Clustering-based Numerosity Reduction for Cloud Workload Forecasting

Andrea Rossi[1,2] , **Andrea Visentin**[2] , **Steven Prestwich**[2] and **Kenneth N. Brown**[2]
[1]SFI CRT in Artificial Intelligence, University College Cork, Cork, Ireland
[2]School of Computer Science, University College Cork, Cork, Ireland
{a.rossi, s.prestwich, k.brown}@cs.ucc.ie, andrea.visentin@ucc.ie

Abstract

Finding smaller versions of large datasets that preserve the same characteristics as the original ones is becoming a central tool in Machine Learning, especially when computational resources are limited. In this paper, we apply clustering techniques for wisely selecting a subset of datasets for training models for time series prediction of future workload in cloud computing. We train Bayesian Neural Networks (BNNs) and state-of-the-art probabilistic models to predict machine-level future resource demand distribution and evaluate them on unseen data from virtual machines in the Google Cloud data centre. Experiments show that selecting the training data via clustering approaches such as Self Organising Maps allows the model to achieve the same accuracy in less than half the time, requiring less than half the datasets rather than selecting more data at random. Moreover, BNNs can capture uncertainty aspects that can better inform scheduling decisions, which state-of-the-art time series forecasting methods cannot do. All the considered models achieve prediction time performance suitable for real-world scenarios.

1 Introduction

As many companies are migrating their services to the cloud, the demand for resources in cloud computing platforms is continuously growing, with a market size of USD 545.8 billion in 2022 and expected to grow to USD 1,240.9 billion by 2027 [Markets and Markets, 2022]. To make more efficient use of limited resources, predicting near-term demand is becoming ever more important. Providers aim to maximise their profit while correctly predicting customers' requests and managing resources efficiently while limiting the environmental impact of CO2 emissions and pollution [Achar, 2022].

Time series prediction of future workload in cloud computing has been widely studied for more than twenty years [Dinda and O'Hallaron, 2000; Di *et al.*, 2012; Calheiros *et al.*, 2014; Song *et al.*, 2018]. Recent research has shifted to probabilistic forecasts to provide a more comprehensive picture and to help resource scheduler decisions [Minarolli *et al.*, 2017; Mohammadi Bahram Abadi *et al.*, 2018]. Time series probabilistic forecast allows providers to consider uncertainty and noise of the demand while guaranteeing high accuracy.

The emergence of Deep Learning (DL) and Big Data has introduced new approaches for enhancing the ability to generalize using large datasets to train more sophisticated and precise prediction models [Bi *et al.*, 2021; Rossi *et al.*, 2022]. However, when computational resources for training Deep Learning (DL) models are limited, it is important to carefully select the data used for training. The goal is to find a smaller subset of the original dataset that retains its characteristics. This allows a model to be trained with the same accuracy as when using a larger dataset but with reduced computational cost and faster training time. Unsupervised clustering[1] has been proven to be a powerful tool for dividing workload datasets into several groups and training independent forecasting models for each of them [Gao *et al.*, 2020; Yu *et al.*, 2016].

In this paper, we use clustering approaches to group workload datasets into groups that share similar patterns. Then, we select a subset of datasets from each cluster for training DL probabilistic predictive models to accurately and jointly forecast CPU and memory usage demand distributions and compare it to a random selection of the non-clustered training sets. We analyse the performance of a Hybrid Bayesian Neural Network (HBNN) [Rossi *et al.*, 2022], and state-of-the-art predicting models DeepAR [Salinas *et al.*, 2020] by Amazon and Temporal Fusion Transformer (TFT) [Lim *et al.*, 2021] by Google.

From the output probability distribution, we compute confidence intervals. We use their upper bounds to estimate the total predicted resources needed to meet the demand with a level of confidence and relate it to the percentage of correctly predicted requests. We evaluate the predictive methods using 29-day load datasets of 2,500 machines from a Google Cloud cell [Wilkes, 2020].

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

- We investigate the performance of state-of-the-art time series forecasting architectures such as DeepAR and TFT that, to the best of our knowledge, are applied for the first time in the context of workload prediction in

[1]To avoid confusion, we will use 'cluster' to refer to a subcollection of data identified in the workload trace files, and 'cell' to refer to a collection of computational nodes.

cloud computing.

- We provide an analysis and comparison of a wide range of clustering methods suitable for dealing with cloud workload time series.
- We show how a clustering approach leverages the prediction capabilities of DL probabilistic forecasting models, offering a trade-off between accuracy and limited computational resource. An accurate selection of training datasets allows a provider to reach the same accuracy in less than half the time, requiring less than half the datasets.
- We preprocess the workload datasets and make them available for reproducibility.

The rest of the paper is organized as follows. Section 2 describes the background and related work in this field. In Section 3, we describe the forecasting models. Section 4 presents the experimental results based on a real-world Google Cloud workload trace. Finally, Section 5 concludes the paper and discusses future work.

2 Background and Related Work

In this section, we discuss the problem of data reduction, specifically on the clustering approaches analyzed in this paper, with a focus on cloud computing applications.

2.1 Data Reduction

There are several methods for data reduction, including feature selection, numerosity reduction, and dimensionality reduction [Wibbeke *et al.*, 2022]. These methods aim to obtain a smaller version of the dataset while preserving its information value. In workload forecasting in Cloud Computing, feature selection is difficult due to a lack of information in unlabelled workload traces [Ali and Kecskemeti, 2023], and dimensionality reduction techniques can lack explainability [Rojat *et al.*, 2021].

This work focuses on numerosity reduction, which aims to reduce the number of training samples. Coresets are a popular approach for reducing training data, but their formation can be complex [Huang *et al.*, 2021]. Simple random sampling [Watson *et al.*, 2018] is an easy way to reduce training data size, but it does not guarantee that all possible patterns are selected. Stratified sampling [Parsons, 2014] overcomes this problem by dividing samples into subgroups before sampling, but the exact number of clusters is generally unknown in cloud workload traces [Ali and Kecskemeti, 2023].

2.2 Unsupervised Clustering

Clustering is an unsupervised method that aims to divide a set of unlabeled data items into uniform groups or *clusters*. The goal is to create clusters where points are most similar within the group and least similar to those in other groups. K-Means is a popular method that partitions the dataset into K disjoint clusters by minimizing the sum of the squared distances between the data points and their cluster centroids.

Time series can be seen as data in high-dimensional spaces where each timestamp is a feature. However, time series can suffer from the curse of dimensionality [Bellman, 1966], where the increase in distance measurement negatively impacts distance-based algorithms such as K-means clustering. Many approaches have been proposed to deal with time series clustering.

Principal Component Analysis

Principal Component Analysis (PCA) is one of the most popular statistical methods for reducing dataset dimensionality [Pearson, 1901]. The data are transformed linearly into a new orthogonal coordinate system with fewer dimensions that preserves most of the variance of the original data. Due to its high interpretability, especially when data are mapped into two dimensions to identify clusters of correlated data points, it became a popular tool in time series analysis [Lakhina *et al.*, 2004; Li, 2019]. PCA can be used to reduce the dimensionality of the time series before applying K-Means.

Variational Recurrent Auto-Encoder

The Variational Recurrent Auto-Encoder (VRAE) is a model that maps time series data to a latent vector representation of probability distribution in an unsupervised fashion [Fabius and Van Amersfoort, 2014]. It combines the representation learning power of a variational auto-encoder with the ability of recurrent neural networks to deal with sequential data. The network's parameters are learned by jointly optimizing the sum of the Kullback-Leibler divergence between the distribution learned in the latent space and a Gaussian distribution plus the reconstruction error. The variational component of the VRAE can handle larger time series data without slowing down the learning process. This model has been widely applied in anomaly detection and forecasting time series applications [Pereira and Silveira, 2018; Zheng *et al.*, 2021] and can be used to reduce the dimensionality of time series and apply K-Means on the latent space.

Self-Organising Maps

A Self-Organizing Map (SOM) or Kohonen map is an unsupervised machine learning technique introduced by Kohonen [Kohonen, 1990]. It is an artificial neural network trained with competitive learning to create a lower-dimensional representation of the input data. The objective of the training is to convert an input space into a two-dimensional map space. SOMs are inspired by the brain map for visual and sensory information and can produce a map as a method for dimensionality reduction that is a nonlinear generalization of PCA. SOMs can be used as a clustering technique and have been widely applied in time series contexts, including forecasting applications [Cherif *et al.*, 2011; Sarlin and Eklund, 2011; Barreto, 2007].

Dynamic Time Warping

In the context of time series, the centroid can be computed as the arithmetic mean of each timestamp of all the time series grouped into a cluster. However, Euclidean distance has often been proven less effective and accurate than Dynamic Time Warping (DTW) [Petitjean *et al.*, 2014], which can handle time series of different lengths and is invariant to dilation in time. DTW Barycenter Averaging (DBA) is used [Petitjean *et al.*, 2011] as the distance metric. DTW offers the opportunity to apply the K-Means algorithm directly to the time

series data at the cost of extra complexity. It has been used in various time series applications, including sensor-based signal classification in healthcare [Varatharajan *et al.*, 2018] and forecasting tasks [Malarya *et al.*, 2021; Prasetyo *et al.*, 2021; Yu *et al.*, 2016].

3 Workload Prediction Models

Our aim is to help a resource manager to configure available computational resources by using an estimation of the future demand for resources from a predictive model based on the historical workload data.

For the purposes of this paper, we have narrowed our analysis of predictive capabilities to three DL probabilistic forecasting models outlined in this section. This decision was made due to the significant computational resources required for extensive experimentation. However, we believe these models are sufficient to validate our results.

3.1 HBNN

A Bayesian Neural Network (BNN) is a type of neural network with probabilistic weights optimized using Bayes-by-backprop [Blundell *et al.*, 2015]. A variation of BNN, called Hybrid BNN (HBNN), only has a Bayesian last layer as a trade-off between training time and epistemic uncertainty modelling [Kristiadi *et al.*, 2020], i.e. the uncertainty due to the small size of the datasets or the lack of knowledge of the model parameters [Hüllermeier and Waegeman, 2021]. This model has been successfully used for workload prediction in cloud computing [Rossi *et al.*, 2022]. The full model includes a 1D convolution layer, two Long Short-Term Memory (LSTM) layers, and three dense layers, with only the last layer being Bayesian. The output layer is two Gaussian distributions for CPU and memory demand forecasts. The loss function is the negative log-likelihood, and the model captures aleatory uncertainty due to the noise in the input data. The network is implemented using Keras and TensorFlow Probability libraries.

3.2 DeepAR

DeepAR is a time series probabilistic forecasting model using auto-regressive RNNs proposed by Amazon [Salinas *et al.*, 2020]. It is trained on one or more target time series produced by the same or related processes. The model approximates these processes and uses them to forecast the evolution of the target time series. The model estimates the parameters of a Gaussian distribution to produce a probabilistic forecast and is trained by minimizing the log-likelihood function via the Adam algorithm. The model is implemented using the PyTorch Forecasting library with hyperparameter optimization using Optuna. The output of the models consists of seven quantiles of the distribution from which the mean and standard deviation of the Gaussian distribution are calculated.

3.3 Temporal Fusion Transformer

Temporal Fusion Transformer (TFT) is an attention-based Deep Neural Network designed by Google for time series forecasting [Lim *et al.*, 2021]. It can be applied to multi-horizon forecasting and includes components such as an LSTM encoder-decoder layer and a variable selection network for feature selection. The multi-head attention mechanism highlights the most relevant parts of the input window for prediction. The network is optimized using quantile loss as the loss function and is jointly optimized using multiple time series from different distributions. TFT has outperformed DeepAR in electricity demand and traffic forecasting applications. The output of the models consists of seven quantiles of the distribution from which the mean and standard deviation of the Gaussian distribution are calculated.

4 Experiments

The studies in this section investigate the effectiveness of using clustering-based training data selection for cloud workload prediction. The optimal number of clusters for each clustering technique is determined experimentally, and a subset of datasets is selected either randomly or with representatives from each cluster. This subset is used to train predictive models, with the number of datasets varied to compare the models in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE). The models are also compared based on confidence intervals to evaluate the quality of probabilistic forecasts, and runtime performance is evaluated.

4.1 Dataset

The machine-level workload datasets used in the experiments come from the real-world Google Cloud Trace 2019 [Wilkes, 2020], which records resource utilization of tasks for 29 days on eight cells worldwide. The data is compiled from the Instance Usage table of Google trace using Google BigQuery, with missing records ignored and average CPU and memory demand aggregated in 5-minute intervals, similarly to [Herbst *et al.*, 2014; Kumar and Singh, 2019].

The datasets are used to train Deep Learning models to predict future resource demand. Each dataset has 8352 data points, with CPU and memory burden multiplied by a weight proportional to the run time period inside the window, as done in [Janardhanan and Barrett, 2017; Rossi *et al.*, 2022]. The distribution of the datasets is Gaussian, and the time series are stationary according to the Augmented Dickey-Fuller test.

4.2 Experimental Setup

The experiments were conducted using Ubuntu 20.04, a 2.60GHz Intel®Xeon®Gold 6240 CPU, and a 48 GB NVIDIA Quadro RTX 8000 GPU, using historical workload data from Google Cloud Trace 2019. The workload is predicted for a time interval of 10 minutes ahead, which is sufficient for most applications [Mao *et al.*, 2010; Baldan *et al.*, 2016]. The datasets are split into training and test sets, with the first 80% (23 days) of each dataset used for training and the remaining 20% (6 days) used for evaluation. An exhaustive hyperparameter search is conducted to find the best hyperparameters for the Deep Learning models. More details on the implementation, hyperparameters, and clustering techniques can be found in the open-source GitHub repository[2].

[2]https://github.com/andreareds/clustered-reduction-cloud-workload

4.3 Clustering Evaluation

This section describes the evaluation of clustering, where the exact number of clusters is unknown. The performance of the algorithm is evaluated by varying the number of clusters, K. The elbow curve method [Thorndike, 1953] is a popular method for finding the best K, based on the distortion score, which is the sum of squared distances of points from cluster centres. As K increases, the distortion score decreases, but overfitting can occur. The elbow curve method selects K where there is an elbow in the distortion curve as a function of K. The Silhouette score [Rousseeuw, 1987] confirms the results of the elbow method.

The methods used to select the datasets are as follows:

- **Random**: The Random method selects training datasets randomly without using clustering methods.
- **PCA**: The PCA method reduces the dimension of the datasets using the first two principal components and applies K-Means to the latent space, with the best K found to be 7 using the elbow method.
- **VRAE**: The VRAE method trains a VRAE algorithm to minimize reconstruction error and uses the encoder to reduce dataset dimensionality, with K-Means applied to the encoded space and the best K found to be 5.
- **SOM**: The SOM method trains multiple SOMs with varying map sizes and selects the best size using the elbow method, with the best configuration found to be a 2x3 map with 6 clusters.
- **DTW**: The DTW method applies DTW directly on the workload datasets based on the DBA distance function, with resampling used to reduce computation cost and the best K found to be 4.

Clusters are computed using the first 80% of each dataset to avoid leakage in forecasting, while the last 20% is used as the test set.

4.4 Point estimate Accuracy

After clustering the time series, a certain number of datasets L are selected to train the predictive models, with dimensionality reduction used for clustering only. As a baseline, L datasets are randomly selected among all available ones, while when the series are clustered, an equal number of datasets are selected for each cluster such that the total number is L. The models are trained by varying L between 50 and 2000 and tested on datasets not used for training. The first evaluation is in terms of MSE and MAE, computed with respect to the mean of the predicted Gaussian distribution. Table 1 shows the MSE and MAE for CPU and memory demand prediction with L = 500, L = 1000, and L = 2000 for all predicted models trained with the clustering techniques.

First, we can see that independently of the clustering techniques used for training, the HBNN has an MSE and MAE at least five times higher than the ones conducted by the state-of-the-art methods DeepAR and TFT. This is because the HBNN is a much simpler network, incapable of learning patterns from several time series, compared to more complex architectures. DeepAR and TFT achieve similar performance in MSE and MAE for all the considered clustering techniques for both CPU and memory demand.

While TFT previously outperformed DeepAR in electricity demand and traffic forecasting [Lim *et al.*, 2021], for our cloud prediction, it is not possible to conclude which is the best DL model. We believe it is due to the absence of covariates of the resource demand to train the model, an essential component used by the variable selection network of the TFT.

From these experiments, we can see how all the clustering-based selection techniques significantly boost the performance of DeepAR and TFT compared to a random selection of the training data when increasing the number of training datasets, showing the benefits of clustering the datasets before training when the computational resources are limited.

On the contrary, HBNN does not benefit much from a clustered selection of the training datasets. DTW and SOM are the best approaches among all the clustering techniques. This is because DTW has been designed for dealing with time series, while the other approaches are more general and data structure independent. Instead, PCA and VRAE achieve an unsatisfactory score from a point estimate point of view for DeepAR and TFT because the clustering is not applied directly to the time series but to the learned latent space. However, these dimensionality reduction approaches seem more effective for simpler models such as HBNN.

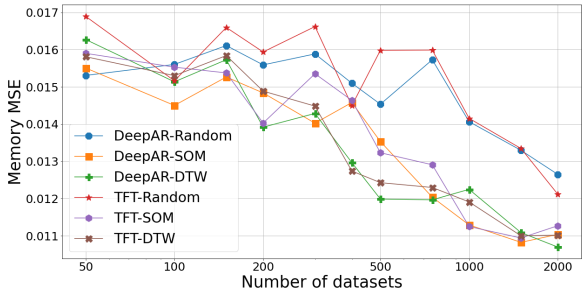

Figure 1: MSE comparison of DeepAR and TFT for the memory demand. DTW and SOM are the best clustering-based selection approaches and allow the models to achieve the same MSE with fewer training datasets compared to a random selection. When the number of datasets is high, the MSE of all the approaches converge. The x-axis is in a logarithmic scale.

To visualise how the performance varies, we plot the MSE of the memory prediction for DeepAR and TFT for all the clustering approaches by varying the number of datasets (whose scale is logarithmic) in Figure 1. From the plot, we can see that once the number of datasets increase sufficiently, all the approaches, including the baseline, converge to similar values of the metrics. However, the clustering approaches allow the models to reach the same performance with much less data and, consequently, fewer computational resources, compared to a random selection of the datasets. This is because clustering helps differentiate training set patterns more effectively than randomly selecting datasets. For example, DeepAR-Random reaches 0.014 with 1,000 training datasets, while DeepAR-DTW requires 200 training datasets only.

		Training Size 500				Training Size 1000				Training Size 2000			
		CPU		Memory		CPU		Memory		CPU		Memory	
Model	**Clustering**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
HBNN	**Random**	0.13	0.291	0.103	0.255	**0.085**	**0.229**	0.12	0.278	0.075	0.217	0.079	**0.212**
	PCA	**0.098**	**0.253**	0.122	0.274	0.09	0.233	0.073	0.216	**0.071**	0.217	**0.071**	0.215
	VRAE	0.158	0.314	0.099	0.253	0.157	0.27	0.081	0.228	0.073	**0.215**	0.081	0.231
	SOM	0.114	0.271	0.124	0.281	0.163	0.319	0.107	0.257	0.082	0.22	0.085	0.226
	DTW	0.106	0.259	**0.07**	**0.21**	0.125	0.283	**0.07**	**0.206**	0.078	0.228	0.081	0.219
DeepAR	**Random**	**0.014**	**0.094**	0.015	0.091	0.014	0.093	0.014	0.09	0.015	0.097	0.013	0.082
	PCA	0.026	0.13	0.014	0.088	0.025	0.126	0.014	0.088	0.013	0.088	**0.011**	**0.077**
	VRAE	0.024	0.124	0.016	0.095	0.014	0.095	0.014	0.09	0.014	0.094	0.013	0.081
	SOM	**0.014**	0.096	0.014	0.088	0.017	0.103	**0.011**	**0.08**	**0.012**	**0.085**	**0.011**	0.079
	DTW	0.022	0.12	**0.012**	**0.082**	**0.013**	**0.09**	0.012	0.083	0.013	0.089	**0.011**	0.078
TFT	**Random**	0.02	0.114	0.016	0.096	0.014	0.093	0.014	0.09	0.014	0.097	0.012	0.084
	PCA	0.024	0.124	0.014	0.088	0.021	0.116	0.013	0.086	0.013	0.087	**0.011**	**0.078**
	VRAE	0.016	0.101	0.015	0.092	0.014	0.094	0.015	0.093	0.014	0.094	0.013	0.082
	SOM	0.017	0.104	0.013	0.087	0.017	0.103	**0.011**	**0.08**	**0.012**	**0.085**	**0.011**	0.08
	DTW	**0.014**	**0.094**	**0.012**	**0.084**	**0.012**	**0.089**	0.012	0.082	0.013	0.089	**0.011**	0.079

Table 1: MSE and MAE for CPU and memory demand with different training sizes (500, 1000 and 2000). In bold, for each model, the best clustering technique.

4.5 Confidence Intervals Evaluation

In this section, we evaluate the forecasting models with the metrics defined in [Rossi *et al.*, 2022]. This evaluation aims to assess the prediction's quality if we want to exploit the prediction for resource allocation and virtual machine provisioning problems. Given the mean and the standard deviation of a Gaussian distribution, we can compute confidence intervals of the prediction and quantify the number of predicted resources based on the upper bound (UB) of the confidence interval to compare the predictive models.

The considered metrics are as follows:

- **Success Rate (SR)**: the percentage of correctly predicted requests, i.e. the percentage of actual requests that are within the UB of the prediction.
- **Overprediction (OP)**: the percentage of overprediction computed as the difference between the UB of the confidence interval and ground truth for those predictions which exceed the real demand of resources.
- **Underprediction (UP)**: the percentage of underprediction computed as the difference between ground truth and the UB of the confidence interval for those predictions which are below the real demand of resources.

We display numerical results for SR, OP and UP in Table 2 using 500 datasets in training for HBNN, DeepAR and TFT models for a 95% confidence interval. Under these metrics, a clustering-based selection improves the performance compared to training with a random selection of the datasets, with DTW and SOM performing best overall, especially for state-of-the-art probabilistic methods.

There are important differences between HBNN and state-of-the-art models. HBNN reaches an SR very close to the target confidence level (2% points above the target), while DeepAR and TFT are over 15% points below. This can be seen graphically in Figure 2, where we measure the SR with a training size of 1000 datasets while varying the confidence level of the predicted probability distribution of memory for the best clustering approaches and the baseline. This is due to the predicted standard deviation of the model. While the Bayesian layer of the HBNN is capable of modelling the epistemic uncertainty of the prediction and the uncertainty of the prediction is quantified by the magnitude of the standard deviation, DeepAR and TFT are overconfident in the prediction, with a much lower standard deviation, capable of modelling only the noise of the data. A similar discussion can be extended to the UP because this metric is related to the percentage of unmatched requests, with a portion of those much lower for the HBNN compared to the other two methods, making it more reliable for high Quality of Service (QoS) applications. Interestingly, the UP of the CPU demand is lower for DeepAR and TFT, meaning that the upper bound of the prediction is quite close to the ground truth. For this reason, the prediction requires further processing to improve its performance. On the other hand, a high standard deviation makes the UB of the prediction much higher, thus increasing the OP, which is up to 5 times lower for DeepAR and TFT compared to the HBNN.

Regarding the clustering approaches, as in the previous results on the point estimate accuracy, DTW and SOM approaches are the best clustering methods to select the datasets for training, with a better SR, OP and UP of up to 6% compared to a random selection of the datasets for state-of-the-art method. The HBNN, again, is not benefiting much from a clustering-based selection. The differences between the clustering methods decrease as the number of datasets used in training increases.

When we use the majority of the datasets in training, it does

		CPU			Memory		
Model	**Clustering**	**SR (%)**	**OP (%)**	**UP (%)**	**SR (%)**	**OP (%)**	**UP (%)**
HBNN	**Random**	98.06	220.67	**19.38**	99.85	**125.11**	**5.08**
	PCA	98.83	179.26	27.99	99.19	136.20	10.79
	VRAE	98.29	180.61	23.17	98.91	151.32	11.30
	SOM	**97.62**	220.24	64.51	**98.15**	140.13	16.38
	DTW	98.25	**173.61**	27.02	98.92	140.82	9.88
DeepAR	**Random**	72.65	42.86	18.60	74.61	29.66	16.42
	PCA	71.88	43.85	18.92	74.22	29.29	15.10
	VRAE	72.49	40.71	18.45	74.66	29.66	15.92
	SOM	76.28	**35.08**	16.05	77.44	**25.74**	11.97
	DTW	**76.85**	38.90	**15.94**	**77.63**	26.55	**11.42**
TFT	**Random**	72.56	42.9	18.85	75.45	30.35	16.36
	PCA	72.84	45.56	19.24	76.04	29.83	15.25
	VRAE	73.72	41.91	18.47	76.92	30.62	16.22
	SOM	78.42	**36.75**	16.22	**80.81**	27.55	11.81
	DTW	**78.43**	39.70	**15.96**	80.09	**27.27**	**11.08**

Table 2: CPU and Memory demand prediction statistics (Success Rate, Overprediction and Underprediction in percentage) of HBNN, DeepAR and TFT with 500 training datasets at 95% confidence. In bold, for each model, the best clustering technique.

not matter whether the datasets are selected randomly or via a clustering approach. This can be seen in Figure 3 for the OP of memory for DeepAR and TFT models while varying the number of datasets. This proves that when a high amount of data is available, an accurate selection of training data that considers different patterns, i.e. different distributions of the training input, is essential to achieve a good trade-off between performance and computational cost. DTW and SOM are the clustering approaches which reduce both the UP and OP for both DeepAR and TFT models, confirming that time series-based clustering approaches are more suitable for grouping similar datasets according to their patterns.

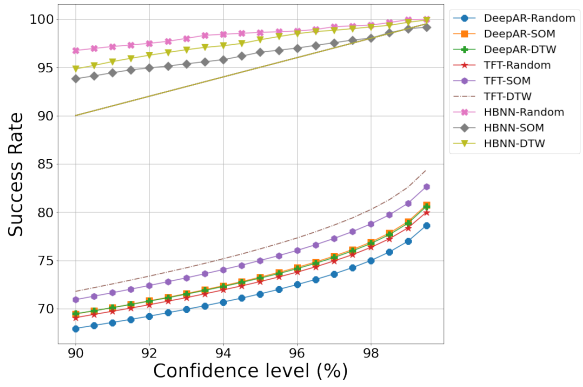

Figure 2: Memory SR comparison of DeepAR, TFT and HBNN as a function of the confidence level with models trained with 1000 datasets. The model is the most accurate when the curve is closest to the line $y = x$. DTW and SOM-based selections improve the SR compared to the random selection of datasets.

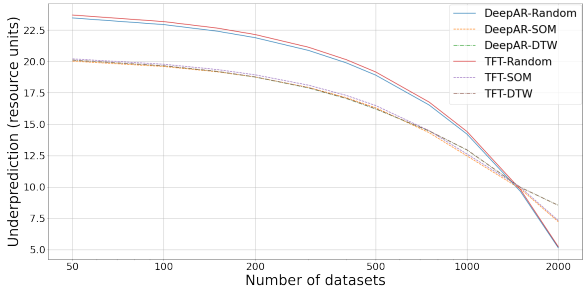

Figure 3: Memory OP comparison of DeepAR and TFT. If DTW and SOM are applied, a much lower number of datasets is required to get the same OP compared to a random selection of the datasets. The x-axis is in a logarithmic scale.

4.6 Runtime Performance Analysis

Another critical aspect of the predictive models in Cloud Computing applications is related to the runtime performance, which is critical for the deployability of the models in real-world scenarios. In particular, we are interested in measuring both the time necessary to train a model and the time to make a prediction once the model is trained.

First, we measure the training time of the models while using the selected datasets from the clusters. We plot the training time by varying the number of training datasets L in Figure 4. We consider the average training time, which is affected by the amount of input data and not by how its selection is performed. Increasing the number of training datasets, i.e. the training data, increases the time for the model to converge, with some fluctuations whose causes include the random initialization of the network and the batch order fed in the backpropagation optimization. However, from the plot, it is clear that DeepAR converges two times faster, independently of the clustering approach used.

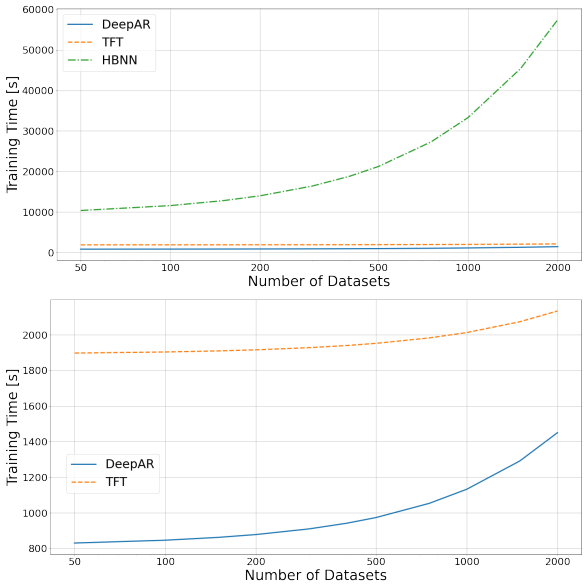

Figure 4: Training time curve as a function of the number of datasets. On the top, all three models are plotted. On the bottom, we plot only DeepAR and TFT to show the differences between the two models. The training time increases when we increase L, but TFT grows slower when the number of datasets increases, compared to the other methods. The x-axis is in a logarithmic scale.

The Bayesian layer is much slower to train because of the reparametrization trick applied in each epoch, and it scales much less well compared to the state-of-the-art methods, which have been designed to be jointly trained with multiple time series. Moreover, the libraries used to implement HBNN and state-of-the-art methods are different, with Keras much slower than PyTorch. If we relate the training time to the accuracy, we can see that clustering approaches allow us to reach the same performance with a much lower number of training datasets. For example, to achieve the same MSE, a random selection requires 1,000 datasets, compared to 200 or 300 when we apply SOM (see Figure 1), with a reduction of the computational cost, confirming the advantages of clustering methods compared to a random selection.

We also measure the inference time of HBNN, DeepAR and TFT to predict one sample. All the DL models are fast, with 0.1 ms required for prediction using the HBNN. DeepAR takes 2.16 ms, while TFT requires 20.10 ms. The differences depend on the complexity of the architecture. Although the inference time among the three models is significantly different, with the HBNN 200 times faster than the TFT, it is on the order of ms, negligible compared to the 10 minutes ahead forecast.

Inference time is also independent of the clustering approach because once the model is trained, the time depends only on the number of parameters and the complexity of the network. This runtime performance analysis allows us to determine that, with similar performance in terms of point estimate accuracy and the considerations on the confidence intervals computed from the predicting probability distributions, DeepAR is preferable to the TFT because both the training time and the inference time are lower.

Furthermore, we measure the fitting and prediction time of the four clustering approaches. Each technique includes a different training procedure. PCA includes reducing the feature space and computing the K-Means on the top of the latent space. Similarly, VRAE consists of the joint training of the encoder and decoder, the encoder is then used to map the time series to a latent space, and K-Means is applied on top of it. SOM requires the training of the neurons of the map that will be used for the prediction. Finally, the DTW is comparable to the K-Means algorithms, with a different distance metric which is computationally more expensive. We list the time performance in Table 3. SOM is the fastest method under both training and inference time, while the training phase for DTW is computationally expensive. We can conclude that SOM offers the best trade-off between accuracy and runtime performance.

Clustering Method	Fitting Time [s]	Inference Time [s]
PCA	158.6	0.965
VRAE	185.6	1.744
SOM	**7.8**	**0.177**
DTW	2010.7	0.682

Table 3: Fitting and inference time of the clustering approaches. In bold, the best fitting and inference time.

5 Conclusion

Resource demand prediction is a challenging task for cloud computing managers, and when computational resources are limited, an accurate reduction of training data is necessary.

This paper analyses the accuracy of DL probabilistic models in predicting future demand distribution of CPU and memory in the latest Google Cloud Trace, comparing state-of-the-art models with BNNs that capture prediction uncertainty.

Clustering algorithms are applied to aggregate datasets with similar patterns and select good representatives among available datasets, reducing training time compared to random selection. Results show that clustering-based selection achieves the same accuracy with fewer data and time series clustering methods such as DTW and SOMs are more effective. Confidence intervals are computed from the output distribution to relate prediction confidence to the rate of correctly predicted requests. State-of-the-art methods achieve better point estimate accuracy but are often overconfident.

At the same time, HBNN can quantify epistemic uncertainty, providing a trade-off between accuracy and successful resource requests at the cost of allocating more resources. The models' runtime performance is analysed for practical deployment in real-world scenarios, with the DeepAR method trained with datasets selected with SOMs offering the best trade-off between cost and accuracy.

In future work, we will extend the clustering algorithms for dealing with time series and extend these to a multi-step ahead prediction. Finally, the aim is to use the predictions in

resource allocation algorithms to preconfigure the machine in cloud cells in advance, providing a high QoS level.

Acknowledgments

This publication has emanated from research supported in part by Science Foundation Ireland under Grant Nos. 18/CRT/6223 and 12/RC/2289-P2, by the EU Horizon projects Glaciation (GA No. 101070141) and TAILOR (GA No. 952215), and the Google Cloud Research Credits program with the award GCP203677602.

References

[Achar, 2022] Sandesh Achar. Cloud computing: Toward sustainable processes and better environmental impact. *Journal of Computer Hardware Engineering (JCHE)*, 1(1), 2022.

[Ali and Kecskemeti, 2023] Shallaw Mohammed Ali and Gabor Kecskemeti. Sequal: an unsupervised feature selection method for cloud workload traces. *The Journal of Supercomputing*, pages 1–19, 2023.

[Baldan *et al.*, 2016] Francisco J Baldan, Sergio Ramirez-Gallego, Christoph Bergmeir, Francisco Herrera, and Jose M Benitez. A forecasting methodology for workload forecasting in cloud systems. *IEEE Transactions on Cloud Computing*, 6(4):929–941, 2016.

[Barreto, 2007] Guilherme A Barreto. Time series prediction with the self-organizing map: A review. *Perspectives of neural-symbolic integration*, pages 135–158, 2007.

[Bellman, 1966] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[Bi *et al.*, 2021] Jing Bi, Shuang Li, Haitao Yuan, and MengChu Zhou. Integrated deep learning method for workload and resource prediction in cloud systems. *Neurocomputing*, 424:35–48, 2021.

[Blundell *et al.*, 2015] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[Calheiros *et al.*, 2014] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using ARIMA model and its impact on cloud applications' QoS. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.

[Cherif *et al.*, 2011] Aymen Cherif, Hubert Cardot, and Romuald Boné. Som time series clustering and prediction with recurrent neural networks. *Neurocomputing*, 74(11):1936–1944, 2011.

[Di *et al.*, 2012] Sheng Di, Derrick Kondo, and Walfredo 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*, pages 1–11. IEEE, 2012.

[Dinda and O'Hallaron, 2000] Peter A Dinda and David R O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4):265–280, 2000.

[Fabius and Van Amersfoort, 2014] Otto Fabius and Joost R Van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

[Gao *et al.*, 2020] Jiechao Gao, Haoyu Wang, and Haiying Shen. Machine learning based workload prediction in cloud computing. In *2020 29th international conference on computer communications and networks (ICCCN)*, pages 1–9. IEEE, 2020.

[Herbst *et al.*, 2014] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. *Concurrency and computation: practice and experience*, 26(12):2053–2078, 2014.

[Huang *et al.*, 2021] Lingxiao Huang, K Sudhir, and Nisheeth Vishnoi. Coresets for time series clustering. *Advances in Neural Information Processing Systems*, 34:22849–22862, 2021.

[Hüllermeier and Waegeman, 2021] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[Janardhanan and Barrett, 2017] Deepak Janardhanan and Enda 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)*, pages 55–60, 2017.

[Kohonen, 1990] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[Kristiadi *et al.*, 2020] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International conference on machine learning*, pages 5436–5446. PMLR, 2020.

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

[Lakhina *et al.*, 2004] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM computer communication review*, 34(4):219–230, 2004.

[Li, 2019] Hailin Li. Multivariate time series clustering based on common principal component analysis. *Neurocomputing*, 349:239–247, 2019.

[Lim *et al.*, 2021] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[Malarya *et al.*, 2021] Ankita Malarya, Karthik Ragunathan, Mani Bharath Kamaraj, and Vignesh Vijayarajan. Emerging trends demand forecast using dynamic time warping.

In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 402–407. IEEE, 2021.

[Mao *et al.*, 2010] Ming Mao, Jie Li, and Marty Humphrey. Cloud auto-scaling with deadline and budget constraints. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 41–48. IEEE, 2010.

[Markets and Markets, 2022] Markets and Markets. Cloud computing market by service model, by deployment model, organization size, vertical and region - global forecast to 2027, 2022.

[Minarolli *et al.*, 2017] Dorian Minarolli, Artan Mazrekaj, and Bernd Freisleben. Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. *Journal of Cloud Computing*, 6(1):1–18, 2017.

[Mohammadi Bahram Abadi *et al.*, 2018] Reza Mohammadi Bahram Abadi, Amir Masoud Rahmani, and Sasan Hossein Alizadeh. Self-adaptive architecture for virtual machines consolidation based on probabilistic model evaluation of data centers in cloud computing. *Cluster Computing*, 21:1711–1733, 2018.

[Parsons, 2014] Van L Parsons. Stratified sampling. *Wiley StatsRef: Statistics Reference Online*, pages 1–11, 2014.

[Pearson, 1901] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[Pereira and Silveira, 2018] Joao Pereira and Margarida Silveira. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1275–1282. IEEE, 2018.

[Petitjean *et al.*, 2011] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3):678–693, 2011.

[Petitjean *et al.*, 2014] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *2014 IEEE international conference on data mining*, pages 470–479. IEEE, 2014.

[Prasetyo *et al.*, 2021] Joko Prasetyo, Noor Akhmad Setiawan, and Teguh Bharata Adji. Clustering based oil production rate forecasting using dynamic time warping with univariate time series data. In *2021 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, pages 204–208. IEEE, 2021.

[Rojat *et al.*, 2021] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. Explainable artificial intelligence (xai) on timeseries data: A survey. *arXiv preprint arXiv:2104.00950*, 2021.

[Rossi *et al.*, 2022] Andrea Rossi, Andrea Visentin, Steven Prestwich, and Kenneth N Brown. Bayesian uncertainty modelling for cloud workload prediction. In *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*, pages 19–29. IEEE, 2022.

[Rousseeuw, 1987] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

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

[Sarlin and Eklund, 2011] Peter Sarlin and Tomas Eklund. Fuzzy clustering of the self-organizing map: some applications on financial time series. In *International Workshop on Self-Organizing Maps*, pages 40–50. Springer, 2011.

[Song *et al.*, 2018] Binbin Song, Yao Yu, Yu Zhou, Ziqiang Wang, and Sidan Du. Host load prediction with long short-term memory in cloud computing. *The Journal of Supercomputing*, 74(12):6554–6568, 2018.

[Thorndike, 1953] Robert L Thorndike. Who belongs in the family. In *Psychometrika*. Citeseer, 1953.

[Varatharajan *et al.*, 2018] Ramachandran Varatharajan, Gunasekaran Manogaran, Malarvizhi Kumar Priyan, and Revathi Sundarasekar. Wearable sensor devices for early detection of alzheimer disease using dynamic time warping algorithm. *Cluster Computing*, 21:681–690, 2018.

[Watson *et al.*, 2018] Jane Watson, Noleine Fitzallen, Jill Fielding-Wells, and Sandra Madden. The practice of statistics. *International handbook of research in statistics education*, pages 105–137, 2018.

[Wibbeke *et al.*, 2022] Jelke Wibbeke, Payam Teimourzadeh Baboli, and Sebastian Rohjans. Optimal data reduction of training data in machine learning-based modelling: A multidimensional bin packing approach. *Energies*, 15(9):3092, 2022.

[Wilkes, 2020] John Wilkes. 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.

[Yu *et al.*, 2016] Yongjia Yu, Vasu Jindal, I-Ling Yen, and Farokh Bastani. Integrating clustering and learning for improved workload prediction in the cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 876–879. IEEE, 2016.

[Zheng *et al.*, 2021] Zhong Zheng, Long Wang, Luoxiao Yang, and Zijun Zhang. Generative probabilistic wind speed forecasting: A variational recurrent autoencoder based method. *IEEE Transactions on Power Systems*, 37(2):1386–1398, 2021.