate. We validate our models on extensive datasets from Google and Alibaba cloud clusters. Results show that modelling the uncertainty of predictions positively impacts performance, especially on service level metrics, because uncertainty quantification can be tailored to desired target service levels that are critical in cloud applications. Moreover, transfer learning benefits performance in scenarios where models are built on data from the same provider.

Keywords Bayesian neural networks · Cloud computing · Workload prediction · Uncertainty · Deep learning · Transfer learning

1 Introduction

Cloud computing services have gained enormous popularity in the last few years because they reduce the cost of business and increase the productivity of industries [1]. It is expected that cloud computing demand will rise steadily in the future, especially due to the recent advances in Artificial Intelligence (AI) and Big Data, whose workloads require massive computational resources [2]. To maintain a useful service while respecting energy and sustainability constraints, cloud providers must allocate and manage computing resources such as processing power, memory and storage to meet demand while minimising over-provisioning. Given the time and energy needed to wake-up or maintain resources, efficient operation requires accurate predictions of future demand.

✉ Andrea Rossi
a.rossi@cs.ucc.ie

Andrea Visentin
a.visentin@ucc.ie

Diego Carraro
diego.carraro@insight-centre.org

Steven Prestwich
s.prestwich@cs.ucc.ie

Kenneth N. Brown
k.brown@cs.ucc.ie

1 Centre for Research Training in Artificial Intelligence, Cork, Ireland

2 School of Computer Science & IT, University College Cork, Cork, Ireland

3 Insight Centre for Data Analytics, University College Cork, Cork, Ireland

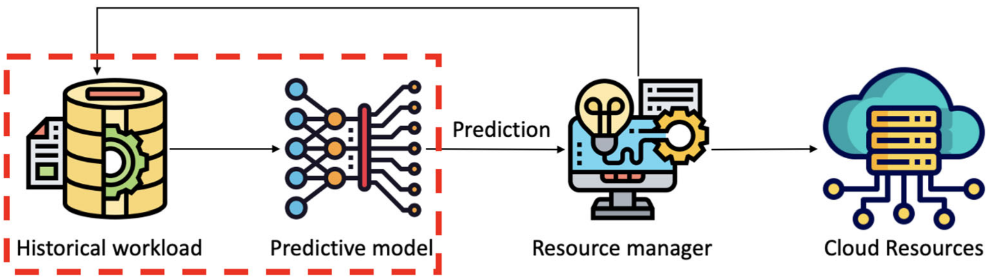

Fig. 1 The workload prediction scheme in cloud computing. Our work focuses on building and evaluating future workload predictions (i.e. red box)

Typically, a workload prediction scheme develops as a loop (see Fig. 1). Its main component is predictive model, trained on historical workload data, that provides future workload forecast. Based on such predictions, a resource manager (human experts or or automated algorithms) can, for example, implement resource allocation policies and elastic scaling or perform workload balancing across resources. Over time, the workload history evolves and feeds back as newly available data to update the predictive model.

Workload forecasting is a challenging task due to the dynamic of workload patterns, which exhibit spikes and irregularities [3]. Researchers have tackled the problem as a time series analysis task and proposed statistical methods such as ARIMA, Machine Learning (ML) approaches including linear regression and decision trees, and recently Deep Learning (DL) architectures such as Long Short-Term Memory (LSTM). The majority of these models provide point forecast predictions of the future workload, but they do not consider the uncertainty of such predictions. However, predicting a probability distribution instead allows one to account for the uncertainty of the model due to the noise in the data (i.e. aleatory uncertainty) and due to the lack of representativeness of the training data (i.e. epistemic uncertainty). Moreover, it can be leveraged by cloud managers to make more informed decisions when optimizing resource allocation, approaching the problem based on the degree of a prediction's confidence. For example, when the uncertainty is high, the safety margin resources allocated are increased compared to when the models have more confidence in the predictions. The literature has proposed uncertainty-aware solutions for workload prediction in cloud computing systems [4–6], but the domain is still under-investigated and focuses mostly on modelling only the aleatoric uncertainty.

The literature does not assess whether the proposed predictors *generalize* well to out-of-distribution data. In ML, this problem is generally tackled with Transfer Learning (TL) [7], where a model is trained on one or several source domain(s) (i.e. in-distribution datasets) and is used or adapted to predict a different but related target domain (i.e. out-of-distribution dataset).[1] In the cloud computing scenario, transfer learning is useful, for example, when a cloud provider establishes a new data centre in a previously untapped region, and so limited historical workload data is available to train a workload predictor for that region. In this case, one possibility to obtain a workload forecast for the new region is to leverage a predictor trained on data from other regions, i.e. one that can transfer its predictive capabilities to the new region.

As data centres grow in complexity and serve increasingly diverse workloads, accurate workload forecasting has become critical for optimizing performance and resource allocation. In the past, focusing on a few metrics like CPU usage was sufficient, but modern data centres must consider a broader range of variables, including GPU usage, memory demand, disk accesses, and storage utilization. When forecasting multiple resources, the system can predict their demand independently (univariate approach) or jointly (multivariate approach).

In this paper, we present a set of transfer learning scenarios tailored for cloud workload prediction. We then adapt uncertainty-aware deep learning forecasting models to jointly predict computational resources and memory required. We build such investigation on our previous work [4], where we proposed a Bayesian Neural Network (i.e. HBNN) and a Probabilistic LSTM (i.e. LSTMD). The former architecture captures the epistemic and aleatoric uncertainty of the prediction, while the latter captures only the aleatoric uncertainty. The contributions and the findings of our work are summarised as follows:

- We investigated and designed adapted versions of HBNN and LSTMD architectures (introduced in [4]) which provide an effective balance between additional representation power to fit the increasingly complex

[1] In the literature, this is sometimes referred to as Domain Adaptation.

bivariate prediction and the lack of generalisation caused by overfitting.
- We propose a set of scenarios that define which data is available for training and fine-tuning of forecasting models. These scenarios simulate real-world situations, such as the opening of a new data centre from an existing provider.
- We preprocess twelve workload traces from Google Cloud 2011 [8] and 2019 [9] and Alibaba Cluster 2018 [10] and 2020 [11], and we open source the resulting datasets.[2] We make this procedure detailed and explicit for reproducibility as, to the best of our knowledge, the current literature is inconsistent and lacks detail.
- In an extensive experimental analysis, we compare how the predictive models behave in real-world traces under the different scenarios introduced herein. The first part evaluates the uncertainty-aware models' univariate and bivariate prediction quality in terms of a wide range of accuracy and service level metrics. Furthermore, we assess their runtime performance. The second phase of the experiments analyses the transfer learning capabilities of the bivariate HBNN trained on multiple domains (datasets) and tested on a different combination of source and target domains (datasets).

The remainder of the paper is structured as follows. Section 2 survey the related work. We describe the methodology presented in this work in Sect. 3. Section 4 reports the results of the experiments, and Sect. 5 outlines the conclusions and future works.

2 Related work

In cloud computing management, workload prediction plays an important role that has been broadly studied over the past twenty years. In this section, we outline relevant literature on statistical, ML and DL forecasting approaches and the role of uncertainty and transfer learning.

2.1 Workload forecasting

Researchers and practitioners have tackled forecasting workloads with statistics-based at first, then traditional ML-based, and finally transitioning to DL-based, considered state-of-the-art in most modern cloud applications.

Statistics-based approaches employ mathematical models to identify repetitive patterns and trends in the data, but they often struggle to capture the high complexity and variability of cloud workloads accurately. ARIMA-based [12] models are simple yet powerful statistical techniques that essentially combine three components: the AutoRegressive (AR) component ensures predictions are linear combinations of historical values of the target variable; the Integrated (I) component helps make the series stationary through differencing operations; and the Moving Average (MA) component leverages past forecast errors to make predictions. ARIMA models are effective at capturing mainly stationary trends and seasonality effectively. They have been used, for example, for predicting the future host workload in distributed systems [13] and for predicting HTTP requests [14]. ARIMA is now a popular baseline approach for evaluation purposes, e.g. [15–17], as more advanced approaches and prediction tasks arose in recent years. Other statistical techniques have been applied to workload forecasting. For example, exponential smoothing [18] and GARCH [19] are applied as baselines to forecast CPU/memory demand (e.g. [20–23] and [24], respectively) for autoscaling applications. The former models leverage an autoregressive component to capture simple patterns and trends, while the latter leverage autoregressive and moving average components and are designed to model and forecast time series where variance changes over time, i.e. heteroskedasticity.

ML-based models have been shown to outperform statistical methods and improve prediction accuracy [25]. Indeed, they offer numerous benefits over traditional statistical methods because they can handle complex and diverse data more efficiently and successfully capture nonlinear patterns in the data. On the other hand, they require manual feature extraction and hyperparameter tuning, which is sometimes expensive. Examples in the literature include logistic regression [26], random forest [27], K-nearest neighbours regression [28], support vector regression [25, 29] and ensemble methods [30].

Although training DL-based architectures is expensive in terms of data availability and computational resources, such models are now considered state-of-the-art because they outperform both statistical and ML-based alternatives in most cloud applications. In particular, sequential models have become a popular choice for their inner time-sensitive nature, effective at capturing long-term dependencies and dynamicity of workload data. One such model is the Recurrent Neural Network (e.g. used in [31, 32]), and another is its advanced versions, i.e. Long Short Term Memory (e.g. used in [23, 33]) and Gated Recurrent Unit (e.g. used in [34]). In this paper, we investigate an LSTM-based model to perform workload forecasting (see section 3.3). When dealing with cloud workload time series data, it is common to see intricate patterns over extended periods. To capture these long-range dependencies, attention mechanisms [35] and transformers are often used.

[2] https://github.com/andreareds/TowardsUncertaintyAwareWorkloadPrediction

Attention mechanisms help models focus on the relevant parts of the input sequence, while transformers enable parallel data processing and eliminate the need for recurrence. These techniques have revolutionized sequence modelling and applied to the cloud workload forecasting problem [36–38]. One type of transformer, the Informer model has been evaluated in predicting CPU usage in cloud data centres, finding that the Informer model, which considers both long-range dependencies and sequence ordering, consistently outperforms vanilla transformers [39]. However, transformers require a huge amount of data that, in many cases, are unavailable or expensive to collect [40].

Finally, researchers have explored combining two or more models above into ensembles to enhance forecasting accuracy and model robustness. Examples of this kind are [41–43]. However, in this paper, we focus on comparing individual architectures and leave the additional investigation of combining such architectures into ensemble models as future work.

2.2 Uncertainty-aware forecasting

The models discussed earlier provide a point prediction of future workload with no indication of confidence in such prediction. This can be problematic because, due to the lack of explainability of the models, these predictions may be either underconfident or overconfident, and we cannot distinguish between them. In critical use cases such as healthcare or autonomous vehicles, incorrect, overconfident predictions can be dangerous. Measuring uncertainty in time series forecasting can be done through various methods [44]. One of them is ensemble forecasting, which involves multiple models, each with slightly varying initial conditions, to produce a range of possible outcomes and their probabilities. This method is commonly used in weather forecasting [45]. Another approach is probabilistic forecasting, which estimates the probability of various future outcomes and the complete set of probabilities represents a probability forecast [44]. These uncertainty quantification methods can also be divided into parametric and non-parametric methods. The former does not make strong assumptions about the underlying probability distributions or functional forms of the data. The latter assumes that the data follow a specific distribution, often a normal distribution. In the literature, there is limited investigation of approaches capable of capturing and quantifying the inherent uncertainties in cloud workload data. In ML, uncertainty is often categorised as either epistemic, i.e. due to insufficient training data, or aleatoric, i.e. caused by stochasticity of observations due to noise and randomness. The design of the approach influences the type of uncertainty that can be modelled [46]. In the workload prediction task, early attempts have been made to capture such uncertainties through confidence intervals, e.g. [47], and non-parametric methods such as kernel estimation method, e.g. [6]. Recently, a popular non-parametric method called Quantile Regression has been applied to workload prediction to model aleatoric uncertainty [48] through quantile distributions. However, interpreting uncertainty estimates in non-parametric models can be more difficult, as they combine multiple sources of uncertainty [49, 50]. In particular, none of these methods aims at quantifying epistemic and aleatoric uncertainties together, and none of them provides bivariate forecasting. To the best of our knowledge, in our previous work [4], we are the first to explore state-of-the-art DL models designed to model both uncertainties under a Bayesian framework. In this paper, we extend this work and provide a more comprehensive investigation of the impact of uncertainty in workload prediction.

2.3 Transfer learning

Transfer Learning (TL) [7] is a broad machine learning paradigm where a model is pre-trained on a source task or domain and then adapted or fine-tuned to a related target task or domain. Transferring knowledge across tasks is generally known as *task transfer*. Transferring knowledge across domains is generally known as *domain adaptation*, *cross-domain* or *domain generalization* [51, 52], very closely related to each other, so that sometimes the literature is inconsistent in its terminology and does not distinguish between them. In this paper, our work is about transferring knowledge across domains and overlaps the three areas so that we use the general transfer learning umbrella terminology. In particular, we investigate when pre-training is performed on *multiple* source domains to learn domain-invariant representations that can be exploited on a different target domain. Generalization can be achieved by fine-tuning (FT) on the target dataset or in a zero-shot fashion.

TL has been widely applied in the context of time series classification [53] and forecasting [54], showing the effectiveness of this approach, including QoS aspects [55] for workload prediction [56]. TL has also been applied in other cloud computing contexts, such as runtime performance prediction [57, 58] and autoscaling [59]. To the best of our knowledge, we are the first to explore transfer learning across domains for workload predictions.

3 Methodology

This section describes the complete framework of the prediction approaches considered in this paper. The task is time series forecasting, in which we try to predict the upcoming resource demand for a given data centre (or cluster, we use them interchangeably here) managed by a provider. We consider the resource traces as the datasets.

Section 3.1 describes the different forecasting approaches for univariate and bivariate predictions. In Sect. 3.2, we present the different transfer learning scenarios considered. The DL forecasting models used are detailed in Sect. 3.3. Section 3.4 presents the training and FT approaches, and Sect. 3.5 describes the dataset used. Finally, Sect. 3.6 details the metrics used in the experimental comparison.

3.1 Univariate/bivariate prediction

Univariate prediction focuses on forecasting a single variable, such as processing unit usage, over time. This approach is generally simpler, requiring fewer computational resources and often less complex models. Additionally, univariate models can be highly accurate for resources that have predictable patterns, such as CPU usage under certain workloads. However, the simplicity of univariate prediction comes at a cost: it fails to capture the interdependencies between different resources. In data centres, resource demands are often interlinked; an increase in CPU usage, for instance, can lead to higher memory or disk access demands. Without accounting for these correlations, univariate predictions can miss critical insights, leading to inaccurate forecasts and inefficient resource allocation.

Bivariate or multivariate predictions, on the other hand, attempt to model two or more variables together. Herein, we focus on jointly predicting processing units and memory usage. Such forecasting, for example, can yield a better understanding of workloads driven by data-intensive applications, helping prevent scenarios where processing units over-provisioning go underutilized if memory or storage are under-predicted and act as a bottleneck. In general, multivariate models provide a more holistic view of data centre workloads, allowing for better forecasts in scenarios where resource usage patterns are interconnected. The main downside of bivariate forecasting is the complexity increase. These models require larger datasets to capture the relationships between variables, and they are computationally more expensive, often requiring more sophisticated machine learning models.

3.2 Transfer learning scenarios

In cloud workload prediction, transfer learning is particularly useful when dealing with different data centres, each considered as a separate dataset with unique workload characteristics, user behaviour, and infrastructure. Rather than building a separate prediction model for each data centre from scratch, transfer learning enables models to adapt knowledge from one (source) data centre to another (target) data centre, significantly improving prediction accuracy and efficiency. In particular, we consider as the source domain all the datasets a machine learning model is trained on and as the target domain the cluster which future workload we need to predict.

We leverage this representative real-world scenario to frame our investigation, where we mimic different data availability conditions to apply transfer learning. In particular, we consider the following scenarios:

- **All Datasets**. In this scenario, the machine learning models are trained on all available datasets, and so the source contains data from different distributions, while the target is from just one of those distributions. This simulates a situation in which a provider is trying to forecast a data centre workload (target domain) and can train its predictors on traces collected internally and those made available by other companies (e.g. open source).
- **Same-distribution Datasets**. In this case, the source domain contains datasets from a single distribution, and the target is also from that distribution. For example, when a provider is developing a forecasting model that has access to internal historical data. The traces can also be collected from other data centres that the same company operates. These datasets may share several common characteristics, as the provider has standardized its hardware and allocation software across its facilities. This consistency can lead to more comparable workload patterns, especially if the data centres host similar services that are subject to analogous demand trends.
- **Different-distribution Datasets**. In this case, the source domain contains datasets from one distribution, but the target contains datasets from a different distribution, and thus is a transfer learning case. In this scenario, a provider operates a single cloud data centre and aims to enhance its forecasting capabilities. To improve its predictive models, the company utilizes publicly available (e.g. open source) datasets related to similar operations from other organizations, such as workload patterns from comparable cloud services or industry benchmarks. By applying transfer learning, the company can leverage the patterns and insights learned from these external datasets, such as common usage spikes during specific times of the day.

Another important factor to consider is whether historical data is available for the specific data centre. We assess whether past demand data exists or not:

- **With Historical Data**: If time series data for resource demands (such as processing units, memory, etc.) in the cluster to be predicted is available. Since we can consider a level of continuity in the services hosted by the data centre, the patterns observed in past data are highly relevant and can enhance the accuracy of predictions. This scenario often occurs when a provider is developing a new workload prediction system, allowing the specific historical data to be used for both training and fine-tuning the models.

- **Zero-shot Learning**: When no historical data is available for the cluster, it typically means a new site has been launched or there has been significant restructuring in the operations or services offered. This situation requires zero-shot learning, where TL is essential to adapt models based on similar data from other clusters or external sources.

The experimental analysis extensively covers the impact of all these scenarios on the models' forecasting accuracy.

3.3 Forecasting models

This section describes the three forecasting models we compare in our work. A graphical representation of their architectures is depicted in Fig. 2.

3.3.1 LSTM

An LSTM-based model that does not provide uncertainty quantification is used as the baseline of our experiments. The network consists of an input layer with a size of 288, corresponding to the past 24 h of workload. The input size has been found experimentally. The input sequence is filtered by a succession of 1D convolutional (1DConv) layers (between one and three). The convolution is followed by an LSTM layer, widely adopted as effective at learning long-term dependencies and sequential patterns in time-series or sequence data [60]. Combining convolutional and LSTM layers has been proven effective in time series forecasting [61, 62]. A sequence of dense layers, whose number varies with the training set's size and the prediction type, follows the LSTM layer. We optimize the network weights through the Mean Squared Error (MSE) loss function.

3.3.2 HBNN

Similarly to our previous work [4], we employ a Bayesian Last Layer network to capture epistemic and aleatoric uncertainties. HBNN follows the same architecture of the LSTM-based described in 3.3, with the following modifications. A Bayesian layer, designed to model the epistemic uncertainty, is positioned after the first sequence of dense layers. In the Bayesian layer, each weight, instead of being deterministic, has an associated probability distribution that can be trained via variational inference.

Mathematically, let **W** represent the weights in the Bayesian layer. Instead of having fixed weights, we model

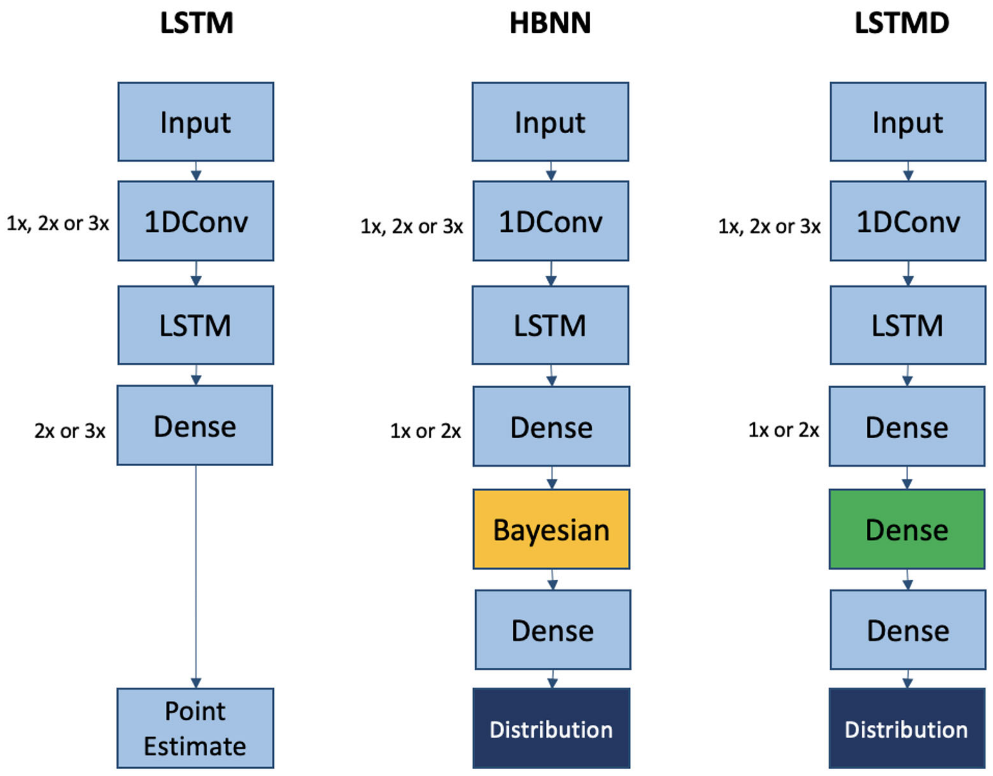

Fig. 2 Network architectures of the three models we compare. The presence of the Bayesian layer and the type of output layer characterizes the three architectures. We use labels 1x, 2x, 3x besides a specific layer to indicate the presence of a sequence of 1, 2 and 3 occurrences of this layer

$\mathbf{W}$ as a distribution $q(\mathbf{W}|\theta)$, where θ are the parameters of the distribution. During training, we aim to approximate the true posterior distribution $p(\mathbf{W}|\mathcal{D})$ given the data $\mathcal{D}$ by minimizing the Kullback–Leibler (KL) divergence between $q(\mathbf{W}|\theta)$ and $p(\mathbf{W}|\mathcal{D})$. This is achieved through variational inference, where we maximize the evidence lower bound (ELBO):

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{W}|\theta)}[\log p(\mathcal{D}|\mathbf{W})] - \text{KL}(q(\mathbf{W}|\theta)||p(\mathbf{W})) \quad (1)$$

The Bayesian layer captures epistemic uncertainty, which arises from the model's lack of knowledge. By modelling the weights as distributions, we account for the uncertainty in the model parameters. This means that when the input is propagated through this layer, the distributions of the neurons are sampled, making the output probabilistic. This layer is followed by a dense layer that outputs the parameters (mean and standard deviation) of one or two Gaussian distributions for univariate and bivariate predictions, respectively. This layer has two neurons in the univariate case and four neurons in the bivariate case. The output Gaussian distributions capture the *aleatoric uncertainty*, which is the inherent noise in the data.

The training of the HBNN is performed using the Bayes by Backpropagation algorithm. This algorithm modifies the standard backpropagation to update the parameters of the weight distributions. Specifically, for each weight w_i in the Bayesian layer, we maintain a mean μ_i and a standard deviation σ_i. During each forward pass, we sample a weight w_i from the distribution $q(w_i|\mu_i, \sigma_i)$. The gradients of the loss w.r.t. μ_i and σ_i are then computed using the reparameterization trick, which allows us to backpropagate through the sampling process. The reparameterization trick expresses the sampled weight as:

$$w_i = \mu_i + \sigma_i \cdot \epsilon_i \quad (2)$$

where $\epsilon_i \sim \mathcal{N}(0, 1)$ is a standard normal random variable. The gradients are then used to update μ_i and σ_i via gradient descent.

Compared to [4], in the experiments of this paper, we extended the network architecture with additional layers to account for the greater complexity of training the model on more datasets and acquiring transfer learning capabilities. To allow higher flexibility across different problems (univariate/bivariate) and the size of the training dataset, we model the architecture by dynamically setting a wider set of parameters (e.g. number of neurons, activation functions, etc.) in an extensive hyperparameters search. We optimize the weights through the negative log-likelihood loss function:

$$\mathcal{L} = -\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{W}) \quad (3)$$

where N is the number of data points, y_i is the target variable, and $\mathbf{x}_i$ is the input feature vector.

3.3.3 LSTM distributional

The LSTM Distributional (LSTMD) network [4] is similar to the HBNN in terms of its overall architecture, but it differs in a few key aspects. The primary difference is that the LSTMD network does not include a Bayesian layer. Instead, it uses a traditional dense layer, which means that it does not capture epistemic uncertainty.

In the LSTMD network, the weights in the dense layer are deterministic, meaning they are fixed values learned during training. This contrasts with the Bayesian layer in the HBNN, where the weights are modelled as probability distributions to account for uncertainty in the model parameters. As a result, the LSTMD network only captures aleatoric uncertainty, which is the inherent noise in the data, through its output distribution. Similarly to the approach of the HBNN, we extended the network architecture with additional layers and evaluated a wider set of hyperparameters. We use the same loss function for training (negative log-likelihood).

3.4 Prediction settings and fine-tuning

For each dataset, the first 80% data points are used as the training set (20% of which is used as the validation set), and the remaining 20% as the test set. When training with multiple datasets, the training time frames of all datasets are aligned to ensure that one trace cannot exploit information of a particular timestamp in another trace.

We tune the following hyperparameters with the Talos library on the validation set: the number of neurons for each layer, the batch size, the activation functions, the learning rate, the number of kernels in the 1DConv layer and the optimizer with its momentum and decay coefficients. We train the models with their best hyperparameters ten times for each scenario using various random seeds as initialisation to assess the optimisation algorithm's convergence, and we use early stopping on the validation set to avoid overfitting. We forecast the 5-minute interval of demand 10 min in the future, where 10 min is sufficient for most real-world applications [16, 63], e.g. resource allocation, vertical scaling etc. All three architectures are implemented in Keras; for the HBNN and LSTMD we also used TensorFlow probability. We share a GitHub repository that contains further information on the models' architecture and the search space for the hyperparameters.

To enhance the generalization capabilities of the models, we employ a fine-tuning operation. This method involves adjusting the pre-trained model parameters using a small subset of data from the target domain. By doing so, the model can better adapt to new domains, whether they come from the same provider or different providers or even different clusters from the same provider. The fine-tuning process is crucial as it allows the model to retain its learned knowledge while improving its performance on the new domain-specific data.

When deploying the fine-tuning approach specifically, we use the first 80% of the dataset for fine-tuning, while the remaining 20% is used for testing to avoid overfitting, as similarly done in the first part of the experiments. The hyperparameters are set during the training phase prior to the fine-tuning phase, as described at the beginning of this section.

3.5 Datasets

The datasets we use in our experiments are extracted from Google Cloud Trace 2011 and 2019, Alibaba Cluster Trace 2018 and Alibaba Cluster Trace 2020. In the following, we describe such data and how we preprocess it. The resulting twelve datasets can be downloaded from the GitHub repository we share.

3.5.1 Google cloud trace 2011 and 2019

The Google Cloud Trace 2011 (GC11) [8] and 2019 (GC19) [9] are publicly available datasets collected from the Google Cloud Platform and contain details about the resource utilisation of some cluster cells. In particular, Google Cloud Trace 2011 is composed of 29 days of resource usage collected in May 2011 from 12,500 machines in a single cluster cell; Google Cloud Trace 2019 is composed of data of 29 days from eight different cluster cells accounting for around 10,000 machines each, distributed across different geographical regions around the world. Similarly to other works in literature [41, 64, 65], we preprocess Trace 2011 and Trace 2019 with the following procedure, and for the Trace 2019, we also employ Google BigQuery due to the dataset size, i.e. about 2.4TiB compressed. For each cluster, we create a time series dataset that includes the average CPU and average memory usage for all the machines with a 5-minute interval. Missing records are neglected for simplicity. For the program tasks that run only partially in a 5-minute window, we multiply the average resource by a weight corresponding to the fraction of the window in which the task is in execution. Data is finally scaled in the range [0, 1] using a MinMax scaling strategy for speeding up the convergence of the training. We obtain nine datasets of roughly 8k data points per resource (CPU and CPU memory usage).

3.5.2 Alibaba cluster trace 2018 and 2020

The cluster trace 2018 (AC18) includes the workload history for CPU and memory utilisation of about 4,000 machines in eight consecutive days. The 2020 version (AC20) is a longer trace of around two months and 1,800 machines that contain over 6,500 GPUs. From this trace, we craft two datasets, one related to the CPU and memory usage and one for the GPU and GPU memory usage. We preprocess these traces with the same procedure we used for the Google traces. We obtain one dataset of roughly 2k data points per resource (CPU and CPU memory usage) for Alibaba 2018; and two datasets of roughly 14k data points per resource (GPU, CPU and their memory usage) for Alibaba 2020.

In Fig. 3, we plot one representative dataset distribution per cluster to show they are all Gaussian distributions but of different parameters.

3.6 Metrics

The forecasts are evaluated in terms of their accuracy and their impact on service level metrics.

All prediction models are compared based on their point estimate accuracy. The metrics employed for this evaluation are Mean Squared Error (MSE) and Mean Absolute Error (MAE), widely used to assess time series forecasting approaches. In the case of HBNN and LSTMD models, the error is computed w.r.t. to the mean of the predicted distribution, while for the LSTM, the error is calculated based on the point prediction. We chose to use MSE and MAE metrics in our experiments to evaluate the accuracy of our predictions and compare the LSTM model to the LSTMD and HBNN models. These metrics provide a straightforward way to quantify the difference between the predicted

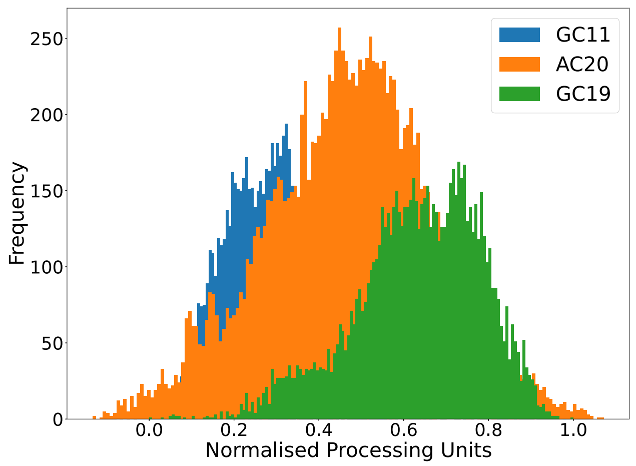

Fig. 3 Processing units distributions from different clusters. We plotted only one dataset per cluster, and we excluded AC18 to make the plot more readable

and actual values. Additionally, we compute the Quantile Loss (QL) for the probabilistic models (HBNN and LSTMD). The inclusion of QL is crucial as it allows us to evaluate the performance of these models at higher confidence levels, which target the high quality of service levels typically required in cloud environments. The quantile loss is defined mathematically as:

$$QL_\alpha(y,\hat{y}) = \begin{cases} \alpha(y-\hat{y}) & \text{if } y \geq \hat{y} \\ (\alpha-1)(y-\hat{y}) & \text{if } y < \hat{y} \end{cases} \quad (4)$$

where y is the actual value, $\hat{y}$ is the predicted value and α is the quantile level.

We further assess the models using the service level metrics defined in [4], i.e. where predictions are tailored to a specific target service level. In particular, we focus on the *Success Rate* (SR) and *Total Predicted Resources* (TPR). SR is the percentage of future demand within the confidence interval, i.e. whether the target confidence level is met. For example, when a 95% service level is set, we would like 95% of the requests to be within the upper bound of the prediction. TPR quantifies the overall amount of predicted resources that would be given as input to the resource manager, i.e. the sum of all the upper bounds of the predictions, in percentage w.r.t. the real demand.

In the case of HBNN and LSTMD, which predict a probability distribution, the final prediction values are computed w.r.t. the upper bound of the confidence interval with a target service level varying from 90% to 99.5%. For LSTM, we could infer a confidence interval from its point estimate, i.e. the output of the network plus a fixed threshold, e.g. 5%, as done in literature [5, 6]. Instead, to avoid penalising the baseline too much, we calculate such an interval dynamically with the following procedure. We check the SR value scored by HBNN (LSTMD), and we adjust the interval to achieve the same SR value. In other words, we set the desired SR value first, and we calculate the interval accordingly. An extended analysis focused on all the metrics presented in [4] is available in the appendix.

4 Experimental results

In this section, we evaluate the models in terms of point estimate accuracy, the efficiency of the predicted resources and runtime performance. In each subsection, we first assess the extension to bivariate models and those trained with multiple datasets. We then discuss the results based on the TL approach. Experiments are performed with a CPU Intel®Xeon®Gold 6240 at 2.60GHz and GPU NVIDIA Quadro RTX 8000 with 48 GB of memory where Ubuntu 20.04 is installed.

4.1 On models comparison

By predicting multiple resource metrics, data centres can better anticipate resource bottlenecks, improve load balancing, minimize energy consumption, and ensure service quality. The first part of the experiments focuses on evaluating the uncertainty-aware predictive models, where we compare univariate and bivariate predictors trained and tested on one or twelve of the datasets described in the previous section for a total of four prediction configurations. Each configuration is defined by a specific prediction task (univariate or bivariate) and specific training data (single dataset or multiple datasets). Therefore, we distinguish each model by adding a specific prefix to its name: U and B for univariate and bivariate models, respectively, and S and M when trained on a single dataset or multiple (twelve) datasets, respectively. For example, M-B-HBNN refers to an HBNN trained with multiple datasets for a bivariate prediction. Also, it is worth noting that univariate models predict just one resource at a time, and bivariate models simultaneously predict both resources (i.e. processing units and memory). The goal of this experiment is twofold: understand if the joint prediction of the two targets outperforms the univariate case and quantify how impactful the addition of different traces is. Bivariate prediction is computationally harder, but it could give better results, especially if the joint distribution can not be decomposed into 2 univariate ones.

Table 1 shows the results of processing units and memory forecast for all models' combinations, i.e. single/multiple datasets and univariate/bivariate. Also, each model's performance is the average MSE and MAE computed across the twelve datasets.

Table 1 Average MSE/MAE comparison for processing units and memory prediction. In bold, the best model overall, and in italics, the best model in their groups

Model	Processing units		Memory	
	MSE	MAE	MSE	MAE
S-U-LSTM	0.0041	0.0457	0.0042	0.0425
S-U-LSTMD	**0.0040**	*0.0455*	**0.0041**	**0.042**
S-U-HBNN	0.0047	0.0502	0.0044	0.0455
S-B-LSTM	*0.0047*	*0.0500*	*0.0054*	*0.0487*
S-B-LSTMD	0.0061	0.0556	0.0088	0.0631
S-B-HBNN	0.0299	0.1285	0.0386	0.1533
M-U-LSTM	0.0044	0.0474	0.0048	0.0447
M-U-LSTMD	*0.0041*	*0.0464*	*0.0044*	*0.0439*
M-U-HBNN	0.0047	0.0485	0.0052	0.0498
M-B-LSTM	0.0044	0.0479	0.0044	0.0438
M-B-LSTMD	0.0046	**0.0446**	0.0046	0.0446
M-B-HBNN	*0.0042*	0.0471	*0.0043*	*0.0436*

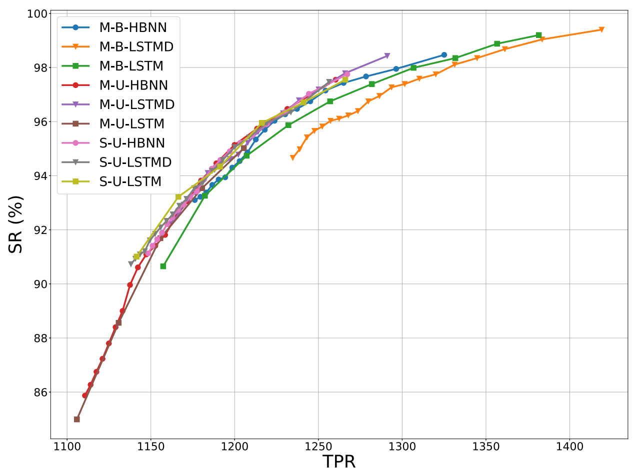

Fig. 4 Total predicted processing units for S-U, M-U and M-B models. The closer the curve to the top left corner, the better the model

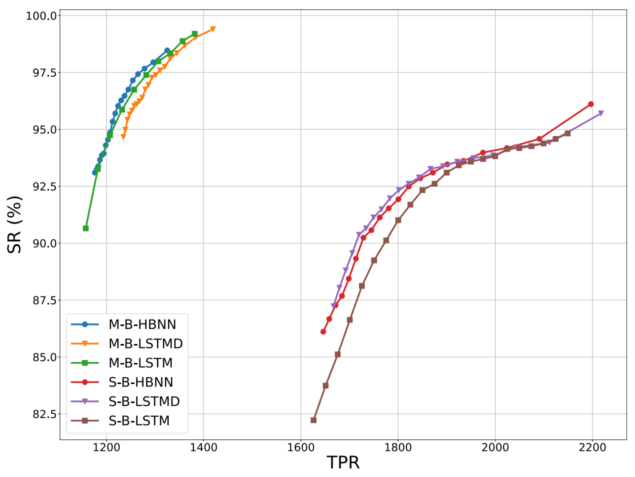

Fig. 5 Total predicted processing units for bivariate models. The ones trained with a single dataset perform very poorly, with a much higher amount of TPR w.r.t. than the models trained with multiple datasets

First of all, results for MSE and MAE are consistent, so both metrics reveal the same patterns in the evaluation. Training a model with multiple datasets improves the accuracy of bivariate models, and the bigger improvement is for HBNN. The opposite happens for univariate models, even if performance degradation is quite limited. Also, univariate models outperform bivariate models, as confirmed by other previous works [66]. This happens especially when the size of the overall training data is small, there is no strong correlation between time series [67] and when the focus is on short-term predictions [68]. Indeed, we analysed the Pearson correlation between the processing units and memory demand, finding a weak correlation between the time series and determining that homoscedasticity holds according to the Breusch-Pagan test [69]. Considering the three architectures separately, LSTMD achieves the best accuracy score across the different training combinations. On the contrary, the S-B-HBNN fails to converge due to the extra complexity given by the Bayesian layer; the S-B-LSTMD struggles and does not achieve the same performance as the univariate, while S-B-LSTM worsens to a lesser extent.

As the results stand, it is not clear whether training multivariate models (cheaper) with multiple datasets (more expensive) has a positive payoff in comparison with training more univariate models (more expensive) with a single dataset (cheaper). Moreover, according to the Diebold-Mariano test [70] at 95% confidence level,[3] the performances of all models are not statistically different (except for the single bivariate models, which are different from all other models). For these reasons, we cannot limit the analysis to these traditional accuracy metrics.

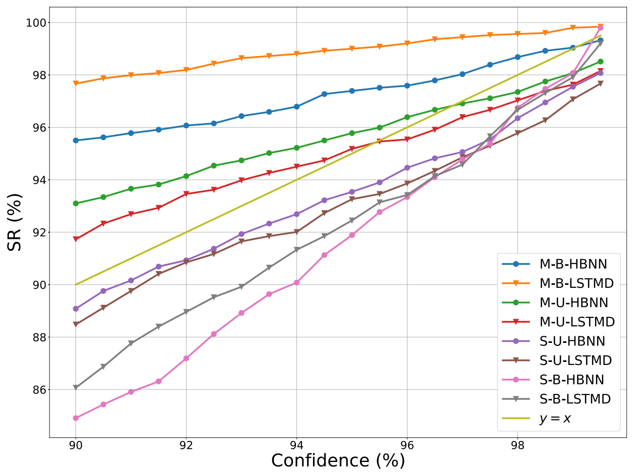

Fig. 6 Target service level versus SR for memory prediction. The plot is built w.r.t. the UB of the confidence interval by varying the target service level. The closer the plot is to the line $y = x$, the more accurate the model under this metric

In Fig. 4, we draw a graphical representation of the TPR versus the SR for processing units demand. The graph is drawn by computing the SR and TPR for the entire range of target service levels. One model outperforms another one when its curve is more to the left than the other's (SRs being equal) and its curve is above the other's (TPRs being equal). To improve the readability of the graph, we removed the curves of the single bivariate models, which achieve poor performance under this metric. Figure 5 focuses on the comparison of bivariate models trained with single and multiple datasets and confirms the results of the accuracy metrics, i.e. that the model benefits significantly if it is trained with more data. In particular, the M-B models achieve between around 25% and 60% saving in terms of

[3] We run a pairwise test for each of the model's combinations.

TPR for both processing units and memory demand, with a higher SR.

Moreover, we evaluate the impact of uncertainty-aware predictions by plotting the confidence levels versus the success rate achieved by the models. A perfect model would achieve an SR equal to the targeted confidence level, i.e. would resemble the bisector line $y = x$. Indeed, we also compute the MSE and MAE of each model's curve w.r.t. to such a bisector line to represent the plot in numerical values and enhance interpretation. The analysis is depicted in Fig. 6. It shows M-U-LSTMD is the model that overall achieves the best accuracy, but there are also interesting differences between each combination of the model, e.g. the HBNN outperforms the LSTMD counterpart in the case of S-U and M-B versions. Depending on the target service level provided to the final customer, the cloud manager could prefer models that achieve a higher SR than the target confidence level rather than a lower one. For instance, we should favour a model that achieves a 96% SR compared to 94% w.r.t. a 95% confidence level.

The applicability of DL models to real-world scenarios strongly depends on the time necessary for training and deploying the model in a cloud resource management setup. The three critical aspects are, therefore, the training time, the fine-tuning time (i.e. how often we update the network weights with newly available data) and the inference time. The training time depends mainly on the size of the training set, the size of the network, and the learning algorithm; the fine-tuning time depends on how often we update the weights of the deep learning model, and the inference time is related to the prediction time once the model is trained. We fix the size of the network (by hyperparameterization) and the learning algorithm, and we measure training time by varying the size of the training set in 20%, 40%, 60% and 80%. We measure fine-tuning time by varying the number of steps among 6, 12, 18 and 24, which correspond to 30, 60, 90 and 120 min frequency. We measure the inference time of predicting one sample. The results are computed as an average of 10 runs for all the model combinations, and we report values in Table 2. As we can see, the models take more or less the same time for the fine-tuning and inference steps, with the HBNN often being the fastest network. The training time, instead, varies with the type of training and the prediction task. The HBNN is the slowest model (because the Bayesian layer is more complex than dense layers), except for the univariate version trained with multiple datasets. However, the training phase is generally infrequent and done offline, e.g. overnight, so it is not a critical factor. At the same time, fine-tuning and inference happen more frequently in resource management operations. We would also like to underline that the results for the S-U versions refer to the training of one single (dataset, resource) pair, which means that we need to run this phase 24 times (12 clusters $\times$ 2 resources). This also applies to the S-B (12 times) and M-U versions (2 times). We conclude that all the models can be practically deployed in real-world scenarios, with the advantage of having a model trained with multiple datasets, which does not require any parallelization of the systems for each possible cluster cell.

To summarize, results show an overall positive impact of uncertainty-aware models over the LSTM baseline, with different trade-offs given by training univariate and multivariate models on single and multiple datasets. Appendix A includes further results on the model comparison.

4.2 On transfer learning capabilities

The second part of the experiments investigates the domain generalization capabilities of the uncertainty-aware models in the scenarios defined in Sect. 3.2. The methodology we

Table 2 Runtime performance analysis. In italics is the best model for each subgroup. The training time is computed by training the model with 20, 40, 60 and 80%, respectively. The fine-tuning time is computed by fine-tuning the network every 6, 12, 18 and 24 steps, which corresponds to 30, 60, 90 and 120 min, respectively. The inference time is the time to predict one sample

Model	Training time [s]				Fine-Tuning time [s]				Inference time [s]
	20%	40%	60%	80%	6	12	18	24	1 sample
S-U-LSTM	*12*	*21*	*29*	*48*	5.38	7.4	12.77	9.29	0.0002
S-U-HBNN	35	140	268	325	*1.57*	*1.63*	*1.7*	*1.64*	0.0058
S-U-LSTMD	15	25	38	52	2.49	2.32	2.22	1.68	*0.00002*
S-B-LSTM	11	21	32	*53*	3.03	*1.61*	*1.34*	*1.54*	0.0002
S-B-HBNN	57	106	154	172	*1.84*	2.01	1.88	2.01	*0.0001*
S-B-LSTMD	*10*	*16*	*25*	237	3.17	4.06	2.98	2.26	*0.0001*
M-U-LSTM	1121	947	869	1050	2.54	2.44	2.37	3.41	*0.0001*
M-U-HBNN	*486*	*425*	*530*	*621*	*1.7*	*1.63*	*1.63*	1.79	*0.0001*
M-U-LSTMD	714	518	706	836	4.92	2.3	5.35	*1.49*	*0.0001*
M-B-LSTM	1441	*1323*	*1603*	1602	*2.29*	*1.73*	11.81	5.45	0.0002
M-B-HBNN	2123	3210	3230	2281	2.54	2.3	2.6	*2.78*	*0.0001*
M-B-LSTMD	*954*	1559	1477	*1592*	3.41	3.0	*2.09*	3.33	*0.0001*

describe can be applied to any of the models proposed in the first part of the experiments, but it is expensive. Thus, in this second part, we select only the M-B-HBNN network, being the best model according to the results of the previous Sect. 4.1. We chose this model because it shows an efficient trade-off between performance (i.e. close to the best performance in terms of accuracy and service level metrics) and complexity of deployment for the experiments (i.e. expensive to train with multiple datasets but one single model that provides predictions for processing units and memory for all datasets). We leave for future work a more comprehensive investigation that also involves the other competitive models, such as some of the LSTMD-based ones. This analysis focuses first on the impact of zero-shot learning (no historical data) and fine-tuning when considering all datasets, while the second delves into the same and different-distribution scenarios.

4.2.1 All datasets

We design four prediction scenarios that differ based on the combinations of datasets used for training (i.e. source domain) and testing (i.e. target domain). All of these are based on the *All Datasets* scenario described in Sect. 3.2.

- **All**: the source domain is all the (twelve) datasets, and the target domain is one of the (twelve) datasets at a time. In other words, we use a portion of all the twelve source datasets (domains) for training, and we use another portion of one of these datasets (domains) for testing.
- **All FT**: we consider *All* as the pretrained model. One by one, we use each of the (twelve) datasets to fine-tune *All* and as a target domain. This scenario assesses whether the fine-tuning process leads to a better weight configuration for the target domain we want to predict.
- **All-but-one**: the model is trained with all but one datasets (source domain). We then predict the remaining dataset (target domain). This experiment further investigates the model's generalisation capabilities on unseen datasets in a zero-shot fashion.
- **All-but-one FT**: we consider *All-but-one* as the pretrained model. One by one, we use each of the excluded datasets to fine-tune *All-but-one* and as a target domain. This investigates the combination of pretraining and fine-tuning on newly available data.

In all scenarios, the final performance is obtained by averaging results across all the twelve target datasets. Figure 7 shows an illustrative example of four of them. It is worth noting that we only build the following most

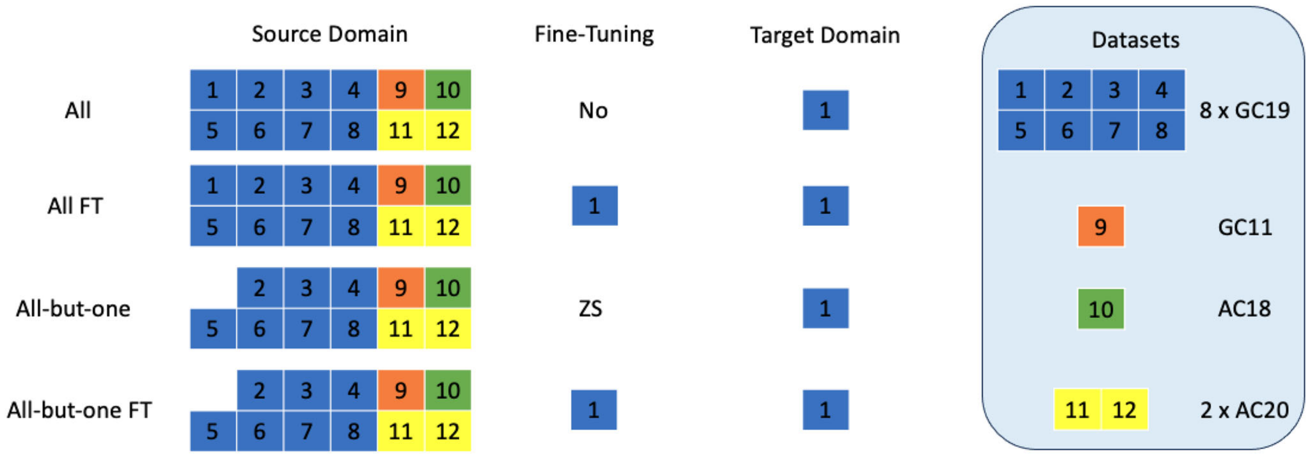

Fig. 7 An illustrative example of four training scenarios. In this example, we consider dataset number 1 of GC19 as the target domain (and FT dataset where applicable), and the source domains are reported accordingly. To note, ZS stands for Zero-Shot, so that, in practice, we do not fine-tune the model

Table 3 Accuracy results for M-B-HBNN trained on scenarios *All*, *All FT*, *All-but-one* and *All-but-one FT*. In bold, the best performance

Scenario	Processing units			Memory		
	MSE	MAE	QL (99%)	MSE	MAE	QL (99%)
All	**0.0042**	**0.0471**	0.110	**0.0043**	**0.0436**	0.084
All FT	0.0058	0.0561	**0.102**	0.0062	0.0058	**0.061**
All-but-one	0.0044	0.0481	0.116	0.0045	0.0448	0.099
All-but-one FT	0.0057	0.0563	0.108	0.0058	0.0538	0.082

representative scenarios among all the possible combinations we could build, and we leave exhaustive experimentation for future work.

In Table 3, we report the accuracy results in terms of MSE, MAE and QL. The model trained under *All* scenario achieves the best accuracy, showing the benefit of training with data that overlap with the target domain (we can consider this scenario as a baseline for these experiments). This shows that including traces from the same provider benefits the model's accuracy. The model trained under *All-but-one* achieves performance close to the one in *All*, proving that knowledge can successfully transfer to the target domains. Fine-tuning improves the uncertainty estimation, reducing the QL in both instances. However, it increases the point accuracy(MAE and MSE); this is likely due to the fact that the model (tuned to reduce the negative log-likelihood) overfits the target dataset or that it forgets the demand patterns learned on other traces but fits in a more accurate way the uncertainty.

We also analyse the TL capabilities of the HBNN regarding the service level metrics. The results are presented in Figs. 9 and 10. We can see that the reduction in point accuracy is influencing these metrics more than the improvement in the QL. Surprisingly, *All-but-one* shows the best performance across all models, confirming the potential of TL on this domain.

Finally, we analyse how uncertainty varies when inference is made on different target datasets. In Fig. 8, we plot the values of the standard deviation (that governs the size of the confidence interval, and its magnitude is proportional to the amount of uncertainty in the prediction) of *All* vs *All-but-one*. The plot shows that the standard deviation values for *All-but-one* are higher than the values for *All*. This is because when the model predicts unseen datasets (missing from the training) the model is more uncertain about its predictions. This further confirms the usefulness of uncertainty as an additional feature to predictions.

4.2.2 Same/different-distribution datasets

The focus of these experiments is to investigate the scenarios presented in Sect. 3.2. In TL-GC19, we only use the eight datasets from GC19 for training, and then we test on each individual dataset. These scenarios assess whether pretraining and FT enhance generalisation capabilities on different domains from the same providers (same-distribution scenario). We consider the impact of not training the model in the target domain (TL-GC19 All-but-one) and the introduction of fine-tuning (represented by the addition of "FT" as suffix). In TL-GC19vsOther, the source domain is GC19 as well, and we test its zero-shot and FT performance on GC11, AC18 and AC20. These scenarios assess whether pretraining and FT enhance generalisation capabilities on different domains from other providers or clusters from the same provider (our-of-distribution scenario).

We report the accuracy results of scenario TL-GC19 (where the source and target dataset are drawn from GC19) in Table 4. The table shows that the baseline *TL-GC19 All* achieves the best performance in terms of point accuracy, while the fine-tuning models achieve the best uncertainty estimation. Error worsens (it doubles for MSE and increases by around 40% for MAE in both resources) when the source and target domains do not overlap, i.e. *TL-GC19 All-but-one*. Thus, although the source and the target are from the same cloud provider, M-B-HBNN struggles to transfer domain knowledge effectively. One possible explanation is that the source dataset (i.e. seven datasets) is

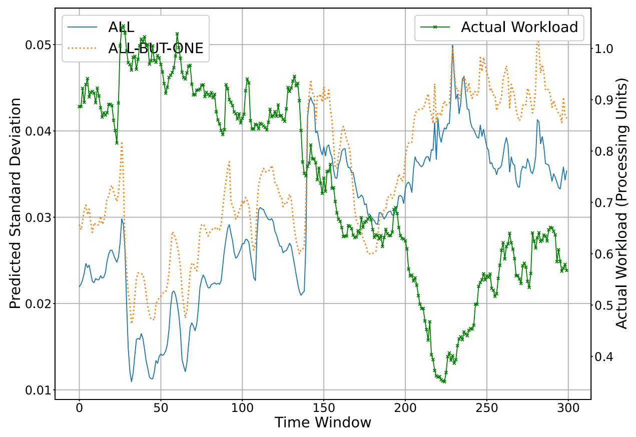

Fig. 8 Standard deviation variability for across inference on the test set *All* (solid blue) and *All-but-one* (dotted orange). The standard deviation is compared to the magnitude of the actual workload (green). Uncertainty for All-but-one is higher for predictions on unseen data

Table 4 Accuracy results for M-B-HBNN trained on the same-distribution scenario *TL-GC19*. In bold, the best performance

Scenario	Processing units			Memory		
	MSE	MAE	QL (99%)	MSE	MAE	QL (99%)
TL-GC19 All	**0.0043**	**0.0474**	0.113	**0.0044**	**0.0444**	0.118
TL-GC19 All FT	0.0059	0.0562	0.060	0.0063	0.0546	0.081
TL-GC19 All-but-one	0.0087	0.0673	0.117	0.0091	0.0681	0.363
TL-GC19 All-but-one FT	0.0097	0.0733	**0.056**	0.0104	0.0701	**0.066**

Table 5 Accuracy results for M-B-HBNN trained on the ifferent-distribution scenario *TL-CG19vsOther*

Target	FT	Processing units			Memory		
		MSE	MAE	QL (99%)	MSE	MAE	QL (99%)
GC11	✗	0.1443	0.3176	0.014	0.2467	0.471	0.021
	✔	0.0588	0.1669	0.029	0.0421	0.1634	0.023
AC18	✗	0.0737	0.2356	0.108	0.0598	0.2242	0.092
	✔	0.0271	0.1114	0.090	0.01	0.0614	0.054
AC20	✗	0.3206	0.4658	0.641	0.09	0.2466	0.191
	✔	0.0522	0.1791	0.227	0.0608	0.1934	0.395

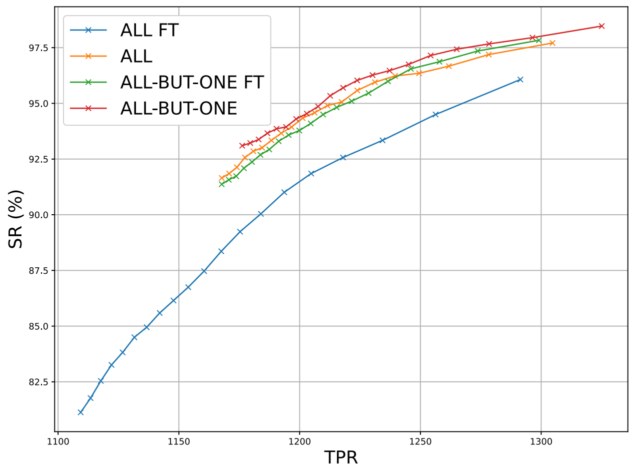

Fig. 9 Success rate in function of the total predicted processing units for *All*, *All FT*, *All-but-one* and *All-but-one FT*. The closer the curve to the top left corner, the better the model

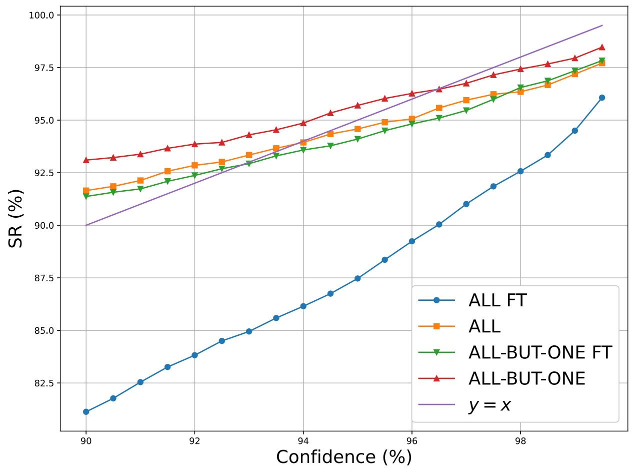

Fig. 10 Target confidence level versus SR for processing units prediction for *All*, *All FT*, *All-but-one* and *All-but-one FT*. The closer the plot is to the line $y = x$, the more accurate the model under this metric

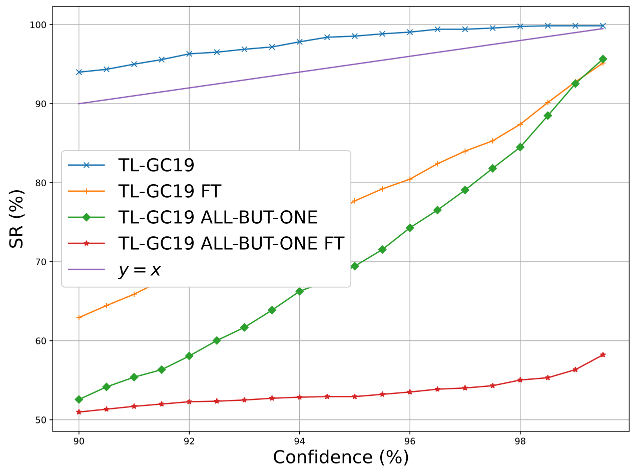

Fig. 11 Target service level versus SR for processing units prediction for *TL-GC19*. The closer the plot is to the line $y = x$, the more accurate the model under this metric

too small to capture the distribution-independent properties of the CG19 provider. The improvement of the uncertainty estimation is in line with the previous results.

Accuracy results of scenario TL-GC19vsOther (where GC19 is the source dataset and the target dataset is one of the other three datasets) are reported in Table 5. The model's error is quite high for all three target domains (in particular for AC20), which means that, again, the model struggles to generalise on unseen domains (this time from different providers). However, fine-tuning on the target domain helps to improve the accuracy performance (especially for processing units on AC20), although error values are an order of magnitude higher than the *All* baseline. Regarding the QL, the fine-tuning results are opposite, with no consistent pattern across the different datasets.

Figure 11 shows the results for scenario *TL-GC19*. Because results for scenario *TL-GC19vsOther* are poor (i.e. M-B-HBNN fails to generalise on unseen cloud providers), we avoid reporting and detailed analysis. From the aforementioned figures, the main takeaways are that (i) *All-but-one* achieves similar performance to *All*, confirming TL capabilities found earlier by the accuracy metrics (ii) the same TL capabilities do not hold when the source and target domains are from GC19, i.e. *TL-GC19*, and (iii) fine-tuning always harm the performance.

Overall, the results of our TL experiments show that M-B-HBNN can transfer domain knowledge effectively only when the source domain used for training is sufficiently big and sufficiently similar to the target domain. On the contrary, it struggles to generalise well when the source domain is small or the target domain is very different from the target domain. Additionally, the model's performance can be adversely affected by limited data, as it may not learn to generalize well to new data. Furthermore, the complexity of the model can lead to overfitting, where the model becomes too specialized to the specific characteristics of the training datasets and fails to perform well on unseen data. Appendix A includes further results on the transfer learning capabilities.

5 Conclusion and future work

Accurate cloud workload forecasting is fundamental for resource managers to optimize their decision-making process. In this paper, we propose a framework and a variety of realistic scenarios and investigate the performance of deep learning probabilistic forecasting models. We first evaluated univariate and bivariate versions of HBNN and LSTMD that can predict processing units (CPU and GPU) and memory usage of workload traces (i.e. datasets) from Google Cloud and Alibaba. Experiments show that training one model on multiple datasets benefits the learning task more than training an individual model on each dataset. While traditional pointwise accuracy metrics do not fully capture model effectiveness, predictions with uncertainty based on confidence levels positively impact cloud service metrics. The runtime of all models is efficient enough for real-world deployment. Notably, the bivariate HBNN trained on multiple datasets (M-B-HBNN) offers a good trade-off between performance and deployment complexity. We further explored the transfer learning capabilities of uncertainty-aware models, focusing on generalization to different-distribution target domains. Experiments reveal that performance degrades as source and target domains differ, especially across different cloud providers, and fine-tuning often does not aid knowledge transfer. Acceptable transfer learning performance is achievable when the source domain is large enough to account for target domain diversity. Further investigation into transfer learning, particularly the relationship between source and target domains, is needed to make the transfer successful.

In future works, we plan to tailor the training to a desired confidence level by explicitly, for example, leveraging a non-symmetrical loss function. We will design and implement a fully Bayesian neural network to capture the full posterior distribution over network weights and outputs, enhancing uncertainty estimation. Additionally, we aim to explore the effectiveness of uncertainty-aware models for multi-step-ahead workload prediction and machine-level workload forecasting. Finally, integrating uncertainty-aware predictions with scheduling, resource allocation, scaling, and balancing in cloud computing environments remains a key area of interest.

Additional results on models comparison

The service level metrics presented in [4] are:

- Success Rate (SR): the percentage of future demand within the confidence interval, i.e. whether the target confidence level is met. For example, when a 95% service level is set, we would like 95% of the requests to be within the UB of the prediction.
- Overprediction (OP): it is related to the predicted resource waste, i.e. the percentage amount of overprediction, defined as the difference between the UB of the prediction and the real demand for the requests within the confidence interval.
- Underprediction (UP): it quantifies the amount of unmatched demand that the customer requested, i.e. the percentage amount of underprediction, defined as the difference between the real demand and the UB of the prediction for the requests greater than the UB.
- Total Predicted Resources (TPR): it quantifies the overall amount of predicted resources that would be given as input to the resource manager, i.e. the sum of all the upper bounds of the predictions, in percentage w.r.t. the real demand.

Table 6 shows values for the service level metrics, where we report only target service levels such as 95% and 99% (as more significant for systems that require high-quality service levels). To help readers interpret such results, we also draw a graphical representation of the TPR versus the SR for processing units demand in Fig. 4.

From the aforementioned table and Fig. 4, we can see that the uncertainty-aware models consistently outperform the LSTM baseline, with HBNN outperforming LSTMD for the processing units and the other way around for the memory. Also, Training with multiple datasets has more advantages over training with single datasets, despite the single univariate model seeming superior in the case of memory prediction at a 99% service level. This confirms that the univariate prediction is easier than the bivariate version.

Additional results on transfer learning capabilities

We also analyse the TL capabilities of M-B-HBNN regarding the service level metrics. Table 7 reports the results for scenarios *All*, *All FT*, *All-but-one* and *All-but-*

Table 6 Results for the service level metrics. We compare each HBNN and LSTMD with an LSTM baseline. Confidence intervals for LSTMs are tailored to the success rates achieved by HBNN and LSTMD. In bold, we report the best model for each subgroup, and in italics, the best model for each pair

		Processing Units					Memory				
Target QoS	Model	SR (%)	OP (%)	UP (%)	TPR (%)	QL[a]	SR (%)	OP (%)	UP (%)	TPR (%)	QL[a]
95%	S-U-LSTM	93.86	*18.08*	0.53	*117.55*	-	93.54	***12.58***	0.39	***112.18***	-
	S-U-HBNN	93.86	18.09	*0.51*	117.58	0.069	93.54	13.59	*0.38*	113.21	0.049
	S-U-LSTMD	93.14	17.11	*0.56*	116.55	0.066	93.26	***11.80***	0.41	***111.39***	0.043
	S-U-LSTM	93.14	***16.57***	0.63	***115.94***	-	93.26	12.43	*0.40*	112.03	-
	S-B-LSTM	91.93	85.88	3.27	182.61	-	91.89	71.49	*1.33*	170.16	-
	S-B-HBNN	91.93	*82.10*	*2.93*	*179.17*	0.315	91.89	*71.47*	1.43	*170.04*	0.258
	S-B-LSTMD	92.61	*84.26*	3.01	*181.25*	0.324	92.45	*71.89*	1.39	*170.50*	0.260
	S-B-LSTM	92.61	89.09	*3.00*	186.08	-	92.45	72.58	*1.24*	171.34	-
	M-U-LSTM	91.41	15.63	0.76	114.87	-	95.78	15.54	*0.27*	115.27	-
	M-U-HBNN	91.41	***15.39***	*0.70*	***114.69***	0.059	95.78	*15.20*	0.30	*114.90*	0.055
	M-U-LSTMD	94.62	*19.55*	*0.43*	*119.12*	0.075	95.18	*14.53*	0.32	*114.21*	0.052
	M-U-LSTM	94.62	19.67	0.46	119.21	-	95.18	14.91	*0.30*	114.61	-
	M-B-LSTM	95.70	22.61	0.39	122.22	-	97.39	19.90	***0.14***	119.76	-
	M-B-HBNN	95.70	*21.58*	***0.38***	*121.20*	0.083	97.39	*19.47*	0.16	*119.31*	0.070
	M-B-LSTMD	96.99	28.25	***0.24***	128.01	0.108	99.00	*24.30*	***0.06***	124.24	0.088
	M-B-LSTM	96.99	*25.84*	0.27	*125.57*	-	99.00	*24.30*	0.06	*124.24*	-
99%	S-U-LSTM	97.03	24.61	0.23	124.37	-	97.55	***18.11***	*0.16*	***117.95***	-
	S-U-HBNN	97.03	*24.04*	*0.23*	*123.81*	0.096	97.55	18.37	0.16	118.21	0.069
	S-U-LSTMD	97.12	***22.61***	*0.22*	*122.34*	0.092	97.77	18.11	*0.15*	118.12	0.061
	S-U-LSTM	97.12	25.55	0.26	125.33	-	97.77	18.80	*0.13*	118.67	-
	S-B-LSTM	95.47	94.09	1.10	190.12	-	97.89	88.29	1.01	185.25	-
	S-B-HBNN	95.47	***91.89***	*1.05*	*188.11*	0.436	97.89	*87.05*	*0.95*	*183.25*	0.376
	S-B-LSTMD	96.91	*90.11*	*0.98*	*186.78*	0.445	98.12	*85.32*	0.91	*181.90*	0.371
	S-B-LSTM	96.91	91.67	1.04	188.67	-	98.12	86.28	*0.93*	182.57	-
	M-U-LSTM	96.01	22.65	0.36	121.99	-	97.56	21.79	*0.18*	120.34	-
	M-U-HBNN	96.01	*21.01*	*0.34*	*121.03*	0.091	97.56	*21.50*	0.22	*119.98*	0.078
	M-U-LSTMD	96.88	23.91	0.33	123.98	0.104	98.00	22.23	*0.18*	121.01	0.075
	M-U-LSTM	96.88	*23.22*	*0.31*	*123.10*	-	98.00	22.51	0.19	121.34	-
	M-B-LSTM	98.00	25.55	0.21	126.33	-	99.12	**22.91**	0.10	123.66	-
	M-B-HBNN	98.00	*24.61*	*0.20*	*125.89*	0.116	99.12	22.90	*0.09*	123.60	0.099
	M-B-LSTMD	98.45	27.44	***0.19***	127.88	0.151	99.33	23.10	***0.08***	123.88	0.125
	M-B-LSTM	98.45	*25.88*	0.22	*125.78*	-	99.33	23.57	0.10	*123.57*	-

[a] Quantile loss is computed only for the probabilistic models

one FT. Table 8 reports the results for scenario *TL-GC19*. Because results for scenario *TL-GC19vsOther* are poor (i.e. M-B-HBNN fails to generalise on unseen cloud providers), we avoid reporting and detailed analysis. From the aforementioned tables, the main takeaways are that (i) *All-but-one* achieves similar performance to *All*, confirming TL capabilities found earlier by the accuracy metrics (ii) the same TL capabilities do not hold when the source and target domains are from GC19, i.e. *TL-GC19*, and (iii) fine-tuning always harm the performance.

Acknowledgements This work was conducted with the financial support of 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 the EU Horizon projects Glaciation (GA No. 101070141) and TAILOR (GA No. 952215), and by the Google Cloud Research Credits program with the award GCP203677602. Andrea Rossi is the corresponding author.

Author contributions Author contribution Conceptualization, A, R.; methodology, A, R.; software, A, R.; formal analysis, A, R., A.V.; investigation, A, R.; resources, S, P., K, B.; data curation, A, R.; writing-original draft preparation, A, R., A, V.; writing-review and editing, A, R., A, V., D, C.; visualization, A, R.; supervision, A, V., S,

Table 7 Service level metrics results of M-B-HBNN for scenarios *All*, *All FT*, *All-but-one* and *All-but-one FT*. In bold, the model with the best SR for each confidence level

		Processing units					Memory				
Confidence Level	Model	SR(%)	OP(%)	UP(%)	TPR(%)	QL	SR(%)	OP(%)	UP(%)	TPR(%)	QL
95%	All	**94.58**	20.53	0.51	120.02	0.079	95.74	16.57	0.25	116.31	0.060
	All FT	87.47	16.63	1.16	115.47	0.064	85.87	9.74	0.83	108.92	0.035
	All-but-one	95.7	21.58	0.38	121.2	0.083	97.39	19.47	0.16	119.32	0.070
	All-but-one FT	94.1	20.33	0.47	119.86	0.078	**95.14**	16.94	0.28	116.65	0.061
97%	All	95.95	22.9	0.39	122.5	0.090	**97.11**	18.53	0.19	118.35	0.068
	All FT	91.01	19.61	0.84	118.77	0.077	90.85	11.94	0.55	111.39	0.044
	All-but-one	**96.75**	24.18	0.28	123.9	0.095	98.03	21.8	0.11	121.69	0.080
	All-but-one FT	95.46	22.59	0.35	122.24	0.088	96.15	18.63	0.21	118.41	0.069
99%	All	97.19	27.44	0.24	127.19	0.110	98.35	22.29	0.11	122.18	0.084
	All FT	94.5	25.45	0.44	125.0	0.102	96.07	16.32	0.26	116.06	0.061
	All-but-one	**97.95**	29.15	0.15	128.99	0.116	**99.04**	26.23	0.06	126.17	0.099
	All-but-one FT	97.35	26.94	0.2	126.74	0.108	97.71	21.86	0.13	121.73	0.082

Table 8 Service level metrics results of M-B-HBNN for scenario *TL-GC19*. In bold, the model with the best SR for each confidence level

		Processing units					Memory				
Confidence Level	Model	SR(%)	OP(%)	UP(%)	TPR(%)	QL	SR(%)	OP(%)	UP(%)	TPR(%)	QL
95%	TL-GC19	**98.55**	10.85	0.075	112.04	0.013	**97.18**	12.29	0.13	109.12	0.080
	TL-GC19 FT	77.7	4.96	0.85	105.34	0.035	89.36	7.89	0.41	104.45	0.051
	TL-GC19 All-but-one	69.44	4.07	1.05	104.26	0.102	74.44	4.75	1.12	100.59	0.307
	TL-GC19 All-but-one FT	52.93	13.62	22.08	92.78	0.029	63.79	46.63	10.35	133.24	0.031
97%	TL-GC19	**99.42**	12.53	0.02	113.75	0.088	**98.19**	14.07	0.1	110.94	0.094
	TL-GC19 FT	84.0	6.04	0.59	106.69	0.043	92.9	9.25	0.28	105.94	0.062
	TL-GC19 All-but-one	79.07	5.22	0.65	105.81	0.107	83.06	6.35	0.7	102.61	0.327
	TL-GC19 All-but-one FT	54.02	14.45	20.67	95.01	0.037	65.03	48.93	9.27	136.63	0.042
99%	TL-GC19	**99.86**	15.74	0.01	116.96	0.113	**98.84**	17.47	0.06	114.38	0.118
	TL-GC19 FT	92.76	8.28	0.29	109.23	0.060	96.6	11.93	0.14	108.76	0.081
	TL-GC19 All-but-one	92.54	7.74	0.23	108.74	0.117	92.9	9.7	0.26	106.41	0.363
	TL-GC19 All-but-one FT	56.34	16.1	18.1	99.23	0.056	69.08	53.47	7.4	143.03	0.066

P., K, B.; project administration, K, B.; funding acquisition, K, B. All authors have read and agreed to the published version of the manuscript

Funding Science Foundation Ireland 18/CRT/6223, Science Foundation Ireland 16/RC/3918, Science Foundation Ireland 12/RC/2289-P2, EU Horizon Glaciation 101070141, EU Horizon TAILOR (GA No. 952215), Google Cloud Research Credits program GCP203677602

Data availability The data used in the experiments are available on the GitHub repository https://github.com/andreareds/TowardsUncertaintyAwareWorkloadPrediction

Code availability The code used in the experiments are available on the GitHub repository https://github.com/andreareds/TowardsUncertaintyAwareWorkloadPrediction

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Müller, S.D., Holm, S.R., Søndergaard, J.: Benefits of cloud computing: literature review in a maturity model perspective. Commun. Assoc. Inf. Syst. **37**(1), 42 (2015)
2. Marr, B.: Tech Trends in Practice: The 25 Technologies that Are Driving the 4th Industrial Revolution. John Wiley & Sons, ??? (2020)
3. Tirmazi, M., Barker, A., Deng, N., Haque, M.E., Qin, Z.G., Hand, S., Harchol-Balter, M., Wilkes, J.: Borg: the next generation. In: Proceedings of the Fifteenth European Conference on Computer Systems, pp. 1–14 (2020)
4. Rossi, A., Visentin, A., Prestwich, S., Brown, K.N.: Bayesian uncertainty modelling for cloud workload prediction. In: 2022 IEEE 15th International Conference on Cloud Computing (CLOUD), pp. 19–29 (2022). IEEE
5. Minarolli, D., Freisleben, B.: 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, pp. 119–124 (2014). IEEE
6. Minarolli, D., Mazrekaj, A., Freisleben, B.: Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. J. Cloud Comput. **6**(1), 1–18 (2017)
7. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. J. Big data **3**(1), 1–40 (2016)
8. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format+ schema. Google Inc., White Paper **1** (2011)
9. Wilkes, J.: Google cluster-usage traces v3. Technical report, Google Inc., Mountain View, CA, USA (April 2020). Posted at https://github.com/google/cluster-data/blob/master/ClusterData2019.md
10. Jiang, C., Qiu, Y., Shi, W., Ge, Z., Wang, J., Chen, S., Cerin, C., Ren, Z., Xu, G., Lin, J.: Characterizing co-located workloads in alibaba cloud datacenters. IEEE Transactions on Cloud Computing (2020)
11. Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., Li, Y., Zhang, L., Lin, W., Ding, Y.: MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. In: 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 945–960 (2022)
12. Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice. OTexts, ??? (2018)
13. Dinda, P.A., O'Hallaron, D.R.: Host load prediction using linear models. Clust. Comput. **3**, 265–280 (2000)
14. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using arima model and its impact on cloud applications' qos. IEEE Trans. Cloud Comput. **3**(4), 449–458 (2014)
15. Vazquez, C.: Time Series Forecasting of Cloud Data Center Workloads for Dynamic Resource Provisioning. The University of Texas at San Antonio, ??? (2015)
16. Baldan, F.J., Ramirez-Gallego, S., Bergmeir, C., Herrera, F., Benitez, J.M.: A forecasting methodology for workload forecasting in cloud systems. IEEE Trans. Cloud Comput. **6**(4), 929–941 (2016)
17. Kumar, J., Singh, A.K.: Performance assessment of time series forecasting models for cloud datacenter networks' workload prediction. Wirel. Pers. Commun. **116**, 1949–1969 (2021)
18. Brown, R.G.: Exponential Smoothing for Predicting Demand. Little, ??? (1956)
19. Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. J. Econom. **31**(3), 307–327 (1986)
20. Di, S., Kondo, D., Cirne, W.: 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, pp. 1–11 (2012). IEEE
21. Yang, Q., Peng, C., Zhao, H., Yu, Y., Zhou, Y., Wang, Z., Du, S.: A new method based on psr and ea-gmdh for host load prediction in cloud computing system. J. Supercomput. **68**, 1402–1417 (2014)
22. Di, S., Kondo, D., Cirne, W.: Google hostload prediction based on bayesian model with optimized feature combination. J. Parallel Distrib. Comput. **74**(1), 1820–1832 (2014)
23. Wang, T., Ferlin, S., Chiesa, M.: Predicting CPU usage for proactive autoscaling. In: Proceedings of the 1st Workshop on Machine Learning and Systems, pp. 31–38 (2021)
24. Barati, M., Sharifian, S.: A hybrid heuristic-based tuned support vector regression model for cloud load prediction. J. Supercomput. **71**(11), 4235–4259 (2015)
25. Gao, J., Wang, H., Shen, H.: Machine learning based workload prediction in cloud computing. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–9 (2020). IEEE
26. Ajila, S.A., Bankole, A.A.: Cloud client prediction models using machine learning techniques. In: 2013 IEEE 37th Annual Computer Software and Applications Conference, pp. 134–142 (2013). IEEE
27. Shishira, S., Kandasamy, A.: A novel feature extraction model for large-scale workload prediction in cloud environment. SN Comput. Sci. **2**(5), 354 (2021)
28. Didona, D., Quaglia, F., Romano, P., Torre, E.: Enhancing performance prediction robustness by combining analytical modeling and machine learning. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, pp. 145–156 (2015)
29. Zhong, W., Zhuang, Y., Sun, J., Gu, J.: A load prediction model for cloud computing using pso-based weighted wavelet support vector machine. Appl. Intell. **48**, 4072–4083 (2018)
30. Banerjee, S., Roy, S., Khatua, S.: Efficient resource utilization using multi-step-ahead workload prediction technique in cloud. J. Supercomput. **77**, 1–28 (2021)
31. Zhang, W., Li, B., Zhao, D., Gong, F., Lu, Q.: 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), pp. 104–109 (2016). IEEE
32. Duggan, M., Mason, K., Duggan, J., Howley, E., Barrett, E.: Predicting host CPU utilization in cloud computing using recurrent neural networks. In: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 67–72 (2017). IEEE

33. Bi, J., Li, S., Yuan, H., Zhou, M.: Integrated deep learning method for workload and resource prediction in cloud systems. Neurocomputing **424**, 35–48 (2021)
34. Xu, M., Song, C., Wu, H., Gill, S.S., Ye, K., Xu, C.: esdnn: deep neural network based multivariate workload prediction in cloud computing environments. ACM Trans. Int. Technol.(TOIT) **22**(3), 1–24 (2022)
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
36. Xi, H., Yan, C., Li, H., Xiao, Y.: An attention-based recurrent neural network for resource usage prediction in cloud data center. In: Journal of Physics: Conference Series, vol. 2006, p. 012007 (2021). IOP Publishing
37. Arbat, S., Jayakumar, V.K., Lee, J., Wang, W., Kim, I.K.: Wasserstein adversarial transformer for cloud workload prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 12433–12439 (2022)
38. Bao, J., Yang, C., Xia, N., Shen, D.: Long-term workload forecasting in grid cloud using deep ensemble model. In: 2022 Tenth International Conference on Advanced Cloud and Big Data (CBD), pp. 42–47 (2022). IEEE
39. Garg, S., Ahuja, R., Singh, R., Perl, I.: An effective deep learning architecture leveraging birch clustering for resource usage prediction of heterogeneous machines in cloud data center. Cluster Computing, 1–21 (2024)
40. Rossi, A., Visentin, A., Prestwich, S., Brown, K.N.: Clustering-based numerosity reduction for cloud workload forecasting. In: International Symposium on Algorithmic Aspects of Cloud Computing, pp. 115–132 (2023). Springer
41. Liu, B., Lin, Y., Chen, Y.: Quantitative workload analysis and prediction using google cluster traces. In: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 935–940 (2016). IEEE
42. Zia Ullah, Q., Hassan, S., Khan, G.M., et al.: Adaptive resource utilization prediction system for infrastructure as a service cloud. Computational intelligence and neuroscience **2017** (2017)
43. Kim, I.K., Wang, W., Qi, Y., Humphrey, M.: Forecasting cloud application workloads with cloudinsight for predictive resource management. IEEE Transactions on Cloud Computing (2020)
44. Foldesi, L., Valdenegro-Toro, M.: Comparison of uncertainty quantification with deep learning in time series regression. In: NeurIPS 2022 Workshop on Robustness in Sequence Modeling (2022)
45. Gneiting, T., Raftery, A.E.: Weather forecasting with ensemble methods. Science **310**(5746), 248–249 (2005)
46. Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U.R., et al.: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Inf. Fusion **76**, 243–297 (2021)
47. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using arima model and its impact on cloud applications' qos. IEEE Trans. Cloud Comput. **3**(4), 449–458 (2015). https://doi.org/10.1109/TCC.2014.2350475
48. Mammen, P.M., Bashir, N., Kolluri, R.R., Lee, E.K., Shenoy, P.: Cuff: A configurable uncertainty-driven forecasting framework for green ai clusters. In: Proceedings of the 14th ACM International Conference on Future Energy Systems, pp. 266–270 (2023)
49. Kong, L., Sun, J., Zhang, C.: Sde-net: Equipping deep neural networks with uncertainty estimates. arXiv preprint arXiv:2008.10546 (2020)
50. Carraro, D., Rossi, A., Visentin, A., Prestwich, S., Brown, K.N.: Performance and energy savings trade-off with uncertainty-aware cloud workload forecasting. In: 2023 IEEE 31st International Conference on Network Protocols (ICNP), pp. 1–6 (2023). IEEE
51. Dou, Q., Castro, D., Kamnitsas, K., Glocker, B.: Domain generalization via model-agnostic learning of semantic features. Advances in neural information processing systems **32** (2019)
52. Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., Yu, P.: Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering (2022)
53. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.-A.: Transfer learning for time series classification. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 1367–1376 (2018). IEEE
54. Ye, R., Dai, Q.: A novel transfer learning framework for time series forecasting. Knowl.-Based Syst. **156**, 74–99 (2018)
55. Hao, J., Yue, K., Zhang, B., Duan, L., Fu, X.: Transfer learning of bayesian network for measuring qos of virtual machines. Appl. Intell. **51**(12), 8641–8660 (2021)
56. Liu, C., Jiao, J., Li, W., Wang, J., Zhang, J.: Tr-predictior: an ensemble transfer learning model for small-sample cloud workload prediction. Entropy **24**(12), 1770 (2022)
57. Li, Y., Ma, J., Cao, D.: Cross-domain workloads performance prediction via runtime metrics transferring. In: 2020 IEEE International Conference on Joint Cloud Computing, pp. 38–42 (2020). IEEE
58. Nguyen, H.H., Matthews, B., Elahi, I.: Deep domain adaptation for runtime prediction in dynamic workload scheduler. In: Neural Information Processing Systems, pp. 1097–1105 (2012)
59. Zhang, L., Zheng, W., Li, C., Shen, Y., Guo, M.: Autrascale: An automated and transfer learning solution for streaming system auto-scaling. In: 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 912–921 (2021). IEEE
60. Song, B., Yu, Y., Zhou, Y., Wang, Z., Du, S.: Host load prediction with long short-term memory in cloud computing. J. Supercomput. **74**(12), 6554–6568 (2018)
61. Leka, H.L., Fengli, Z., Kenea, A.T., Tegene, A.T., Atandoh, P., Hundera, N.W.: A hybrid cnn-lstm model for virtual machine workload forecasting in cloud data center. In: 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 474–478 (2021). IEEE
62. Patel, E., Kushwaha, D.S.: A hybrid cnn-lstm model for predicting server load in cloud computing. The Journal of Supercomputing, 1–30 (2022)
63. Mao, M., Li, J., Humphrey, M.: Cloud auto-scaling with deadline and budget constraints. In: 2010 11th IEEE/ACM International Conference on Grid Computing, pp. 41–48 (2010). IEEE
64. Kumar, J., Singh, A.: An efficient machine learning approach for virtual machine resource demand prediction. Int. J. Adv. Sci. Technol. **123**, 21–30 (2019)
65. Herbst, N.R., Huber, N., Kounev, S., Amrehn, E.: Self-adaptive workload classification and forecasting for proactive resource provisioning. Concurr. Comput.: Pract. Exp. **26**(12), 2053–2078 (2014)
66. He, Z., Zhao, C., Huang, Y.: Multivariate time series deep spatiotemporal forecasting with graph neural network. Appl. Sci. **12**(11), 5731 (2022)
67. Du Preez, J., Witt, S.F.: Univariate versus multivariate time series forecasting: an application to international tourism demand. Int. J. Forecast. **19**(3), 435–451 (2003)
68. Chayama, M., Hirata, Y.: When univariate model-free time series prediction is better than multivariate. Phys. Lett. A **380**(31–32), 2359–2365 (2016)
69. Breusch, T.S., Pagan, A.R.: A simple test for heteroscedasticity and random coefficient variation. Econometrica: Journal of the econometric society, 1287–1294 (1979)
70. Diebold, F.X., Mariano, R.S.: Comparing predictive accuracy. J. Business Eco. Stat. **20**(1), 134–144 (2002)

Andrea Rossi is a PhD student at the SFI Centre for Research Training in Artificial Intelligence at University College Cork. He got a bachelor's in Information Engineering and a master's in Computer Engineering at the University of Padua (Italy). His research is focused on time series analysis and predictive models for resource management problems in cloud computing environments.

Andrea Visentin has a BSc and an MSc in Computer Engineering from the University of Padua (Italy) and completed a PhD at the Insight Centre for Data Analytics at UCC. He is currently a permanent lecturer at the School of Computer Science \& IT. Moreover, he is a researcher at the Confirm Centre for Smart Manufacturing and the Insight Centre for Data Analytics.

Diego Carraro has a BSc and an MSc in Computer Engineering from the University of Padua (Italy) and completed a PhD at the Insight Centre for Data Analytics at UCC. He was a researcher at the Confirm Centre for Smart Manufacturing and the Insight Centre for Data Analytics.

Steven Prestwich is a Lecturer in the School of Computer Science \& IT, University College Cork, Ireland. He has a PhD in Computer Science from the University of Manchester (UK) and an MA in mathematics from the University of Oxford (UK). He is currently an investigator in the Insight Centre for Data Analytics and the Confirm smart manufacturing centre. His current work includes deep learning, forecasting and supply chain optimisation and constraint acquisition.

Kenneth N. Brown is a Professor in the School of Computer Science \& IT, Deputy Director of Insight@UCC and he co-leads the research challenge on decision making. He is a Funded Investigator on Confirm, the national centre for smart manufacturing. He is a PI and executive committee member of Enable, the inter-centre spoke on smart cities and IoT. His research interests are in the areas of artificial intelligence, optimisation and constraint-based reasoning.