

Title	Modelling uncertainty in cloud workload forecasting: a Hybrid Bayesian Neural Network approach
Authors	Rossi, Andrea
Publication date	2024
Original Citation	Rossi, A. 2024. Modelling uncertainty in cloud workload forecasting: a Hybrid Bayesian Neural Network approach. PhD Thesis, University College Cork.
Type of publication	Doctoral thesis
Rights	© 2024, Andrea Rossi. - https://creativecommons.org/licenses/by/4.0/
Download date	2026-10-23 12:07:22
Item downloaded from	https://hdl.handle.net/10468/16964

Modelling Uncertainty in Cloud Workload Forecasting: A Hybrid Bayesian Neural Network Approach

Andrea Rossi

MSC

118106018

**Thesis submitted for the degree of
Doctor of Philosophy**



NATIONAL UNIVERSITY OF IRELAND, CORK

COLLEGE OF SCIENCE, ENGINEERING AND FOOD SCIENCE

SCHOOL OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

CENTRE FOR RESEARCH TRAINING IN ARTIFICIAL INTELLIGENCE

June 2024

Head of School: Professor Utz Roedig

Supervisors: Professor Kenneth N. Brown
Dr. Steve Prestwich
Dr. Andrea Visentin

Contents

List of Figures	iii
List of Tables	viii
Acknowledgements	xiii
Abstract	xvi
1 Introduction	1
1.1 Motivation and Background	1
1.2 Research Questions	5
1.3 Contributions	7
1.4 Publications	10
1.5 Outline of the Dissertation	11
2 Workload Prediction in Cloud Computing	13
2.1 Predictive Models	16
2.1.1 Statistical Methods	16
2.1.2 Machine Learning Methods	20
2.1.3 Deep Learning Methods	22
2.2 Uncertainty in Cloud Workload Forecasting	26
2.2.1 Epistemic and Aleatoric Uncertainty	27
2.2.2 Forecasting Uncertainty with Deep Learning	29
2.2.3 Bayesian Inference	31
2.2.4 Probabilistic Forecasting Models	36
2.3 On Times Series Clustering	40
2.4 Conclusion	45
3 Cloud Workload Datasets Preprocessing	47
3.1 Cloud Workload Traces	48
3.2 Preprocessing	50
3.3 Preprocessed Datasets	53
3.3.1 Google Cloud Trace 2011	54
3.3.2 Google Cloud Trace 2019	55
3.3.3 Alibaba Cluster Trace 2018	56
3.3.4 Alibaba Cluster Trace 2020	58
3.4 Analysis	60
3.4.1 Stationarity Test	60
3.4.2 Pearson Correlation Coefficient	61
3.4.3 Data Distribution	62
3.5 Conclusion	62
4 Hybrid Bayesian Neural Network for Modelling Uncertainty in Cloud Workload Forecasting	66
4.1 Hybrid Bayesian Neural Network	67
4.2 Experiments	72
4.2.1 Methodology	72
4.2.2 Forecasting Models and Optimization	73

4.2.3	Metrics	75
4.3	Results and Discussion	79
4.3.1	Point Estimate Accuracy	79
4.3.2	Confidence Interval Evaluation	79
4.3.3	Runtime Performance Analysis	86
4.4	Conclusion	88
5	Bivariate HBNN and Transfer Learning for Uncertainty-Aware Cloud Workload Forecasting	90
5.1	Towards Bivariate Cloud Workload Forecasting	91
5.1.1	Methodology	92
5.1.2	Results	94
5.2	Towards Transfer Learning for Cloud Workload Forecasting . . .	102
5.2.1	On Transfer Learning for Cloud Computing	104
5.2.2	Methodology	105
5.2.3	Results	107
5.3	Conclusion	111
6	Clustering-based Numerosity Reduction	115
6.1	Numerosity Reduction for Times Series Data	116
6.2	Experiments	119
6.2.1	Methodology	119
6.2.2	Forecasting Models and Optimization	121
6.2.3	Clustering methods	122
6.3	Results and Discussion	123
6.3.1	Point Estimate Accuracy	123
6.3.2	Confidence Interval Evaluation	126
6.3.3	Runtime Performance Analysis	128
6.4	Conclusion	133
7	Conclusions	135
7.1	Future Work	138
A	Preprocessed Datasets	A1
A.1	Cluster-level Datasets from Google Cloud Trace 2019	A1
A.2	Memory distribution of cluster-level preprocessed datasets . . .	A6
B	Bivariate HBNN and Transfer Learning Results	B8
B.1	Additional Results on Bivariate HBNN	B8
B.2	Additional Results on Transfer Learning for HBNN	B10
C	Clustering-based Numerosity Reduction Results	C14
C.1	Point Estimate Accuracy	C14
C.2	Confidende Interval Evaluation	C15

List of Figures

1.1	Resource management scheme in cloud computing. This dissertation is focused on the forecasting part (highlighted in the red box).	3
2.1	Variational Recurrent Auto-Encoder architecture [FVA14]. The network is trained end-to-end to minimise the reconstruction error. After the training is completed, the encoder maps the time series data to distributions in the latent space.	42
2.2	Self-Organising Map architecture [SFSHTM20]. The network is trained with competitive learning, where each neuron competes with each other to become the winner neuron, i.e. the neuron that is activated in response to certain input.	43
3.1	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2011.	55
3.2	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell A.	57
3.3	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell B.	57
3.4	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the machine with ID 000020732905 from cluster cell A.	58
3.5	Average CPU and memory (RAM) workload of the preprocessed Alibaba Cluster Trace 2018.	59
3.6	Average CPU and memory workload of the preprocessed Alibaba Cluster Trace 2020.	59
3.7	Average GPU and GPU memory workload of the preprocessed Alibaba Cluster Trace 2020.	60
3.8	Histogram of the normalised processing units of the compute cells A to D from the preprocessed GC19.	63
3.9	Histogram of the normalised processing units of the compute cells E to H from the preprocessed GC19.	64
3.10	Histogram of the normalised processing units of the compute cells from the preprocessed GC11, AC20 and AC18.	65
4.1	Graphical representation of the DL architecture proposed. For each of the three models (LSTM, LSTMD and HBNN), we apply minimal changes to the layer to appreciate the impact of each component in terms of accuracy and uncertainty modelling capabilities.	71
4.2	Prediction target of our models. Each square represents a 5-minute interval of the cloud workload. In this dissertation we forecast 10 minutes ahead.	73

4.3	Visual representations of the metrics used in the confidence interval evaluation. The TPR is given by the sum of the black dots, while the SR is the percentage of points below the black line. .	77
4.4	SR vs confidence level for CPU workload. The points in the plot are the result of the average of the 8 datasets from GC19. . . .	83
4.5	SR vs confidence level for memory workload. The points in the plot are the result of the average of the 8 datasets from GC19. .	83
4.6	SR vs TPR for CPU workload. The points in the plot are the result of the average of the 8 datasets from GC19.	84
4.7	SR vs OP for CPU workload. The points in the plot are the result of the average of the 8 datasets from GC19.	85
4.8	SR vs TPR for memory workload. The points in the plot are the result of the average of the 8 datasets from GC19.	85
4.9	SR vs OP for memory workload. The points in the plot are the result of the average of the 8 datasets from GC19.	86
4.10	Actual CPU workload vs predicted standard deviation for a 24-hour window for the dataset A from GC19	87
4.11	Actual CPU demand vs predicted standard deviation for a 24-hour window for the dataset H from GC19	87
4.12	Standard deviation of the HBNN and LSTMD models as a function of the prediction length (forecast horizon). The uncertainty increases as the prediction length extends.	88
5.1	LSTM, LSTMD and HBNN architectures used for experiments with bivariate predictions. In this case, we increase the number of convolution and dense layers to better deal with multiple time series.	93
5.2	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.	98
5.3	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.	99
5.4	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.	100
5.5	An illustrative example of four training scenarios for the second part of the experiments. In this example, we consider dataset number 1 by GC19 as the target domain (and we use it as the FT dataset in All FT and All-but-one cases). Depending on the scenario, The first 80% of dataset number 1 is included in the training set and/or used for fine-tuning. The target domain is the final 20% of the dataset to avoid leakage. This approach is repeated for all the 12 datasets.	107

5.6	Total predicted processing units for <i>All</i> , <i>All FT</i> , <i>All-but-one</i> and <i>All-but-one FT</i> . The closer the curve to the top left corner, the better the model.	111
5.7	Target confidence level versus SR for processing units prediction for <i>All</i> , <i>All FT</i> , <i>All-but-one</i> and <i>All-but-one FT</i> . The closer the plot is to the line $y = x$, the more accurate the model under this metric.	112
5.8	Target service level versus SR for processing units prediction for <i>TL-GC19</i> . The closer the plot is to the line $y = x$, the more accurate the model under this metric.	113
5.9	Standard deviation variability for across inference on the test set <i>All</i> and <i>All-but-one</i> . Uncertainty for <i>All-but-one</i> is higher for predictions on unseen data.	114
6.1	Selection of the training datasets: random selection vs clustering approach. Using random selection, traces are chosen randomly from the pool of datasets. In the clustering approach, traces are first grouped into clusters, and then sampling is applied to each cluster.	121
6.2	Illustration for elbow curve applied on the DTW K-means clustering algorithm. The optimal number of clusters corresponds to the elbow of the curve, which is 4 in this example.	123
6.3	MSE comparison of DeepAR and TFT for the memory demand. DTW and SOM are the best clustering-based selection approaches, allowing the models to achieve the same MSE with fewer training datasets than 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.	125
6.4	MSE comparison of HBNN for the memory demand. PCA is the best clustering-based selection approach and allows the HBNN 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.	126
6.5	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.	128
6.6	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.	130

6.7	Training time curve as a function of the number of datasets. On the left, all three models are plotted. On the right, 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. . . .	131
A.1	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell A.	A1
A.2	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell B.	A2
A.3	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell C.	A2
A.4	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell D.	A3
A.5	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell E.	A3
A.6	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell F.	A4
A.7	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell G.	A4
A.8	Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell H.	A5
A.9	Histogram of the normalised memory units of the compute cells A to D from the preprocessed GC19.	A6
A.10	Histogram of the normalised memory units of the compute cells E to H from the preprocessed GC19.	A7
A.11	Histogram of the normalised memory units of the compute cells from the preprocessed GC11, AC20 and AC18.	A7
B.1	Total predicted memory units for S-U, M-U and M-B models. The closer the curve to the top left corner, the better the model. . .	B8
B.2	Total predicted memory 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.	B9
B.3	Total overprediction for processing units for S-U, M-U and M-B models. The closer the curve to the top left corner, the better the model.	B9
B.4	Total overprediction for 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.	B10
B.5	Total predicted processing units for <i>GC19</i> , <i>GC19 FT</i> , <i>GC19-but-one</i> and <i>GC19-but-one FT</i> . The closer the curve to the top left corner, the better the model. <i>GC19-but-one FT</i> performs very poorly.	B12

B.6	Total overprediction of processing units for <i>GC19</i> , <i>GC19 FT</i> , <i>GC19-but-one</i> and <i>GC19-but-one FT</i> . The closer the curve to the top left corner, the better the model. <i>GC19-but-one FT</i> performs very poorly.	B13
B.7	Standard deviation variability for across inference on the test set <i>GC19</i> and <i>GC19-but-one</i> . Uncertainty for <i>GC19-but-one</i> is higher for predictions on unseen data.	B13
C.1	MSE comparison of DeepAR and TFT for the processing units 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.	C14
C.2	MSE comparison of HBNN for the processing units 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.	C15
C.3	Processing unit 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.	C16
C.4	Memory SR comparison of DeepAR and TFT 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.	C16
C.5	Memory SR comparison of 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.	C17
C.6	Processing unit 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.	C17
C.7	Processing unit OP comparison of HBNN. 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.	C18

List of Tables

2.1	Summary of the uncertainty quantification methods discussed in this section and their use in cloud workload forecasting domain.	31
3.1	Summary of workload traces in the literature used for workload forecasting and list of related publications. In bold the traces used in this dissertation.	51
3.2	Example of general workload dataset before preprocessing. . .	52
3.3	Example of the resulting dataset after preprocessing. For each time interval (of 5 minutes each), we have the measurement of the average resources used in the period.	54
3.4	Preprocessed traces summary. For each trace, we identify the number of timepoints and the type of resources included in each dataset.	64
4.1	MSE and MAE comparison of CPU and memory demand forecasting for LSTM, HBNN, LSTMD, ARIMA and GARCH.	79
4.2	Confidence interval evaluation for CPU workload of the HBNN, LSTMD and LSTM with fixed thresholds. For a given confidence level (95%, 97% or 99%), the best result for each confidence level from the 4 models is in bold; otherwise, the best of a pairwise comparison is in italics. The threshold is expressed as a percentage of the prediction.	80
4.3	Confidence interval evaluation for memory workload of the HBNN, LSTMD and LSTM with fixed thresholds. For a given confidence level (95%, 97% or 99%), the best result for each confidence level from the 4 models is in bold; otherwise, the best of a pairwise comparison is in italics. The threshold is expressed as a percentage of the prediction.	81
4.4	Confidence interval evaluation for CPU workload of the HBNN, LSTMD and LSTM with learned thresholds. For a given confidence level (95%, 97% or 99%), the best result from the 4 models is in bold.	82
4.5	Confidence interval evaluation for memory workload of the HBNN, LSTMD and LSTM with learned thresholds. For a given confidence level (95%, 97% or 99%), the best result from the 4 models is in bold.	82
4.6	Runtime performance results of the LSTM, LSTMD and HBNN models. We report the inference time and different training time and fine-tuning time performances, varying the size of the training data.	89
5.1	Average MSE/MAE comparison for processing units and memory prediction for models trained with a single dataset. In bold, the best model.	94

5.2	Average MSE/MAE comparison for processing units and memory prediction. In bold, the best model.	96
5.3	Results for the service level metrics. We run a pairwise comparison of HBNN and LSTMD with the corresponding LSTM baseline. Confidence intervals for LSTMs are tailored to the success rates achieved by HBNN and LSTMD, respectively. In bold we report the best model for each confidence level.	97
5.4	Average MSE and MAE comparison for resource prediction accuracy. The MSE and MAE are computed between the SR achieved by the models and the confidence level. In bold, the best model.	98
5.5	Runtime performance analysis. In bold is the best model for each subgroup. The training time is computed by training the model with 20, 40, 60 and 80% of the training set, respectively. The fine-tuning of the network is done every 6, 12, 18 and 24 steps, corresponding to 30, 60, 90 and 120 minutes, respectively. The inference time is the time to predict one sample.	101
5.6	Accuracy results for M-B-HBNN trained on scenarios <i>All</i> , <i>All FT</i> , <i>All-but-one</i> and <i>All-but-one FT</i> . In bold, the best performance.	108
5.7	Accuracy results for M-B-HBNN trained on scenario <i>TL-GC19</i> . In bold, the best performance.	109
5.8	Accuracy results for M-B-HBNN trained on scenario <i>TL-GC19vsOther</i>	109
5.9	Service level metrics results of M-B-HBNN for scenarios <i>All</i> , <i>All FT</i> , <i>All-but-one</i> and <i>All-but-one FT</i> . In bold, the model with the best SR for each confidence level.	110
5.10	Service level metrics results of M-B-HBNN for scenario <i>TL-GC19</i> . In bold, the model with the best SR for each confidence level.	110
6.1	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.	124
6.2	CPU and Memory demand prediction statistics (Success Rate, Overprediction and Underprediction in percentage) of HBNN, DeepAR and TFT with 500, 1000 and 2000 training datasets at 95% confidence. In bold, for each model, the best clustering technique.	129
6.3	Time saving for all the models trained with the best clustering approach w.r.t. a random selection of the training datasets, with the baseline trained using 2000, 1500 and 1000 datasets. The training size of the clustering is selected as the minimum size required to achieve an MSE less or equal to the MSE achieved by the baseline.	132
6.4	Fitting and inference time of the clustering approaches. In bold, the best fitting and inference time.	133

B.1	Processing Units and Memory confidence interval statistics of the models for the transfer learning experiments on Google Cloud 2019 traces and tested on Google Cloud 2011 and Alibaba. In bold, the model with the best SR for each confidence level. . . .	B11
B.2	Average MSE and MAE for scenarios <i>All</i> , <i>All FT</i> , <i>All-but-one</i> and <i>All-but-one FT</i> . The MSE and MAE are computed between the SR achieved by the models and the confidence level. In bold, the model overall.	B11
B.3	Average MSE and MAE comparison for scenario <i>GC19</i> . The MSE and MAE are computed between the SR achieved by the models and the confidence level. In bold, the model overall.	B11
B.4	Average MSE and MAE comparison for resource prediction accuracy for the transfer learning experiments on Google Cloud 2019 traces and tested on Google Cloud 2011 and Alibaba. The MSE and MAE computed between the SR achieved by the models and the confidence level. In bold, the model overall.	B12

I, Andrea Rossi, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Andrea Rossi

To my family, who always believed in me
and supported me with unconditional love

Acknowledgements

If I look back to October 2019, when I started this journey, I never could have predicted the twists and turns that awaited me. The sequence of events that came up in recent years included, first of all, the COVID-19 pandemic. This unexpected turbulence ultimately proved invaluable, shaping my research and fostering a deep sense of resilience that I will carry with me throughout my career. And there are many people whom I need to thank for this achievement during my past few years at UCC.

I would like to thank my supervisors, Ken, Steve and Andrea, without whom this thesis wouldn't exist. Thanks for your guidance and inspiration over this journey. It has been a pleasure walking with you in these years. While I only started this journey with Ken and Steve, Andrea joined in a very delicate phase of the PhD during the COVID-19 pandemic, when the enforced social distance made us less human, and the certainties and ambitions began to crumble. Andrea, however, became a guiding light. His support and insightful guidance were instrumental in reawakening my productivity, changing my mindset and reminding me of the ambitious goals I set out to achieve.

A special thanks go to Diego, my friend, flatmate, and colleague for these past years. Living together wasn't just convenient; Diego adopted me into his home, creating a fantastic environment for friendship and productivity. He gave me strong support and guidance that helped me deal with my research challenges at home and outside. Collaborating with him was a pleasure, filled with stimulating and insightful discussions and tips. His companionship and curiosity enriched my personal and academic journey.

Two essential women in this journey deserve special mention: Sofia and Roberta. Sofia is a brilliant researcher who has been a lighthouse for me throughout my PhD. Her support, especially during some challenging moments, was invaluable. Her insightful advice has not only been fundamental to my research but has also been a fantastic learning experience. More importantly, she's also a dear friend I know I can always rely on. Roberta, on the other hand, is my partner-in-crime. Her immense patience has seen her through countless adventures by my side. Her loyal friendship and ability to cheer me up are a treasure. Having Sofia and Roberta in my life has been life-changing, and their friendship has undeniably shaped who I am today.

My PhD journey wouldn't have been the same without the time spent with my fellow PhD companions at UCC, with whom we shared common feelings. To Marco, Buvana, Victor, Andrea, Padraigh, Marlon, Vedat, Jheisson and Ciaran, a huge thank you! I am grateful for the strong network we built and the pleasure of having walked with you on this journey.

Thanks to the Ghettohouse, Andrea, Pasquale, Uday, and Samuele. Sharing a home during the pandemic wasn't always easy, but your presence and support made the difference. Having you all by my side during such a challenging time

was invaluable. I'd also like to thank Davide and Enrico, special friends, for cheering me up throughout the pandemic and beyond.

I would like to express my sincere gratitude to Marco, Andrea, Federico, Federica, Martina and Mohamed for truly making my time in Cork and beyond unforgettable. They filled my days with countless adventures and created a warm and supportive environment, making me feel like I had a family abroad. I would also like to thank Chiara, Emmanuele, Enza, Antonio, Cheyenne, Juanito, and Caoimhe. Though their paths have taken them elsewhere, they hold a special place in my heart.

Thanks to Los Sicanos crew: Alessandra, Fernando, Feffo, Luiza, Teresa, and Marco. We may not be on the same path anymore, but those regular Los Sicanos meet-ups keep the team alive.

Dedico questo lavoro alla mia famiglia, la mia colonna portante che mi ha visto partire lontano (ma non troppo) per nuovi orizzonti e traguardi, a mia mamma Marinella che non ha mai smesso di sperare nel mio ritorno a Rovigo e che mi sostiene con amore incondizionato; alle mie sorelle Claudia, Laura e Ilaria che sono sempre al mio fianco, nei momenti felici e in quelli bui; ai miei nipoti, Francesco e Alessandro, con cui sono cresciuto assieme e che sono contento di aver visto maturare molto, e Luca, la mia gioia più grande; ai miei cognati, Samuele, Antonio e Luca, persone speciali e sagge con cui condivido momenti indimenticabili; agli esseri più belli del mondo, Axel e Gas, che mi riempite di feste ogni volta che torno. Un pensiero speciale va alla zia Antonietta, che ci ha lasciato troppo presto e non ha potuto vedere la fine di questo lavoro. E con un ricordo immenso e pieno d'amore, dedico questo traguardo a mio padre, che non c'è più, ma che vive per sempre nel mio cuore. La sua forza, la sua saggezza e il suo esempio mi guidano ogni giorno. A tutti loro, che sono la mia forza e la mia ispirazione, dedico questo traguardo con infinita gratitudine.

Ai miei amici in Italia, che non posso vedere e sentire come vorrei, ma che sono rimasti e resteranno importanti: Mariangela, Eleonora, Pier Paolo, Clara, Camilla, Filippo, Mattia, Nicola, Sabrina, Stefania, Daniele, Matteo, Alessandro. E al mio gruppo di amici ingegneri che non ha smesso di incontrarsi neanche dopo esserci distribuiti in giro per il mondo: Federico, Giulia, Ilaria, Francesco, Jacopo e Luca.

Finally, I am thankful to all members and staff of the Centre for Research Training in Artificial Intelligence, the Insight Centre, and the Computer Science department at UCC, especially Professor Barry O'Sullivan and Dr Derek Bridge, for leading this vibrant community across universities in Ireland, a format that I found very beneficial for networking and collaborating and also that gave me the opportunity of consolidating my skills with a fantastic industry experience in a great team plenty of interesting research projects and where I felt immediately at home; Peter J. MacHale and Janet Choi for their technical and administrative support over the years. This work has also been supported in part by Science Foundation Ireland under Grant Nos. 18/CRT/6223, 16/RC/3918 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.

Andrea Rossi
Cork, June 2024

Abstract

The exponential growth of cloud services utilisation has rendered robust and accurate workload forecasting paramount. Predicting future computational demands is essential for efficient resource management, cost optimisation, and guaranteeing service level agreement (SLA). At the same time, the increase in cloud resource demand poses new challenges related to data centres' carbon emissions and their environmental impact. However, these predictions are characterised by inherent uncertainty stemming from both fundamental limitations in data and the dynamic nature of cloud usage.

This dissertation delves into workload forecasting in cloud computing. Our primary focus is quantifying predictions' inherent uncertainty, an important but often overlooked aspect. Traditionally, forecasting models provide point estimates, leaving cloud providers blind to the potential range of future demands. This lack of uncertainty awareness can lead to suboptimal resource allocation, unnecessary costs, and SLA violations.

In particular, our work focuses on quantifying the uncertainty of different natures. The epistemic uncertainty arises from limitations in our knowledge or model, while the aleatoric uncertainty stems from the variability in the data itself, reflecting the inherent randomness of cloud workloads. By capturing both these forms of uncertainty, we gain a deeper understanding of the prediction's confidence interval, empowering informed decision-making.

This work introduces a novel Hybrid Bayesian Neural Network (HBNN) model designed to capture both types of uncertainty in workload forecasting by incorporating a Bayesian layer at the network's end. The HBNN surpasses conventional approaches by estimating the future probability distribution to compute the associated confidence intervals. This quantifiable uncertainty empowers the model for more accurate prediction.

The thesis expands upon the HBNN model by exploring its application in bivariate forecasting, simultaneously predicting processing units and memory demands. This approach proves good accuracy at the cost of more training data required. Furthermore, this work investigates the impact of various factors on forecast accuracy, including training data size, unseen data generalisation, and the potential of transfer learning across different cloud environments.

Furthermore, recognising the potential computational hurdles associated with

Bayesian neural networks, this work introduces a clustering-based preprocessing technique. This technique intelligently selects training data points, significantly reducing computational cost while maintaining forecast accuracy. This development enables the HBNN model to be deployed in cloud environments at scale.

The thesis comprehensively evaluates the HBNN model through extensive experiments on real-world cloud workload datasets. The results demonstrate the model's superior performance in quantifying uncertainty, achieving higher accuracy than baseline models and showcasing improved generalisation on unseen data. By quantifying inherent uncertainty, HBNN empowers cloud providers to manage cloud resources more efficiently, reliably and cost-effectively.

Chapter 1

Introduction

1.1 Motivation and Background

Cloud computing is a paradigm that enables individuals and organizations to access a shared and scalable IT resource pool, including servers, storage, and applications, through a network. Nowadays, cloud services have become an integral part of our daily lives, helping us with data storage, document creation, business management, and online gaming. For example, individuals use services like Google Drive or Dropbox to store and back up their photos, documents, and important files. This ensures that their data is safe and allows easy access from any device, anywhere in the world. Additionally, cloud computing has played a crucial role in driving digital trends such as mobile computing, the Internet of Things, big data, and artificial intelligence (AI), accelerating industry development and digital transformation [SS20].

Businesses benefit significantly from cloud computing through cost savings, high performance, and scalability. A company like Netflix, for instance, leverages cloud services to manage vast amounts of data and deliver content seamlessly to millions of users worldwide, scaling up during peak times without needing to invest in physical servers. Small businesses that previously needed to invest heavily in IT infrastructure can now use services like Amazon Web Services or AWS to host their applications and store data without upfront costs, paying only for the resources they use. Cost savings, high performance, and scalability are just some of the benefits that cloud computing offers to businesses and companies. Its adoption can eliminate the need for expensive hardware and software, and avoid the costs of onsite data centres (DCs). Cloud com-

puting services can be easily accessed on demand, which enables businesses to scale up or down as they need. This flexibility and versatility make cloud computing an excellent choice for businesses of all sizes, enabling them to achieve their goals in a more efficient and effective manner.

The COVID-19 pandemic has led to a surge in demand for cloud computing, particularly as working from home has become more common [AAS⁺21]. Zoom, a video conferencing platform, experienced exponential growth in user numbers [PP⁺20]. To manage this surge, Zoom relied heavily on cloud infrastructure to support millions of concurrent video calls, ensuring performance without degradation. This capability was critical as companies and schools transitioned to remote work and learning almost overnight. The market size for cloud computing was projected to reach USD 545.8 billion in 2022 and is expected to grow even further, reaching USD 1,240.9 billion by 2027 [MM22]. At the same time, the size of deep learning (DL) models is growing exponentially, with applications such as the OpenAI's GPT-3 model comprising 175 billion parameters and a memory footprint of 350GB [BMR⁺20]. For this growth, it is estimated that the energy consumption of data centres (which host cloud services) will grow from 292 TWh in 2016 to 353 TWh in 2030 [KW21], and greenhouse gas emissions will increase over 14% in 2040, compared to a 1-1.6% increase in the 2007-2016 [BE18]. Cloud providers face new challenges in this scenario. They need to balance serving a growing number of diverse customers while also minimizing environmental impact and operating and maintenance costs. Effective resource management is essential for remaining competitive in the market while also being environmentally friendly.

Resource management refers to orchestrating resources and services that the cloud offers and seeks to balance the requirements of users and applications with the efficient and effective utilization of resources [MNH⁺15]. The size of modern DCs, the diversity of resource types and their interdependencies, the erratic and unpredictable workload, and the differing objectives of various actors within a cloud ecosystem all pose significant difficulties with managing resources in cloud computing [JS15]. Specific challenges include ensuring energy efficiency, minimizing service level agreement (SLA) violations, balancing loads, managing network loads, maximizing profits, dealing with hybrid clouds, and handling mobile cloud computing. Workload management is a crucial component of resource management. To enhance the performance and quality of service (QoS) of cloud systems, modelling, analysis and forecasting of the workload is required.

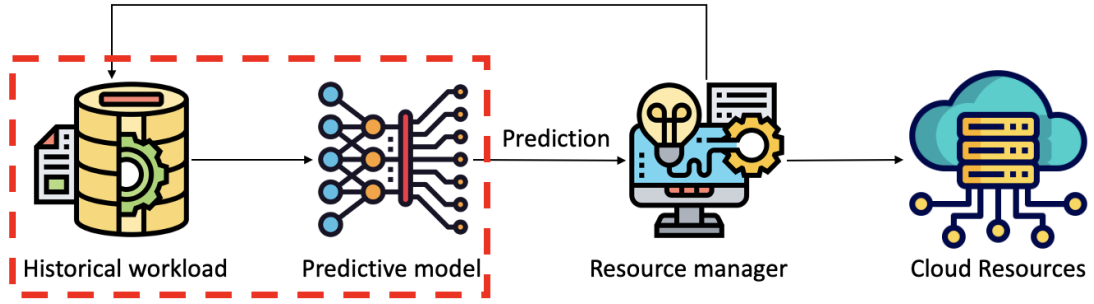


Figure 1.1: Resource management scheme in cloud computing. This dissertation is focused on the forecasting part (highlighted in the red box).

Figure 1.1 shows the resource management scheme in cloud computing. The pipeline starts with the historical workload of the resources used to train a predictive model, which is responsible for providing an accurate prediction of the future demand to the resource manager. The latter is the main actor in this scheme, and it is in charge of the orchestration of the resource in a way that the system requirements, such as the QoS levels, are met. This process involves several interconnected steps to ensure efficient resource utilization: data collection, data preprocessing, feature extraction, model training, workload prediction, resource allocation, and monitoring and feedback. For example, a cloud provider like Google Cloud Platform collects data on server utilization, network traffic, and storage usage to monitor and optimize performance. This is followed by cleaning and transforming the collected data, extracting relevant features, training a forecasting model using historical data, and predicting future workloads. Dynamically allocating resources based on these predictions ensures optimal performance and cost-efficiency. Continuously monitoring performance and providing feedback improves the prediction model and resource management strategies. This dissertation focuses on the forecasting part of the scheme, which is highlighted in the picture with a red box. To address the prediction part, several approaches have been suggested that employ machine learning (ML), data mining, and mathematical techniques [MK20]. Most of the work focuses on predicting the future demand for resources, i.e., processing and memory units, improving the accuracy of the models with innovative architectures and training algorithms but limiting the analysis to point predictions, i.e., single value estimates for a specific future time point.

Making a prediction without considering uncertainty can be limiting. While a single-point prediction might be the most likely scenario, it offers no insight into potential variations or risks. For instance, consider a cloud service

provider forecasting resource demands without uncertainty modelling. This could lead to over-provisioning (wasting resources and increasing costs) or under-provisioning (causing service outages and customer dissatisfaction). Uncertainty modelling addresses this by providing a probabilistic range of future demands, allowing for more informed resource allocation decisions that account for the inherent variability in cloud workloads.

In many fields, such as DL, having a reliable representation of uncertainty is crucial, especially in areas where safety is a top priority [HW21]. Uncertainty can be categorized into aleatoric uncertainty (resulting from inherent randomness in the data) and epistemic uncertainty (resulting from a lack of knowledge or information). Without measuring uncertainty, it can be challenging to determine the reliability of a point prediction. For example, in cloud computing, deploying new services without understanding the uncertainty in demand can result in inefficient scaling decisions, affecting the QoS. Researchers have proposed various methods for uncertainty quantification and tested their effectiveness in various applications, such as computer vision (e.g., object detection and self-driving cars), image processing (e.g., image restoration), medical image analysis (e.g., medical image classification and segmentation), Natural Language Processing (NLP) (e.g., text classification, social media texts, and recidivism risk-scoring), bioinformatics, and more [APH⁺21].

While in cloud workload forecasting, there have been studies on predictive models that account for uncertainty, they have mostly relied on probabilistic forecasting principles, including non-parametric methods [MMF17] such as quantile regression [MBK⁺23]. Although quantile regression can estimate the conditional quantiles of the predicted variable, it does not provide information about the uncertainty in the model parameters, i.e., the epistemic uncertainty, and does not exploit prior knowledge of the underlying data distribution. On the other hand, Bayesian inference and the use of Bayesian Neural Networks (BNNs) offer a more principled and robust approach to reasoning about uncertainty, allowing for the incorporation of prior knowledge and the quantification of uncertainty in both model predictions (aleatoric) and parameters (epistemic) [ZYW20], at the cost of a bigger computational overhead.

In this dissertation, we fill some of the gaps in the literature highlighted in this section. While uncertainty is a well-established notion, our work brings new insights by applying a BNN for the first time in this domain. BNNs have been utilised for uncertainty modelling in various applications, but their use

in cloud workload prediction is entirely novel. To address the time complexity associated with BNNs, we propose the use of a DL architecture for predicting future demand for cloud resources that incorporates BNNs principles only in the last part of the network, while considering both the epistemic and aleatoric uncertainty.

We test the robustness and generalisation capabilities of this model when applied to predict the workload of unseen cloud computing DCs and improve the runtime performance with a clustering-based training data selection. For example, uncertainty modelling can help DC operators plan for unexpected spikes in demand due to unforeseen events, thereby avoiding downtime and ensuring a high level of QoS. Our findings open up insights for practitioners to design new and more uncertainty-aware predictive models to improve cloud operations. We believe that incorporating confidence measurements in cloud predictions can lead to better scheduling decisions and resource utilization, especially when aiming for high QoS levels. For instance, by understanding the uncertainty in demand forecasts, a cloud provider can better allocate resources during peak usage times and reduce costs during off-peak periods. Moreover, this approach has been demonstrably effective and introduces a methodology for accounting for uncertainty that can be extended to applications beyond cloud workload prediction.

1.2 Research Questions

This thesis addresses several critical research questions in the domain of cloud workload forecasting, focusing on uncertainty modelling. The primary research questions explored are:

What datasets are suitable for training forecasting models, and what preprocessing steps are necessary?

Identifying the right datasets is essential because the data's quality and relevance directly impact the forecasting models' accuracy and reliability. In cloud computing, workloads can vary significantly, and using comprehensive real-world datasets like Google Cloud Trace and Alibaba Cluster Trace ensures that the models are trained on realistic and diverse scenarios. Proper preprocessing is crucial to eliminate noise and standardize the data, which enhances the model's learning process and performance.

How can both epistemic and aleatoric uncertainty be modelled in cloud workload forecasting?

Modelling uncertainty is important for making reliable predictions. Epistemic uncertainty, which arises from the model's lack of knowledge, and aleatoric uncertainty, which comes from inherent randomness in the data, both need to be addressed to improve the robustness of the models. By incorporating these uncertainties, we can provide more accurate predictions with confidence intervals, which are essential for effective decision-making in cloud resource management. This helps in planning and allocating resources more efficiently, reducing the risk of over-provisioning or under-provisioning.

What methods can enhance the forecasting and generalization capabilities of the models?

Enhancing the generalization capabilities of forecasting models is crucial because it ensures that the models perform well on new, unseen data. This is particularly important in cloud computing, where workloads can change rapidly and unpredictably. Techniques like transfer learning allow models to leverage knowledge from related tasks or domains, improving their adaptability and robustness. This means that the models can provide accurate forecasts even when faced with different types of workloads or changes in usage patterns.

How can the training time of data-intensive models be reduced?

Reducing the training time of models is essential for making them practical and scalable. Training deep learning models can be computationally expensive and time-consuming, which can be a barrier to their widespread adoption. By exploring clustering techniques, we can simplify the training process by grouping similar workloads together. This not only speeds up the training process but also helps the models focus on the most relevant data patterns, improving their efficiency and effectiveness. Faster training times mean that models can be updated more frequently, keeping them accurate and responsive to the latest data.

These research questions are designed to address key challenges in cloud workload forecasting, providing a comprehensive framework for developing more

accurate, reliable, and efficient forecasting models. By answering these questions, we can improve the performance and applicability of forecasting models, ultimately enhancing the management and optimization of cloud resources.

1.3 Contributions

The work in this dissertation is about uncertainty in time series forecasting of future workload in cloud computing. In the following, we list our main contributions to this topic and how they relate to the chapters of this dissertation.

Uniform workload datasets preprocessing

There are numerous available datasets in the literature regarding cloud computing workload decisions, which have been released by major global service providers such as Google and Alibaba. However, some of these resources have complex structures that make them difficult to preprocess. Additionally, the steps involved in preprocessing these datasets are often vague and inconsistent. To address this issue, we have preprocessed various datasets from Google and Alibaba cloud traces and made them available for training our predictive models. We have also provided a detailed and replicable procedure for preparing workload data in this format (Chapter 3).

Uncertainty modelling with the HBNN

We introduce a new model called Hybrid Bayesian Neural Network (HBNN) to forecast the probability distribution of the future demand for computational resources (Chapter 4). To the best of our knowledge, this is the first time that BNNs have been applied to workload forecasting in cloud computing. The output of our model is a Gaussian distribution, with the standard deviation representing the level of uncertainty in our prediction. To evaluate the accuracy of our models, we have developed a comprehensive methodology. Our results show that the HBNN provides the same level of point accuracy as standard networks, but the standard deviation can be used to compute tighter confidence intervals without the need for further post-processing of the prediction. Thanks to the new approach, we can correctly predict a tighter confidence interval of the prediction, i.e. a reduced amount of predicted resources sufficient to satisfy the demand. This suggests that uncertainty would be beneficial when making scheduling decisions.

Bivariate HBNN

We augment the predictive capabilities of the HBNN by accurately forecasting the demand for processing and memory units simultaneously, extending the model from univariate to bivariate (Chapter 5). We also show that in a similar way other methods can be extended to bivariate predictions. The Bayesian layer makes the neural network more robust, but it faces difficulties in convergence, especially when dealing with complex problems like bivariate prediction (predicting two resources at the same time) compared to univariate prediction (predicting only one resource). This can be overcome by training a model with more data, i.e. multiple time series. The results indicate that the HBNN can be extended to bivariate predictions and can provide estimates of uncertainty when trained with enough data.

Transfer Learning for the HBNN

In our research on workload prediction in cloud computing, we explored Transfer learning (TL) for the first time in the literature (Chapter 5). We trained the HBNN using workload data from diverse cloud service providers, assessing whether the model generalises well on unseen data. Our results indicate that with sufficient data, it is possible to pretrain a neural network to predict future demand on unseen workload data, with better performance when the source and the target domains come from the same cloud service provider. This approach when tailored for cloud workload forecasting would allow cloud resource managers to quickly deploy predictive models on new systems without waiting for extensive historical data for a negligible cost in accuracy.

Numerosity reduction for workload prediction

We develop a framework to train predictive models using a clustering-based algorithm to reduce the size of the training set (Chapter 6). This involves grouping workload traces with similar patterns into clusters before selecting data for model training. Results show that by choosing the appropriate training data, we can achieve the same accuracy performance with significantly reduced computational cost for training the HBNN and other advanced predictive models, with a consequent reduction of energy consumption and CO₂ emissions.

Open Source

To facilitate reproducibility and encourage further research, all the code developed for this dissertation is made available as open-source. This includes the access to the preprocessed datasets used for training and evaluating our models. Additionally, we provide the implementations of the HBNN model and the deep learning architectures used in our experiments. These resources can be found in the following GitHub repositories:

- https://github.com/andreareads/cloud_workload_datasets

This repository provides the preprocessed datasets used in the experiments and detailed preprocessing scripts. It ensures that other researchers can easily access and use the same data for replication studies. More details can be found in Chapter 3.

- <https://github.com/andreareads/UncertaintyCloudWorkload>

This repository contains the code and models developed for predicting cloud workload uncertainty, focusing on the integration of the HBNN implementation introduced in this thesis for enhanced uncertainty quantification. More details can be found in Chapter 4.

- <https://github.com/andreareads/TowardsUncertaintyAwareWorkloadPrediction>

This repository includes the research and implementation efforts aimed at developing more uncertainty-aware predictive models for cloud workloads, including the experiments for bivariate and transfer learning. More details can be found in Chapter 5.

- <https://github.com/andreareads/clustered-reduction-cloud-workload>

This repository focuses on the clustering-based reduction techniques applied to cloud workload prediction, aiming to improve runtime performance and model efficiency. More details can be found in Chapter 6.

By sharing these resources, we aim to contribute to the transparency and openness of research in this domain, enabling others to build upon and extend our work.

1.4 Publications

Aspects of the work described in this dissertation have been published as follows:

- 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) (pp. 19-29). IEEE, 2022.
- Andrea Rossi, Diego Carraro, Andrea Visentin, Steven Prestwich and Kenneth N. Brown: *Forecasting Workload in Cloud Computing: Towards Uncertainty-Aware Predictions and Transfer Learning*, accepted for publication in the *Cluster Computing, The Journal of Networks, Software Tools and Applications* by Springer Nature.
- Andrea Rossi, Andrea Visentin, Steven Prestwich and Kenneth N. Brown: *Clustering-based Numerosity Reduction for Cloud Workload Forecasting*, in International Symposium on Algorithmic Aspects of Cloud Computing (pp. 115-132). Cham: Springer Nature Switzerland, 2023 and accepted for presentation at The Second Workshop of Artificial Intelligence for Time Series Analysis (AI4TS): Theory, Algorithms, and Applications (IJCAI'23).

The following publications and submitted manuscripts are derived from work carried out during the PhD study, but not presented in this dissertation:

- Kate Murray, Andrea Rossi, Diego Carraro, and Andrea Visentin: *On Forecasting Cryptocurrency Prices: A Comparison of Machine Learning, Deep Learning, and Ensembles* in *Forecasting*, 5(1), pp. 196-209, 2023.

This paper investigates statistical machine learning and deep learning forecasting methods in the context of cryptocurrency price prediction. It provides a detailed comparison of various models, including traditional machine learning algorithms, deep learning approaches, and ensemble methods. The study highlights the strengths and weaknesses of each approach and presents insights into their practical applicability for forecasting in highly volatile markets [MRCV23].

- Diego Carraro, Andrea Rossi, Andrea Visentin, Steven Prestwich and Kenneth N. Brown: *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). IEEE Computer Society,

2023.

This research extends the work of the thesis by addressing the resource allocation problem in cloud computing. The study explores various strategies for efficient GPU allocation to meet the demands of diverse cloud workloads. It includes an in-depth analysis of the challenges and solutions associated with optimizing GPU usage, ensuring high performance and cost-effectiveness in cloud environments [CRV⁺23].

- Veronica Biolcati, James Woolley, Élodie Lévêque, Andrea Rossi, Anna Hoffmann, Andrea Visentin, Pádraig Ó Macháin, Daniela Iacopino, *Establishing the original order of the poems in Harward's Almanac using paleography, codicology, X-ray fluorescence spectroscopy, and statistical analysis* in *Heritage Science*, 11(1), pp. 1-14, 2023.

In this paper, clustering algorithms are applied to classify documents within Harward's Almanac, a historical manuscript. The research focuses on categorizing the documents based on their content and features. The study provides valuable insights into the use of machine learning for historical document analysis, demonstrating the effectiveness of clustering methods in organizing and interpreting complex datasets [BWL⁺23].

1.5 Outline of the Dissertation

The remainder of this dissertation is organised as follows.

In Chapter 2, we review the forecasting methods in workload demand prediction in cloud computing. After this review, we survey the uncertainty quantification methods in DL. We conclude the chapter with an overview of time series clustering.

In Chapter 3, to provide a platform for other researchers and to support reproducibility, we preprocess the real-world datasets from Google Cloud and Alibaba Cloud systems that we use in the experiments and describe each step explicitly and accurately to overcome the inconsistent and often not detailed versions in the literature.

In Chapter 4, we present the HBNN, our model that accounts for the uncertainty in univariate time series forecasting. We analyse the components that characterise this model. Then, through experiments on eight datasets, we show the

advantages of uncertainty modelling and compare it to a standard predictive DL architecture.

In Chapter 5, we analyse how well the HBNN model can generalize to different datasets and train it for bivariate forecasting. Then, we examine how accurately the models can predict in a domain generalisation context that gives the opportunity for deploying pre-trained models.

In Chapter 6, we explore the use of time series clustering to reduce the computational cost of the training of the forecasting models. In particular, we compare four clustering algorithms for time series to reduce the numerosity of the training set to train the HBNN and state-of-the-art probabilistic forecasting models.

Chapter 7 draws conclusions and discusses open research paths that could be explored in the future.

All the experiments in this dissertation have been conducted using a CPU Intel®Xeon®Gold 6240 at 2.60GHz and GPU NVIDIA Quadro RTX 8000 with 48 GB of memory in Ubuntu 20.04.

Chapter 2

Workload Prediction in Cloud Computing

Workload prediction is a technique used to improve efficiency and reduce the operational cost of the cloud. It can be exploited in many aspects of cloud resource management, such as proactive resource allocation, load balancing, and virtual machine (VM) migration [SS21]. Proactive resource management involves allocating resources in advance based on predicted future workload. This approach helps to avoid challenges like under/overloading of physical machines, resource wastage, and QoS violations. Load balancing is another application that benefits from workload forecasting, as it allows for even distribution of the workload across different servers, preventing any single server from becoming a bottleneck. Workload forecasting also guides the decision-making process for VM migration, ensuring efficient utilization of resources. In this way, cloud service providers can optimize the energy consumption of DCs by sleeping or awakening servers, leading to cost savings and reduced environmental impact. Accurate workload forecasting also helps in meeting the required QoS level, preventing SLA violations, and ensuring rapid response to user demands [MK20].

In workload forecasting, the time is divided into periods $t \in T$. Considering a dataset consisting of the historical workload data, where y_t is the resource usage (e.g., processing units, memory units) at time t . We can formulate the workload forecasting as the task of learning a function $f(t, \theta)$ parameterized by θ that maps a window of the historical workload to a prediction of the upcoming one:

$$f : (\mathbf{y}_{t-K}, \dots, \mathbf{y}_{t-1}, \theta) \rightarrow \hat{\mathbf{y}}_t \quad (2.1)$$

where $\hat{\mathbf{y}}_t$ is the predicted value of the workload at time t , and K is the number of past values given as input. The function f is learned by minimizing a loss function $L(\mathbf{y}_t, f(\mathbf{y}_{t-K}, \dots, \mathbf{y}_{t-1}, \theta))$, which quantifies the discrepancy between the predicted and the true workload. Therefore, workload forecasting can be mathematically formulated as a regression problem that finds the parameter θ that minimizes the loss over given datasets. We can optimize the model using the gradient descent algorithm and a loss function such as the mean squared error (MSE) or mean absolute error (MAE), which are defined as follows:

$$\text{MSE} = \frac{1}{N+1} \sum_{t=0}^N (\hat{\mathbf{y}}_{t+k} - \mathbf{y}_{t+k})^2 \quad (2.2)$$

$$\text{MAE} = \frac{1}{N+1} \sum_{t=0}^N |\hat{\mathbf{y}}_{t+k} - \mathbf{y}_{t+k}| \quad (2.3)$$

where N is the number of predictions and $\hat{\mathbf{y}}_{t+k}$ and $\mathbf{y}_{t+k}$ are the prediction and ground truth at time $t+k$, respectively.

If the variable $\hat{\mathbf{y}}_t$ represents only one type of resource, then the problem of predicting the workload is called univariate forecasting. On the other hand, if the output has multiple dimensions, the problem is referred to as multivariate forecasting. Additionally, if the objective is to predict $\hat{\mathbf{y}}_t$, it is known as one-step-ahead forecasting while, if the goal is to predict $(\hat{\mathbf{y}}_t, \hat{\mathbf{y}}_{t+1}, \dots, \hat{\mathbf{y}}_{t+H})$, then it is referred to as multi-step-ahead forecasting, where H is the prediction length or horizon.

An important yet underinvestigated aspect of workload forecasting is the uncertainty of the prediction. This involves generating probabilistic forecasts (i.e., where $\hat{\mathbf{y}}_t$ is a probability distribution instead of a value) with confidence intervals or quantiles that can provide an informative picture to the resource manager, helping the decision-making process. Accurately measuring uncertainty is essential for making safe decisions and comprehending the potential costs of every scheduling decisions [CJBR21].

Recent advancements in cloud workload forecasting have focused on leveraging sophisticated time series forecasting models and ML techniques to enhance accuracy and efficiency. For instance, Google Cloud Platform employs predic-

tive autoscaling, which uses historical and real-time data to forecast future load and automatically adjust resources, ensuring optimal performance and cost-effectiveness [Goo24]. Similarly, Alibaba Cloud has developed models which use historical data and DL algorithms to predict future resource needs and automatically adjust capacity, addressing issues like elastic lag and cold starts [Ali24]. A significant part of these advancements is the incorporation of uncertainty in forecasts, allowing for more robust and resilient resource management. By accounting for uncertainty, cloud providers can better handle unexpected surges in demand and optimize resource allocation, leading to improved service quality and reduced operational costs. These practices underscore the industry's shift towards proactive resource management and the adoption of ML models to improve cloud service quality and scalability.

In this dissertation, we focus on uncertainty modelling in workload forecasting in cloud computing environments. In this chapter, we review the main time series forecasting techniques used in workload prediction, uncertainty modelling in time series analysis and time series clustering algorithms. Our review is based on a rigorous selection process to ensure comprehensiveness and relevance. We selected papers using specific search terms such as "*workload forecasting*", "*workload prediction*", "*cloud computing*" "*time series analysis*", "*uncertainty modelling*" and "*uncertainty quantification*" to identify relevant literature. Papers were sourced from reputable databases including IEEE Xplore, ACM Digital Library, and Google Scholar. We prioritized publications in well-regarded journals and conferences in the fields of cloud computing and ML. The number of citations was considered as an indicator of a paper's impact and relevance. Starting from key seed papers, we followed trails of references to discover additional important works. Papers were included based on their relevance to workload forecasting and their contribution to advancing the field, particularly those introducing novel methodologies or significant enhancements to existing techniques, expanding the research to state-of-the-art models in time series forecasting and time series clustering. Exclusion criteria included papers that did not directly address workload forecasting or lacked sufficient experimental validation.

In Section 2.1, we classify the predictive models in the literature. In Section 2.2, we discuss the uncertainty in cloud workload forecasting with a focus on types of uncertainty and how to model it. Moreover, we review the Bayesian inference via BNNs and the state-of-the-art probabilistic forecasting models. Finally, in Section 2.3, we review a few approaches for time series clustering and Section

2.4 concludes the chapter.

2.1 Predictive Models

The workload in cloud computing refers to the computing requirement needed to ensure the system is running effectively, and it is represented with time series, i.e. sequences of data points that occur in successive order over some period of time. Each time series represents the output obtained from monitoring the system resources, e.g. processing units or memory, over time. In recent years, workload prediction in cloud computing has been extensively studied, resulting in an important data collection effort and the development of various approaches to tackle this issue. These approaches can be grouped into three categories: statistics-based, classic ML-based, and DL-based. These categories represent the technological progression of these techniques over time, from basic statistical methods to advanced neural networks (NNs). Statistics-based approaches employ mathematical models to identify patterns and trends in workload data, but they often struggle to manage the complexity and variability of cloud workloads. Classic ML-based approaches use various algorithms to learn from historical data and make predictions, but they necessitate manual feature extraction and parameter tuning. DL-based approaches utilize multi-layer NNs to learn the features and relationships of workload data automatically, but they demand large amounts of training data and computational resources. This section offers a succinct and non-exhaustive overview of these forecasting models, emphasizing their key features and applications.

2.1.1 Statistical Methods

Statistical methods are widely used for time series forecasting due to their ability to model complex patterns in data. They are among the first methods that have been proposed for this task. Here, we briefly describe the most popular statistical methods used in workload prediction in cloud computing.

ARIMA

AutoRegressive Integrated Moving Average (ARIMA) is a powerful statistical modelling technique that enables the analysis and prediction of time series data. ARIMA models are particularly useful when working with stationary time series data, characterized by consistent mean and variance over time and often exhibit

patterns like trend and seasonality. With ARIMA, it is possible to forecast future values of a time series based on its historical data.

The ARIMA model is a more advanced version of the ARMA model, which is used to analyze time series data and make predictions about future values. The "AR" in ARIMA stands for autoregressive, meaning that the studied variable is regressed on its own past values. The order of the autoregressive component, denoted as p , is the number of lag terms to consider. The AR model can be expressed as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \quad (2.4)$$

where c is a constant (intercept), y_t and ϵ_t are the values of the time series and the white noise error term at time t , respectively, and ϕ_i are the autoregressive coefficients.

The "MA" stands for moving average, indicating that the regression combines error terms from various past times. The order of the moving average component, denoted as q , is the number of error terms to consider. The MA model can be expressed as:

$$y_t = \mu + \epsilon_t + \vartheta_1 \epsilon_{t-1} + \vartheta_2 \epsilon_{t-2} + \cdots + \vartheta_q \epsilon_{t-q} \quad (2.5)$$

where μ is the mean of the time series, ϵ_t is white noise error term at time t , and ϑ_i are the moving average coefficients.

The "I" stands for integrated and represents the difference between data values and their previous values, with this differencing process potentially being repeated multiple times. The order of the integrated component, denoted as d , represents the number of times that differentiation is applied to reach stationarity. A time series is considered stationary if its statistical characteristics, such as the mean, variance, and autocorrelation, do not change over time [HA18]. We denote the value of the time series at time t after d differentiation as $y_t^{(d)}$. These features are designed to help the model fit the data as well as possible. From its components, ARIMA models are typically denoted as $\text{ARIMA}(p, d, q)$, where p , d , and q are non-negative integers representing the order of the autoregressive model, the degree of differencing, and the order of the moving-average model, respectively [HA18]. Combining the three components, the model can be expressed as:

$$y_t^{(d)} = c + \phi_1 y_{t-1}^{(d)} + \phi_2 y_{t-2}^{(d)} + \cdots + \phi_p y_{t-p}^{(d)} + \vartheta_1 \epsilon_{t-1} + \vartheta_2 \epsilon_{t-2} + \cdots + \vartheta_q \epsilon_{t-q} + \epsilon_t \quad (2.6)$$

There are also more powerful variations of this model, such as SARIMA (Seasonal ARIMA) and SARIMAX, where seasonality and exogenous variables, i.e. external regressors, are considered in the forecasting model. It is worth noting that the workload of the cloud server tends to have a non-seasonal random pattern due to its burst [TBD⁺20], which can make a seasonal model less effective in predicting its workload. In contrast, other workload types, such as HTTP server requests, exhibit strong seasonality in their pattern [WSS23].

ARIMA models have been widely investigated in workload forecasting, starting from predicting the future host workload in distributed systems (Digital Unix) [DO00]. In their study, [CMRB14] put forth a module for workload prediction based on the ARIMA model. This module allows for the dynamic provisioning of VMs in an elastic cloud environment to serve predicted requests while considering QoS parameters like response time and rejection rate. The model was trained using data from Wikipedia web servers, which exhibit seasonal patterns, making it an effective predictor.

As more public datasets on workload in cloud computing became available in the following years, ARIMA emerged as a widely-used benchmark for assessing advanced models for workload forecasting [Vaz15, XYB⁺15, BRGB⁺16, KS21].

GARCH

Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) is a statistical model that analyzes time-series data while considering serial autocorrelation in variance errors [Bol86]. These models assume that the error term's variance follows an autoregressive moving average process, making them useful for tasks such as predicting the volatility of financial asset returns. The GARCH model is identified by its order p and q , so we refer to it as GARCH(p, q).

The value of the dependent variable at time t , i.e. y_t , is defined as:

$$y_t = \mu + \epsilon_t \quad (2.7)$$

where μ is the expected value of y_t and ϵ_t the error term at time t computed as

follows:

$$\epsilon_t = \sigma_t e_t \quad (2.8)$$

where σ_t and $e_t \sim N(0, 1)$ are the standard deviation and the white noise at time t , respectively.

The variance σ_t^2 is computed as follows:

$$\sigma_t^2 = \omega + \sum_{i=1}^p \alpha_i \sigma_{t-i}^2 + \sum_{j=1}^q \beta_j \epsilon_{t-j}^2 \quad (2.9)$$

with $\omega > 0$, $\alpha_i, \beta_i \geq 0$, parameters of the model to ensure the positivity of the variance.

From this formulation, we can see that y_t can be drawn from a Gaussian distribution of mean equal to μ and variance equal to σ_t^2 , i.e.:

$$y_t \sim N(\mu, \sigma_t^2) \quad (2.10)$$

This model focuses on modelling the variance of the output while a simple AR component is applied for the mean component.

GARCH has been used as a baseline in workload forecasting in cloud computing, such as [BS15], for modelling the volatility in workload data.

Other methods

There are many other statistical techniques available for workload forecasting. Two of the most common methods are exponential smoothing (ETS) and Kalman filters. ETS is a time series forecasting method that can be applied to univariate data and handle data with systematic trends or seasonal patterns. It generates forecasts that are weighted averages of previous observations, with older observations carrying less weight [Bro56]. On the other hand, Kalman filters are recursive Bayesian filters that can estimate the state of a linear dynamic system based on a series of noisy measurements [Kal60].

ETS and its variants have been widely used for workload forecasting in cloud computing. Examples include [DKC12, YPZ⁺14, DKC14, WFC21] where ETS

has been used as a baseline in workload prediction of CPU demand for applications such as autoscaling. Similarly, Kalman filters have been applied for resource provisioning and elasticity mechanisms in cloud computing applications [HJL⁺14, HDPW16].

2.1.2 Machine Learning Methods

ML-based algorithms offer numerous benefits over traditional statistical methods for predicting workloads in cloud computing. Statistical methods are based on the assumption that the underlying relationships in the time series are stationary, meaning that they do not change over time. ML methods, on the other hand, are not based on this assumption. This makes ML methods suitable also for forecasting non-stationary workloads. Furthermore, ML algorithms can handle complex and diverse data more efficiently despite being less interpretable and requiring more data for training than statistical methods [CTS19]. Here, we briefly describe the most popular ML methods used in workload prediction in cloud computing.

K-Nearest Neighbours Regression

K-nearest neighbours (KNN) regression is ML method that works by finding the K most similar past workloads to the current workload and predicting the future workload based on the average of the K past workloads [Mit97]. The similarity between two workloads is typically measured using a distance metric, such as Euclidean or Manhattan distance on the features. The features might include historical workload data, time of day, day of the week, and other relevant factors. One of the main advantages of KNN is its simplicity and the fact that it makes no assumptions about the underlying data distribution, making it a flexible method that can adapt well to changes in the cloud environment. However, it can be computationally intensive, particularly with large datasets, and the model's performance can be significantly impacted by the choice of K and distance metric. KNN is one the first ML methods used in workload forecasting, and it has been applied in middleware platforms [DQRT15].

Random Forest

Random forest (RF) regression is a type of ensemble learning algorithm. Unlike traditional models that rely on a single decision tree, RF builds a "forest" – a collection of numerous, uncorrelated decision trees. Each tree is trained on a

random subset of features and data points drawn with replacement from the original dataset [Bre01]. RF regression is well-suited for cloud workload forecasting because it can handle the complexity of cloud computing workloads, the scarcity of data, and the non-stationarity of workloads. This dynamic model can be updated with new data to adjust to changing workload patterns, making it a valuable tool in the constantly evolving cloud computing landscape. RF works best when it is fed with features extracted from the data rather than raw data. This aligns with the approach taken by [SK21b], where the RF is one of the ML methods used for forecasting CPU, disk and memory utilisation in cloud computing environments.

Support Vector Regression

Support vector regression (SVR) is a type of kernel-based regression, which means that it uses a kernel function to transform the data into a higher dimensional space, where it is easier to learn a linear relationship in the data [DBK⁺96]. SVR works by finding a hyperplane in the transformed space that minimizes the error between the predicted and actual workload values, i.e. by finding a function that deviates from the actual workload value by no more than a certain threshold called support. It is robust to outliers, but the choice of kernel function and the tuning of parameters can significantly impact the model's performance. In this regard, the performance of the SVR can be improved when the tuning of the hyperparameters is suitable [ZZSG18] and it has been shown to outperform ARIMA and improve the prediction accuracy [GWS20].

Other methods

There are many other ML algorithms available for workload forecasting. Other popular examples in the literature include linear regression [AB13] and gradient boosting algorithms such as XGBoost [DQRT15, CLY⁺21]. Linear regression is a supervised learning algorithm that can predict the value of a continuous variable, such as CPU usage or memory usage, based on the values of other variables, such as the number of active users or the time of day [Fre09]. Linear regression is a simple algorithm to train and interpret, but it can be less accurate than other ML algorithms for complex patterns, such as cloud workloads. Gradient boosting is an ensemble learning algorithm that combines the predictions of multiple weak learners to produce a more accurate prediction [Fri01]. Gradient boosting algorithms such as XGBoost, a specific implementa-

tion of the gradient boosting algorithm known for its accuracy and scalability, can handle the complexity and dynamism of cloud computing workloads. It is one of the most sophisticated algorithms that outperform other ML models like linear regression, KNN and SVR [BRK21].

2.1.3 Deep Learning Methods

ML methods, while powerful tools, may struggle to capture long-term patterns in the data, and often, these methods require feature engineering, a process of manually extracting and crafting informative features from the raw data, to achieve optimal performance [WBB⁺22]. The use of DL methods has become increasingly popular for addressing the difficulties presented by the intricate nature of cloud workload data, especially since the advent of the first large datasets from the cloud computing domain. Unlike traditional ML approaches, DL excels at automatically extracting features directly from the raw data. This eliminates the need for extensive manual feature engineering and allows the model to learn complex, non-linear relationships within the data. In the following sections, we will go deeper into the most popular DL approaches employed for workload prediction in cloud computing environments.

Neural Network-based models

The initial exploration into workload forecasting with DL models utilized feed-forward NNs [PNKJ11]. These models demonstrated significant promise, capturing non-linear relationships between input features and workload demands, leading to precise short-term predictions. However, as the complexity of workload data grew, limitations of feedforward NNs emerged [QZG16]. Feedforward NNs laid the groundwork for more complex models, showcasing the potential of DL in capturing non-linear relationships and providing precise short-term predictions. They serve as an early point of reference for evaluating the effectiveness of newer, more sophisticated architectures.

To address this challenge, Deep Belief Networks (DBNs) were introduced. DBNs possess a unique architecture that allows them to learn intricate dependencies and feature representations within the data [ZDY⁺17, KGR⁺24]. DBNs are stacked layers of a specific type of neural network known called Restricted Boltzmann Machines (RBMs) and that are trained one layer at a time in an unsupervised manner. Each layer learns to represent the data in a progressively more abstract and informative way. For instance, in the context of cloud

workload forecasting, a DBN might learn low-level features like CPU utilization patterns in the first layer. Subsequent layers could then build upon these features, potentially identifying relationships between CPU usage and memory demands, or even user behaviour patterns influencing workload spikes. This enhanced learning capability translated to improved forecasting capabilities, particularly for capturing long-term trends and complex patterns in workload data. Thus, DBNs demonstrate a significant advancement over previous models.

However, the effectiveness of ML models hinges not only on the architecture itself but also on careful hyperparameter tuning and weight initialization. Researchers have investigated various optimization algorithms, including differential evolution [KS18, KS19a], particle swarm optimization [MDB⁺18, KP24], genetic algorithms [PR21], and other metaheuristics [KS19b]. These methods can be used to fine-tune model parameters, resulting in improved overall predictive performance. The advancements in optimization algorithms illustrate the importance of parameter tuning in enhancing model accuracy and reliability.

Sequential models

The time-sensitive nature of cloud workload data presents a unique challenge for forecasting models. NNs might struggle to capture the sequential patterns and dependencies inherent in such data. To address this, sequential models have emerged as a powerful tool for cloud workload forecasting.

One of the fundamental building blocks in this domain is the Recurrent Neural Network (RNN) [ZLZ⁺16, DMD⁺17]. RNNs excel at processing sequential data, allowing them to effectively analyze the order and relationships between data points in a time series that fits the temporally ordered nature of cloud workload data. This capability makes RNNs suitable for capturing trends and dependencies within the data. RNNs serve as a foundational model for evaluating the performance of advanced sequential models in handling temporal dependencies.

However, RNNs can suffer from the vanishing gradient problem, hindering their ability to learn long-term dependencies. To overcome this limitation, advancements have led to the development of more sophisticated architectures. LSTM networks address the vanishing gradient problem by incorporating specialized memory cells that can learn and retain information over longer periods [JB17, SYZ⁺18]. This allows LSTMs to effectively capture both short-term and

long-term dependencies within the workload data, leading to more accurate forecasts. LSTM networks mark a pivotal improvement in handling dependencies in sequential data.

The exploration of RNN architectures continues with models like Bidirectional LSTMs (BiLSTMs) [BLYZ21, WFC21, UBY23]. BiLSTMs process data in both forward and backward directions, allowing them to capture context from both the past and future, potentially leading to even more comprehensive understanding of the workload patterns. Gated Recurrent Units present another variant, offering a computational efficiency advantage over LSTMs while maintaining strong performance in capturing long-term dependencies [XSW⁺22].

Beyond these individual models, recent research has explored the synergy between Convolutional Neural Networks (CNNs) and LSTMs. CNNs excel at extracting spatial features from data, and their integration with LSTMs leverages this strength to capture both local patterns within workload data and the broader temporal dependencies [LFK⁺21, KMDA21, OHU21, PK22]. This hybrid approach has proven to be particularly effective in cloud workload forecasting tasks. More recently, Temporal Convolutional Networks (TCNs) have emerged as a powerful alternative for time series forecasting tasks. TCNs employ convolutional layers to capture temporal dependencies in data, utilizing dilation and causal convolutions to ensure that predictions are only influenced by past information, preserving the temporal order [LFV⁺17]. TCNs have demonstrated superior performance in capturing long-range dependencies and handling variable-length sequences, making them well-suited for complex time series data [HZ19]. In the context of cloud workload forecasting, TCNs can effectively model intricate temporal patterns and relationships between various workload metrics, such as CPU, memory usage, and network traffic. By leveraging these capabilities, TCNs provide more accurate and robust predictions, enabling cloud service providers to optimize resource allocation and improve service quality [Abo22].

Overall, cloud workload forecasting has benefited greatly from the advancements in sequential models, offering a robust toolkit for capturing the complexities of cloud workload data and generating accurate forecasts.

Transformer-based models

Cloud workload data often exhibits intricate patterns that unfold over extended periods. Traditional sequential models might struggle to capture these long-

range dependencies effectively. To address this challenge, recent advancements in DL have introduced novel techniques like attention mechanisms and transformers.

Attention mechanisms, as proposed by Vaswani *et al.* in 2017 [VSP⁺17], offer a groundbreaking approach to sequence modelling. These mechanisms allow models to focus on the most relevant parts of an input sequence, dynamically assigning weights to different segments based on their significance for the prediction task. This targeted focus on informative parts of the data sequence empowers models to capture long-range dependencies more effectively. Transformers further revolutionize sequence modeling by integrating the attention mechanism and enabling parallel processing of data. Unlike traditional RNNs that process data sequentially, transformers can analyze the entire sequence simultaneously. This parallel processing capability not only improves efficiency but also eliminates the vanishing gradient problem that can hinder RNNs in capturing long-term dependencies.

The combination of attention mechanisms and transformers has demonstrably revolutionized sequence modelling. Researchers have successfully applied these techniques to the cloud workload forecasting problem [ZZCG19, XYLX21, QYLC21, SZL⁺21, AJL⁺22, CZY22, PB23, DKMS23]. By incorporating attention and transformers into forecasting models, researchers have achieved significant improvements in capturing long-range dependencies within cloud workload data, leading to more accurate and robust forecasts. These advancements highlight the transformative impact of attention and transformer models on sequence data processing.

Ensemble models

Over the years, researchers have been exploring how combining various DL architectures can enhance forecasting accuracy and model robustness. They have introduced ensemble models integrating multiple DL, ML, and statistical models into a single framework. These models leverage the unique strengths of individual models, capturing various aspects of workload patterns and providing more dependable predictions. Several studies [DKC14, EVG15, EVG15, LLC16, ZUHK⁺17, KWQH20, BRK21, KSK⁺24] have demonstrated the effectiveness of ensemble models.

Using the ensemble approach in data analysis enables a thorough examination of intricate relationships and patterns. This reduces the possibility of overfit-

ting and improves overall performance. By combining the benefits of various DL architectures, the forecasting model becomes more adaptable and capable of handling diverse workload scenarios. This enhances the effectiveness of workload forecasting in cloud computing [VKSR21].

For more predictive models in cloud computing that have not been included in this chapter due to their specific focus on time series forecasting, which falls outside the scope of uncertainty modelling for cloud workload prediction, we refer the reader to the following survey papers [HAA⁺17, DLCÖ19, MK20, AK23b, KS23, KJS24].

Numerous models have been proposed for cloud workload forecasting, employing techniques such as data mining, time series analysis, and DL techniques. This survey particularly emphasizes models that are widely used in the literature, focusing on those where uncertainty quantification components can be easily integrated. These models aim to predict future workloads to enable proactive resource allocation and management. The performance metrics used in these studies include predictive accuracy and computational efficiency. For instance, the studies collectively highlight that models leveraging DL techniques generally outperform those based on simple statistical methods in terms of accuracy. However, these models often require more computational resources and longer training times. The reviewed studies provide a comprehensive evaluation of various cloud workload forecasting models, identifying their strengths and limitations. These insights are invaluable for developing more accurate and efficient forecasting models with enhanced uncertainty quantification capabilities.

2.2 Uncertainty in Cloud Workload Forecasting

The models discussed so far all share a common characteristic - they provide a point prediction of future workload without any indication of the confidence of the forecast. This can be problematic because, especially in DL, 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 and overconfident predictions can be dangerous. To ensure safe and informed decision-making, models should provide an output that accurately describes the level of certainty

in their results. This means that models should include information about their uncertainty so that users can determine whether the output can be trusted. If the model produces an output with high uncertainty, the algorithm can request additional information or input from a human to make the decision-making process smoother.

Cloud workload can be highly unpredictable, and without considering uncertainty, it can be inaccurate and misleading. This can lead to several problems, such as overprovisioning, which can waste money and resources; underprovisioning, which can lead to performance problems and outages; and inefficient resource utilization, leading to higher costs and lower performance. Managing uncertainty is vital, and it enables organizations to make more informed decisions, allocate resources effectively, and adapt to changing conditions in dynamic environments [KKM⁺21].

However, uncertainty quantification is challenging due to several sources of uncertainty, including the diversity and dynamic nature of cloud workloads. Cloud workloads exhibit diverse characteristics and patterns, making generalising forecasting models across different applications and usage scenarios challenging. Furthermore, cloud workloads constantly change due to user behaviour, application updates, and external events, making capturing long-term trends and patterns difficult [MAH18].

We also need to highlight that cloud computing is a relatively new technology, and the availability of historical data for workload forecasting is often limited, making it challenging to build accurate predictive models. Data can be noisy and incomplete, containing missing values, outliers, and inconsistencies, further complicating the task of uncertainty quantification.

In ML, uncertainty is generally classified as epistemic and aleatoric, which are explained in the following section.

2.2.1 Epistemic and Aleatoric Uncertainty

Epistemic uncertainty refers to the uncertainty of a model caused by insufficient training data [HW21]. Adding more relevant and high-quality data for training can reduce this type of uncertainty. It is important to note that the effectiveness of data in reducing epistemic uncertainty depends on its appropriateness and annotations [HLZ21]. When the input of the model is far from the training data, models are expected to exhibit high epistemic uncertainty (*data variability*).

In contrast, they should show low epistemic uncertainty for data points near the training data. If a narrow dataset is used as a baseline, the model can only perform well for inputs within that range. However, the model may lack knowledge of inputs beyond the training range and, therefore, fail to categorize data correctly [ZLP⁺22].

Additionally, complicated models containing a multitude of parameters can result in increased epistemic uncertainty. When there is a lack of sufficient data, it may not be possible to definitively determine the values of all the parameters, which can lead to uncertainty surrounding their accuracy (*procedural variability*). Epistemic uncertainty is a significant issue in real-world applications, mainly when there is a large quantity of data but poor quality. Epistemic uncertainty can be used to adapt forecasting models to changing data patterns and improve their accuracy over time. Indeed, by monitoring the level of epistemic uncertainty, forecasting models can be updated or adjusted to reflect new trends or changes in the data distribution [ARA23].

Aleatoric uncertainty is related to the inherent stochasticity in the process. When we capture data, it is never a perfect representation of reality because it is always contaminated by noise and randomness. This noise cannot be controlled and is accumulated across all observations, resulting in aleatoric uncertainty for the model. Unfortunately, unlike epistemic uncertainty, aleatoric uncertainty cannot be reduced with additional observations. This is because additional data will still include noise captured during observation. Aleatoric uncertainty is not a property of the model but rather an inherent property of the data distribution. It is also known as data uncertainty and can be captured by probabilistic classification and regression models through maximum likelihood estimation [DKD09].

Generally, the sum of epistemic and aleatoric uncertainty is referred to as predictive uncertainty [GHO17]. The design of the approach influences the type of uncertainty that can be modelled [APH⁺21]. There is a broad literature on how to model the uncertainty in DL, including ensembles [PHB18] and Bayesian approaches [CN20, MPV21] that can be adapted to time series forecasting tasks. In the following section, we survey the techniques used for modelling and quantifying uncertainty in cloud workload forecasting.

2.2.2 Forecasting Uncertainty with Deep Learning

Uncertainty quantification techniques for time series forecasting provide a range of tools for capturing and interpreting uncertainty in model predictions. The choice of approach depends on the specific characteristics of the data, computational considerations, and the types of uncertainty that need to be accounted for. Various methods are employed to measure the level of uncertainty in time series prediction, such as ensembles and probabilistic forecasting [FVT22].

Ensemble forecasting enhances the robustness and accuracy of predictions by training multiple models and combining their outputs [AV17]. This technique captures uncertainty by considering diverse sources of variability. The variability and the spread of predictions among ensemble members serve as a natural indicator of uncertainty. Ensemble forecasting offers robustness by reducing overfitting and improving generalization. However, managing and training multiple models can be computationally intensive, and the size and diversity of the ensemble members influence the effectiveness of ensembles. For this reason, ensembles have become very popular for improving the accuracy of cloud workload forecasting [ZLZ⁺20, KWQH20, BRK21], without focusing on the uncertainty quantification aspects that are mainly tackled by probabilistic forecasting.

Probabilistic forecasting, instead of providing a single forecast, estimates the probability of various future outcomes. This approach generates a distribution of potential values, accounting for the inherent unpredictability and variability in real-world scenarios, making it an ideal choice for workload forecasting in cloud computing environments. Probabilistic forecasting is a comprehensive concept that includes any predictive technique that provides a probability distribution of future outcomes.

Probabilistic forecasting can be classified into *parametric* and *non-parametric* approaches. Parametric methods assume a specific structure or distribution for the data, while non-parametric methods do not. Parametric methods are typically more efficient and can provide more accurate forecasts for data that follows the assumed distribution. However, they may be less flexible in capturing complex patterns in the data that deviate from the assumed distribution. Non-parametric methods are more adaptable to a wider range of time series patterns but may require more data and computational resources compared to parametric approaches. The choice between parametric and non-parametric methods depends on the characteristics of the time series data, the specific fore-

casting task, and the balance between flexibility and computational efficiency. For time series with well-defined patterns and limited complexity, parametric methods can provide efficient and accurate forecasts. For more complex and non-stationary time series, non-parametric methods offer greater flexibility and adaptability. Examples include Bayesian inference [BT11] and density kernel estimation [Wę18], which are parametric and non-parametric methods, respectively.

Bayesian inference can incorporate prior beliefs about cloud workload patterns into the forecasting model that can be based on historical data, expert opinions, or domain knowledge, which is translated in the assumption of a specific shape of the underlying output distribution, e.g. Gaussian. Density kernel estimation is a method that estimates the probability density function of a random variable based on observed data points, and it is particularly useful for capturing the underlying distribution of data without assuming a specific form. In cloud workload forecasting, kernel density estimation has been applied directly to estimate the probability density function of future workload scenarios, offering a non-parametric approach to uncertainty quantification [MMF17].

Another popular non-parametric method called quantile regression (QR) is used to estimate conditional quantiles of the response variable based on the predictor variables. This method provides a complete understanding of the potential outcomes' distribution for a given input and gives a detailed knowledge of the uncertainty associated with the model's predictions. QR is often used in time series forecasting, and most state-of-the-art models predict a set of quantiles of the output distribution. It has recently been used for uncertainty prediction in workload prediction in cloud computing [MBK⁺23]. However, interpreting uncertainty estimates in non-parametric models can be more difficult, as they combine multiple sources of uncertainty [KSZ20].

Another classification of probabilistic forecasting is between Bayesian and non-Bayesian methods. Bayesian methods incorporate prior beliefs about the parameters of the model, while non-Bayesian methods do not. Bayesian methods can provide more robust and reliable forecasts, especially for small datasets, by incorporating prior knowledge about the data. However, they may be more computationally expensive and require the specification of prior beliefs, which can be challenging in some cases. Non-Bayesian methods are simpler and more computationally efficient but may be less robust in cases where prior knowledge is available. BNNs are a specific application of NNs that incorporate the

Table 2.1: Summary of the uncertainty quantification methods discussed in this section and their use in cloud workload forecasting domain.

Method	Classification	Type of uncertainty	Cloud workload forecasting applications
Ensemble Forecasting (*)	non-parametric, non-Bayesian	epistemic	[ZLZ ⁺ 20, KWQH20, BRK21]
Density Kernel Estimation	non-parametric, non-Bayesian	aleatoric	[MMF17],
Quantile Regression	non-parametric, non-Bayesian	aleatoric	[MBK ⁺ 23]
Bayesian Neural Network (via Monte Carlo Dropout)	non-parametric, Bayesian	epistemic	$\times$
Bayesian Neural Network (via Variational Inference)	parametric, Bayesian	epistemic	$\times$

(*) Uncertainty aspects are still underinvestigated

Bayesian principles, while QR and density estimation methods do not rely on Bayesian principles.

Finally, uncertainty quantification methods for time series forecasting can be classified based on the type of uncertainty that they can model, i.e. aleatoric only, epistemic only or both. Density estimation methods and QR model only the aleatoric uncertainty [ZZT⁺22]. Ensemble forecasting captures the diversity among models, which represent different hypotheses, and helps to quantify epistemic uncertainty. Bayesian inference updates prior beliefs based on new workload data, resulting in posterior distributions that represent both the inherent randomness in the data (aleatoric uncertainty) and the uncertainty in the model itself (epistemic uncertainty). However, in the cloud workload prediction context, Bayesian inference is still underinvestigated. Table 2.1 summarises and classifies the uncertainty quantification methods described in this section.

In DL, BNNs use Bayesian inference principles to model both the epistemic and aleatoric uncertainty and generate probabilistic predictions. There are various ways to implement BNNs, including via variational inference (VI). The following section will go through the Bayesian inference approach and how it is applied via BNNs and VI.

2.2.3 Bayesian Inference

Bayesian inference is a crucial framework in statistics and probabilistic reasoning that enables us to revise our beliefs about uncertain quantities (known as parameters) after studying observed data. This method is built on Bayes' theorem, which outlines the connection between the conditional probabilities of

two events. Specifically, in the context of Bayesian inference, it links the likelihood of a hypothesis (which refers to parameter values) given the observed data to the likelihood of the data given the hypothesis, the prior probability of the hypothesis, and the marginal probability of the data [DKC12].

Given a dataset $\mathcal{D}$, we annotated it as $\mathcal{D} = \bigcup_{i=1}^{|\mathcal{D}|} (x_i, y_i)_i$ for simplicity. In the case of a general time series dataset x_i corresponds to $(y_{t-K}, \dots, y_{t-1})$ (past values) and y_i to $\hat{y}_t$ (value(s) to predict). To estimate an unknown parameter θ from $\mathcal{D}$, we combine our prior beliefs with the likelihood of the data to obtain a posterior distribution. This distribution represents our updated beliefs after considering the data. Before observing any data, we express our prior beliefs about θ as a probability distribution, called the prior distribution, denoted by $p(\theta)$. Before data is collected, the prior distribution incorporates any existing knowledge or beliefs about θ . The likelihood function, denoted by $p(\mathcal{D}|\theta)$, describes the probability of observing the data from $\mathcal{D}$ given a specific value of θ . It quantifies how well θ explains the observed data.

Based on the new observed data, the beliefs about the parameter need to be adjusted. To do so, the posterior distribution $p(\theta|\mathcal{D})$ can be computed. This distribution gives the probability distribution of the parameter θ , given the observed data $\mathcal{D}$. To obtain this distribution, Bayes' theorem can be applied:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \quad (2.11)$$

where the term $p(\mathcal{D})$ represents the marginal likelihood or evidence. It acts as a normalizing constant, ensuring that the posterior distribution sums up to 1.

After obtaining the posterior distribution, future data points can be predicted. The predictive distribution, which is represented by $p(\mathcal{D}^*|\mathcal{D})$, shows the probability distribution of new data points $\mathcal{D}^*$ based on the observed data $\mathcal{D}$. To obtain it, we integrate the likelihood function with respect to the posterior distribution.

$$p(\mathcal{D}^*|\mathcal{D}) = \int p(\mathcal{D}^*|\theta)p(\theta|\mathcal{D})d\theta \quad (2.12)$$

where $p(\mathcal{D}^*|\theta)$ is the likelihood function for the new data points $\mathcal{D}^*$ given a parameter θ . However, in practical applications, the integrals required to compute the posterior and predictive distributions are often too complex to calculate.

Applying Bayesian principles in DL can be hard due to the numerous parameters involved in NNs and the complexity of the models [BCKW15]. Despite this, many methods have been created to apply Bayesian concepts to DL models, which can improve the accuracy of predictions and provide a more reliable outcome. This approach can be seen in the development of BNNs, which use Bayesian principles in a unique way to provide probabilistic forecasts, unlike traditional deterministic NNs.

Bayesian Neural Networks

A BNN is a type of NN that utilizes Bayesian inference to calculate the uncertainty of its predictions. In a regular NN, the weights and biases are typically learned by optimizing a loss function through backpropagation. However, in a BNN, the weights and biases are considered random variables with probability distributions. Bayesian inference is then employed to update these distributions as new data is received. This enables the model to estimate the uncertainty in its predictions [JLB⁺22].

BNNs are a valuable tool for probabilistic modelling, and they are gaining more attention in research and applications that require accurate uncertainty measurement. Nonetheless, BNNs pose particular challenges, including higher computational complexity and the requirement for more advanced training algorithms compared to standard NNs [WBM⁺20].

One of the main concepts of BNNs is to assign prior distributions to the model's parameters (such as weights and biases) and modify them based on the observed data using Bayesian inference. This process results in the posterior distribution of the parameters, representing updated beliefs about the model's parameters based on the data. A BNN functions as a probabilistic model, denoted as $p(y_i|x_i, \mathbf{w})$, which calculates the probability of each potential output by utilizing the set of weights $\mathbf{w}$.

The weights can be learnt by maximum likelihood estimation (MLE), the MLE weights are given by:

$$\mathbf{w}^{MLE} = \arg \max_{\mathbf{w}} \log p(\mathcal{D}|\mathbf{w}) = \arg \max_{\mathbf{w}} \sum_i \log p(y_i|x_i, \mathbf{w}) \quad (2.13)$$

To achieve this, we typically use gradient descent, specifically backpropagation [BCKW15]. We make the assumption that $\log p(\mathcal{D}|\mathbf{w})$ is differentiable in $\mathbf{w}$.

We can introduce regularization by placing a prior on the weights $\mathbf{w}$ and then finding the maximum a posteriori (MAP) weights.

$$\mathbf{w}^{MAP} = \arg \max_{\mathbf{w}} \log p(\mathbf{w}|\mathcal{D}) = \arg \max_{\mathbf{w}} \log p(\mathcal{D}|\mathbf{w}) + \log p(\mathbf{w}) \quad (2.14)$$

where the term $\log p(\mathbf{w})$ is a regularization measure. When $\mathbf{w}$ is assigned a Gaussian prior, it results in L2 regularization (also known as weight decay). On the other hand, assigning a Laplace prior to $\mathbf{w}$ leads to the recovery of L1 regularization.

In Bayesian inference, the posterior distribution of the weights for NNs is calculated as $p(\mathbf{w}|\mathcal{D})$, based on the training data. This computation is equivalent to using an ensemble of an infinite number of NNs, which is not feasible for practical purposes due to its complexity. Therefore, there is a need for approximation methods [Gra11].

There are different techniques available for training BNNs, including Markov Chain Monte Carlo (MCMC) methods, Monte Carlo Dropout, and VI.

MCMC methods are used to sample from the posterior distribution of the parameters [MRR⁺53], but they can be computationally expensive and impractical for large networks. Monte Carlo Dropout is a simpler and more efficient way to approximate Bayesian inference in NNs [GG16]. During training, dropout is applied to the network by randomly setting some activations to zero. Multiple forward passes with different dropout masks are used to obtain a distribution over model predictions, which can be interpreted as uncertainty estimates.

VI approximates the posterior distribution using a tractable distribution like a Gaussian distribution. The objective is to find the approximating distribution parameters closest to the true posterior distribution. This approach is less accurate but more computationally efficient than MCMC methods, making it more suitable for real-world applications [SKW15].

Variational Inference

VI is a statistical and ML method that uses probabilities to estimate complex probability distributions, particularly in Bayesian inference scenarios. Essentially, VI transforms the task of determining the precise posterior distribution of unobserved variables into an optimization challenge [BKM17].

The goal is to estimate the posterior distribution of the weights based on the training data, which is denoted as $P(\mathbf{w}|\mathcal{D})$. Unfortunately, for many complex models, calculating the exact posterior distribution may not be feasible due to computational limitations or analytical difficulties.

To tackle the challenge of approximating the true posterior distribution, VI introduces a simpler and more manageable family of probability distributions called the "variational family" represented by $q(\mathbf{w}|\theta)$. The goal is to determine the closest member of the variational family to the true posterior using a distance metric. As a result, VI aims to identify the optimal parameter θ^* that minimizes the Kullback-Leibler (KL) divergence between the distribution on the weights $q(\mathbf{w}|\theta)$ and the true Bayesian posterior distribution of the weights. The KL divergence is defined as:

$$\mathbf{KL} [q(\mathbf{w}|\theta)||p(\mathbf{w}|\mathcal{D})] = \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{p(\mathbf{w})p(\mathcal{D}|\mathbf{w})} d\mathbf{w} \quad (2.15)$$

Therefore, the optimal parameter θ^* is given by

$$\begin{aligned} \theta^* &= \arg \min_{\theta} \mathbf{KL} [q(\mathbf{w}|\theta)||p(\mathbf{w}|\mathcal{D})] \\ &= \arg \min_{\theta} \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{p(\mathbf{w})p(\mathcal{D}|\mathbf{w})} d\mathbf{w} \\ &= \arg \min_{\theta} \underbrace{\mathbf{KL} [q(\mathbf{w}|\theta)||p(\mathbf{w})]}_{\text{complexity cost}} - \underbrace{\mathbb{E}_{q(\mathbf{w}|\theta)} [\log p(\mathcal{D}|\mathbf{w})]}_{\text{likelihood cost}} \end{aligned} \quad (2.16)$$

The equation 2.16 is a cost function comprising two components. The first component, the likelihood cost, depends on the data. The second component, the complexity cost, depends on the prior distribution, i.e. the complexity of the variational family.

The KL divergence can be equivalently expressed with the evidence lower bound (ELBO) or negative variational free energy:

$$\begin{aligned}
\theta^* &= \arg \min_{\theta} \mathbf{KL} [q(\mathbf{w}|\theta) \| p(\mathbf{w}|\mathcal{D})] \\
&= \arg \min_{\theta} \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{p(\mathbf{w}|\mathcal{D})} d\mathbf{w} \\
&= \arg \min_{\theta} \mathbb{E}_{q(\mathbf{w}|\theta)} \left[\log \frac{q(\mathbf{w}|\theta)}{p(\mathbf{w}|\mathcal{D})} \right] \\
&= \arg \min_{\theta} \underbrace{\mathbb{E}_{q(\mathbf{w}|\theta)} \log q(\mathbf{w}|\theta) - \mathbb{E}_{q(\mathbf{w}|\theta)} \log p(\mathbf{w}, \mathcal{D})}_{-\text{ELBO}(q)} + \underbrace{\log p(\mathcal{D})}_{\text{evidence}}
\end{aligned} \tag{2.17}$$

The evidence relates to the probability of the observed data. The ELBO is essential in VI as it acts as an objective function to lead the optimization process in discovering the most accurate approximating distribution. Because the KL divergence is always positive, the evidence is greater or equal to the ELBO, i.e.

$$\log p(\mathcal{D}) \geq \text{ELBO}(q) \tag{2.18}$$

Therefore, by maximizing the ELBO, we can minimize the KL divergence between the variational distribution and the true posterior, resulting in a more accurate approximation of the posterior distribution $p(\mathbf{w}|\mathcal{D})$. In practice, BNNs can be optimized through VI using the Bayes by Backprop algorithm [BCKW15].

2.2.4 Probabilistic Forecasting Models

In the following sections, we will introduce several cutting-edge probabilistic models that do not rely on Bayesian principles, many of which employ QR. These models are specially designed for training with multiple time series data, advancing the field of probabilistic forecasting. Because of their stochastic nature, they all can be used to model the uncertainty of the prediction.

DeepAR

DeepAR is a time series probabilistic forecasting model using autoregressive RNNs proposed by Amazon [SFGJ20] and constitutes a milestone for the time series forecasting community. One or more target time series produced by the same or related processes serve as the DeepAR algorithm's training input. The technique uses this input dataset to train a model that approximates these processes and utilizes it to forecast the evolution of the target time series. Each sample consists of a set of adjacent points of the time series representing the

context, i.e., how far the network can see, and the prediction window, which defines how far the model can predict the future. The advantage of this approach is that it can be jointly trained by sharing information across time series, capturing patterns of similar time sequences generated by the same process. Instead of directly using LSTM for the prediction, the model estimates the mean forecast as well as quantiles representing different levels of uncertainty. DeepAR's ability to jointly train across multiple time series and model uncertainty through quantiles makes it a benchmark for probabilistic forecasting in various applications, including cloud workload forecasting.

Temporal Fusion Transformer

Temporal Fusion Transformer (TFT) is an attention-based Deep NN designed by Google [LALP21]. It is another state-of-the-art method for time series forecasting. It can be applied to multi-horizon forecasting, where the multi-head attention allows the heads to share some weights that may be interpreted in terms of seasonality analysis. The model includes several components, such as an LSTM encoder-decoder layer and a variable selection network for feature selection when covariates are passed as network input.

The multi-head attention mechanism is exploited to highlight the most relevant parts of the input window to make the prediction. The optimization of this network generally uses quantile loss as the loss function, which allows modelling the uncertainty of the forecast. As for DeepAR, TFT is jointly optimized using multiple time series from different distributions. TFT has outperformed DeepAR in electricity demand and traffic forecasting applications, making it a leading model for complex forecasting tasks that require attention mechanisms and uncertainty modelling.

To conclude this review, we highlight other state-of-the-art probabilistic models that have gained popularity in time series forecasting, especially in benchmarking applications such as electricity demand forecasting and traffic forecasting. However, only some of them have been applied in the context of workload prediction in cloud computing.

Other Models Used in Cloud Workload Forecasting

- **Facebook Prophet** [TL18]: is a prediction tool created by Facebook's Core Data Science team. It is an open-source program that uses an additive

model to fit non-linear trends with yearly, weekly, and daily seasonality and holiday effects. While it is most effective with time series data with strong seasonal effects and multiple seasons of historical information, it has also been utilized for workload forecasting in cloud computing [DAMZ21]. Prophet's ease of use and effectiveness in capturing seasonality make it a practical tool for time series forecasting.

- **N-BEATS** [OCCB19]: is a model that uses NN to make probabilistic predictions for univariate time series. The architecture is based on backward and forward residual links with a deep stack of fully-connected layers. It is a fast and interpretable model that works well in various domains without requiring modifications. N-BEATS' robust performance and interpretability establish it as a reliable choice for univariate time series forecasting in diverse applications.
- **N-HiTS** [COO⁺23]: is a neural hierarchical interpolation model designed for time series forecasting. It tackles two significant obstacles in long-term forecasting: the unpredictability of the forecasts and their computational complexity. N-HiTS utilizes unique hierarchical interpolation and multi-rate data sampling methods, which allow the model to generate predictions sequentially. It also breaks down the input signal and creates a forecast through synthesis. N-BEAT and N-HiTS were employed concurrently in an ensemble for cloud workload forecasting [BYXS22]. N-HiTS' innovative approach to hierarchical interpolation and long-term forecasting challenges offers a sophisticated method for addressing complex forecasting problems in cloud workloads.

Other Models Not Used in Cloud Workload Forecasting

- **DeepVAR** [SBSC⁺19]: is an upgraded version of DeepAR that can handle multiple time series and detect how they affect each other. Compared to DeepAR, DeepVAR is specifically designed to manage multivariate time series data. While DeepAR can also handle multivariate data, DeepVAR is better at capturing the target values' covariance and the interactions between multiple time series. Its capability to predict multiple related time series together makes it highly suitable for multivariate forecasting tasks.
- **MQTransformer** [EPM20]: is a forecasting model that leverages transformer architectures initially developed for NLP. It incorporates context-

dependent and feedback-aware attention mechanisms, enhancing probabilistic forecasting accuracy. The model's decoder-encoder attention allows it to align with the context and analyze its historical data to improve its forecasts. Additionally, MQTransformer's positional encoding allows the NN to learn seasonality patterns. MQTransformer's advanced attention mechanisms position it as an effective tool for models requiring context-aware forecasting.

- **Informer** [ZZP⁺21]: is a self-attention encoder-decoder framework that utilizes a self-attention mechanism to capture long-range dependencies in time series data more effectively than traditional RNNs. Informer employs a global attention mechanism to dynamically select the most relevant time steps. This combination of self-attention mechanisms enables the Informer to capture complex temporal patterns accurately, setting a high standard for long-range dependency modelling.
- **TimesNet** [WHL⁺22]: is a novel deep neural network architecture for time series analysis based on the idea of transforming 1D time series data into 2D tensors, which allows the network to capture both intra-period and inter-period variations in the data. TimesNet has been shown to achieve state-of-the-art results on a variety of time series analysis tasks, demonstrating that this model outperformed several other state-of-the-art models on short-term forecasting, long-term forecasting, imputation, classification, and anomaly detection.
- **PatchTST** [NNSK22]: is a model designed for long-term forecasting with transformers. It vectorizes time series into patches of a given size and encodes the resulting sequence of vectors via a transformer that then outputs the prediction length forecast via an appropriate head. The model is based on two key components: segmentation of time series into subseries-level patches and channel-independence where each channel contains a single univariate time series that shares the same embedding and transformer weights across all the series. PatchTST has shown significant improvements in long-term forecasting accuracy compared to other state-of-the-art transformer-based models.

2.3 On Times Series Clustering

Clustering is an unsupervised method that divides 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. The most popular clustering algorithm is K-means, which is a method that partitions the dataset into K clusters by minimizing the sum of the squared distances between the data points and their centroids [M⁺67]. In the K-means algorithm, assigning a data point to a cluster is typically based on minimizing the Euclidean distance, defined as:

$$d_{\text{Euclidean}}(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.19)$$

where $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_n\}$ are two time series with the same length n . Specifically, a data point is assigned to the cluster whose centroid has the closest Euclidean distance to that point, and then the cluster centroids are updated based on the new assignments. This process is repeated until the assignments stabilize or a maximum number of iterations is reached.

Clustering can also be applied to time series data. Time series clustering has proven to be a versatile technique, finding applications across various domains. For instance, it has been used in healthcare to identify similar patterns of disease progression and suggest treatment strategies [HDM15]. It has also been employed in finance to detect fraud and assess risk by grouping similar financial time series and pinpointing anomalies [CLKK16]. Additionally, time series clustering has been leveraged in smart city applications, such as traffic optimization, energy consumption management, and public safety [BDN⁺20].

However, time series clustering poses several unique challenges compared to traditional clustering approaches. Time series data inherently exhibit temporal dependencies, meaning that the values at a given point in time are often influenced by the values in previous time steps, complicating the clustering process. Additionally, because of the curse of dimensionality, time series become more equidistant from each other in the high-dimensional space and this phenomenon, known as the concentration of the norm, results in traditional distance measures like Euclidean distance becoming less meaningful, negatively impacting distance-based algorithms such as K-means clustering. Many appro-

aches have been proposed to deal with time series clustering, including applying dimensionality techniques before applying the standard K-means algorithm.

Principal Component Analysis

Principal Component Analysis (PCA) is one of the most popular statistical methods for reducing dataset dimensionality [Pea01]. The data are transformed linearly into a new orthogonal coordinate system with fewer dimensions that preserves most of the variance of the original data. The new orthogonal axes are called principal components. The mathematical formulation of PCA involves calculating the covariance matrix of the data and then finding the eigenvectors and eigenvalues of this matrix. The eigenvectors correspond to the principal components, and their corresponding eigenvalues represent their respective variances. The top k eigenvectors with the largest eigenvalues represent the k principal components that capture the most variance in the 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 [LCD04, Li19]. PCA can be used to reduce the dimensionality of the time series before applying K-means [AAEBS23].

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, learned in an unsupervised fashion [FVA14]. This architecture combines the representation learning power of a variational auto-encoder with the ability of RNN to deal with sequential data. Whereas in a conventional Variational Auto-Encoder, the encoder and decoder are standard feed-forward neural networks, in the VRAE, they are RNNs. The network's parameters are learned by jointly optimizing the sum of the KL divergence between the distribution learned in the latent space and a normal distribution plus the reconstruction error, which is the difference between the original data and the reconstructed one. An illustration of the VRAE is depicted in Fig. 2.1. The variational component of the VRAE can deal with larger time series data without requiring more training epochs. This model has been widely applied in anomaly detection and forecasting time series applications [PS18, ZWYZ21]. VRAE can be used to reduce the dimensionality of the time series and apply K-means on the latent space.

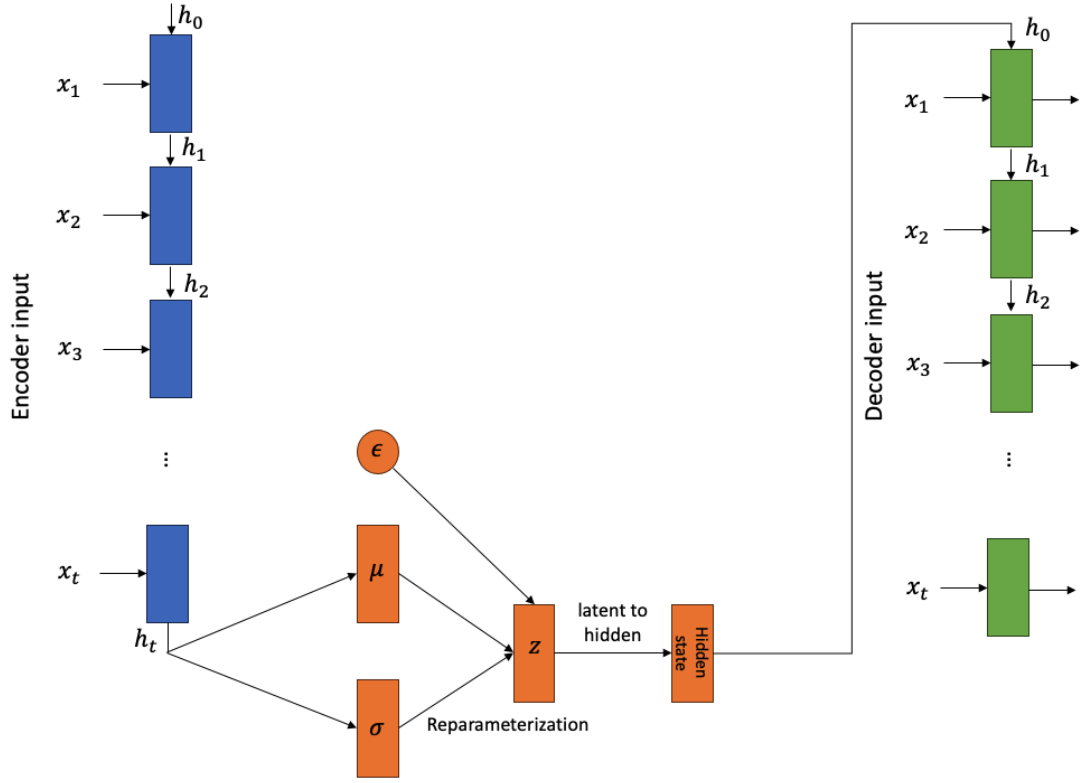


Figure 2.1: Variational Recurrent Auto-Encoder architecture [FVA14]. The network is trained end-to-end to minimise the reconstruction error. After the training is completed, the encoder maps the time series data to distributions in the latent space.

Self-Organising Maps

A Self-Organising Map (SOM) or Kohonen map is an unsupervised ML technique introduced by Kohonen [Koh90]. It is an artificial neural network that differs from standard neural networks because it is trained with competitive learning (unsupervised). First, the SOM is trained to create a lower-dimensional input data representation. Second, mapping uses the created map to categorize further input data. In most situations, the training's objective is to convert an input space into a two-dimensional map space. SOMs are inspired by the brain map for visual and sensory information, which responds similarly when to particular input sequences. In this scenario, the neurons compete among themselves: when the input is fed to the network, the network computes the distance between the input and each weight. An illustration of the SOM is depicted in Fig. 2.2.

This ability to produce a map is a method for dimensionality reduction that is a nonlinear generalization of PCA. Because each map node can be seen

as a cluster medoid, we can use SOM as a clustering technique. SOMs have been widely applied in time series contexts, including forecasting applications [Bar07, CCB11, SE11]

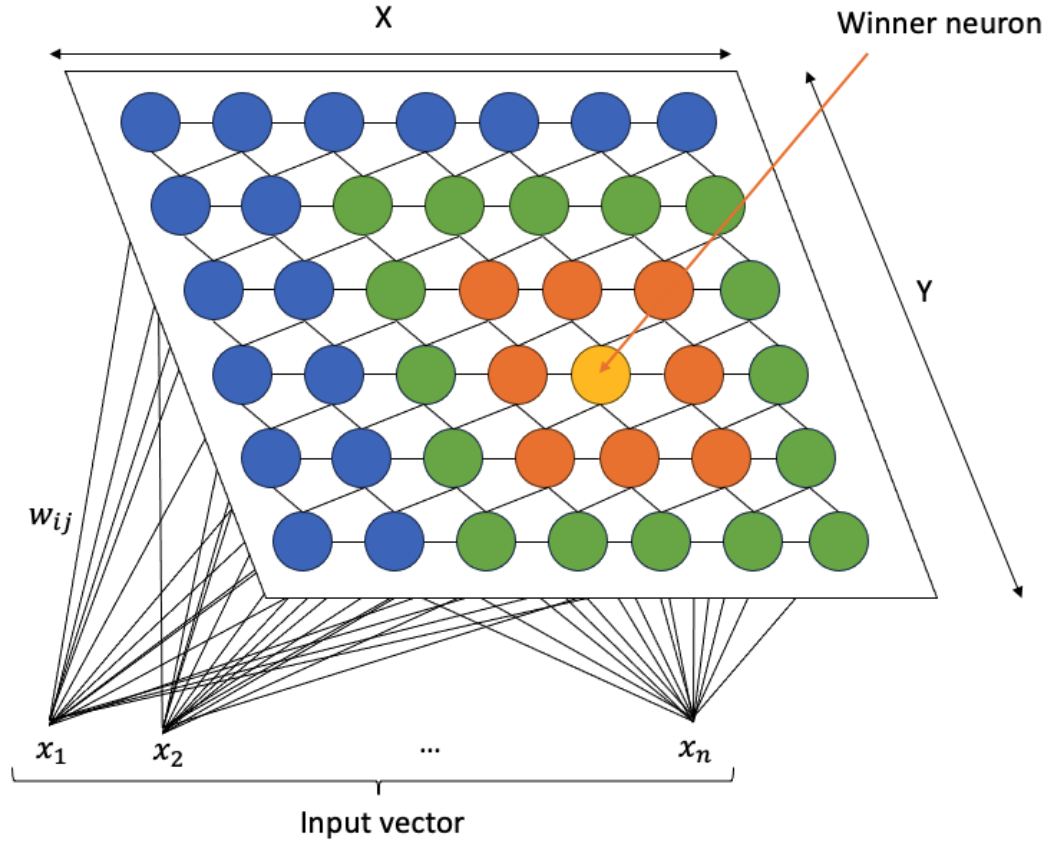


Figure 2.2: Self-Organising Map architecture [SFSHTM20]. The network is trained with competitive learning, where each neuron competes with each other to become the winner neuron, i.e. the neuron that is activated in response to certain input.

Dynamic Time Warping

In the context of time series, the centroid can be computed as the arithmetic mean of each time series timestamp grouped into a cluster. However, Euclidean distance has often been proven less effective and accurate than elastic metrics [PG15, PLEF20]. A popular one is the Dynamic Time Warping (DTW) [PFW⁺14], which can handle time series of different lengths and is invariant to dilation in time. The objective is to find the optimal match, i.e. the warping path between two times series. Given $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_m\}$ are two time series of length n and m , respectively, we can arrange a n -by- m grid, where the point (i, j) of the grid correspond to the data point (x_i, y_j) . The

DTW formulation for (i, j) is defined as:

$$d_{\text{DTW}}(i, j) = d_{ij} + \min\{d_{\text{DTW}}(i-1, j), d_{\text{DTW}}(i, j-1), d_{\text{DTW}}(i-1, j-1)\} \quad (2.20)$$

where d_{ij} is the distance between points x_i and y_j and it is generally the Euclidean distance. DTW offers the opportunity to apply the K-means algorithm directly to the time series data at the cost of extra complexity. DTW Barycenter Averaging (DBA) [PKG11] is generally used to compute the centroids, i.e. time series that minimize the sum of DTW distances to a cluster of time series. The element b_t of the DTW barycenter $B = \{b_1, b_2, \dots, b_n\}$ for the cluster of p time series $\chi = \{X_1, X_2, \dots, X_p\}$ is defined as:

$$b_t = \arg \min_{x_t} \sum_{i=1}^p d_{\text{DTW}}(x_t, x_t^{(i)}) \quad (2.21)$$

where $d_{\text{DTW}}(x_t, x_t^{(i)})$ is the DTW distance between x_t and the corresponding value $x_t^{(i)}$ of the i -th time series. DTW has been used in various time series applications, including sensor-based signal classification in healthcare [VMPS18] and forecasting [YJYB16, MRKV21, PSA21, SSMB24].

Other Methods

Other time series clustering methods include Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and hierarchical clustering.

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters [Lia05]. The process generally starts by treating each object as a singleton cluster. Then, pairs of clusters are successively merged until all clusters have been merged into a single cluster that contains all time series, or it can be interrupted when a predefined number of clusters is reached. Distance metrics like DTW are used to compute the similarity between two time series. Hierarchical clustering has been widely applied in time series classification for data streams [RGP08] and several applications in energy consumption forecasting at different levels, such as household, city, national levels [RLM⁺20, CMA23].

DBSCAN [KRA⁺14] is an unsupervised density-based clustering algorithm. Unlike hierarchical clustering, which requires the number of clusters to be predefined, DBSCAN can automatically detect high and low-density areas: the high-density regions are identified as clusters, and low-density regions are considered as noise. To determine high-density and low-density regions, we consider two parameters: search distance and minimum feature per cluster. If a region meets both criteria, it is classified as a high-density region. Regions that do not meet these two criteria are considered low-density regions. DBSCAN has been widely used in various applications in time series analysis, particularly in anomaly detection [Che19, JBP22].

2.4 Conclusion

The chapter gave an overview of the fundamental workload forecasting model that can be used in cloud computing environments. The chapter then discussed the various techniques proposed to handle the uncertainty surrounding cloud workload prediction. The review examines the strengths and weaknesses of different approaches, analysing how researchers have tackled the challenge of modelling uncertainty in this field. However, the chapter also highlights some critical research gaps in this area.

The review identified that traditional statistical methods, while simple and interpretable, struggle with capturing complex, non-linear relationships in cloud workload data. ML techniques, although promising in learning patterns from historical data, can be sensitive to the quality and quantity of training data. DL approaches, despite their superior performance in handling large-scale data and intricate patterns, often face challenges such as overfitting and high computational requirements.

One significant finding is that the existing models often fail to model both two essential types of uncertainty: epistemic uncertainty, which arises from limitations in the model's knowledge or training data, and aleatoric uncertainty, which stems from the inherent randomness or noise in the data itself. By not accounting for these different sources of uncertainty, the models may provide an incomplete picture of the potential future workload. Moreover, uncertainty quantification in cloud workload forecasting is still underexplored. The ability to accurately quantify uncertainty can significantly impact the effectiveness and reliability of workload predictions. While some techniques exist, further

research is necessary to develop more robust and comprehensive methods to quantify epistemic and aleatoric uncertainty. This would enable cloud providers to make better-informed decisions. Additionally, while some advanced methods have been proposed, they often come with limitations, such as high computational complexity and the need for extensive tuning.

In this dissertation, we address these research gaps by proposing a novel approach to model both aleatoric and epistemic uncertainty to predict future cloud workloads. We leverage Bayesian Neural Networks for uncertainty modelling, aiming to provide a balanced trade-off between predictive accuracy and computational efficiency. Additionally, we explore time series clustering for numerosity reduction, enhancing the scalability of the proposed methods. In the context of this thesis, the Bayesian uncertainty modelling approach is positioned within the landscape of state-of-the-art models. By incorporating uncertainty quantification, this approach aims to offer a balanced trade-off between predictive accuracy and computational efficiency, addressing some of the limitations identified in these studies.

Finally, this chapter has provided an overview of time series clustering, delving into its background concepts, algorithms and applications. This comprehensive understanding of the field serves as a springboard for the remainder of this thesis, where we will further develop and validate the proposed approaches in subsequent chapters. By integrating these insights, we set the stage for proposing innovative solutions that address both the need for accuracy and the practical challenges of computational efficiency in cloud workload forecasting.

Chapter 3

Cloud Workload Datasets Preprocessing

Forecasting models rely heavily on real-world data to design, train, evaluate, and fine-tune their practical applications. Many publicly available cloud computing workload datasets have been created to meet this need, which capture the usage patterns, resource demands, and behaviours of applications and users in cloud environments. These datasets are essential in facilitating research, benchmarking, and developing data-driven solutions for cloud optimisation.

Dataset providers use different techniques to clean, anonymise, and format raw data before making it available for analysis. However, the absence of standardised guidelines and documentation, often due to heterogeneous cloud infrastructures, can lead to different interpretations of data and make it hard to preprocess the datasets uniformly, reproduce research findings, compare results across studies, and ensure data quality.

This problem is propagated to the preprocessing step, generally performed by researchers. Inconsistent descriptions of preprocessing methods can cover various areas, such as combining information, extracting features, and reducing noise. These differences in how data is prepared can lead to variations in the dataset characteristics, which may impact research results' accuracy and reproducibility. Although these datasets offer valuable insights into cloud computing practices, the lack of consistency in preprocessing descriptions highlights the importance of standardised methods and clear documentation to ensure that research findings can be replicated and trusted. By addressing these challenges, researchers can create more robust studies in cloud computing and develop

practical solutions for improving cloud services.

In this chapter, we first survey the most popular cloud workload datasets in the literature. Then, we describe the preprocessing steps we apply to workload datasets, underlying the inconsistency and often incomplete description of these steps in the literature. Finally, we describe in detail the datasets we preprocessed and used in our experiments for the rest of the dissertation, providing some insights into the preprocessed datasets. For the rest of this dissertation, we will refer to a dataset as a set of time series describing a phenomenon over time, e.g. the average CPU and memory workload of a cloud cluster cell or a machine in a cluster cell, while we will refer to trace as the original data found in the literature and/or provided by cloud computing service providers.

3.1 Cloud Workload Traces

The literature has explored a large variety of datasets over the years. As mentioned in the previous chapters, the first attempts to explore workload prediction have been applied to HTTP servers, including NASA [KS18, YS21] and Wikipedia [HHKA13, CMRB14] and distributed systems, such as the Unix systems released in 1997 [DO00, YPZ⁺14]. The limitations of these datasets are given by the fact that HTTP and distributed systems are often characterised by multiple seasonalities (daily, weekly, etc.) [Yoa13], making the time series forecasting task much easier than predicting the future workload in cloud computing systems [CHM⁺19].

Simulated and synthetic datasets have become increasingly popular with the emergence of CloudSim [CRB⁺11]. CloudSim is a framework that enables users to simulate and model cloud computing systems and applications. It provides support for creating virtualized cloud environments, allocating resources, and executing workloads. By using CloudSim to generate simulated workload data, one can evaluate the performance and efficiency of various cloud scenarios and policies without having to use real cloud infrastructures. For instance, CloudSim can be used to generate simulated workload data such as the workload of random cloudlets, which consists of a set of cloudlets (units of computation) that have random arrival times, durations, and amounts of data processed [PR21, WFC21], or the TPC-W [Men02] workload that simulates the activities of a business-oriented transactional web server, like an e-commerce site. It measures the number of web interactions processed per second by a system under

different workload profiles [IKLL12, AB13]. One of the most popular datasets available on CloudSim is the PlanetLab [CCR⁺03] workload dataset, which is a real dataset that contains a set of CPU utilization traces from PlanetLab VMs. PlanetLab was a global research network that supported the creation of new network services. These traces were collected during 10 random days in March and April of 2011 [QZG16, FPL⁺16].

The Google Cloud Trace 2011 (GC11) [RWH11] is one of the earliest real-world publicly available cloud workload traces. This is why it became the most popular trace used in workload forecasting experiments, with dozens of articles published in more than one decade, including [DKC12, YPZ⁺14, DKC14]. This trace has been invaluable in understanding and improving cloud infrastructure. Recently, Google has released an advanced and updated version of the trace, the Google Cloud Trace 2019 (GC19) [Wil20], which is still underinvestigated in the literature [CPD23]. Because we are using these traces in the rest of the dissertation, we will give a deeper description of this trace in Section 3.3

Two popular datasets used in research are the Facebook Hadoop workloads and the GWA-T-12 Bitbrains dataset. The Facebook Hadoop workloads [CAK12] contains segments of one hour from Facebook Hadoop traces, which were published as part of the UC Berkeley AMP Lab’s SWIM project [CGGK11]. This trace provides information about the arrival times, durations, and amounts of data processed by the tasks in Hadoop, an open-source framework that enables distributed storage and processing of large datasets across computing clusters. Researchers have utilized this trace in various studies, such as [KWQH20, JLKW20, AJL⁺22]. The GWA-T-12 Bitbrains dataset [SVBI15] is a collection of performance metrics from 1,750 VMs in a distributed DC managed by Bitbrains, a service provider specializing in managed hosting and business computation for enterprises. The dataset is divided into two traces of different sizes, and each file in the dataset represents the performance metrics of a VM. The dataset has been used in [ZUHK⁺17, LFK⁺21, BRK21].

Several cloud service providers, including Microsoft and Alibaba, have made real-world traces from their cloud environments publicly available for research purposes. Microsoft has released the Microsoft Azure 2017 trace [CBM⁺17] and the Microsoft Azure 2019 trace [SFG⁺20]. The former is a trace of the VM workload of Microsoft Azure collected in 2017 and comprises information about approximately 2 million VMs and 1.2 billion utilisation readings. This trace is a subset of the VM workload in one of Azure’s geographical regions. Several

works have used this trace, including [JLKW20, DAMZ21]. The Microsoft Azure 2019 trace is a newer and larger version, containing information about roughly 2.6 million VMs and 1.9 billion utilisation readings. It has been used in [SK21a, SK21b, AJL⁺22]. Both can be used for research on cluster scheduling, resource management, workload characterisation, and performance modelling. On the other hand, Alibaba released the Alibaba Cluster Trace 2018 (AC18) [JQS⁺20] that became one of the most used datasets in the cloud workload prediction literature [ZZCG19, WFC21, VKSR21]. Later, Alibaba released a new version, the Alibaba Cluster Trace 2020 (AC20) [WXY⁺22], also containing information about the workload of a GPU cluster system for two months. It is still not frequently used, with only a few works using it, including [MBK⁺23]. We will give more information about these two traces in Section 3.3.

While most of the literature is focused on predicting the average CPU workload of a cloud computing system or a VM, more recently, there has been greater attention to the GPU clusters, with a few datasets publicly available for research. This includes the Microsoft Philly trace [JVP⁺19], containing representative training workloads of DL models on Microsoft Philly clusters with jobs submitted between August and December 2017. Another one is the SenseTime Helios trace [HSY⁺21] that contains public releases of workload traces from SenseTime Helios DC. The Helios DC is dedicated to developing DL models. It contains 8 independent GPU clusters and over 12,000 GPUs in total. The 6-month workload trace from 4 representative GPU clusters, Earth, Saturn, Uranus, and Venus, are publicly released in this trace. The utilisation of those two GPU traces in experiments in workload prediction is limited to [MBK⁺23].

To conclude, we summarise the traces described in this Section in Table 3.1, including a list of research works in the literature that use them in cloud workload prediction applications.

3.2 Preprocessing

According to the description of the preprocessing phases in the literature, we have formalized a procedure for processing the workload datasets. This procedure can be extended to future datasets, and we have taken care to make it replicable. We have based our choices on the consensus of the literature, ensuring that the procedure is well-motivated.

All the traces we preprocessed contain the measurement of the average resource

Table 3.1: Summary of workload traces in the literature used for workload forecasting and list of related publications. In bold the traces used in this dissertation.

Name	Type	Year	Reference Paper	Research Papers
NASA	HTTP servers	1996	[AW96]	[KS18],[YS21],[KSB21],[KS21]
Wikipedia		2009	[UPVS09]	[HHKA13], [CMRB14]
UNIX systems	Distributed systems	1997	[DO00]	[DO00],[YPZ ⁺ 14],[SYZ ⁺ 18],[ZZCG19],[JLKW20]
PlanetLab		2003	[CCR ⁺ 03]	[QZG16],[FPL ⁺ 16],[DMD ⁺ 17],[MDB ⁺ 18],[ZYY ⁺ 18],[KSB21]
GWA-T-12 Bitbrains		2015	[SVBI15]	[ZUHK ⁺ 17],[LFK ⁺ 21],[BRK21],[KMDA21], [DQY21], [OHU21],[PK22]
Facebook Hadoop		2011	[CGGK11]	[KWQH20],[JLKW20],[AJL ⁺ 22]
CloudSim datasets	Synthetic data	2003	[CRB ⁺ 11]	[PR21],[WFC21]
CloudSim TPC-W		2003	[CRB ⁺ 11]	[Men02]
Google Cloud 2011	Real world cloud traces	2011	[RWH11]	[DKC12],[YPZ ⁺ 14],[DKC14],[THHS14],[HJL ⁺ 14],[BS15],[EVG15],[YZY ⁺ 15],[Vaz15],[ZLZ ⁺ 16],[LLC16],[ASKO16],[FPL ⁺ 16],[BRGB ⁺ 16],[YJYB16],[IM16],[JB17],[ZDY ⁺ 17],[SYZ ⁺ 18],[ZZSG18],[KS19a],[KS19b],[SH20],[GWS20],[KWQH20],[JLKW20],[KSB21],[BLYZ21],[KS21],[BLYZ21],[QYLC21],[PK22],[AS22],[MTSA22],[XSW ⁺ 22],[RBL ⁺ 22],[AJL ⁺ 22]
Google Cloud 2019		2019	[Wil20]	[CPD23]
Microsoft Azure 2017		2017	[CBM ⁺ 17]	[JLKW20],[DAMZ21],[SZL ⁺ 21],[AJL ⁺ 22]
Microsoft Azure 2019		2019	[SFG ⁺ 20]	[SK21a],[SK21b],[AJL ⁺ 22]
Alibaba Cluster 2018		2018	[JQS ⁺ 20]	[ZZCG19],[WFC21],[VKS21],[CLY ⁺ 21],[CZ21],[SZL ⁺ 21],[PK22],[ZLW ⁺ 22],[YGL ⁺ 22],[CZY22],[XSW ⁺ 22],[AJL ⁺ 22]
Alibaba Cluster 2020		2020	[WXY ⁺ 22]	[MBK ⁺ 23]
Microsoft Philly		2019	[JVP ⁺ 19]	[MBK ⁺ 23]
SenseTime Helios		2021	[HSY ⁺ 21]	[MBK ⁺ 23]

utilisation of *jobs* or *tasks* or *instances* that are running in a *machine* belonging to a certain *cloud compute cell* in a certain time interval. The traces contain information on the scheduling decisions of the cloud resource manager. From those traces, we assume we can derive a table similar to Table 3.2, generally by combining information from different tables. In particular, we are interested in the following attributes:

- **Machine ID:** ID of the machine whose statistics are measured
- **Job ID:** ID of the job assigned to the machine with ID *Machine ID*
- **Status:** job status, e.g. running, pending, failed, killed, etc.
- **Start time:** the timestamp of the start time of the measurement
- **End time:** the timestamp of the end time of the measurement
- **Average CPU/GPU units:** the average CPU or GPU units used in the time interval between *start time* and *end time*
- **Average memory units:** the average memory (RAM) or GPU memory

units used in the time interval between *start time* and *end time*

Table 3.2: Example of general workload dataset before preprocessing.

Machine ID	Job ID	Status	Start time	End time	Average CPU/GPU units	Average memory units
1200	8795	Running	50000500000	50000557000	5652.5	424.5
1205	5435	Running	50000604000	50000609000	45676.4	454.0
3625	8641	Failed	50000820000	50000830000	10155.7	787.0
3789	4564	Pending	50000904000	50000927000		
3789	4564	Running	50000927000	50001012000	546476.4	458.0

Given the table and the attributes as in the example, the preprocessing procedure can be summarised as follows:

- We consider the measurements of the average utilisation of resources as done in most of the works (e.g. [WFC21, SYZ⁺18]). Generally, the traces present spikes such as the minimum and maximum utilisation over a time period and can be used in the forecasting [DKC12].
- We consider only the workload in execution, i.e., the jobs with status *Running* in the interval in which the average utilisation of resources is measured, excluding jobs that are queued, failed, etc. In most works, the jobs considered in the workload are not specified.
- As we commonly found in the literature (e.g. [HHKA14, KS19b, LLC16]), we aggregate the average utilisation of processing units and memory in intervals of 5 minutes. This can easily be extended to multiples of 5 minutes by averaging consecutive intervals. Other works, such as [MBK⁺23], use intervals of 1 minute.
- Missing data are ignored for simplicity. The majority of the papers do not specify how to deal with missing records.
- To extract the average CPU and memory workloads for each time window of a given cluster, we consider that a job can run only partially on the 5-minute window of interest. For this reason, we multiply the average CPU and memory workload by a weight corresponding to the run time interval within that window as done in [JB17]. As a result, a task with a one-minute run time in a five-minute window does not count as running for the entire period.

We applied this preprocessing to a few of the aforementioned traces. The pseudocode of the procedure for preprocessing each trace is shown in Algorithm 1.

It is worth noting that step (3) involves extracting the information from each task submitted in a certain machine in a certain cluster cell. However, this operation might involve aggregating information from different tables, depending on the underlying structure of each trace. An example of the resulting dataset after the preprocessing is shown in Table 3.3. In the resulting dataset, intervals are consecutive and have the same length (5 minutes), and the average resource units include the overall workload in that time interval. This aggregation can be performed at the machine level (we include only the jobs with the same Machine ID) or at the cluster level (we include all the jobs from all the machines).

Algorithm 1 Preprocessing Steps for Workload Datasets

- 1: **Input:** Raw dataset containing measurements of average resource utilization.
 - 2: **Output:** Preprocessed dataset with relevant attributes and aggregated metrics.
 - 3: **procedure** PREPROCESSDATASET(*rawDataset*)
 - 4: Initialize *preprocessedDataset* as an empty table.
 - 5: **for** each record in *rawDataset* **do**
 - 6: Extract *Machine ID*, *Job ID*, *Status*, *Start time*, *End time*, *Average CPU/GPU units*, *Average memory units* from record.
 - 7: **if** *Status* = 'Running' **then**
 - 8: Add the record to *preprocessedDataset*.
 - 9: **for** each 5-minute interval in *preprocessedDataset* **do**
 - 10: Initialize *intervalCPUUtilization* and *intervalMemoryUtilization* to 0.
 - 11: **for** each record in the 5-minute interval **do**
 - 12: Calculate *weight* as $(End\ time - Start\ time) / 5\ minutes$.
 - 13: Multiply *Average CPU/GPU units* and *Average memory units* by *weight*.
 - 14: Add weighted values to *intervalCPUUtilization* and *intervalMemoryUtilization*.
 - 15: Store the aggregated values in *preprocessedDataset*.
 - 16: Return *preprocessedDataset*.
-

3.3 Preprocessed Datasets

The selection of workload datasets for processing is critical in any data-driven research or analysis. While the goal for the chosen datasets is to be relevant and representative of real workloads, the selection is critical because the data quality directly impacts the effectiveness of the trained model. In this context, we have chosen to preprocess and use GC11 and AC18, real-world cloud

Table 3.3: Example of the resulting dataset after preprocessing. For each time interval (of 5 minutes each), we have the measurement of the average resources used in the period.

Time Interval	Start time (ms)	End time (ms)	Average CPU/GPU units	Average memory units
1	50000500000	50000800000	1.753E12	5.753E11
2	50000800000	50001100000	1.923E12	6.323E11
3	50001100000	50001400000	1.345E12	5.985E11
4	50001400000	50001700000	1.763E12	5.772E11
5	50001700000	50002000000	1.901E12	6.051E11

traces widely explored in the literature (see Table 3.1). We also preprocessed their newer versions GC19 and AC20. These traces have earned prominence by offering rich and diverse insights into various cloud computing applications. Preprocessing these datasets is worthwhile as it helps ensure the data is clean, consistent, and formatted so that the DL model can understand and utilise effectively. In selecting datasets, we carefully considered their relevance to our research focus on cloud computing, ensuring they aligned closely with our specific goals. High-quality and comprehensive data was a priority, as it is essential to derive accurate insights. Additionally, the complexity and efficiency of preprocessing were taken into account to avoid errors and inconsistencies. Consequently, the preprocessing requirements for other datasets such as the Microsoft traces were significantly more complex and time-consuming, which could have introduced additional challenges and potential inconsistencies in our analysis. Therefore, we opted for datasets that better suited our research objectives and provided a more robust foundation for our data-driven analysis.

3.3.1 Google Cloud Trace 2011

The GC11 is a comprehensive collection of information that offers a detailed snapshot of the operations carried out in a Google compute cell. A Google compute cell refers to a group of machines connected by a high-bandwidth network and controlled by a cluster-management system known as Borg [TBD⁺20], which is Google’s internal cluster management software. Google released this trace as part of its effort to provide researchers with real-world data from production clusters.

The trace contains details about jobs, tasks, machines, events and resource usage. It provides data from a single 12.5k-machine Borg cell from May 2011

with 29 days of workload. The trace contains information such as machine events, collection events, instance events, and resource usage. Machine events describe the state changes of machines in the cell, such as adding, removing, updating, or failing. Collection events describe the life cycle of collections in the cell, such as submitting, scheduling, starting, ending, or killing. Instance events describe the life cycle of instances in the cell, such as creating, placing, beginning, ending, or evicting. Resource usage describes the CPU and memory resources consumed by instances on each machine over time.

Google obfuscated, normalised, and scaled the data to protect confidentiality. For example, most text fields are hashed, resource sizes are linearly transformed using a scaling w.r.t. the biggest machine in the cluster cells and numerical identifiers are mapped onto a sorted series using sequential numbers.

The original trace is available via Google Cloud Storage, and we preprocessed it locally. The preprocessed dataset is depicted in Figure 3.1.

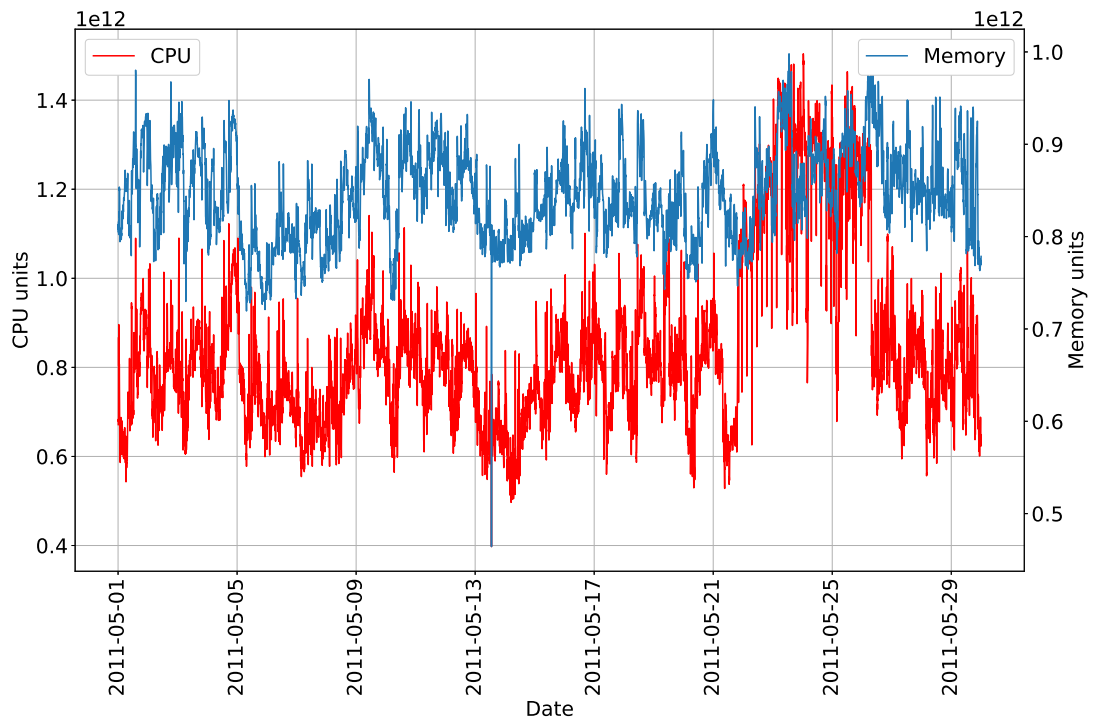


Figure 3.1: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2011.

3.3.2 Google Cloud Trace 2019

The GC19 is a newer released trace that captures the workload and resource utilisation of eight different Google compute cells (from A to H) geographically

distributed worldwide. The median number of machines per cell is approximately 10,000. The trace begins on Wednesday, May 1, 2019, and records 29 days of resource usage of different jobs. This version contains more information with a slightly different structure. Here, Borg supports two kinds of resource requests: jobs and alloc sets. Jobs describe computations a user wants to run, and alloc sets represent resource reservations that jobs can run in. Jobs and alloc sets are collectively called collections. Their components (tasks and alloc instances) are called instances. Like GC11, the trace contains information such as machine events, collection events and resource usage. Moreover, Google obfuscated and scaled the data for privacy issues.

The original trace is huge, with a compressed size of around 2.4 TiB. As a result, the data is accessible through Google BigQuery¹, a fast SQL tool that allows storage and fast analysis of large amounts of data, eliminating the need for local resources and allowing for processing and analysis. However, this approach is more expensive due to the cost of the service. To the best of our knowledge, we are the first to utilise this trace in cloud workload forecasting, and we have preprocessed it and made it available for future studies in this area.

We created eight datasets, one for each cluster cell, from the trace. Figures 3.2 and 3.3 illustrate the preprocessed version of the trace for cluster cells A and B, respectively. For the sake of completeness, we include the plots of the cluster cells from GC19 in Appendix A.1. Additionally, we preprocessed 2,500 machine-level datasets that represent the average workload of individual machines in cluster cell A. Figure 3.4 displays the preprocessed datasets for one of those machines in cluster cell A.

3.3.3 Alibaba Cluster Trace 2018

The AC18 is a trace that contains the information of a production cluster of Alibaba Group, one of the largest e-commerce platforms in the world. The trace covers 8 days and includes about 4,000 machines. The trace contains information about jobs, tasks, machines, events, resource usage, and constraints. It also consists of the directed acyclic graph information of the batch workloads, which provides insights into the dependencies and execution order of the tasks. The original trace is publicly available on GitHub, and we preprocessed it locally. The preprocessed 8-days dataset is depicted in Figure 3.5. This is the only

¹<https://cloud.google.com/bigquery>

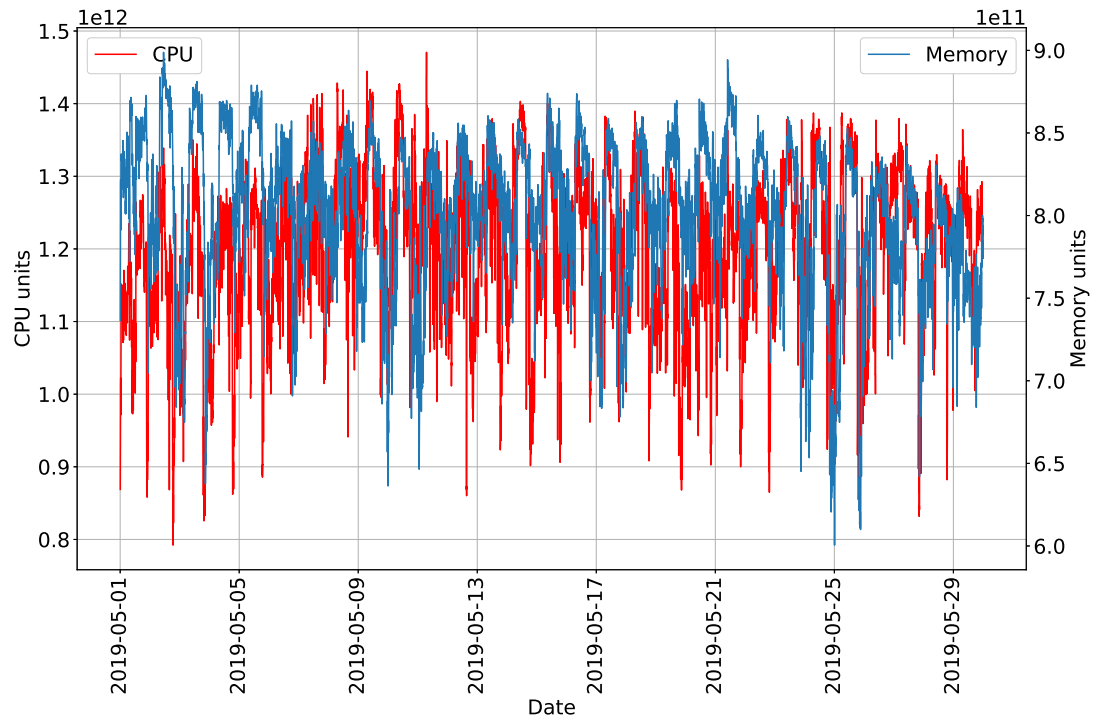


Figure 3.2: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell A.

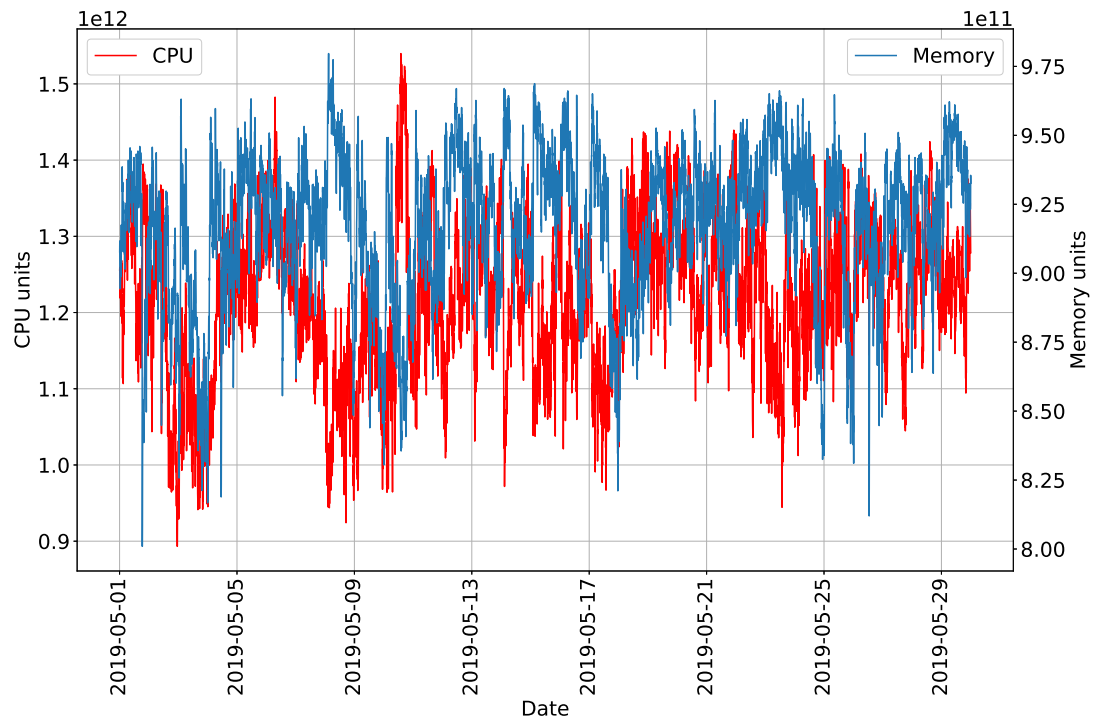


Figure 3.3: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell B.

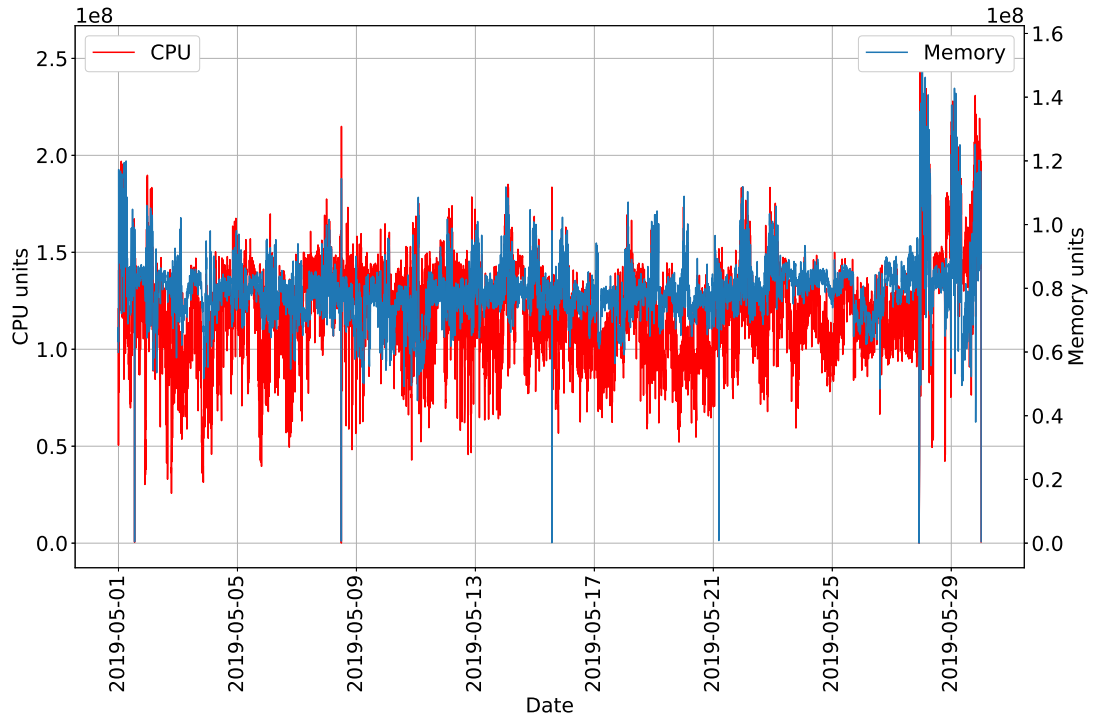


Figure 3.4: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the machine with ID 000020732905 from cluster cell A.

dataset that exhibits a stronger daily seasonality from the second to the last day of the series. However, it is a much smaller dataset compared to the others.

3.3.4 Alibaba Cluster Trace 2020

The AC20 is an updated version of a previous trace containing information about a production cluster owned by Alibaba Group. The trace covers a period of approximately two months and includes over 6,500 GPUs on about 1,800 machines provided by the Alibaba Platform for artificial intelligence (AI). These machines are used for their AI/ML workloads in Alibaba Machine-Learning-as-a-Service. What makes this trace novel is that it includes information not only about average CPU and memory workloads but also about the workload of the GPUs in their cloud computing system. In addition, the trace provides sensor data for these GPUs, including temperature, power, and utilisation. The original trace is publicly available on GitHub, and we have preprocessed it locally. We created two datasets, one for CPU and memory (RAM) workload and one for the GPU and GPU memory workload. Figures 3.6 and 3.7 depict the preprocessed version of these datasets.

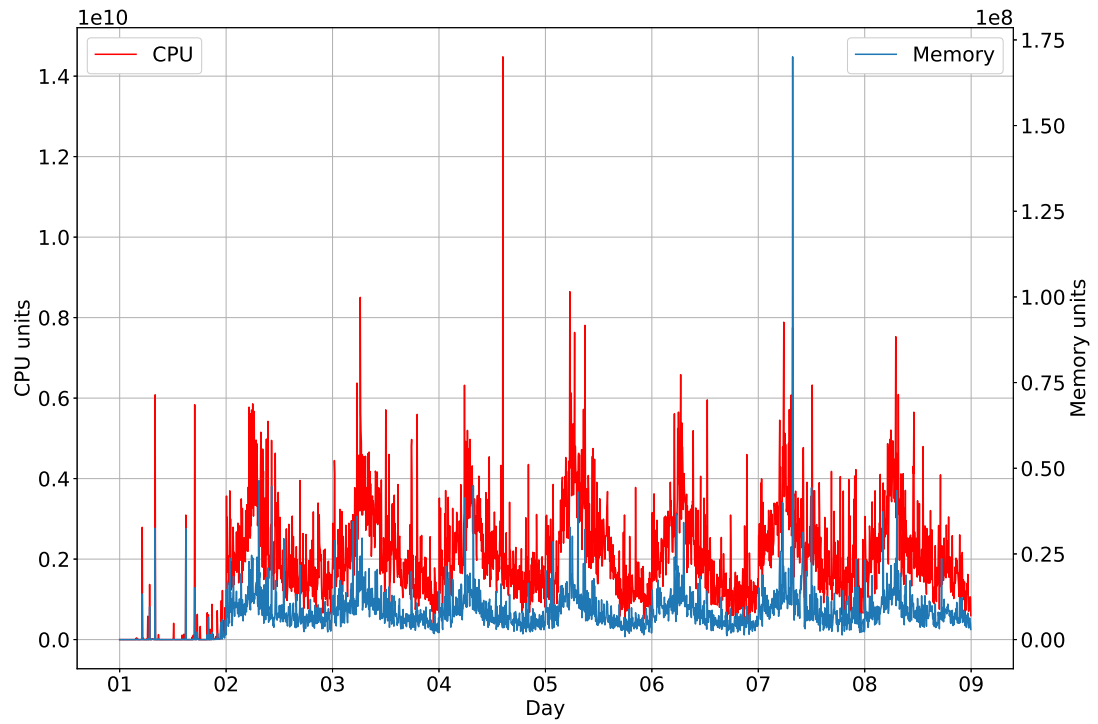


Figure 3.5: Average CPU and memory (RAM) workload of the preprocessed Alibaba Cluster Trace 2018.

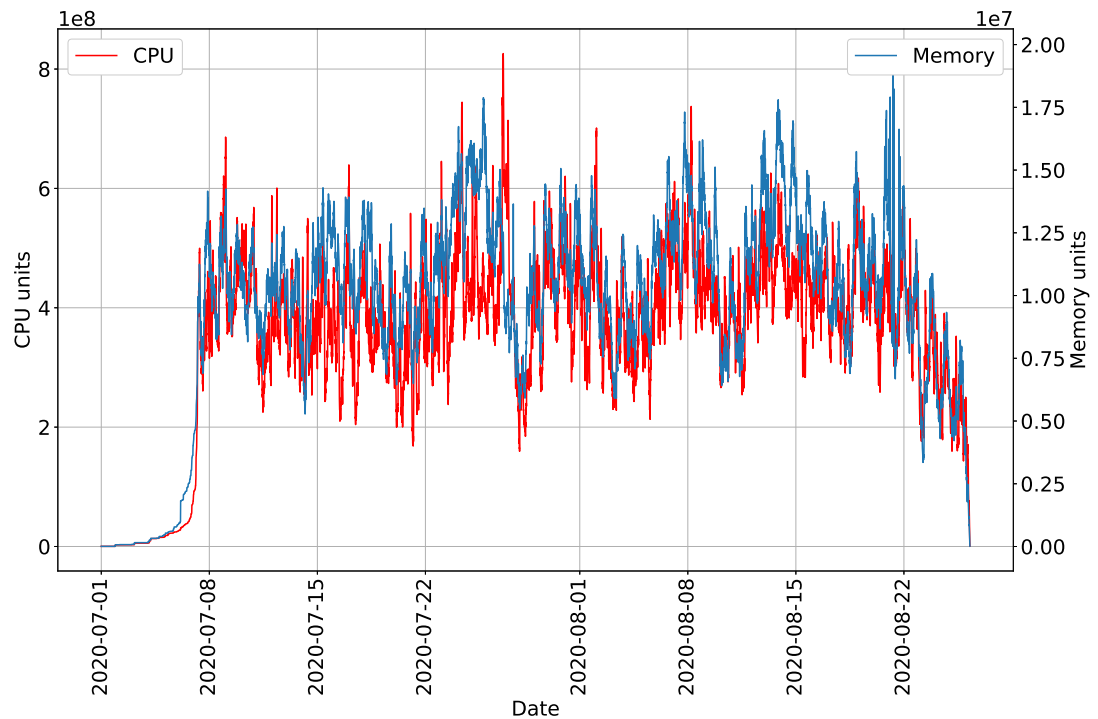


Figure 3.6: Average CPU and memory workload of the preprocessed Alibaba Cluster Trace 2020.

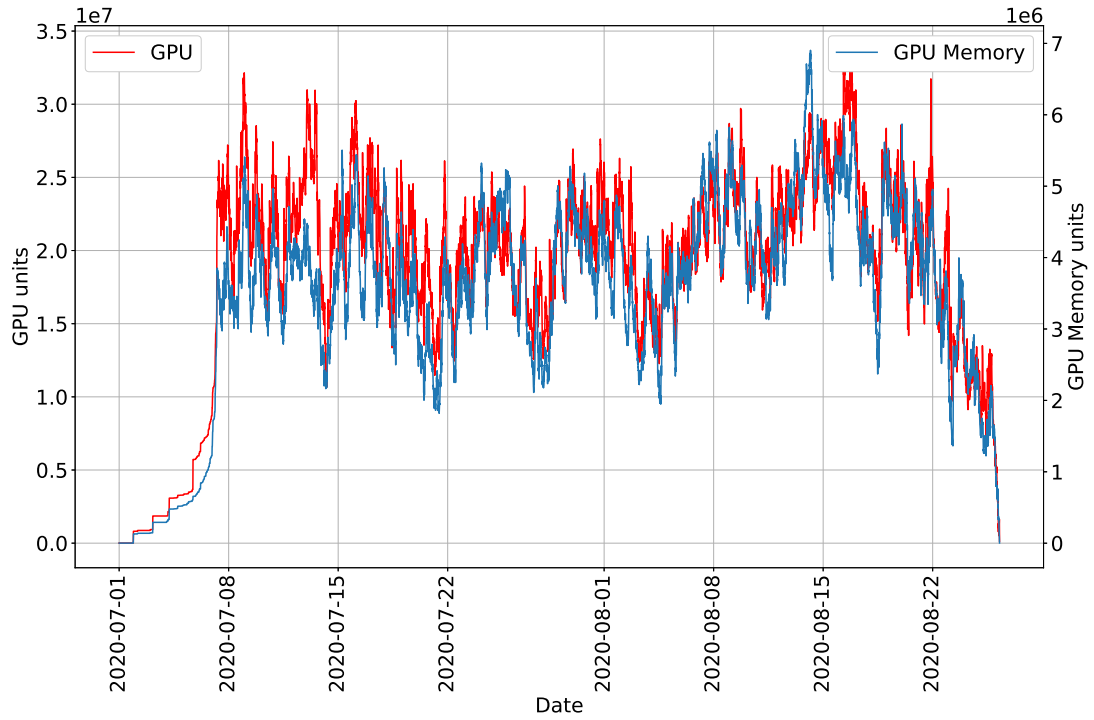


Figure 3.7: Average GPU and GPU memory workload of the preprocessed Alibaba Cluster Trace 2020.

3.4 Analysis

After preprocessing the traces, we can perform an initial analysis to establish experimental assumptions for the rest of the dissertation. This section will delve into the essential components of the analysis and statistical tests applied to workload datasets to derive meaningful insights to design a correct methodology for evaluating the forecasting models. In particular, we are interested in understanding some characteristics of the resulting time series after the preprocessing, such as stationarity, correlation and data distribution.

3.4.1 Stationarity Test

Stationarity is a fundamental concept in time series analysis. A time series is considered stationary if its statistical characteristics, such as the mean, variance, and autocorrelation, do not change over time [HA18]. Stationarity is crucial because many time series forecasting models assume it. The augmented Dickey-Fuller (ADF) test [DF79] can determine whether a time series is stationary. This test is commonly used to identify whether a time series exhibits non-stationary behaviour or a unit root. If the ADF test indicates non-stationarity, it

may require differencing or other transformations to make it suitable for time series modelling. Establishing stationarity is vital in analysing time series data because it enables the application of predictive models that assume a stable and consistent underlying data structure. After preprocessing our data, we applied the ADF test to all the time series data and obtained compelling results. All our time series showed stationarity at a significance level set at less than 1%. This result means that the statistical properties of these time series, including mean, variance, and autocorrelation, remain constant over time, making them suitable for various time series modelling and forecasting techniques. The ADF test successfully validated the stationary nature of our data, providing a solid foundation for further analysis and modelling endeavours.

3.4.2 Pearson Correlation Coefficient

A critical aspect in time series forecasting is whether different time series in the same dataset, e.g., average CPU and average memory of a machine, are correlated such that they can be jointly used for training the predicting models and forecast. This is important in many applications; for example, electricity consumption is generally correlated to the day's temperature. Correlation can be computed using the Pearson correlation coefficient, which evaluates the linear relationship between two time series. This coefficient ranges from -1 to 1, with values closer to -1 indicating a strong negative linear relationship, values closer to 1 showing a strong positive linear relationship and values closer to 0 meaning no correlation. This test is useful to gain valuable insights into how two time series move together or diverge over time.

However, for the validity of this correlation test, the data must satisfy the assumptions of stationarity and homoscedasticity, i.e. constant variance across all periods. Violations of homoscedasticity, i.e. heteroscedasticity, can lead to biased and inefficient estimates, which can compromise the reliability of the Pearson correlation. While we already tested the stationarity of the time series, the Breusch-Pagan test [BP79] is used to identify heteroscedasticity in a regression model.

We found that homoscedasticity for all the datasets. We then calculated the Pearson correlation between processing and memory units for each dataset and found a weak correlation between each pair of time series. This methodology not only quantified the linear relationship between the two variables but also confirmed the robustness of our findings. This contributes to a more reliable

and accurate understanding of our data.

3.4.3 Data Distribution

Understanding and visualising the shape of data distribution is a crucial task in VI. It is typically accomplished by creating a histogram that shows the frequency of normalised resource units. This graphical representation provides a comprehensive view of the data distribution, revealing patterns, trends, and outliers that may not be immediately apparent in the raw data.

The shape of the data distribution, as obtained from the visualisations, plays a pivotal role in VI because it informs our approach to modelling the data and shaping the conclusions we draw from our analysis. For example, recognising a Gaussian shape in the data distribution can lead us to employ statistical models or techniques specifically tailored to this type of distribution.

Figures 3.8 and 3.9 illustrate the distribution of normalised processing units of the 8 compute cells from GC19. Similarly, Figure 3.10 shows the distribution of GC11, AC20, and AC18. It becomes evident from these figures that all the time series can be approximated by a Gaussian distribution. However, each series is characterised by unique parameters, specifically the mean μ and standard deviation σ . These parameters define the specific shape of each Gaussian distribution and are critical in guiding our understanding of the data. For the sake of completeness, we include the plots of the distribution of memory units of the 12 datasets in Appendix A.2.

3.5 Conclusion

Cloud workload datasets have a significant role in developing predictive models. The literature offers many workload datasets derived from various workload traces and forms the basis for creating and evaluating robust and reliable predictive models. However, before these datasets can be used, they undergo a rigorous preprocessing phase. This phase involves designing and implementing a specific procedure tailored to each dataset, ensuring the data integrity is maintained, inconsistencies are eliminated, and the data is prepared for further analysis.

The processed data is then openly shared with the research community, promoting collaboration and collective advancement in the field. Table 3.4 briefly

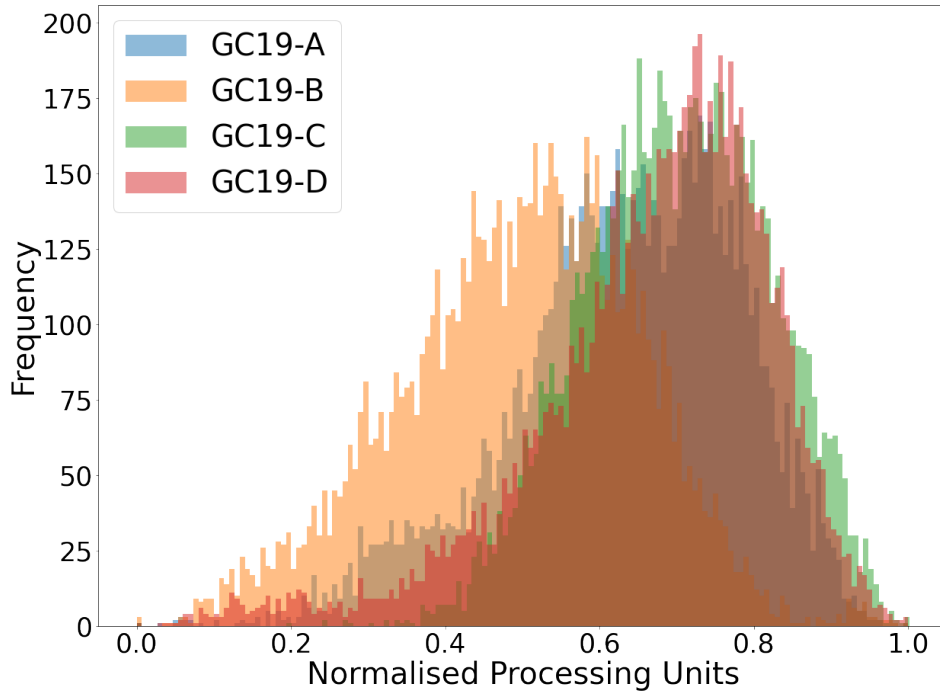


Figure 3.8: Histogram of the normalised processing units of the compute cells A to D from the preprocessed GC19.

summarises the resulting preprocessing datasets. All the preprocessed datasets described above are available for download on GitHub². The repository contains the datasets grouped in separate folders and contains the code and the SQL queries run for preprocessing.

In this chapter, we have presented and meticulously selected several datasets used in the rest of the dissertation. All the time series of the preprocessed datasets are stationary; there is no linear correlation between the processing units and memory units of the same cluster cell/machine, and Gaussian distributions can approximate the data distributions. The stationarity of the time series ensures the reliability and suitability for time series modelling and forecasting. The weak Pearson correlation indicates that the processing and memory units should be considered independently, enhancing the accuracy of predictive models. Finally, the Gaussian nature of the data distribution allows for the application of statistical models tailored for Gaussian-distributed data, ensuring robust and reliable analysis.

²https://github.com/andrearedds/cloud_workload_datasets

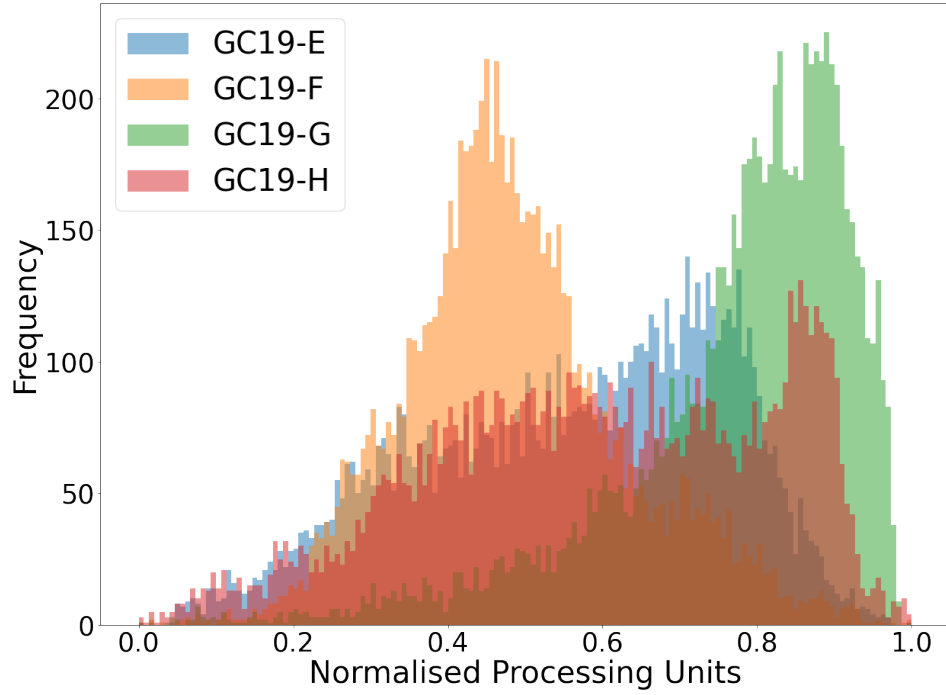


Figure 3.9: Histogram of the normalised processing units of the compute cells E to H from the preprocessed GC19.

Table 3.4: Preprocessed traces summary. For each trace, we identify the number of timepoints and the type of resources included in each dataset.

Trace	Level	# datasets (per resource)	# timepoints (time length)	CPU	Memory (RAM)	GPU	GPU Memory
GC19	Cluster	8	8,352 (29 days)	✓	✓	✗	✗
GC19	Machine	2,500	8,352 (29 days)	✓	✓	✗	✗
GC11	Cluster	1	8,352 (29 days)	✓	✓	✗	✗
AC18	Cluster	1	2,304 (8 days)	✓	✓	✗	✗
AC20	Cluster	1	16,208 (~2 months)	✓	✓	✓	✓

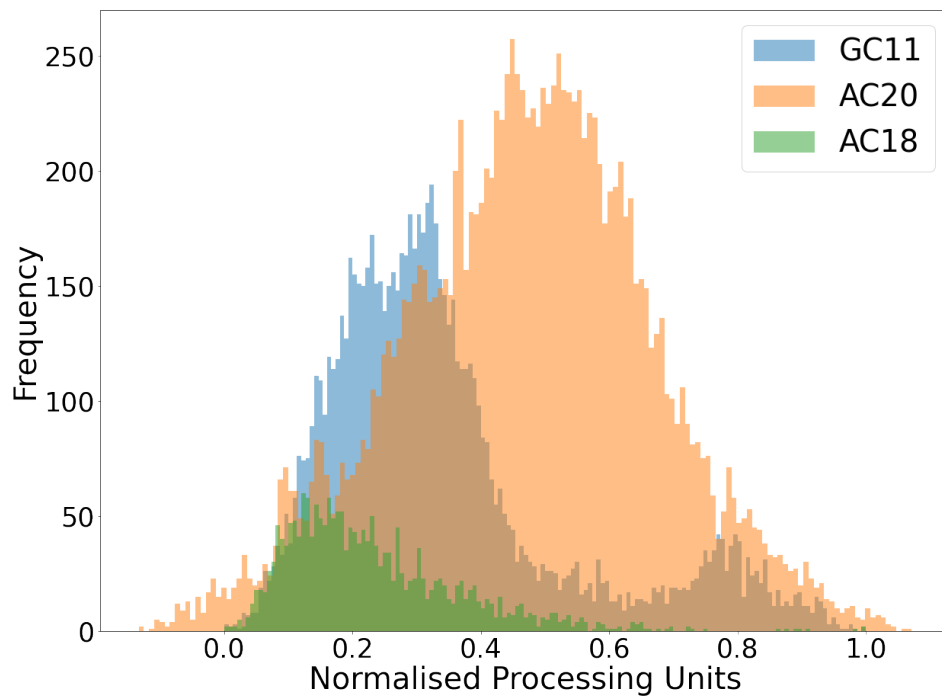


Figure 3.10: Histogram of the normalised processing units of the compute cells from the preprocessed GC11, AC20 and AC18.

Chapter 4

Hybrid Bayesian Neural Network for Modelling Uncertainty in Cloud Workload Forecasting

Workload prediction in cloud computing is challenging because cloud workloads are highly uncertain due to their dynamic and unpredictable nature. Taking uncertainty into account is key for leveraging the robustness of predictive models. In Chapter 2, we provided a comprehensive survey of existing forecasting models and discussed various techniques for uncertainty quantification in the literature. These discussions highlighted the limited investigation of approaches capable of capturing and quantifying the inherent uncertainties in cloud workload data.

In response to this need and inspired by [KHH20], this chapter introduces our new architecture for Hybrid Bayesian Neural Network (HBNN), that we designed to tackle both the epistemic and aleatoric uncertainty, incorporating Bayesian principles.

We explain how we design the architecture of the model in Section 4.1. The chapter also discusses how the HBNN overcomes the challenges of uncertainty quantification in the context of cloud workloads. To evaluate the performance of the HBNN, we conduct several experiments using real-world cloud workload datasets presented in Chapter 3. Section 4.2 defines the methodology used in these experiments, defining how we compare the HBNN with existing forecasting models. In Section 4.3, we thoroughly discuss the results, highlighting the implications of using the HBNN in cloud workload forecasting. The results

of our experiments demonstrate the positive impact of Bayesian layers in the model and the efficacy of the HBNN in capturing and quantifying uncertainties. Finally, in Section 4.4, we summarize the key findings and emphasize the significance of the HBNN in advancing the state-of-the-art in cloud workload forecasting and uncertainty modelling.

4.1 Hybrid Bayesian Neural Network

In Chapter 2, we provided an overview of the most popular forecasting models in the context of cloud workload prediction. We discussed a range of techniques which have been used to forecast future workload, including statistical [VKJ15, WFC21, CMRB14], ML [DKC12, HJL⁺14, GWS20], and DL [JB17, SYZ⁺18, DMD⁺17] approaches. Among these models, LSTM networks have become the preferred approach and one of the most prominent models in this category. LSTMs have shown an exceptional ability to handle sequential data and time series, making them a well-suited choice for various predictive tasks. This is due to the forget gate and input gate, which can avoid pathological data and achieve high accuracy in predictions [GWS20].

Inspired by a DL architecture for univariate forecasting presented in [JB17], we design an LSTM-based architecture by varying the numbers and types of layers in the network. From now on, we will refer to this model as the LSTM model. To predict future resource usage, e.g. memory, the network takes a sequence of past workload values as input. We added a 1-dimensional CNN (1DConv) layer as the input layer. This combination has shown great promise in improving forecasting accuracy and capturing complex patterns in time series data [FFW⁺19, LFK⁺21, KMDA21, OHU21, PK22]. The 1DConv layer is followed by an LSTM layer, which in turn is followed by two dense layers. For univariate one-step ahead forecasting, the final dense layer has just one neuron, which outputs the predicted value of the future workload (point estimate of the next time frame). The resulting model is a DL model that takes past workload as input and predicts future workload as output.

Most prior work in cloud computing shares such an input and output in this task. However, one major gap in the literature is the often-overlooked issue of quantifying uncertainty. The presence of uncertainty in predictions is crucial in decision-making processes, yet it has not received sufficient attention in existing research. This is a deficiency because a single value cannot capture the uncer-

tainty within time series data, highlighting the need for a more comprehensive approach to forecasting. While certain studies have considered the uncertainty factors [CMRB14, MF14, MMF17], most of the research in cloud computing has primarily focused on point estimate predictions, i.e. single values.

To overcome the limitation of conventional forecasting models, we propose the LSTM Distributional (LSTMD) model. Unlike traditional models that provide a single-point estimate, the LSTMD predicts two parameters, namely the mean μ and the standard deviation σ , of a Gaussian distribution ($\mathcal{N}(\mu, \sigma^2)$). This is important because time series data often exhibit complex patterns that a single value cannot capture. By modelling the underlying distribution of the data, the LSTMD provides a more accurate representation of the uncertainty associated with predictions, including the noise in the time series.

The Gaussian assumption of the output is based on an analysis of the preprocessed datasets shown in Section 3.4 where we found that the Gaussian distribution approximates the underlying real distribution of the workload best. This conclusion is supported by several factors. Empirically, the Gaussian distribution's parameters, such as mean and variance, closely match the statistical properties observed in the dataset, accurately capturing the central tendency and dispersion of the workload data. Theoretically, the Central Limit Theorem suggests that, given a sufficiently large number of independent and identically distributed random variables, their normalized sum tends toward a Gaussian distribution, which is particularly relevant for aggregated workload data. The Gaussian distribution also simplifies the mathematical formulation and computation within the Bayesian layer, facilitating more efficient and analytically tractable inference due to its well-known properties and closed-form solutions for various operations. This approach is similar to the quantile regression method but with a stronger assumption on the shape of the output (quantile regression is distribution agnostic). The architecture of this model is very similar to that of the LSTM model, with the main difference being the output distribution. Therefore, the negative log-likelihood is used as a loss function.

The LSTMD model has two objectives: first, to provide more comprehensive and useful information that can be used in scheduling decisions, and second, to appreciate the contribution of the model's components in the cloud workload prediction task.

The use of LSTMD helps address the challenges associated with the stochastic nature of data. However, it is important to distinguish between epistemic and

aleatoric uncertainty. Epistemic uncertainty arises due to a lack of knowledge or information. BNNs can help disentangle and quantify these distinct sources of uncertainty by incorporating Bayesian principles into the neural network architecture. Chapter 2 explains how BNNs can provide a more nuanced understanding of predictive uncertainty. The probabilistic nature of BNNs allows for better handling of outliers and noise in the data, resulting in more reliable and stable predictions. This increased robustness can be especially beneficial in real-world applications where data quality may vary or be subject to anomalies. Creating a fully Bayesian network by making all the components of the network Bayesian would address all the challenges related to the robustness of the model.

It is important to note that using BNNs has certain limitations. Firstly, they require much more data than traditional NNs. Obtaining a large amount of data is often not feasible, and collecting new data can be an expensive operation. Additionally, optimizing the network via Bayes by Backprop is more computationally expensive than using the standard gradient descent algorithm.

When designing neural networks, it is important to balance various factors such as computational overhead, data granularity, accuracy, and the ability to model uncertainty. One way to achieve this balance is by selectively applying Bayesian principles. Typically, this involves using the Bayesian framework only for the last layer of the neural network [KHH20]. By doing this, we can take advantage of the benefits of uncertainty quantification provided by Bayesian methods without dealing with excessive computational overhead. This trade-off allows us to maintain a practical balance between all these factors.

We aim to create the HBNN by modifying the LSTM architecture. Specifically, we replace the deterministic fully connected layer with a Bayesian layer. This layer is associated with a probability distribution of the weights for each neuron and has a Gaussian prior distribution for regularization. Additionally, the model predicts the parameters mean (μ) and standard deviation (σ) of a Gaussian distribution. To determine the likelihood of the actual demand (targets) based on the model's distribution, we use the negative log-likelihood as the loss function since the output is a distribution and not a point estimate.

A graphical representation of LSTM, LSTM and HBNN models, highlighting differences, is shown in Figure 4.1. In particular, the details, the mathematical formulation and the time complexity of each layer are as follows:

- **Input Layer:** Accepts past workload values $X = [x_1, x_2, \dots, x_t]$, where x_t

represents the workload at time t . The complexity is $O(1)$ since it just accepts the input data.

- **CNN Layer (1DConv):** Captures local patterns within the time series data. It takes X as input, outputs $\text{Conv1D}(X)$, where $\text{Conv1D}(X) = W_{\text{conv}} \cdot X + b_{\text{conv}}$, with W_{conv} and b_{conv} weights and bias, respectively. The complexity is $O(n \cdot k)$, where n is the length of the input sequence and k is the kernel size.
- **LSTM Layer:** Handles sequential data and captures long-term dependencies. It takes $\text{Conv1D}(X)$ as input, outputs h_t . The LSTM layer equations include the following, where σ is the sigmoid function, $\tanh$ is the hyperbolic tangent function, W and U are weights, and b is the bias.

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\ h_t &= o_t \cdot \tanh(C_t) \end{aligned}$$

The complexity is $O(n \cdot d^2)$, where d is the number of LSTM units. This accounts for the internal operations within the LSTM cell.

- **Dense Layer:** Applies a non-linear transformation. It takes h_t as input and outputs $\text{Dense}(h_t)$ where $\text{Dense}(h_t) = W_{\text{dense}} \cdot h_t + b_{\text{dense}}$ and W_{dense} and b_{dense} are weights and bias. The complexity is $O(d \cdot p)$, where d is the number of LSTM units and p is the number of units in the dense layer. In the case of LSTM and LSTMD, an additional dense layer with p_2 number of units and similar complexity is added to the network, while in the HBNN is replaced by a Bayesian layer.
- **Bayesian Layer:** Incorporates Bayesian principles to model uncertainty. It takes $\text{Dense}(h_t)$ as input and outputs a distribution over weights. Each weight w in the layer is associated with a Gaussian distribution $w \sim \mathcal{N}(\mu_w, \sigma_w^2)$. The layer's output is given by:

$$y = \sigma(W \cdot h + b), \quad W \sim \mathcal{N}(\mu_W, \sigma_W^2), \quad b \sim \mathcal{N}(\mu_b, \sigma_b^2)$$

The complexity is $O(p \cdot p_2)$, where p is the number of units in Dense Layer

and p_2 is the number of units in the Bayesian layer. The computation involves sampling from distributions and additional overhead from maintaining distributions for weights.

- **Output Layer:** Predicts the mean (μ) and standard deviation (σ) for the Gaussian distribution in the case . Takes $\text{Dense}(h_t)$ as input, outputs μ and σ . The complexity is $O(p_2)$ for both mean and standard deviation predictions for the HBNN and LSTMD, while is $O(p_2)$ for the LSTM with a point prediction.

In conclusion, this section has shown how we derived our proposed model for addressing uncertainty in time series forecasting, introduced the LSTMD model as a means to capture and model the aleatoric uncertainty through probability distributions and proposed the HBNN also to quantify the epistemic uncertainty and enhance model robustness. Moreover, the trade-off between runtime performance and the depth of Bayesian integration in the network architecture has been highlighted as a key consideration in practical implementation with a final architecture, which is Bayesian only in the last layer.

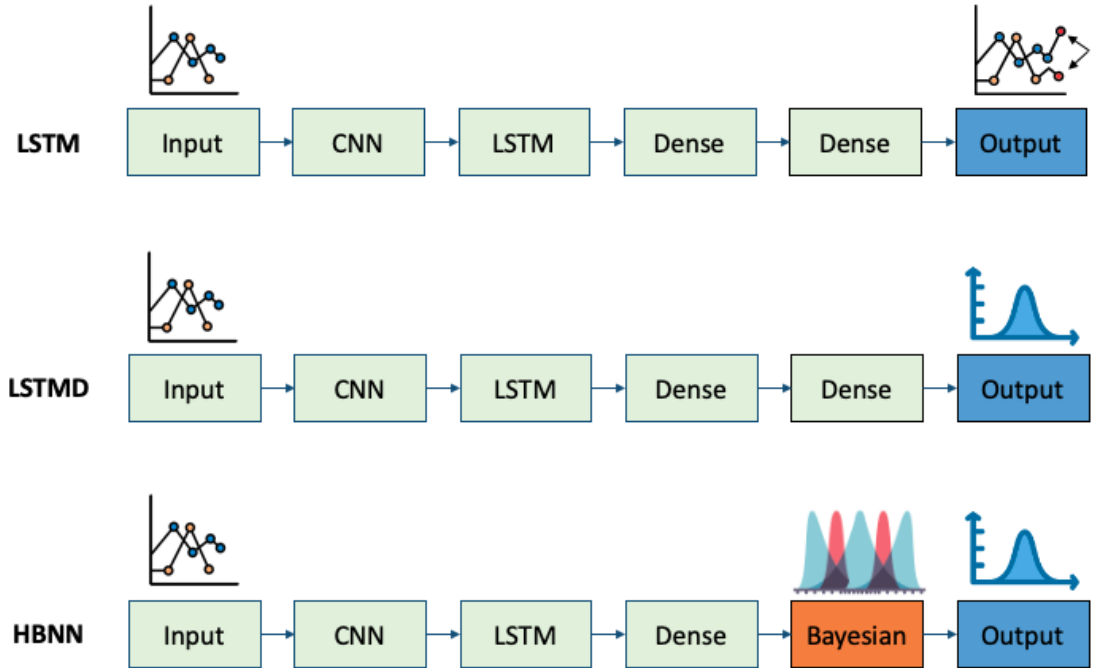


Figure 4.1: Graphical representation of the DL architecture proposed. For each of the three models (LSTM, LSTMD and HBNN), we apply minimal changes to the layer to appreciate the impact of each component in terms of accuracy and uncertainty modelling capabilities.

4.2 Experiments

The goal of the offline experiments presented in this section is to assess the contribution of the components proposed in the LSTMD and HBNN architectures and show the potential benefits of considering the uncertainty in cloud workload prediction.

First, we evaluate the Bayesian layer and/or the output distribution effects on the performances in terms of point estimation with MSE and MAE metrics. Then, we analyse the fitting of the predicted probability distributions that include a measure of uncertainty. From the probability distribution, we can compute confidence intervals that we can use in our evaluation and relate it to satisfying a certain QoS. Finally, we evaluate the runtime performance of the forecasting models.

The experiments are run over the preprocessed historical workload of the 8 cluster cells from GC19 presented in Chapter 3.

4.2.1 Methodology

In the cloud workload prediction task, the objective is to predict the future resource workload based on historical data. Generally, the forecasting model is fed with the past values where each timestamp, as from our preprocessing presented in Chapter 3, corresponds to the average utilisation of a resource in a 5-minute interval. In our experiments, we empirically found that the best input size is 144, corresponding to the 12-hour average workload. Since we need to give the resource manager enough time to exploit the forecast for operations such as resource allocation or machine pre-provisioning, we forecast the workload of the 5-minute interval 10 minutes in the future (forecast horizon or prediction length) with respect to the historical data because 10 minutes are enough for most applications [MLH10, BRGB⁺16]. Applying the direct multi-step ahead forecasting strategy where an independent model is trained for each forecast time step, we forecast only the average workload in the time interval between 10 and 15 minutes ahead as depicted in Fig. 4.2. While the resource optimisation stage would require a complete multi-step ahead prediction for consecutive time intervals, our analysis is focused on evaluating the forecasting model with the uncertainty of its prediction, for which one interval is sufficient.

For each of the eight datasets in GC19, we train the forecasting models by splitting the datasets into two parts. The first 80% of each dataset is used as

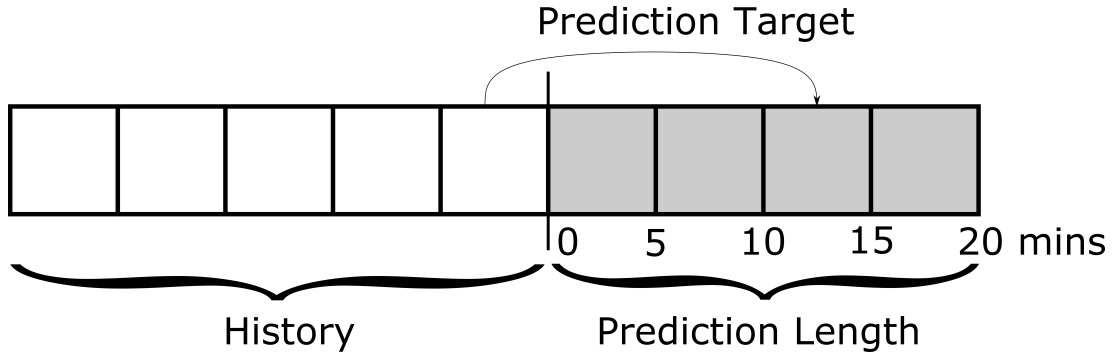


Figure 4.2: Prediction target of our models. Each square represents a 5-minute interval of the cloud workload. In this dissertation we forecast 10 minutes ahead.

the training set (corresponding to about 23 days). The remaining part is used as the test set (about 6 days). The networks are optimized and trained for each cluster to predict the future workload of one resource at a time. Before training, data is scaled in the range $[-1, 1]$ because the scaling strategy can speed up the convergence when using the gradient descent algorithm [HPK11]. More details on the implementation of the predictive models can be found in the shared repository¹.

We optimise the hyperparameters and train five forecasting models described in Section 4.2.2, two of which are statistical methods. We train them 10 times for each cluster and resource with random initialization to validate the results. We select the best-performing model in terms of MSE. The metrics used for the evaluation are described in Section 4.2.3.

4.2.2 Forecasting Models and Optimization

The 8 preprocessed datasets from GC19 are used to train five forecasting models: two statistical methods ARIMA and GARCH, described in Section 2.1.1, and the three DL models presented in Section 4.1: LSTM, LSTMD and HBNN. In this section, we want to provide an overview of the optimisation and training of these models.

In the ARIMA model, three parameters are used to model the time series. They are associated with the auto-regressive, the integrated and the moving average parts of the model and are estimated via MLE. We use it because it is a well-known baseline in workload prediction [CMRB14]. We found the best order

¹<https://github.com/andreareds/UncertaintyCloudWorkload>

of the model for each dataset via grid search by splitting the training set into two parts where the second, corresponding to the 20% of the data, is used as the validation set. We selected the best parameters according to the Akaike Information Criterion (AIC)[Aka98]. This model is implemented through the Python library *statsmodel*². Because of the size of the datasets (and the 5-minute granularity), ARIMA does not converge with all the data available. For this reason, a sliding window of the past 288 timestamps is used to retrain the model at each timestamp.

The GARCH model aims to capture the volatility of a time series [Bol86] on the assumption of Gaussianity. We use it for two reasons. It was previously used as a baseline for cloud workload prediction in Cloud Computing in [BS15] and because this model focuses on modelling the variance of the output, while for the mean component, a simple least square method is applied. So the output is similar to that of the LSTMD and HBNN, i.e. the mean and the variance of a Gaussian distribution. Similarly to ARIMA, we found the best parameters using a grid search based on the AIC score of the validation set. The model's parameters are estimated via MLE. We also experimentally found that keeping a sliding window of the past history instead of the whole dataset is faster and provides better results. We used a sliding window of 288 timestamps (1 day), and we retrained the model at each forecast step. The model is implemented using *arch*³ library in Python.

Regarding the DL models, the motivation behind their design has been described in Section 4.1. The type and the number of layers in the LSTM model (implemented in *Keras*⁴) were found via the hyperparameter tuning using the *Talos*⁵ optimization library in one of the datasets. After defining those parameters, we modified the last dense layer and the output layer to derive the LSTMD and HBNN architectures. The other hyperparameters of the networks, including the number of neurons in each layer and the learning rate, are optimised via hyperparameter search for each model and dataset pair. The Bayesian layer of the HBNN is implemented using *TensorFlow probability*⁶. As mentioned before, the MSE is used as the loss function for the optimisation of the LSTM through the Adam algorithm [KB14], while LSTMD and HBNN use the negative log-likelihood. In all the hyperparameter searches and training of the DL models,

²<https://www.statsmodels.org/stable/index.html>

³<https://arch.readthedocs.io/en/latest/>

⁴<https://keras.io/>

⁵<https://github.com/autonomio/talos>

⁶<https://www.tensorflow.org/probability>

we used the last 20% of the training set as the validation set for applying an early stopping policy with a patience of 30 epochs. Therefore, the number of epochs varies for each model and dataset pair.

In selecting baselines for our study, the decision to forgo quantile regression in favour of the LSTMD resulted from carefully considering both methodological and practical factors. While quantile regression provides valuable insights into the entire distribution of a predicting variable, our primary focus was on capturing complex dependencies and uncertainties inherent in the data, with the assumption of Gaussianity in the training data, which speeds up the overall performance of the model. Moreover, predicting a Gaussian distribution allows any confidence level required, while quantile regression requires that the target quantiles be defined before the training. For these reasons, we believe that LSTMD is a reasonable candidate for modelling the aleatoric uncertainty only while minimising the architectural differences with the HBNN model.

4.2.3 Metrics

Point Forecast Evaluation

The first set of experiments compares the model from a point prediction point of view. We evaluate the models in terms of MSE and MAE, widely used metrics in time-series analysis [KSPA21]. In the case of the baseline models, the computation of the errors is straightforward because ARIMA, GARCH, and LSTM produce a single output at each prediction. In the case of HBNN and LSTMD, whose output are the Gaussian distribution parameters, we can consider only the mean μ and ignore the variance [SBY⁺17]. The MSE and the MAE provide a first way to evaluate the predictive models, but they do not account for the impact of probabilistic forecasting. For this reason, the analysis cannot be limited only to these metrics.

Confidence Interval Evaluation

When working with a Gaussian distribution, we can calculate confidence intervals to estimate a parameter, like the population mean, with a desired confidence level. A one-sided confidence interval is a range of values constructed around the mean of the distribution when we are only interested in one direction of the distribution (either the lower or upper tail).

In particular, we aim to compute an upper bound (UB) of the prediction such

that we can correctly forecast the future workload with high probability. In this way, we can relate the output of our forecasting models with the SLA to achieve a specific QoS. Accurate forecasting is the initial step in the complex resource management of the cloud computing system [ACC⁺14].

An upper one-sided confidence interval $\hat{y}^{UB}$ of the predicted Gaussian distribution $\hat{y}$ with mean μ and standard deviation σ with a confidence level of $1 - \alpha$ is calculated as:

$$\hat{y}^{UB} = \mu + Z \cdot \sigma \quad (4.1)$$

where Z is the critical value from the standard normal distribution corresponding to the desired confidence level (e.g., for a 95% upper one-sided confidence interval, Z is approximately 1.645).

Given a one-sided upper confidence bound with a certain confidence level (e.g. 95%), we should be able to correctly forecast with a probability corresponding to that confidence level (e.g. with a 95% probability).

Defining N as the size of the test set, y_i as the target of the i -th instance of the test set and $\hat{y}_i^{UB}$ as the UB of the prediction $\hat{y}_i$ of the i -th instance of the test set, we evaluate the forecasting model based on these metrics:

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

$$SR = \frac{|A|}{N} \cdot 100 \quad \text{where} \quad A = \{\hat{y}_i^{UB} | \hat{y}_i^{UB} \geq y_i, \quad i = 1, \dots, N\} \quad (4.2)$$

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

$$OP = \sum_i^N OP_i = \sum_i^N (\hat{y}_i^{UB} - y_i) \quad \text{s.t.} \quad \hat{y}_i^{UB} \geq y_i, \quad i = 1, \dots, N \quad (4.3)$$

- **Underprediction (UP):** Sum of unmatched demand for the requests that

are above the UB of the prediction, i.e.

$$\text{UP} = \sum_i^N \text{UP}_i = \sum_i^N (y_i - \hat{y}_i^{UB}) \quad \text{s.t.} \quad \hat{y}_i^{UB} < y_i, \quad (4.4)$$

$$i = 1, \dots, N$$

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

$$\text{TPR} = \sum_i^N \hat{y}_i^{UB} \quad (4.5)$$

A graphical representation of the metrics above is depicted in Fig. 4.3.

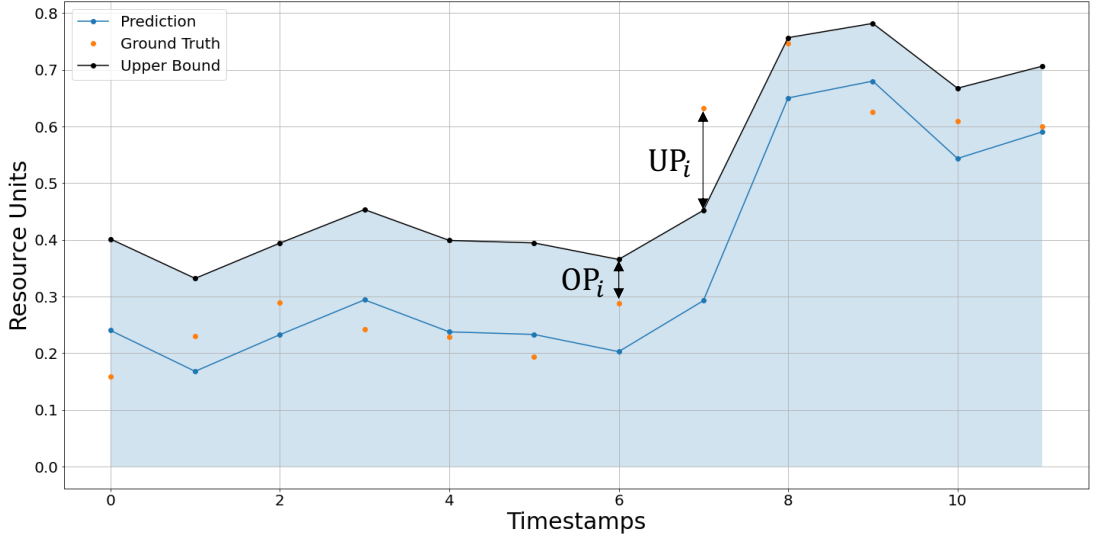


Figure 4.3: Visual representations of the metrics used in the confidence interval evaluation. The TPR is given by the sum of the black dots, while the SR is the percentage of points below the black line.

These metrics aim to evaluate the probabilistic aspects of the forecasting models. We aim to predict a number of resources that, if available, should be able to predict correctly with a given probability. In particular, SR corresponds to the percentage of satisfied requests. However, we also want to reduce the costs of operating a cluster by avoiding overprediction. The OP measures the excess of resources predicted but not needed to serve the upcoming requests, while the UP measures the part of unmatched demand that would have to be processed in the next period. Finally, TPR measures the sum of the predicted bounds; this can be considered as an approximation of the cluster's variable operating cost since a higher prediction causes the scheduler to allocate more resources, and so increases the electricity consumption.

The LSTM does not capture the concept of uncertainty in its point estimate. To compute a UB of the upcoming demand prediction, we add an extra number of resources (e.g. 5% of the total available resources) to the point prediction of the model. This threshold approach has been previously used in the literature [MMF17, MF14]. Therefore, the point estimate plus the threshold represents the UB of the prediction for the LSTM. This threshold can be fixed or vary with parameters such as time. Our threshold varies with the confidence level $1 - \alpha$ and is a function of a parameter $\bar{\sigma}$ that we keep constant for all the predictions.

Given $\hat{y}_i$ the i -th prediction of the LSTM, we can compute a confidence interval from the Gaussian distribution with mean $\hat{y}_i$ and standard deviation $\bar{\sigma}$. We computed $\bar{\sigma}$ in two ways. In the first case, we compare the LSTM to the LSTM-D and HBNN separately. In each comparison, we manually adjusted $\bar{\sigma}$ to match the success rate achieved by either the LSTM-D or HBNN models in a dataset (fixed threshold). In this way, we can compare the OP, UP and TPR when the two models have the same SR. In the second case, we learn $\bar{\sigma}$ from the training set (learned threshold). Given a confidence level of $1 - \alpha$, we computed $\bar{\sigma}$ such that we obtain a SR equals to $1 - \alpha$ in the training set.

Runtime Performance

One of the most important aspects of workload prediction is related to its applicability in real-world scenarios where the forecasting model is used to predict the future demand, and the resource manager will use the forecasts to allocate the resources. We measure the runtime performance of the predictive models and compare them in terms of training time, fine-tuning time and inference time. The training phase refers to training the model from scratch, while the fine-tuning phase aims to refresh the networks' weights based on newly available data. We measure the training time for different sizes of the training set, corresponding to the 20%, 40%, 60% and 80% of the whole datasets, respectively. We measure the fine-tuning time where the fine-tuning using newly available data is performed every 6, 12, 18 and 24 timestamps, corresponding to a time period of 30, 60, 90 and 120 minutes, respectively. Finally, we measure the inference time, i.e. the time necessary to compute a new forecast (one sample) given the history as input.

4.3 Results and Discussion

4.3.1 Point Estimate Accuracy

We report the MSE and MAE results in Table 4.1.

As expected from the literature, the DL models outperform the statistical methods under these metrics. The GARCH for its simple approach of fitting the mean is the model that achieved the worst performance. To have a good fit of a Gaussian distribution, we need to have a good fit of the mean. For this reason and to have more readable results, we will omit the statistical methods in the following experiments. Having a lower point prediction accuracy, they are consistently outperformed by the LSTM model.

The results show that both LSTMD and HBNN achieve slightly better performance for both CPU and memory workload compared to the LSTM. However, this difference is within the statistical error. We run the Diebold-Mariano test, which is a statistical test used to determine whether the difference in forecast accuracy is statistically significant [DM02]. The Diebold-Mariano test compares the predictive accuracy of two forecast models by evaluating the difference in their mean prediction errors. It calculates a test statistic based on the mean and variance of the loss differential, comparing it against critical values from the standard normal distribution. According to this test at a 95% confidence level, the three DL models have the same accuracy. Therefore, it is hard to draw a clear conclusion on which of the DL approaches is better for point prediction.

Table 4.1: MSE and MAE comparison of CPU and memory demand forecasting for LSTM, HBNN, LSTMD, ARIMA and GARCH.

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

4.3.2 Confidence Interval Evaluation

Given the parameter of a Gaussian distribution and a confidence level $1 - \alpha$, we can compute the UB of the confidence interval and compute the metrics. In all the experiments, we focus our attention on confidence levels of 95%, 97% and

99%, which are high values and correspond to those predictions that require a high level of QoS. We separately discuss the results of the fixed and learned threshold approaches.

Fixed Threshold

Tables 4.2 and 4.3 summarise the average SR, OP, UP and TPR with a confidence level of 95%, 97% and 99% of all the datasets for CPU and memory workload, respectively. As mentioned, to compute the correct threshold for the LSTM, for each confidence level $1 - \alpha$, we varied it until we reached the same SR to the model we are comparing with. This threshold can be expressed as a percentage of the prediction, e.g. a threshold of 20% with a prediction of 100 corresponds to an UB of 120. We can see that with the same SR, the OP, the UP, and TPR are lower in the case of the HBNN for the CPU workload. This means we can achieve similar SR performances by predicting lower resource usage. Similarly, the LSTMD outperforms the LSTM in terms of memory and gets a lower UP for the memory workload. LSTMD appears to outperform HBNN on TPR and OP for the memory workload prediction, but it does so by undershooting on the success rate, with poorer performance on UP. Therefore, the HBNN model seems to be superior in terms of prediction quality. It benefits from predicting the distribution; the bounds (prediction's variance) are higher if the demand in the next interval is more uncertain and vice versa.

This experiment proves that both probabilistic models predict a probability distribution which is tighter to the ground truth than the LSTM model. However, in this case, the threshold is automatically adjusted to satisfy the constraint on the SR, which cannot be applied in real-world applications. For this reason, we propose an automatic way to learn the threshold from historical data based on the confidence level and the SR.

Table 4.2: Confidence interval evaluation for CPU workload of the HBNN, LSTMD and LSTM with fixed thresholds. For a given confidence level (95%, 97% or 99%), the best result for each confidence level from the 4 models is in bold; otherwise, the best of a pairwise comparison is in italics. The threshold is expressed as a percentage of the prediction.

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

Table 4.3: Confidence interval evaluation for memory workload of the HBNN, LSTMD and LSTM with fixed thresholds. For a given confidence level (95%, 97% or 99%), the best result for each confidence level from the 4 models is in bold; otherwise, the best of a pairwise comparison is in italics. The threshold is expressed as a percentage of the prediction.

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

Learned Threshold

Given the confidence level $1 - \alpha$, we can compute the threshold that the model should have to achieve a SR equal to $1 - \alpha$ in the most recent historical data, e.g. in the last 3 days of the training test. The goal is to model the uncertainty of future demand by considering the variability of past workload. We consider this as the baseline approach to predict a bound of the forecast as a function of the confidence level when dealing with a point workload predictor such as the LSTM. We calculate two thresholds: a specific one for each cluster whose results are reported as C-LSTM and an average among all the clusters whose results are reported as AVG-LSTM. We then compare its performances with the HBNN and LSTMD models.

Similarly to the fixed threshold, we summarise in Tables 4.4 and 4.5 summarise the average SR, OP, UP and TPR with a confidence level of 95%, 97% and 99% of all the datasets for CPU and memory workload, respectively. The best model should achieve a SR as close as possible to the given confidence level while using fewer resources. If the SR is too low, the demand has not been correctly forecasted too many times. If it is too high, it could have been possible to correctly predict the future workload with fewer predicted resources, so fewer operative costs. The SR of the HBNN is the closest one to the desired value in four out of six experiments, except for the 99% confidence level for CPU and memory demands. This shows that the predicted probability distribution is really close to the realised workload. However, in those situations, the AVG-LSTM reaches an SR closer to the target, but with a significant increment of TPR (about 0.1% more accurate, but with 2.5% extra CPU units). Similarly, the C-LSTM is only 0.2% more accurate but with 12.2% extra memory units.

A more straightforward way to understand how these metrics vary with the confidence level is by providing a visual representation. In Figures 4.4 and

Table 4.4: Confidence interval evaluation for CPU workload of the HBNN, LSTMD and LSTM with learned thresholds. For a given confidence level (95%, 97% or 99%), the best result from the 4 models is in bold.

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

Table 4.5: Confidence interval evaluation for memory workload of the HBNN, LSTMD and LSTM with learned thresholds. For a given confidence level (95%, 97% or 99%), the best result from the 4 models is in bold.

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

4.5, we plot the desired confidence level versus the SR. The plot is constructed by varying the confidence level and computing the corresponding SR for each model. Ideally, we would like to achieve a SR equivalent to the confidence level of $1 - \alpha$. It is noteworthy that overall, the points are located close to the bisector $y = x$, which indicates that the actual SR is similar to the desired confidence level we aim to achieve. This suggests that the predicted probability is accurate and closely reflects the real distribution. In terms of performance, CPU and memory predictions show some differences. Regarding CPU workload, all the models compared consistently lie below the bisector for high confidence levels. Additionally, the points are very close to each other. However, for memory workload, C-LSTM and HBNN are the closest to the bisector. On the other hand, LSTMD always undershoots the target SR, while AVG-LSTM always overshoots it. In both CPU and memory forecasts, HBNN is the model that is closest to the bisector.

We also plot the SR for the CPU workload prediction as a function of the TPR and OP respectively in Figures 4.6 and 4.7. Similarly, we do it for the memory workload prediction in Figures 4.8 and 4.9.

The plots are drawn by progressively incrementing the confidence level and computing the SR, the TPR and OP. If a model correctly predicts the future timestamps with less uncertainty, it should be able to achieve better success rates with fewer resources. The plot shows that the HBNN model is above the

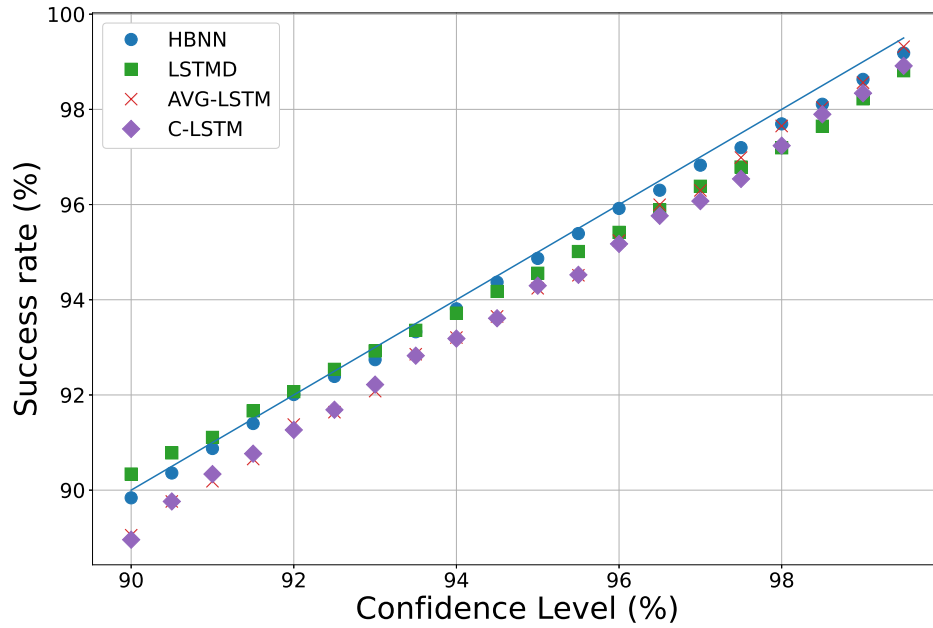


Figure 4.4: SR vs confidence level for CPU workload. The points in the plot are the result of the average of the 8 datasets from GC19.

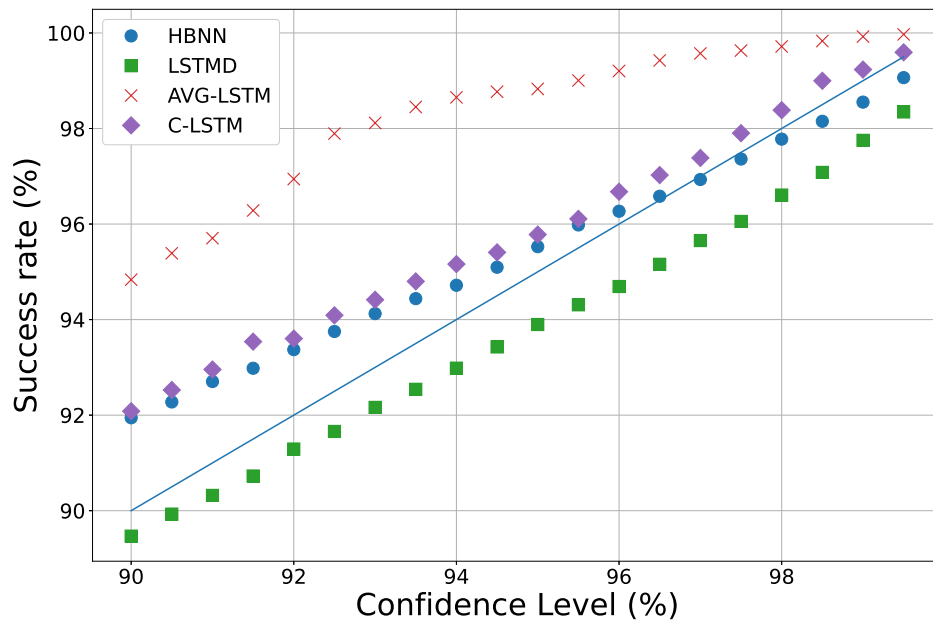


Figure 4.5: SR vs confidence level for memory workload. The points in the plot are the result of the average of the 8 datasets from GC19.

baselines, meaning that with the same number of TPR, the HBNN is capable of satisfying a higher number of requests throughout the analysis, showing the potential benefits of the use of a Bayesian neural model instead of a standard one. Also in this case, there are a few differences between CPU and memory prediction, with the HBNN model performing better for the CPU forecast and the LSTMD performing slightly better for the memory forecast. We believe this is because the memory demand is easier to predict because the memory workload is less variable in its history, and the CPU demand presents more peaks that are harder to predict.

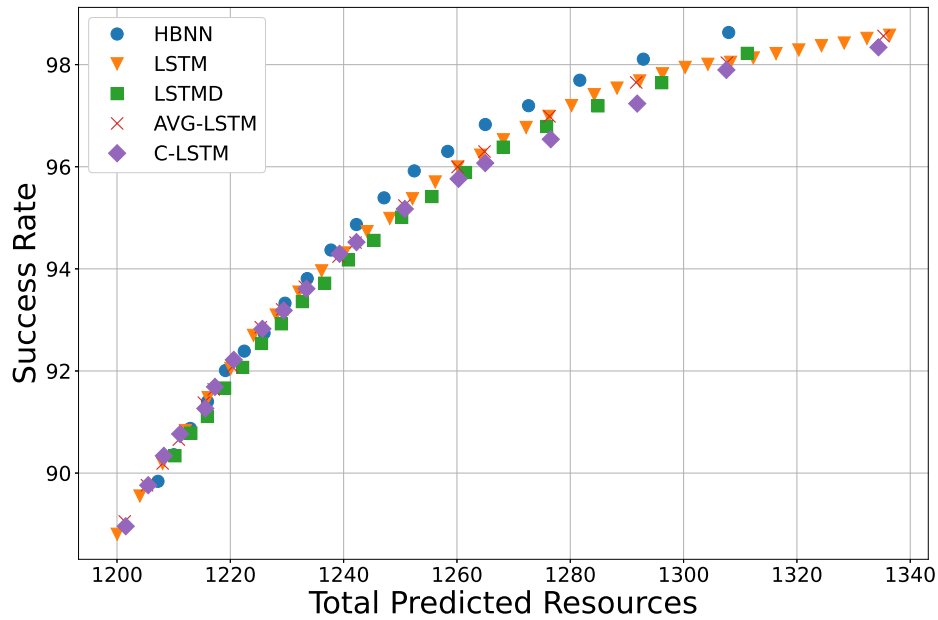


Figure 4.6: SR vs TPR for CPU workload. The points in the plot are the result of the average of the 8 datasets from GC19.

Moreover, we analyze how the uncertainty predicted by the Bayesian network varies across the datasets. Figures 4.10 and 4.11 show the standard deviation and the actual resource demand for the two datasets. For better visualization, we plot only a window of 24 hours. First, we can see that the predicted standard deviation strongly changes over time. In Figure 4.10, the standard deviation and the resources seem to have an inverse relationship; the uncertainty is low when the utilization is high and vice versa. Other clusters display an opposite behaviour, where the higher demand is associated with higher uncertainty; this is particularly clear in the example in Figure 4.11. We believe these are due to the different processes that originate the demand, e.g., users, companies,

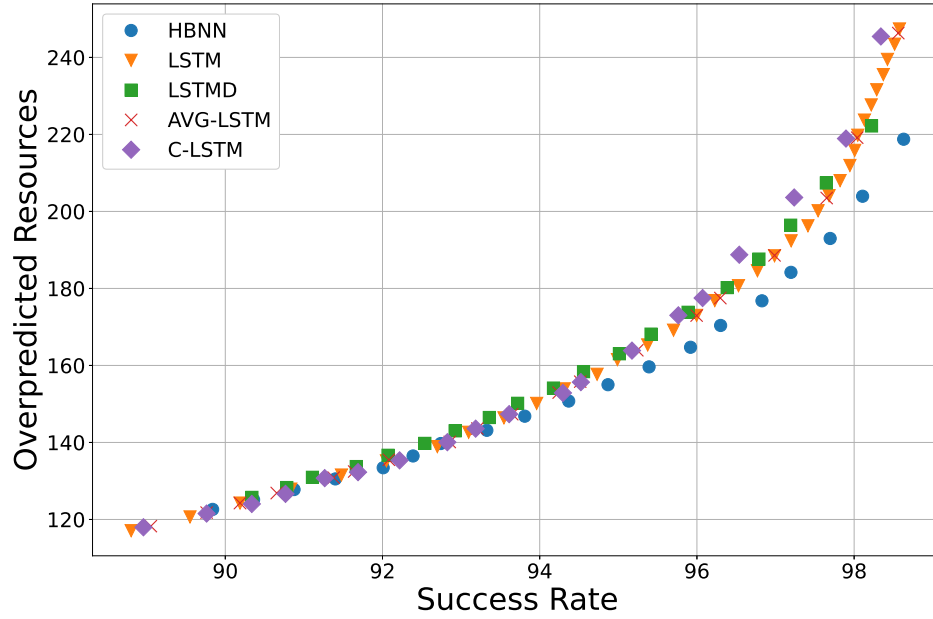


Figure 4.7: SR vs OP for CPU workload. The points in the plot are the result of the average of the 8 datasets from GC19.

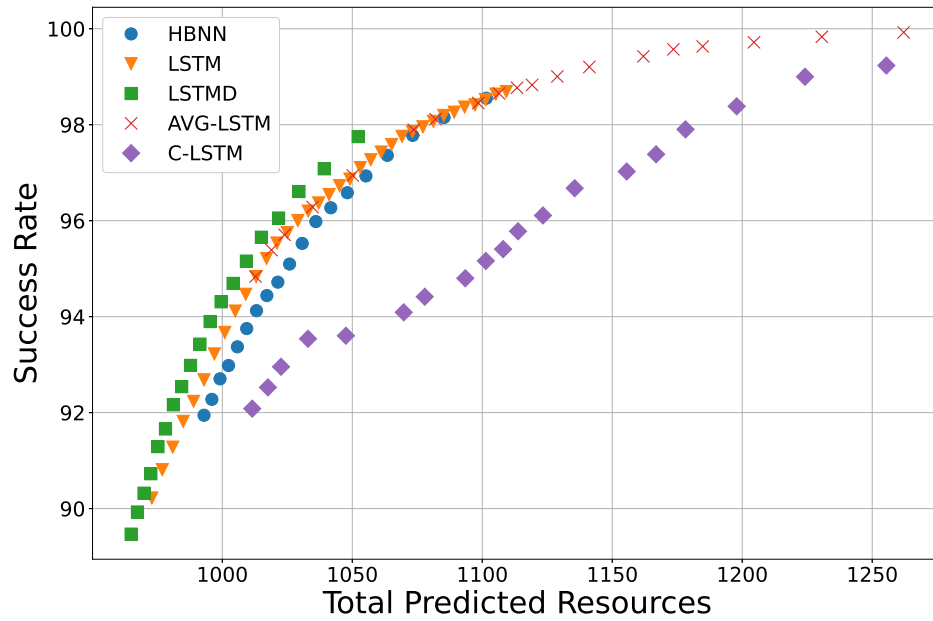


Figure 4.8: SR vs TPR for memory workload. The points in the plot are the result of the average of the 8 datasets from GC19.

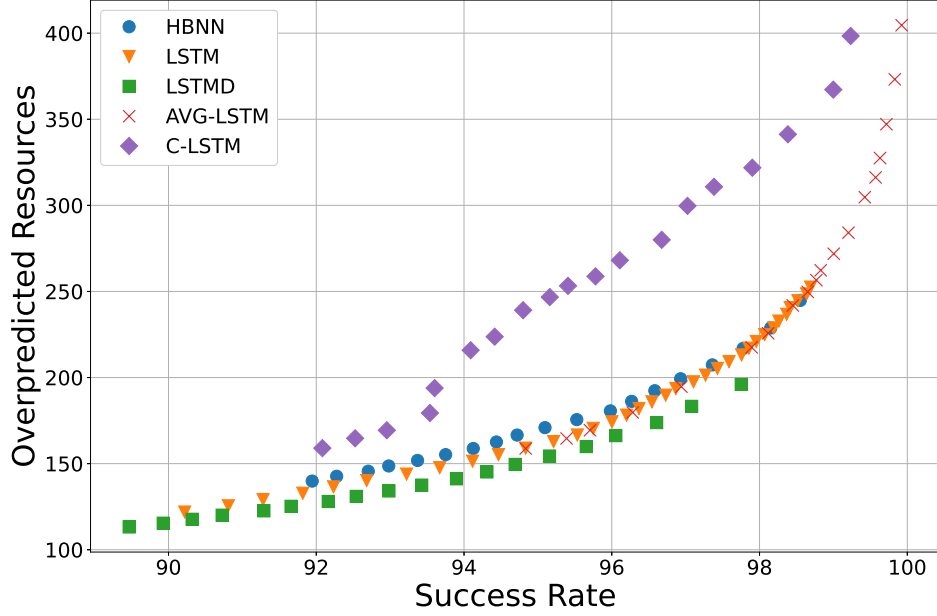


Figure 4.9: SR vs OP for memory workload. The points in the plot are the result of the average of the 8 datasets from GC19.

institutions, and internal jobs. However, it is interesting to see that the network learns these patterns. A common feature is that the standard deviation tends to increase when there is a sharp demand change and tends to stabilize when the trend is steadier.

In Figure 4.12, we plot the average standard deviation of the clusters, which control the uncertainty of the prediction for both HBNN and LSTMD over different prediction lengths. As expected, we can see that the level of uncertainty grows with time: the models reflect the fact that the further in the future the prediction is, the higher the uncertainty.

4.3.3 Runtime Performance Analysis

We report the runtime performance of the LSTM, LSTMD and HBNN in Table 4.6. We can see that the three models do not differ much in fine-tuning and inference time, while for the training, the LSTMD is the fastest model to be trained (the hyperparametrisation search produces smallest networks, i.e. fewer neurons in each layer, for LSTMD, resulting in fewer mathematical operations), followed by LSTM and HBNN. In particular, the difference between LSTMD and LSTM is negligible and the time to train the HBNN increases proportionally to

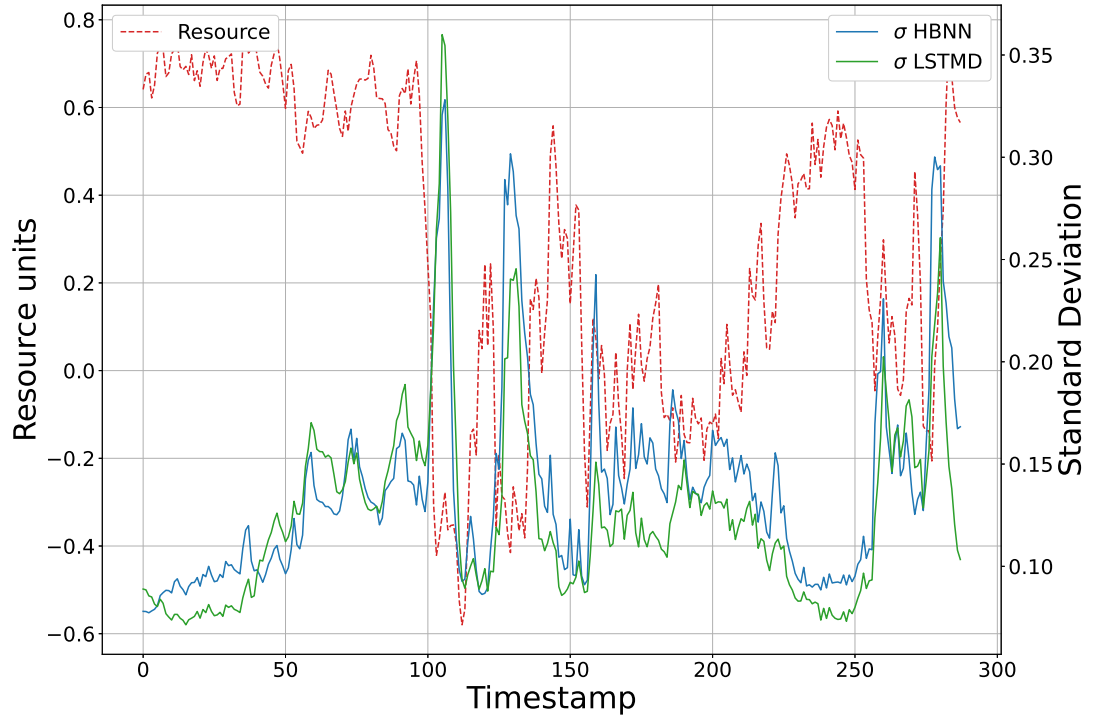


Figure 4.10: Actual CPU workload vs predicted standard deviation for a 24-hour window for the dataset A from GC19

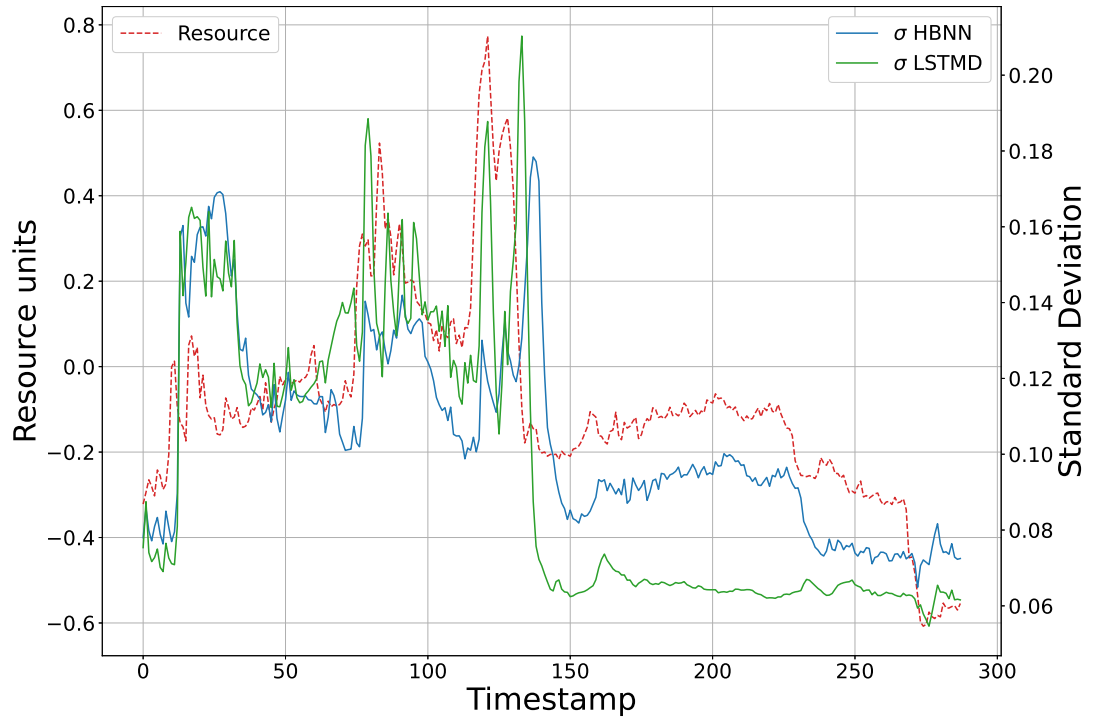


Figure 4.11: Actual CPU demand vs predicted standard deviation for a 24-hour window for the dataset H from GC19

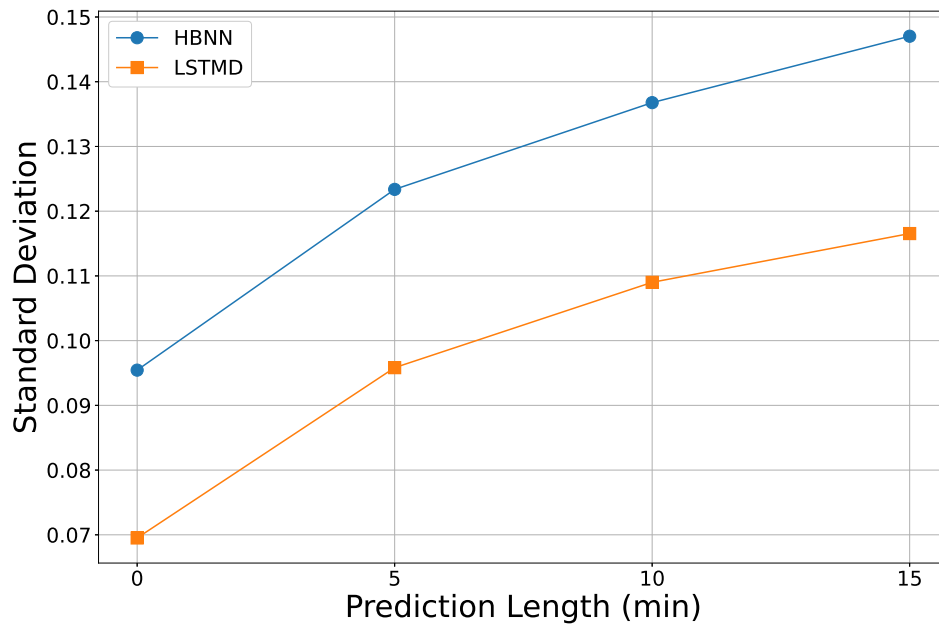


Figure 4.12: Standard deviation of the HBNN and LSTMD models as a function of the prediction length (forecast horizon). The uncertainty increases as the prediction length extends.

the training set size, while for LSTM and LSTMD, the time is almost independent of the training size. All three models are suitable for real-world deployment: training can be done offline or overnight when the workload is generally lower and only when there are big changes in the workload patterns. However, when training from scratch is frequently required and becomes crucial in the application, the LSTMD model achieves the best trade-off between runtime and accuracy.

4.4 Conclusion

In this chapter, we introduced the HBNN model, which predicts the future workload distribution along with a measure of its uncertainty. In particular, HBNN aims to model both aleatoric and epistemic uncertainty. This model is derived from an LSTM architecture with some modifications to make it Bayesian and probabilistic to consider uncertainty. The derivation also includes an intermediate architecture, namely LSTMD, that outputs a distribution while maintaining the characteristics of a deterministic network.

Table 4.6: Runtime performance results of the LSTM, LSTMD and HBNN models. We report the inference time and different training time and fine-tuning time performances, varying the size of the training data.

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

For the first time, an extensive computational study on 8 datasets from GC19 has evaluated the effectiveness of BNNs in workload prediction in cloud computing. The proposed models achieve a point accuracy that is comparable to the DL baseline. The models output a probability distribution that can be used to compute confidence intervals at a certain confidence level. In contrast, deterministic models require additional post-processing of the output, such as adding a threshold to the predicted value to ensure high probability correctness. The HBNN has a higher SR because it commits fewer errors when predicting the same TPR. Similarly, with the same SR, OP and TPR is lower. This is because it reduces the overpredicted resources by adjusting the standard deviation output to fit the uncertainty of the time series. The LSTMD model achieves good performance, too, especially for the memory workload prediction, which presents less variability than the CPU.

A significant novelty of this work is the integration of a Bayesian layer into the neural network to account for uncertainty. By incorporating aleatoric and epistemic uncertainties, the model can better handle the inherent unpredictability in workload predictions. Overall, the model's runtime performance is suitable for practical use in real-world applications. The computational effort the model requires to predict the demand and update its weights is compatible with a practical deployment. The proposed model is an innovative and effective solution for workload prediction in cloud computing that can benefit various industries.

Chapter 5

Bivariate HBNN and Transfer Learning for Uncertainty-Aware Cloud Workload Forecasting

The previous chapter has explored the use of HBNNs for univariate time series forecasting, demonstrating their ability to quantify both the epistemic and aleatoric uncertainty and provide accurate forecasts. However, traditional univariate forecasting approaches often overlook the interdependence between processing units and memory usage, leading to suboptimal forecasting outcomes.

Bivariate forecasting emerges as a promising approach to address the limitations of univariate forecasting by jointly predicting processing units and memory usage. This approach harnesses the dynamic relationships between these resources and can provide more accurate and reliable forecasts [DQY21].

Moreover, 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) [WKW16], 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). In the literature, this is sometimes referred to as domain adaptation [WLL⁺22]. In the cloud computing scenario, TL is useful, for example, when a cloud provider establishes a new data centre in a previously untapped geographical area. Therefore, limited or no historical workload data is available to train a workload predictor for such a region. In this case, one possibility to ob-

tain a workload forecast for the new location is to leverage a predictor trained on data from other locations, i.e. one that can transfer its predictive capabilities to the new region.

In this chapter, we delve into the potential of bivariate forecasting and TL for cloud resource forecasting, particularly in the context of uncertainty quantification using HBNNs. In Section 5.1, we explore how bivariate forecasting can provide more accurate and reliable forecasts of processing units and memory usage while considering the inherent uncertainty in the data. In Section 5.2, we also investigate how TL can leverage large amounts of data to enhance the performance of HBNNs, even without data for the target cluster cell. Section 5.3 concludes the chapter. The experiments are published in [RVC⁺23]. However, this chapter reports a more comprehensive analysis of the results.

5.1 Towards Bivariate Cloud Workload Forecasting

Bivariate forecasting, which involves predicting two variables simultaneously, offers several advantages in cloud workload forecasting over univariate forecasting because it can provide a more comprehensive representation of the uncertainty associated with both processing units and memory usage, allowing for more informed predictions [FS18]. Moreover, jointly predicting multiple variables can help reduce overfitting because the model is forced to consider the relationships between the variables, making it less likely to fit the training data too closely [DQY21].

However, bivariate forecasting models are inherently more complex than univariate forecasting models, and their optimisation can be more challenging [ACOM23]. This makes them more computationally expensive to train and deploy [RT84]. Moreover, having access to only a limited amount of the data can cause convergence issues [ABAS⁺23].

One way to address the convergence issues in bivariate forecasting is to train the model with more data. This can help the model to generalize better and make more accurate forecasts [PAM23]. The first part of the chapter aims to evaluate the uncertainty-aware predictive models, where we compare univariate and bivariate predictors trained and tested on one or more datasets to assess the convergence and generalisation capabilities of bivariate models.

5.1.1 Methodology

The first part of the experiments focuses on evaluating the uncertainty-aware forecasting models, where we compare univariate and bivariate models trained and tested on one or twelve cluster-level datasets for a total of four prediction scenarios. The cluster-level datasets, described in Chapter 3, from GC19 (8 datasets), GC11 (1 dataset), AC18 (1 dataset) and AC20 (2 datasets) are used to train and evaluate the DL models introduced in Chapter 4: HBNN, LSTM and LSTMD.

Scenarios are characterized by specific training data (single or multiple datasets) and a specific prediction task (univariate or bivariate). 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). Furthermore, the model output is set accordingly for bivariate forecasting. In the case of LSTM, the output layers consist of two neurons, one for processing and one for memory units, respectively. The output consists of two Gaussian distributions in the case of LSTMD and HBNN.

In single dataset scenarios, 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 first 80% of each dataset is concatenated, and the training time frames of all datasets are aligned to ensure that one trace cannot exploit information of a particular timestamp in another trace.

The hyperparametrization of multiple datasets and/or bivariate models explores a wider configuration search space due to their increased complexity. Fig. 5.1 shows the architectures used in this set of experiments, where convolutional and dense layers might be repeated multiple times, compared to the simple univariate case in Chapter 4. For each dataset and scenario, the following hyperparameters are tuned 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 also tuned the scaling range and the input size. We found that the best input size is 288, corresponding to

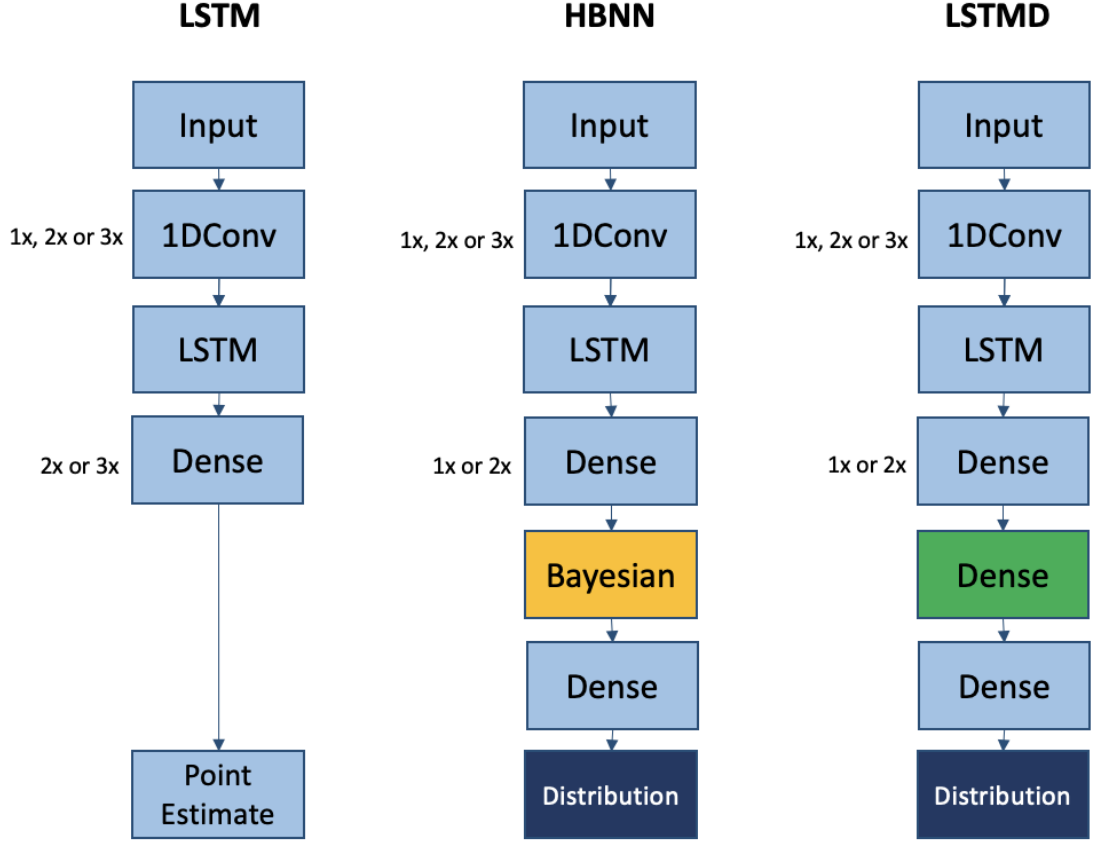


Figure 5.1: LSTM, LSTMD and HBNN architectures used for experiments with bivariate predictions. In this case, we increase the number of convolution and dense layers to better deal with multiple time series.

a 24-hour average workload and before training, data is scaled in the range $[0,1]$. 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 minutes in the future. More details on the implementation of the predictive models can be found in the shared repository¹.

With a similar evaluation framework as for Section 4.2.3, we evaluate the models regarding the point prediction accuracy, confidence interval accuracy and runtime performance. For the point prediction accuracy, i.e. MSE and MAE, in the case of HBNN and LSTMD models, the error is computed w.r.t. the mean of the predicted distribution, while for the LSTM, the error is calculated based on the point prediction. From a confidence interval point of view, we compute Success Rate (SR), Overprediction (OP), Underprediction (UP) and Total Predicted

¹<https://github.com/andreareds/TowardsUncertaintyAwareWorkloadPrediction>

Resources (TPR) with a confidence level of 95%, 97% and 99%. In the case of HBNN and LSTMD, which predict a probability distribution, the final prediction values are computed w.r.t. the upper bound (UB) of the confidence interval with a certain confidence level. 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 [MF14, MMF17]. Instead, we calculate such an interval dynamically, as done in the previous chapter, to avoid penalising the baseline too much with the following procedure. We check the SR value scored by HBNN (or LSTMD), and we adjust the interval to achieve the same SR value. In other words, we set the desired SR value first and calculate the interval accordingly. For the runtime performance, we compute the training time, finetuning time and inference time. We measure the training time by varying the training set size to 20%, 40%, 60% and 80%, the fine-tuning time by varying the number of steps among 6, 12, 18, and 24, corresponding to finetuning every 30, 60, 90, and 120, respectively and the inference time of predicting one sample.

5.1.2 Results

In this section, we evaluate and analyse the performance of univariate and bivariate models.

Point Estimate Accuracy

We first evaluate LSTM, LSTMD and HBNN trained with a single dataset (S) in terms of point estimate accuracy, i.e. MSE and MAE. Table 5.1 shows the results of processing units and memory forecast for univariate and bivariate models training with a single dataset. Each model's performance is the average MSE and MAE computed across the twelve datasets.

Table 5.1: Average MSE/MAE comparison for processing units and memory prediction for models trained with a single dataset. In bold, the best model.

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

First of all, when it comes to training the bivariate model with a single small dataset, we can immediately see the complexity of bivariate forecasting compared to univariate forecasting. While the S-B-LSTMD struggles and does not achieve the same performance as the univariate and the S-B-LSTM worsens to a lesser extent, The S-B-HBNN fails to converge. This is probably due to the extra complexity given by the Bayesian layer, making it impossible to deploy HBNN for bivariate forecasting when trained with a very limited amount of data.

Instead, training with multiple datasets brings positive effects. Table 5.2 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.

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 [HZH22]. This happens especially when the size of the overall training data is small, there is no strong correlation between time series [DPW03] and when the focus is on short-term predictions [CH16]. Indeed, the Pearson correlation analysis between the processing units and memory demand in Section 3.4 found a weak correlation between the time series. Considering the three architectures separately, LSTMD achieves the best accuracy score across the different training combinations.

As the results stand, it is unclear whether training multivariate models (cheaper) with multiple datasets (more expensive) has a positive payoff compared to training more univariate models (more expensive) with a single dataset (cheaper). Moreover, according to the Diebold-Mariano test [DM02] at 95% confidence level, pairwise run for each of the model's combinations, 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.

Confidence Interval Evaluation

We further assess the models using the metrics defined in Section 4.2.3, i.e. where we evaluate the models from a confidence interval point of view. Table 5.3 shows values for the service level metrics, where we report only target

Table 5.2: Average MSE/MAE comparison for processing units and memory prediction. In bold, the best model.

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

service levels such as 95%, 97% 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. 5.2. 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. From the aforementioned table and figure, 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. Fig. 5.3 focuses on comparing 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 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

Table 5.3: Results for the service level metrics. We run a pairwise comparison of HBNN and LSTMD with the corresponding LSTM baseline. Confidence intervals for LSTMs are tailored to the success rates achieved by HBNN and LSTMD, respectively. In bold we report the best model for each confidence level.

Target QoS	Model	Processing units				Memory			
		SR (%)	OP (%)	UP (%)	TPR (%)	SR (%)	OP (%)	UP (%)	TPR (%)
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	93.54	13.59	0.38	113.21
	S-U-LSTMD	93.14	17.11	0.56	116.55	93.26	11.80	0.41	111.39
	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	91.89	71.47	1.43	170.04
	S-B-LSTMD	92.61	84.26	3.01	181.25	92.45	71.89	1.39	170.50
	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	95.78	15.20	0.30	114.90
	M-U-LSTMD	94.62	19.55	0.43	119.12	95.18	14.53	0.32	114.21
	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	97.39	19.47	0.16	119.31
	M-B-LSTMD	96.99	28.25	0.24	128.01	99.00	24.30	0.06	124.24
	M-B-LSTM	96.99	25.84	0.27	125.57	99.00	24.30	0.06	124.24
97%	S-U-LSTM	95.22	20.07	0.41	119.66	95.06	13.92	0.31	113.61
	S-U-HBNN	95.22	20.13	0.39	119.79	95.06	15.22	0.28	114.94
	S-U-LSTMD	94.58	19.13	0.43	118.69	94.86	13.31	0.31	113.00
	S-U-LSTM	94.58	19.36	0.45	118.91	94.86	13.79	0.32	113.47
	S-B-LSTM	93.46	95.31	2.54	192.78	94.78	79.89	0.77	179.12
	S-B-HBNN	93.46	91.38	2.21	189.16	94.78	81.18	0.75	180.43
	S-B-LSTMD	93.58	93.57	2.34	191.27	94.58	80.98	0.78	180.20
	S-B-LSTM	93.58	94.22	2.49	193.52	94.58	77.64	0.90	176.74
	M-U-LSTM	93.82	18.27	0.55	117.72	96.91	17.44	0.20	117.24
	M-U-HBNN	93.82	17.91	0.50	117.41	96.91	17.14	0.23	116.91
	M-U-LSTMD	95.95	21.80	0.31	121.49	96.39	16.36	0.25	116.11
	M-U-LSTM	95.95	21.56	0.37	121.19	96.39	16.68	0.23	116.45
	M-B-LSTM	96.75	25.36	0.28	125.07	98.03	21.18	0.11	121.07
	M-B-HBNN	96.75	24.18	0.28	123.90	98.03	21.80	0.11	121.69
	M-B-LSTMD	97.75	31.51	0.16	131.35	99.44	27.38	0.03	127.35
	M-B-LSTM	97.75	29.22	0.18	129.04	99.44	29.26	0.02	129.24
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	97.55	18.37	0.16	118.21
	S-U-LSTMD	96.35	23.01	0.27	122.75	97.07	16.22	0.19	116.03
	S-U-LSTM	96.35	22.80	0.29	122.51	97.07	16.83	0.19	116.64
	S-B-LSTM	94.58	113.27	1.40	211.86	98.07	95.62	0.12	195.50
	S-B-HBNN	94.58	109.11	1.08	208.03	98.07	100.15	0.15	200.00
	S-B-LSTMD	94.42	111.33	1.23	210.10	97.91	98.71	0.16	198.55
	S-B-LSTM	94.42	111.16	1.53	209.63	97.91	94.98	0.15	194.83
	M-U-LSTM	96.47	22.75	0.32	122.43	98.07	20.65	0.16	120.49
	M-U-HBNN	96.47	22.80	0.27	122.53	98.07	20.85	0.16	120.69
	M-U-LSTMD	97.79	26.14	0.17	125.96	97.63	19.86	0.19	119.67
	M-U-LSTM	97.79	27.44	0.18	127.26	97.63	19.37	0.19	119.18
	M-B-LSTM	97.95	29.95	0.16	129.78	99.04	25.86	0.04	125.82
	M-B-HBNN	97.95	29.15	0.15	128.99	99.04	26.23	0.06	126.17
	M-B-LSTMD	99.04	37.73	0.06	137.67	99.80	33.22	0.01	133.21
	M-B-LSTM	99.04	36.54	0.06	136.48	99.80	33.99	0.01	133.98

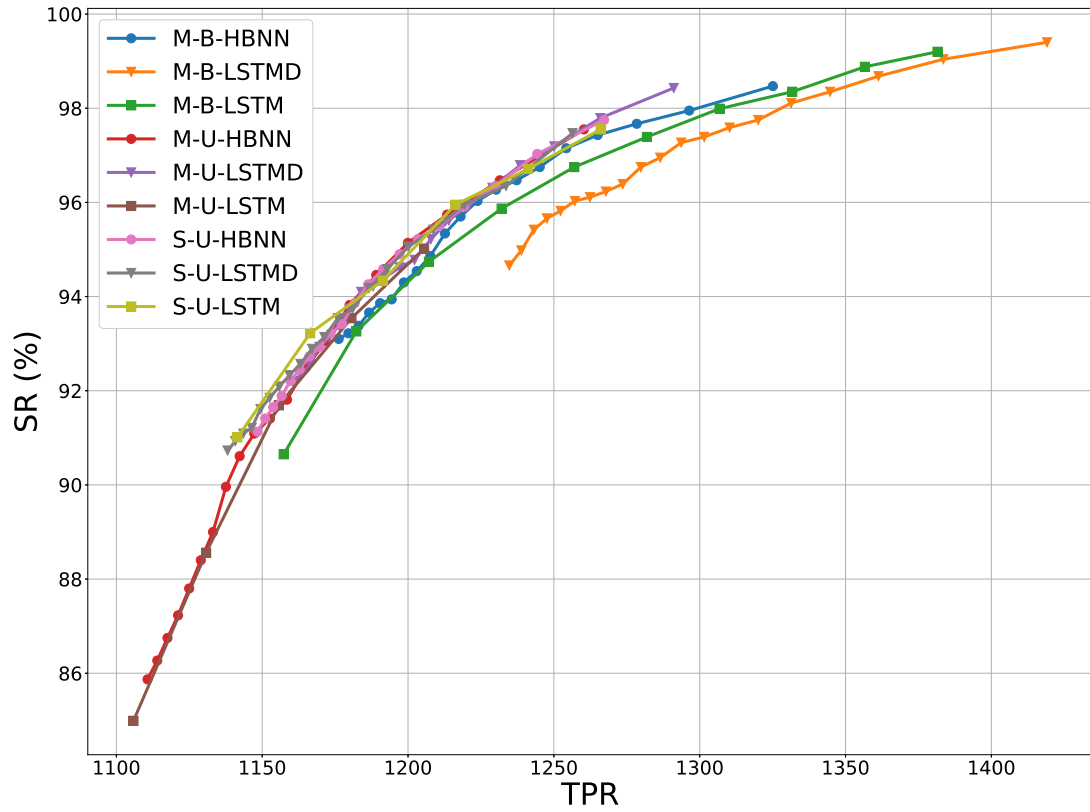


Figure 5.2: Total predicted processing units for S-U, M-U and M-B models. The closer the curve to the top left corner, the better the model.

Table 5.4: Average MSE and MAE comparison for resource prediction accuracy. The MSE and MAE are computed between the SR achieved by the models and the confidence level. In bold, the best model.

Model	Processing units		Memory	
	MSE	MAE	MSE	MAE
S-U-LSTMD	2.96	1.47	3.07	1.67
S-U-HBNN	1.69	1.10	1.80	1.27
S-B-LSTMD	8.15	2.66	6.51	2.34
S-B-HBNN	11.80	3.32	11.81	3.03
M-U-LSTMD	1.24	0.98	1.14	0.92
M-U-HBNN	12.07	3.34	2.42	1.25
M-B-LSTMD	6.89	2.08	21.05	3.94
M-B-HBNN	1.97	1.11	8.66	2.39

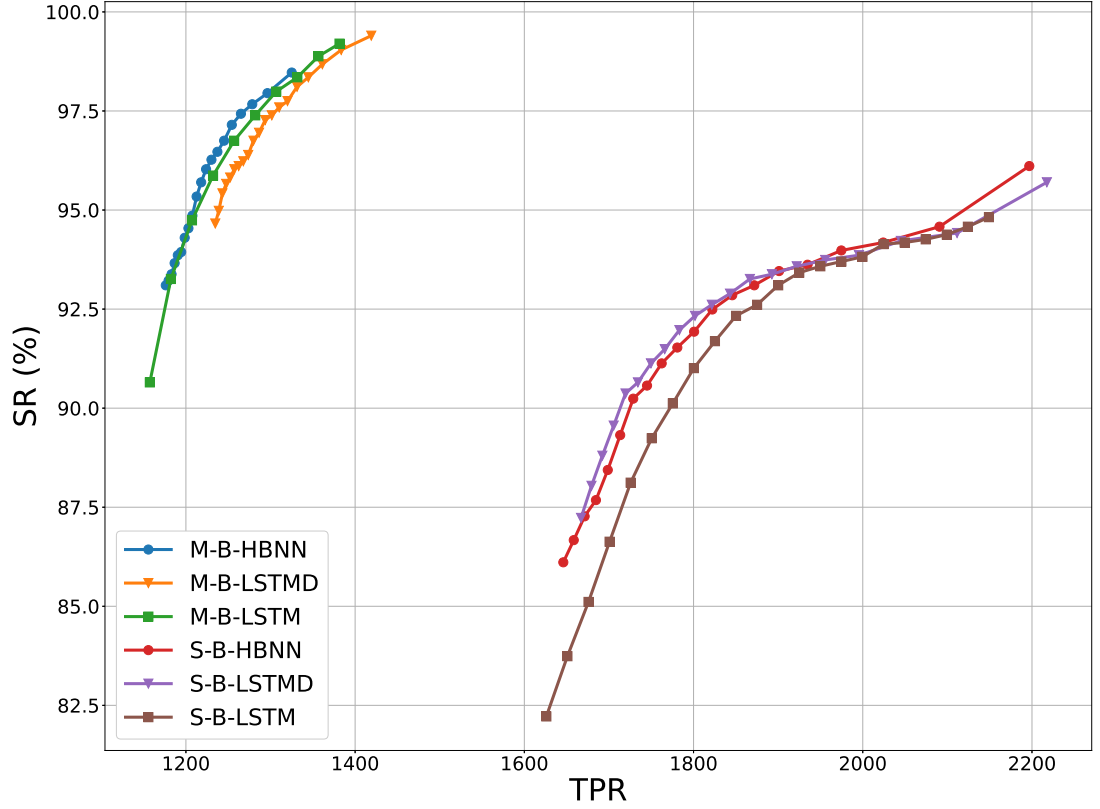


Figure 5.3: Total predicted processing units for bivariate models. The ones trained with a single dataset perform very poorly, with a much higher amount of TPR w.r.t. than the models trained with multiple datasets.

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. 5.4 and Table 5.4. It shows that 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. We should also consider that MSE and MAE are symmetric metrics. 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, even if the MSE and MAE are lower. For instance, we should favour a model that achieves a 96% SR compared to 94% w.r.t. a 95% confidence level, despite the score being the same in terms of MSE and MAE. For this reason, asymmetric metrics would be more suitable to evaluate the model's accuracy combined with the confidence of the prediction. However, the large number of error metrics available in forecast-

ing requires a thoughtful selection process, and using them all is impractical. Standard practice within the field is reporting results using a combination of 2-3 established metrics, often those employed in forecasting competitions. A comprehensive evaluation also requires acknowledging the inherent trade-offs and ongoing debate surrounding metric selection [DF16]. All remaining tables, figures, and detailed results are included in Appendix B.1.

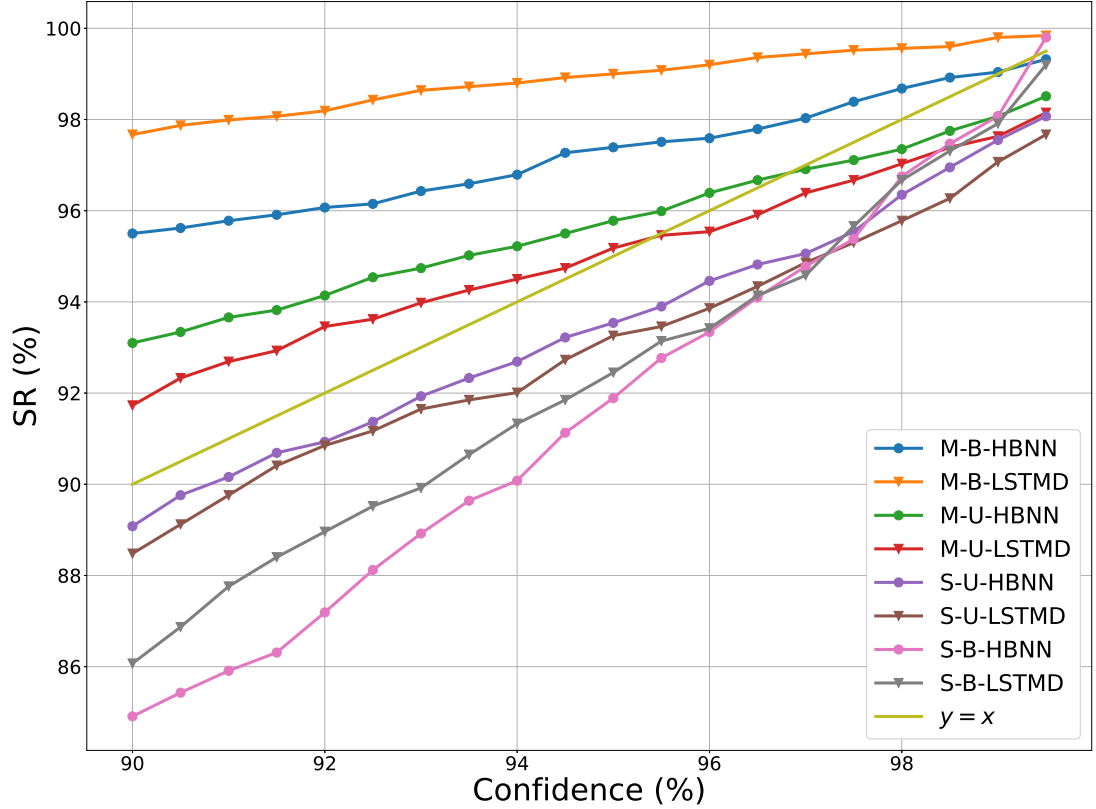


Figure 5.4: Target service level versus SR for memory prediction. The plot is built w.r.t. the UB of the confidence interval by varying the target service level. The closer the plot is to the line $y = x$, the more accurate the model under this metric.

Runtime Performance Analysis

Runtime performance is a critical factor in evaluating the tradeoff between univariate and bivariate forecasting models, especially if we consider that we need to combine two univariate model forecasting two different variables, compared to a bivariate model that can predict both at the same time. However, the complexity of the model can significantly increase the computational cost. It also provides insights into whether the benefits of a more complex bivariate model outweigh the additional computational cost compared to a simpler univariate

Table 5.5: Runtime performance analysis. In bold is the best model for each subgroup. The training time is computed by training the model with 20, 40, 60 and 80% of the training set, respectively. The fine-tuning of the network is done every 6, 12, 18 and 24 steps, corresponding to 30, 60, 90 and 120 minutes, 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

model.

The results are computed as an average of 10 runs for all the model combinations, and we report values in Table 5.5. 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 to optimise than a dense layer), 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 critical. 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, the results show that we can train the HBNN in a bivariate context using multiple datasets while maintaining its uncertainty quantification capabilities. Furthermore, results show a 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. Our findings show that HBNN and LSTMD offer a one-model-fits-all solution that provides accurate bivariate predictions with uncertainty quantification on multiple datasets. In particular, training with multiple datasets is advantageous as it eliminates the need to train separate models for each cluster cell.

5.2 Towards Transfer Learning for Cloud Workload Forecasting

In the previous section, we discussed the application of HBNN for bivariate workload prediction in cloud computing. HBNNs have demonstrated to achieve predictive capabilities when trained with multiple datasets, enabling accurate forecasting of future workloads. However, in practice, cloud providers often face the challenge of limited historical data for each workload. This limited data can make it difficult to train HBNNs, which in turn can restrict their forecasting capabilities.

To address this challenge, Transfer learning (TL) emerges as a promising approach. TL involves leveraging knowledge from a source domain to improve performance in a target domain. In workload forecasting, the source domain could be data from other cloud providers or workloads from different applications within the same provider. By transferring the learned knowledge from the source domain, TL can enhance the generalization ability of HBNNs and enable accurate forecasting even with limited data for the target workload. TL 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, TL 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 TL. In particular, we consider the following situations:

- **All Datasets.** In this scenario, the ML 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 TL 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 TL, 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. Specifically, we are interested in investigating the feasibility of training HBNNs with a subset of data from one cluster cell and then using this trained model to predict workloads for other cluster cells.

5.2.1 On Transfer Learning for Cloud Computing

Transfer learning (TL) [WKW16] is a broad ML 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 [DCdCKG19, WLL⁺22], very closely related to each other, so that sometimes the literature is inconsistent in its terminology and does not distinguish between them.

Our work is about transferring knowledge across domains and overlaps the three areas so we use the general TL 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 (ZS) fashion. In the first case, a pre-trained model's weights are trained on the new target dataset, while in the second, a pre-trained model is directly used to predict the new target dataset without modification of the model [PAL⁺22].

TL offers several advantages in cloud computing forecasting, including improving the model's scalability, as it can be deployed across multiple cluster cells without the need to train separate models for each cell, enhancing generalization by leveraging knowledge from other cluster cells and better capturing the underlying trends and patterns in workload data. Moreover, cloud workloads often exhibit similar temporal patterns and correlations across different cluster

cells. TL can capture and apply these shared dynamics to the target cell, even if the data distributions between the source and target domains differ slightly [FPRA20].

TL has been widely applied in the context of time series classification [FFW⁺18] and forecasting [YD18], showing the effectiveness of this approach, including QoS aspects [HYZ⁺21] for workload prediction [LJL⁺22]. TL has also been applied in other cloud computing contexts, such as runtime performance prediction [LMC20, NME12] and autoscaling [ZZL⁺21]. However, there is no literature on the application of TL across domains for workload predictions in cloud computing.

In this section, we design and run several prediction scenarios where we apply zero-shot or fine-tuning on a pre-trained model, each testing the model's performance using a different combination of source and target domains (datasets).

5.2.2 Methodology

These experiments investigate the TL capabilities of the uncertainty-aware models. The methodology we describe can be applied to any of the models analysed in the first part of the chapter, but it is computationally expensive. Thus, in this part, we select only the M-B-HBNN network, which is the best model according to the results of the first part of the experiments. The M-B-HBNN 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 a more comprehensive investigation involving the other competitive models, such as some of the LSTMD-based for future work.

In our experiments, we use the same 12 cluster-level datasets from GC11, GC19, AC18 and AC20. The datasets are split as in the first part, and we use the best hyperparameters found on the optimisation for the M-B-HBNN model. 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. 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.

We design six prediction scenarios that differ based on the combinations of datasets used for training (i.e. source domain) and testing (i.e. target domain). See Fig. 5.5 for an illustrative example of four of them. It is worth noting that we only build the following most representative scenarios among all the possible combinations we could build.

- **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 another portion of one of these datasets (domains) for testing. The final performance is obtained by averaging results across all the twelve target datasets.
- **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. The final performance is obtained by averaging results across all the twelve target domains. This scenario assesses whether the FT 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 dataset (source domain). We then predict the remaining dataset (target domain). The final performance is obtained by averaging results across all the twelve target domains. This experiment further investigates the model's generalisation capabilities on unseen datasets in a zero-shot fashion. The real-world motivation of this scenario, for example, is being able to predict the workload of a new cluster from day one (instead of waiting a few days or weeks to collect historical workload data and train a model to predict this new cluster).
- **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 FT on newly available data. The final performance is obtained by averaging results across all the twelve target domains.
- **TL-GC19:** we only use the eight datasets from GC19 and we build four sub-scenarios, i.e. *GC19 All* resembling *All*, *GC19 All FT* resembling *All FT*, *GC19 All-but-one* resembling *All-but-one*, *GC19 All-but-one FT* resembling *All-but-one FT*. These scenarios assess whether pretraining and FT enhance generalisation capabilities on different domains from the same

providers.

- **TL-GC19vsOther:** the source domain is GC19 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.



Figure 5.5: An illustrative example of four training scenarios for the second part of the experiments. In this example, we consider dataset number 1 by GC19 as the target domain (and we use it as the FT dataset in All FT and All-but-one cases). Depending on the scenario, The first 80% of dataset number 1 is included in the training set and/or used for fine-tuning. The target domain is the final 20% of the dataset to avoid leakage. This approach is repeated for all the 12 datasets.

We evaluate the models in terms of point prediction and confidence interval accuracy, as done in the previous Section 5.1.2. We do not measure the runtime performance because the model is the same in all the experiments, and the training time depends only on the size of the training set.

5.2.3 Results

Similarly to the previous section, we present and analyse the results obtained from the six prediction scenarios depicted in Section 5.1.1, where we probe the TL capabilities of the M-B-HBNN model.

Point Estimate Accuracy

In Table 5.6, we report the accuracy results for *All*, *All FT*, *All-but-one* and *All-but-one FT*. 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. Particularly, *All-but-one* is the second best model under these metrics and opens the opportunity to deploy pretrained models to unseen/new data centres, overcoming the need for specialised data coming from the target domain. Fine-tuning instead (both in *All* and *All-but-one*) makes the accuracy worse; this is likely due to the fact that the model overfits the target dataset or that it makes it forget patterns learned on other traces.

Table 5.6: 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	MSE	MAE
All	0.0042	0.0471	0.0043	0.0436
All FT	0.0058	0.0561	0.0062	0.0058
All-but-one	0.0044	0.0481	0.0045	0.0448
All-but-one FT	0.0057	0.0563	0.0058	0.0538

We report the accuracy results of scenario TL-GC19 (where the source and target dataset are drawn from GC19) in Table 5.7. The table shows that the baseline *TL-GC19 All* performs best. 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, corresponding to almost half of the total data points) is too small to capture the distribution-independent properties of the GC19 provider, while the model’s hyperparameters are tuned for dealing with all the twelve datasets. This, combined with the bivariate forecasting that we showed in the first part of the chapter, requires more data for convergence and makes ZS learning a much harder task [XLSA18].

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 5.8. 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, this time fine-tuning on the target domain helps to improve the performance (especially for AC20), although error values are an order of magnitude higher than the *All* baseline.

Table 5.7: Accuracy results for M-B-HBNN trained on scenario *TL-GC19*. In bold, the best performance.

Scenario	Processing units		Memory	
	MSE	MAE	MSE	MAE
TL-GC19 All	0.0043	0.0474	0.0044	0.0444
TL-GC19 All FT	0.0059	0.0562	0.0063	0.0546
TL-GC19 All-but-one	0.0087	0.0673	0.0091	0.0681
TL-GC19 All-but-one FT	0.0097	0.0733	0.0104	0.0701

Table 5.8: Accuracy results for M-B-HBNN trained on scenario *TL-GC19vsOther*.

Target	FT	Processing units		Memory	
		MSE	MAE	MSE	MAE
GC11	✗	0.1443	0.3176	0.2467	0.471
	✓	0.0588	0.1669	0.0421	0.1634
AC18	✗	0.0737	0.2356	0.0598	0.2242
	✓	0.0271	0.1114	0.01	0.0614
AC20	✗	0.3206	0.4658	0.09	0.2466
	✓	0.0522	0.1791	0.0608	0.1934

Confidence Interval Evaluation

We also analyse the TL capabilities of M-B-HBNN by looking at the confidence intervals that we can compute given the parameters of the predicted Gaussian distributions. Table 5.9, Figures 5.6 and 5.7 report the results for scenarios *All*, *All FT*, *All-but-one* and *All-but-one FT*. Table 5.10 and Fig. 5.8 report 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 a detailed analysis, but we include the results in Appendix B.2. From the aforementioned tables and figure, 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) FT always harm the performance. Particularly, (i) can be noted by comparing the numerical results and closeness of the blue and red lines in the plots; (ii) is evident in plot 5.8 where the blue line corresponding to *TL-GC19* is the only line close enough to the line $y = x$; (iii) can be noted by looking at the way the SR drops when we fine-tune the model and the relative position of the FT models w.r.t. to the not fine-tuned versions in the plots (where the higher are lines, the best are the performance). Overall, the results align with the point accuracy analysis, as in this set of experiments, the models' performances substantially differ from one

another (i.e., the models' accuracies are statistically different).

Table 5.9: 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.

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

Table 5.10: Service level metrics results of M-B-HBNN for scenario *TL-GC19*. In bold, the model with the best SR for each confidence level.

Confidence Level	Model	Processing units				Memory			
		SR (%)	OP (%)	UP (%)	TPR (%)	SR (%)	OP (%)	UP (%)	TPR (%)
95%	TL-GC19	98.55	10.85	0.04	112.04	97.18	12.29	0.13	109.12
	TL-GC19 FT	77.7	4.96	0.85	105.34	89.36	7.89	0.41	104.45
	TL-GC19 All-but-one	69.44	4.07	1.05	104.26	74.44	4.75	1.12	100.59
	TL-GC19 All-but-one FT	52.93	13.62	22.08	92.78	63.79	46.63	10.35	133.24
97%	TL-GC19	99.42	12.53	0.02	113.75	98.19	14.07	0.1	110.94
	TL-GC19 FT	84.0	6.04	0.59	106.69	92.9	9.25	0.28	105.94
	TL-GC19 All but-one	79.07	5.22	0.65	105.81	83.06	6.35	0.7	102.61
	TL-GC19 All-but-one FT	54.02	14.45	20.67	95.01	65.03	48.93	9.27	136.63
99%	TL-GC19	99.86	15.74	0.01	116.96	98.84	17.47	0.06	114.38
	TL-GC19 FT	92.76	8.28	0.29	109.23	96.6	11.93	0.14	108.76
	TL-GC19 All-but-one	92.54	7.74	0.23	108.74	92.9	9.7	0.26	106.41
	TL-GC19 All-but-one FT	56.34	16.1	18.1	99.23	69.08	53.47	7.4	143.03

Finally, we analyse how uncertainty varies when inference is made on different target datasets. In Fig. 5.9, 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.

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,

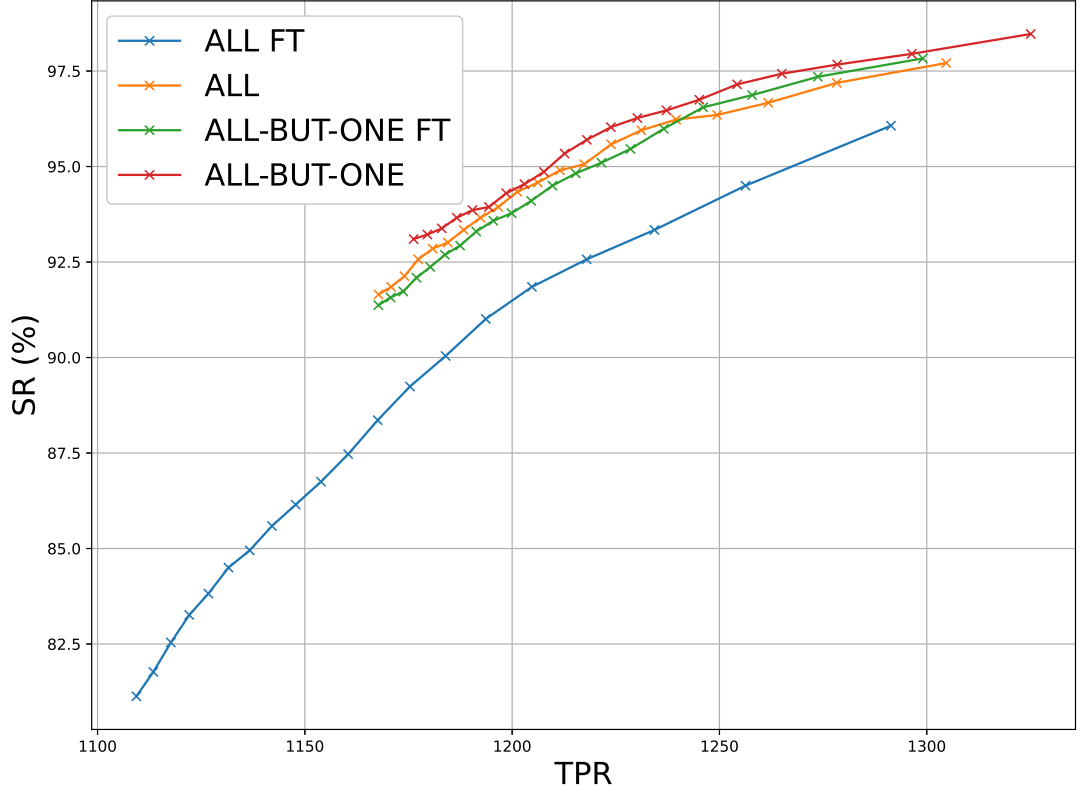


Figure 5.6: Total predicted processing units for *All*, *All FT*, *All-but-one* and *All-but-one FT*. The closer the curve to the top left corner, the better the model.

it struggles to generalise well when the source domain is small, or the target domain is very different from the target domain.

5.3 Conclusion

The HBNN model provides richer information than a simple pointwise prediction, which can help cloud resource managers optimize their decision-making process. However, training a one-fit-all model for predicting both processing and memory units simultaneously on multiple cluster cells is much more challenging.

In this chapter, we first evaluated univariate and bivariate versions of HBNN and LSTM presented in Chapter 4 that can predict processing units 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 many models on single datasets. Moreover, the bivariate HBNN achieves accuracy comparable to its univariate version when trained

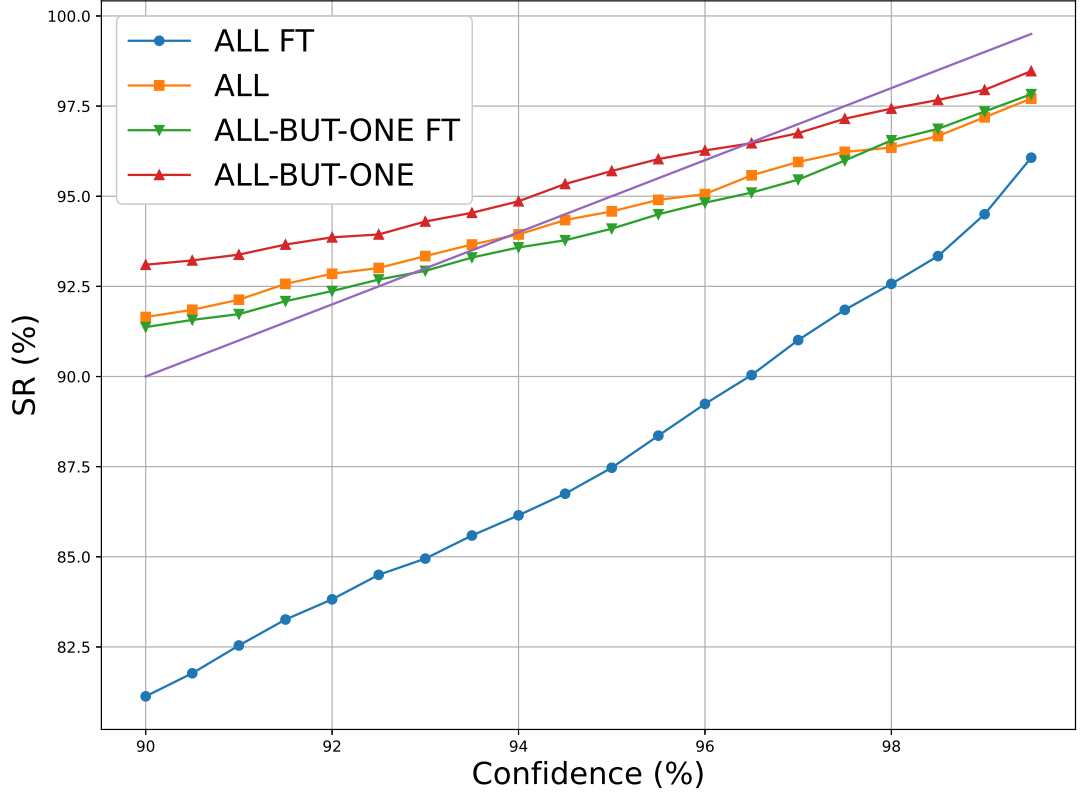


Figure 5.7: Target confidence level versus SR for processing units prediction for *All*, *All FT*, *All-but-one* and *All-but-one FT*. The closer the plot is to the line $y = x$, the more accurate the model under this metric.

with sufficient data. Also, all models' runtime is efficient enough to be deployed in real-world applications. In particular, we found that the bivariate HBNN trained on multiple datasets (i.e. M-B-HBNN) provide a good trade-off between performance and complexity of deployment for the second part of our investigation. Indeed, we further probed the TL capabilities of uncertainty-aware models, as models' generalisation on out-of-distribution target domains is under-investigated in workload forecasting. In practice, we checked whether the M-B-HBNN, trained in ZS or with and without FT scenarios, can transfer knowledge across related (to different extents) domains. Experiments reveal that the more source and target domains are different (particularly when the source and target are from different cloud providers), the more performance degrades, and FT, most of the time, does not help the knowledge transfer, in line with other time series forecasting domains like finance [DR23]. Acceptable TL performances can be achieved when the training source domain is big enough to account for the target domain diversity from the source domain. However, we still believe cloud providers should explore ways of quickly building forecasting

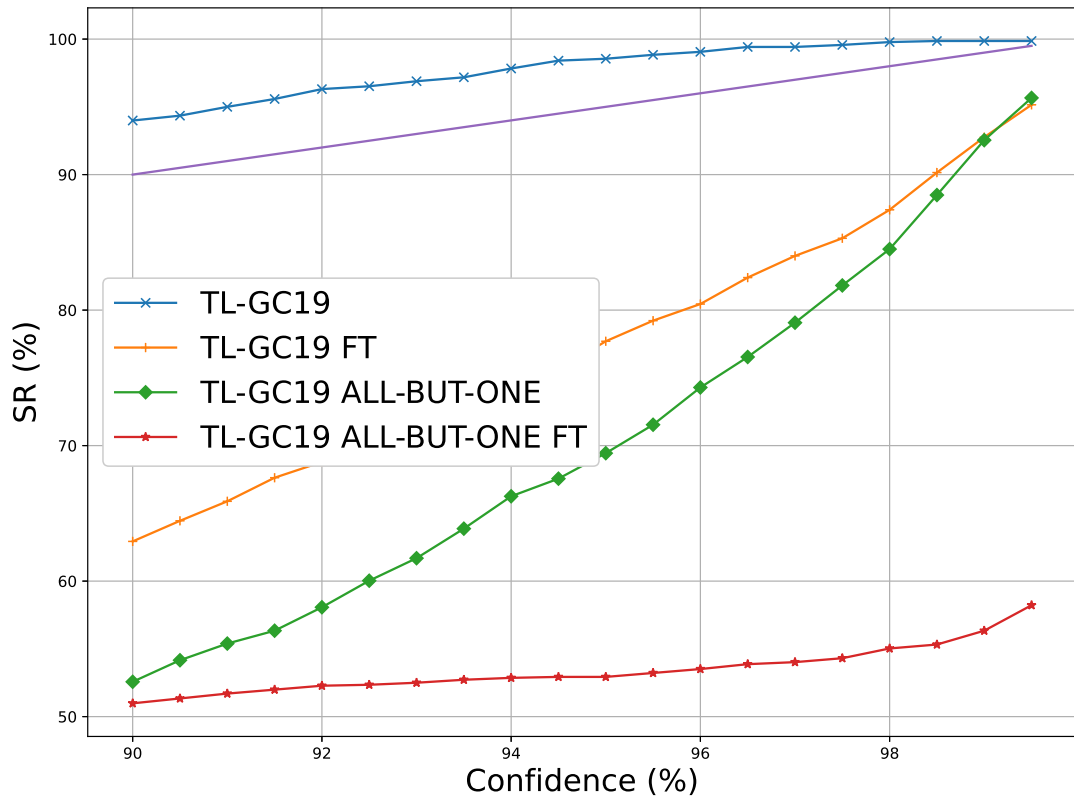


Figure 5.8: Target service level versus SR for processing units prediction for *TL-GC19*. The closer the plot is to the line $y = x$, the more accurate the model under this metric.

models in scenarios where new (unseen) or little data is available. Therefore, to make the transfer successful, further investigation into TL is needed, particularly on the relationship between the source and target domain.

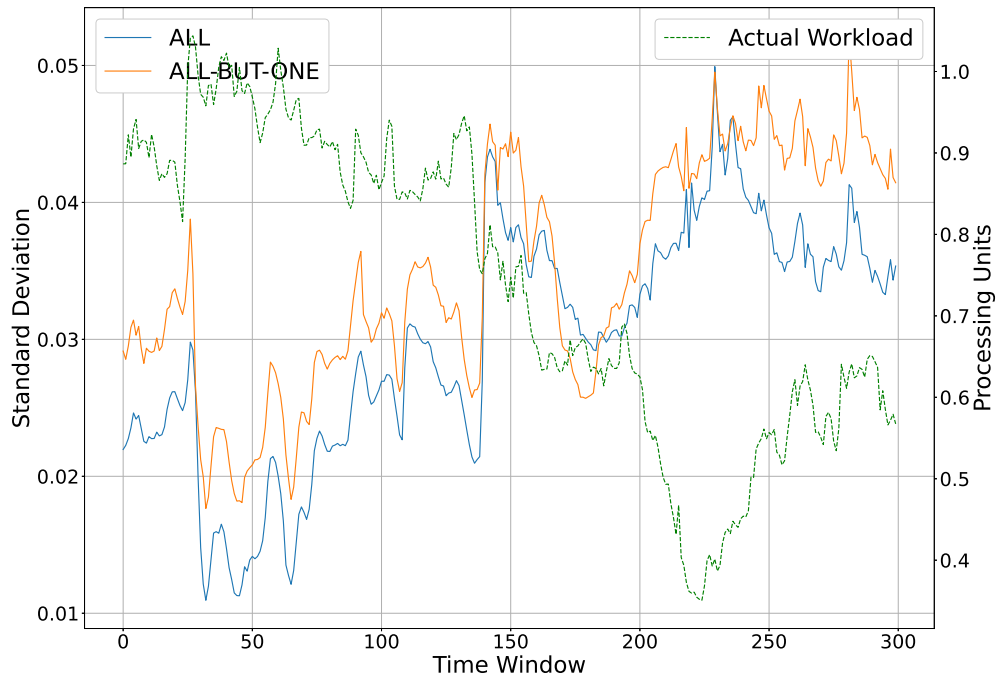


Figure 5.9: Standard deviation variability for across inference on the test set *All* and *All-but-one*. Uncertainty for *All-but-one* is higher for predictions on unseen data.

Chapter 6

Clustering-based Numerosity Reduction

DL models have shown remarkable performance in various tasks, such as image classification and NLP, but their effectiveness heavily relies on the availability of sufficient training data. The previous chapter demonstrated that workload forecasting models, and the HBNN in particular, are also subject to this dependency, as they require a significant amount of data to enable accurate bivariate forecasting.

While more data generally leads to improved model performance, it also introduces challenges related to data redundancy and computational limitations. Redundant data refers to the presence of overlapping or repetitive information within the training dataset, which can hinder the learning process and potentially diminish model accuracy. Simultaneously, deep learning models like HBNN become computationally demanding as the training set size increases, leading to prolonged training times and higher energy consumption.

Reducing the training time of these models is essential for making them practical and scalable. Training DL models can be computationally expensive and time-consuming, which can be a barrier to their widespread adoption.

Many techniques have been proposed to address the challenges of large training datasets. One of these approaches is numerosity reduction, which involves transforming the original data into a lower-dimensional representation while preserving the most important features. This results in improved computational efficiency and reduced energy consumption during training. One effective method for achieving this is through time series clustering.

By leveraging time series clustering techniques, we can simplify the training process by grouping similar workloads together. This not only speeds up the training process but also enhances the models' focus on the most relevant data patterns, thereby improving their efficiency and effectiveness. Faster training times allow for more frequent updates of the models, ensuring they remain accurate and responsive to the latest data.

In this chapter, we explore the application of numerosity reduction techniques, particularly via time series clustering, to enhance the performance and scalability of HBNNs for machine-level bivariate workload forecasting in cloud computing and hundreds of datasets are available for training. Time series clustering is a powerful tool for identifying patterns and relationships within time-dependent data, making it well-suited for reducing redundancies in workload datasets. In Section 6.1, we overview numerosity reduction techniques for times series data. Section 6.2 describes the methodology employed in our experiments, including the data preprocessing steps, the time series clustering algorithm used, and the baselines for assessing forecasting performance. Finally, Section 6.3 summarises the key findings of our research and discusses the implications of our results for enhancing the performance and scalability of HBNNs for workload forecasting in cloud computing. Section 6.4 concludes the chapter.

6.1 Numerosity Reduction for Times Series Data

DL often requires dealing with datasets containing many features or data points. Two key strategies emerge to tackle this challenge: dimensionality reduction and numerosity reduction. Dimensionality reduction focuses on reducing the number of features (dimensions) in the data. This aims to create a more concise representation of the data while preserving the most important information. In contrast, numerosity reduction targets the number of data points themselves. This approach seeks to represent the original data using fewer data points while maintaining the essential characteristics. In other words, if a dataset is represented by m features and n data points, dimensionality reduction algorithms aim to reduce m , while numerosity reductions reduces n .

A time series can be considered as a point in a high-dimensional space, where each dimension corresponds to a data point of that time series. As the length of the time series (i.e., the number of data points) increases, the dimensionality

of the data also increases. The analysis of a high-dimensional space can be quite challenging due to data sparsity. As the number of dimensions increases, the available data becomes more spread out, leading to difficulty in identifying any meaningful patterns. This data sparsity is the main cause of the curse of dimensionality, which demands more computational resources and time to analyze the data effectively [Bel66]. Dimensionality reduction techniques aim to mitigate the curse of dimensionality by reducing the number of dimensions while preserving the essential characteristics of the data and helping capture the temporal dependencies present in the data, which are often lost in the high-dimensional space [VF05, SKPS20]. Several dimensionality reduction methods exist, including feature selection and numerosity reduction [WTBR22].

Feature selection involves selecting a subset of the original features to reduce the dimension of the input. However, engineering and extracting those original features for cloud workload prediction is a complex task, especially crafting features for cloud workload prediction which is complex, especially due to the lack of information on unlabeled workload traces [AK23a], and the existing literature on handcrafted features in workload prediction for cloud computing is also limited [LLS⁺17, SK21b, RBL⁺22, LSM⁺23]. This challenge makes feature selection a difficult task altogether.

On the other hand, numerosity reduction is a technique used in data preprocessing to reduce the number of training points. The goal is to maintain the overall integrity and structure of the data while reducing its size. The literature discusses various techniques for dimensionality and numerosity reduction in time series analysis, but some are less suitable than others when applied to cloud workload forecasting. There are still several open challenges on dimensionality reduction [RKKK22], including developing methods that are robust to outliers and anomalies while also providing interpretable results [RPF⁺21].

One of the most popular dimensionality reduction techniques is aiming to map the high-dimensional space into a lower-dimensional space by transforming the input features of the time series. Principal Component Analysis (PCA) and Auto-Encoder (AE) are examples of these approaches. PCA [Pea01] is a widely used linear technique that transforms the original time series into a lower-dimensional space by capturing the highest amount of variance. AE [Kra91] are neural networks that are trained to reconstruct the original input data from a lower-dimensional representation that can be used to learn a compressed and efficient representation of the data. However, the standard approaches to

these dimensionality reduction methods treat each time point as an independent variable and do not consider the temporal dependence between different time points, leading often to inaccurate forecasts.

Another family of dimensionality reduction methods are called temporal aggregation methods [RTBSD13]. Temporal aggregation involves grouping data points into larger intervals, reducing the overall size of the time series. When data is aggregated, essential patterns are retained while the analysis is simplified, making it useful for summarising long-term trends. An example of an aggregation method is Piecewise Aggregate Approximation, which involves segmenting the time series and representing each segment with its mean value [GLP10]. This method reduces the length of the time series, but at the cost of a loss of information. In creating the workload datasets (Chapter 3), we adopted this approach by considering intervals of 5 minutes while aggregating the total workload in that interval. However, these methods reduce the input's complexity, limiting the model's forecasting capabilities and increasing the predicting interval's size. For example, aggregating the input into intervals of 10 minutes would allow us to predict an interval of 10 minutes only, leading to less detailed predictions.

As mentioned before, unlike dimensionality reduction, which compresses data by reducing features, numerosity reduction focuses on selecting alternate forms of data representation to reduce data volume. When aiming to reduce training data, coresets became a popular approach [AHPV⁺05]. Coresets are subsets of the full dataset that guarantee solutions found on them are competitive with those found on the full dataset. Since coresets are guaranteed to represent the full dataset's statistical properties, their formation can be complex, especially when dealing with time series data [HSV21]. An easy way to reduce the training data size is simple random sampling [WFFWM18], where every sample has an equal chance of being selected. However, this approach does not guarantee that all the possible patterns are selected, leading to a less representative dataset. One way to overcome this problem is stratified sampling [Par14], which overcomes this problem by dividing samples into subgroups before sampling. However, identifying the relevant characteristics or attributes that define the subgroups (strata) in cloud workload traces is generally hard due to lack of information [AK23a], so it is difficult to apply it directly to the problem. To overcome this problem, clustering-based approaches can be adopted to group similar time series into clusters with similar patterns in an unsupervised fashion. In this way, stratified sampling can be applied, where each cluster can

be considered as a stratum. Unsupervised clustering has been proven to be a powerful tool for dividing workload datasets into several groups and training independent forecasting models for each of them [GWS20, YJYB16]. Still, it has never been applied for numerosity reduction in workload forecasting, but it showed its successful application to time series classification [XKS⁺06].

In the next section, we define the methodology and experiments employed in this chapter. We will outline the clustering algorithms, evaluation metrics, and experimental setup, ensuring transparency and reproducibility of our findings.

6.2 Experiments

This work focuses on time series clustering-based numerosity reduction, as motivated in Section 6.1, which aims to reduce the number of training samples. This study investigates the effectiveness of using clustering-based training data selection for cloud workload prediction. Through an extensive methodology, we want to assess if the proposed approaches can reduce the training size without compromising the models' accuracy.

First, we experimentally find the optimal number of clusters for each clustering technique and divide the time series into clusters. Then, a number of datasets from each cluster is selected. This subset is used to train cloud workload predictive models, and we train the models with the same number of datasets, selected randomly from all the time series or with representatives in each cluster. Finally, the models are compared in terms of predictive accuracy (MSE, MAE, confidence intervals) and runtime performance. The experiments were conducted using the 2,500 machine-level workload datasets from GC19, described in Chapter 3.

6.2.1 Methodology

This study investigates the effectiveness of using clustering-based training data selection for machine-level workload prediction, using the 2,500 machine-level workload datasets from GC19, described in Chapter 3. The pipeline can be divided into clustering optimisation and forecasting parts.

We consider four clustering approaches based on the methods described in Section 2.3: PCA, VRAE, SOM and DTW. In particular, SOM is the only algorithm that outputs the clusters after its training. PCA and VRAE are used for re-

ducing the dimension of the time series and apply K-means clustering on the latent space, while with DTW, we apply the K-means algorithm with DTW as the distance metric and DBA for computing the centroids. We excluded hierarchical clustering and DBSCAN from the analysis. Hierarchical clustering is a good choice for exploratory data analysis, as it can be used to visualize the relationships between data points and identify clusters of interest, but for more focused clustering tasks, the other methods can produce more accurate clusters [GSA21]. DBSCAN is characterised by automatically selecting the number of clusters K and tuning other parameters. This leads to a different evaluation methodology than the other methods whose optimal number of clusters is selected via the elbow method. For this reason, we leave this analysis as future work. We find the optimal number of clusters for each technique using the elbow method (see Section 6.2.3) and divide the 2,500 machine-level workload datasets from GC19 into clusters.

Once clusters are created, we select a certain number L of datasets. The selection is at random from the whole pool of 2,500 datasets or selecting an equal number of datasets from each cluster such that the total number of selected datasets is L . In the experiments, L varies between 50 and 2000, and datasets in the test set is not used for training.

The selected datasets are used to train three models to predict future CPU and memory demand (bivariate forecasting) of the 5-minute interval 10 minutes ahead. For the prediction, we used HBNN and state-of-the-art probabilistic models, i.e. DeepAR and TFT, presented in Section 2.2.4, and designed for being jointly trained with multiple time series. Section 6.2.2 provides more details on the models' configuration. Among the selected datasets, the first 80% (about 23 days) of each dataset is used for training. The remaining 20% (about 6 days) of the datasets in the test set are used for evaluation. This avoids timestamp overlap between time series in training and testing; and prevents overfitting. The training pipeline is illustrated in Fig. 6.1.

With a similar evaluation framework as for Section 4.2.3, we evaluate the models regarding the point prediction accuracy, confidence interval accuracy and runtime performance. For the point prediction accuracy, i.e. MSE and MAE, the error is computed w.r.t. the mean of the predicted distribution of each model (models are all probabilistic). From a confidence interval point of view, we compute Success Rate (SR), Overprediction (OP) and Underprediction (UP) by varying the confidence level between 90% and 99%. The final prediction values

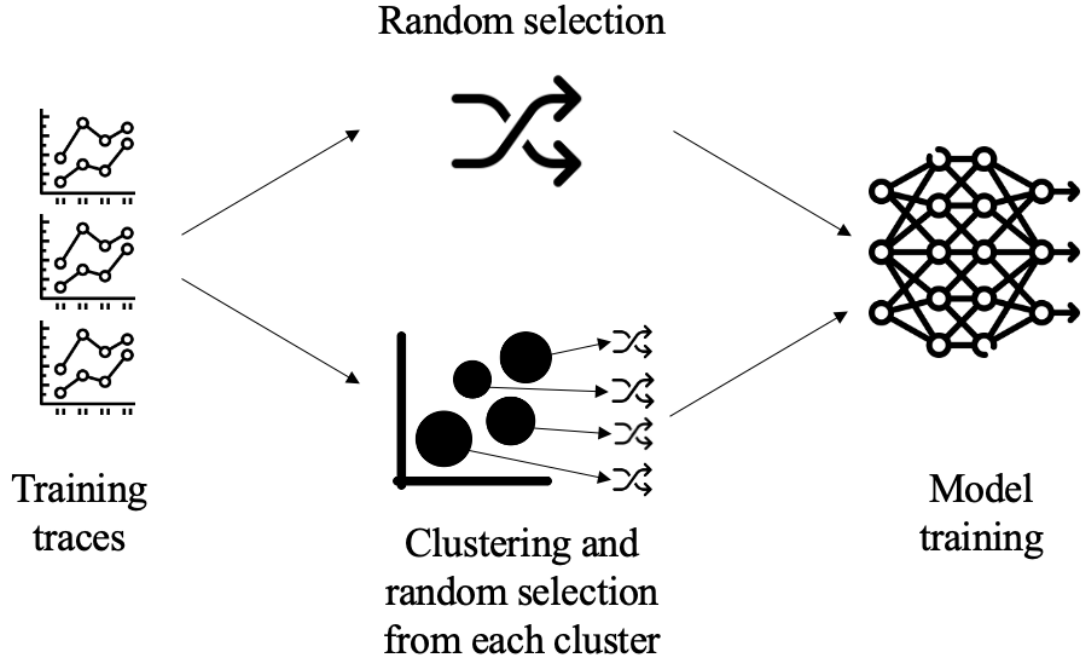


Figure 6.1: Selection of the training datasets: random selection vs clustering approach. Using random selection, traces are chosen randomly from the pool of datasets. In the clustering approach, traces are first grouped into clusters, and then sampling is applied to each cluster.

are computed w.r.t. the upper bound (UB) of the confidence interval with a certain confidence level. For the runtime performance, we compute the training time and inference time. We measure the training time with the training set size varying with L and the inference time of predicting one sample. We also measure the fitting and inference time of the clustering techniques.

6.2.2 Forecasting Models and Optimization

The models considered in this evaluation are the HBNN model presented in Chapter 4 and two state-of-the-art probabilistic forecasting models: DeepAR and the TFT.

An exhaustive hyperparameter search is conducted to find the best hyperparameters for the three DL models. For the HBNN model, the following hyperparameters are tuned with the Talos library: 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, ending up with an architecture similar to the experiments in Chapter 5. We also tuned the scaling range and the input size. We found that the

best input size is 288, corresponding to a 24-hour average workload. Before training, data is scaled in the range $[0,1]$. The optimisation algorithm and loss function are the same as in the previous chapters, i.e., the Adam algorithm and negative log-likelihood.

DeepAR and TFT are implemented in PyTorch, and the Optuna library is used for the hyperparameters search. Models are optimised via the Adam algorithm with quantile loss as the loss function. 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.

More details on the implementation and hyperparameters can be found in the open-source GitHub repository¹.

6.2.3 Clustering methods

This section describes how the optimal number of clusters, which is unknown, has been found. The clustering algorithm's performance is evaluated by varying the number of clusters, K and plotting the elbow curve. The elbow curve method [Tho53] is a popular method for finding the best K , based on the distortion score or within-cluster sum of squares (WCSS), 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. An illustration of the elbow curve for the DTW K-means is given in Fig. 6.2, where, in this case, it is very soft, but it is the point where there is a higher variation of distortion.

We compared these methods used to select the datasets:

- **Random:** The Random method selects training datasets randomly without using clustering methods. This is the baseline of our experiments.
- **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.

¹<https://github.com/andreareds/clustered-reduction-cloud-workload>

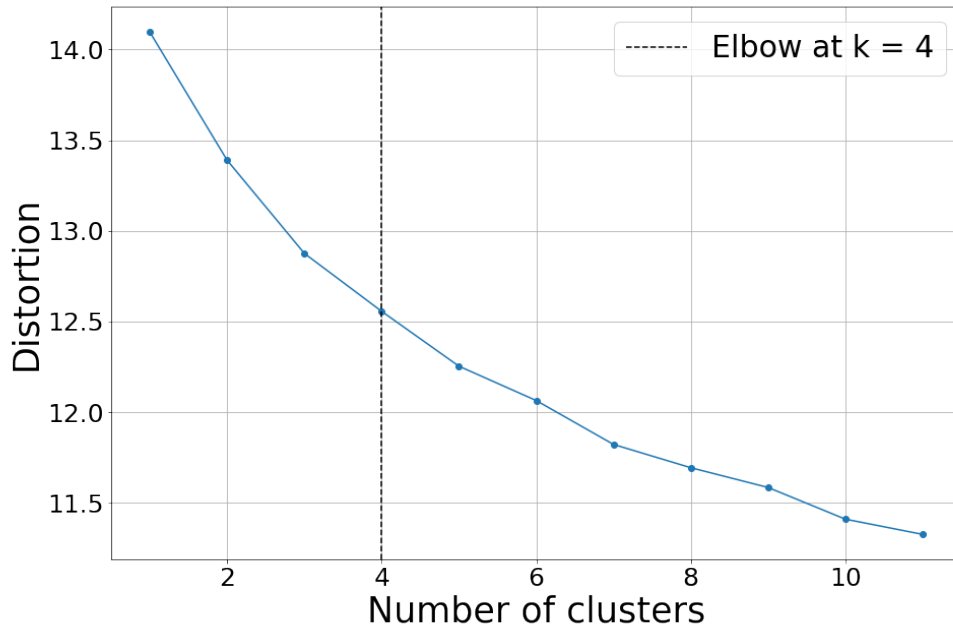


Figure 6.2: Illustration for elbow curve applied on the DTW K-means clustering algorithm. The optimal number of clusters corresponds to the elbow of the curve, which is 4 in this example.

- **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 distance directly on the K-means algorithm and DBA to compute the centroids. To reduce the computational cost, we apply resampling, and the best K is 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.

6.3 Results and Discussion

6.3.1 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

Table 6.1: 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.

Model	Clustering	Training Size 500				Training Size 1000				Training Size 2000			
		CPU		Memory		CPU		Memory		CPU		Memory	
		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

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 6.1 shows the results for CPU and memory demand prediction for all prediction models. We also plot the MSE of the memory prediction for DeepAR and TFT (together) and HBNN for all the clustering approaches by varying the number of datasets (whose scale is logarithmic) in Figures 6.3 and 6.4, respectively. Plots are separated for higher readability. From the plots, we can see that once the number of datasets increases sufficiently, all the approaches, including the baseline, converge to similar metrics values. 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. For completeness, we include all the remaining plots for MAE and CPU in Appendix C.

Experiments show 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

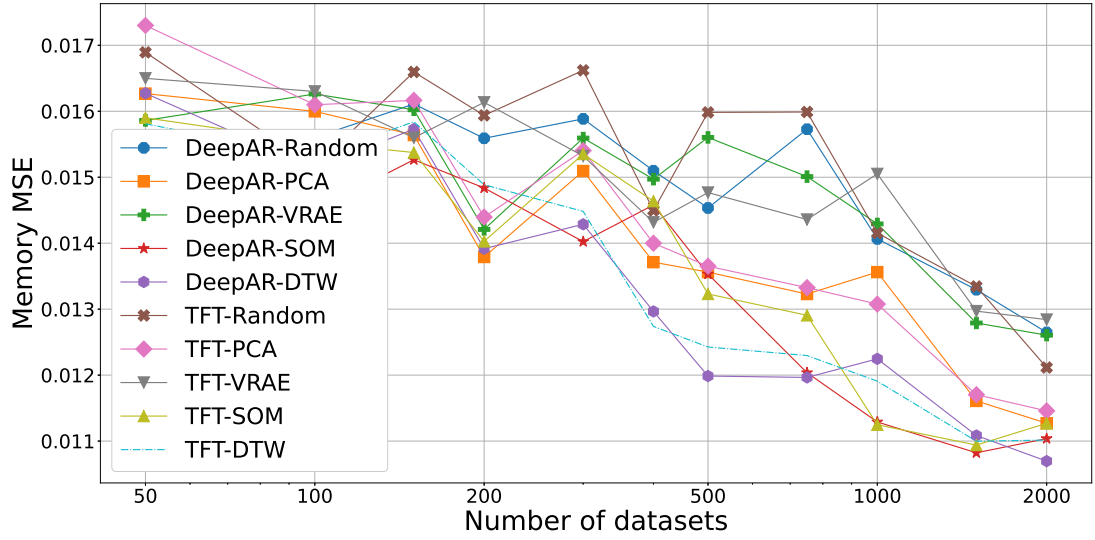


Figure 6.3: MSE comparison of DeepAR and TFT for the memory demand. DTW and SOM are the best clustering-based selection approaches, allowing the models to achieve the same MSE with fewer training datasets than a random selection. When the number of datasets is high, the MSE of all the approaches converge. The x-axis is in a logarithmic scale.

forecasting [LALP21], 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. For example, DeepAR-Random reaches an MSE of 0.014 with 1,000 training datasets, while DeepAR-DTW requires only 200 datasets to reach the same MSE. Moreover, DeepAR-Random trained with 1,000 datasets has a CPU and memory MSE 7% and 14% higher than the respective MSEs achieved by DeepAR-DTW. On the contrary, HBNN does not benefit as much as the other models from a clustered selection of the training datasets.

Overall, 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

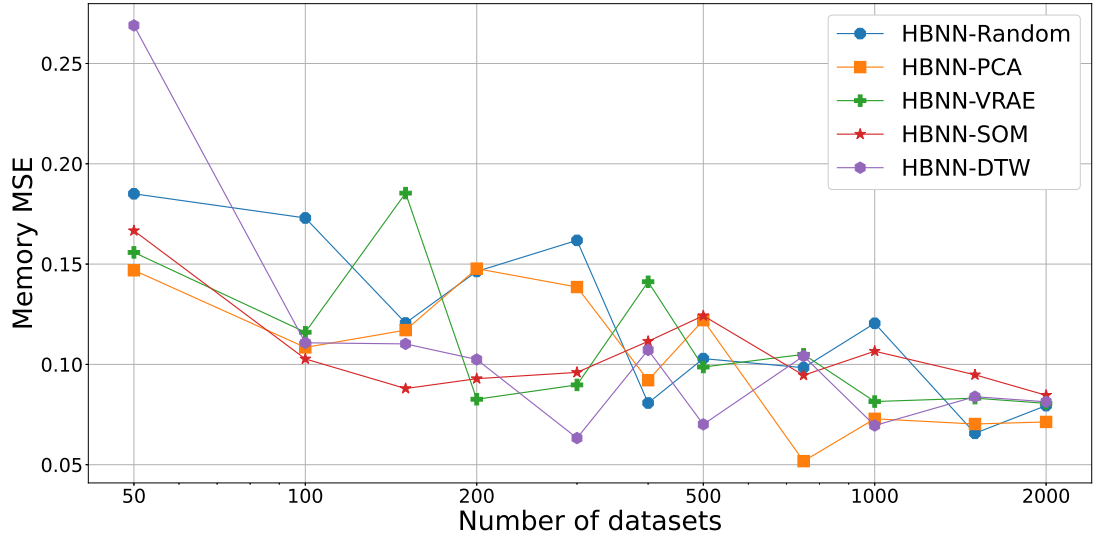


Figure 6.4: MSE comparison of HBNN for the memory demand. PCA is the best clustering-based selection approach and allows the HBNN to achieve the same MSE with fewer training datasets compared to a random selection. When the number of datasets is high, the MSE of all the approaches converge. The x-axis is in a logarithmic scale.

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. This is likely because HBNN is a simpler model compared to DeepAR and TFT, with the best clustering method being dependent on the predictive models utilized.

6.3.2 Confidence Interval Evaluation

This section evaluates the forecasting models with the confidence level metrics defined in 4.2.3: Success Rate (SR), Overprediction (OP) and Underprediction (UP) in percentage. For readability of the results, we omit the computation of Total Predicted Resources because it can be derived by combining OP and UP (as $TPR(\%) = 100 + OP(\%) - UP(\%)$).

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 Fig. 6.5, 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. We display numerical results for SR, OP and UP in Table 6.2 using 500, 1000 and 2000 datasets in training for HBNN, DeepAR and TFT models for a 95% confidence interval.

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 applications where high confidence is required. 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. Again, the HBNN 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 most datasets in training, it does not matter whether the datasets are selected randomly or via a clustering approach. This can be seen by the trend in Fig. 6.6 for the interpolated OP of memory for DeepAR and TFT models while varying the number of datasets. In the plot we excluded PCA and VRAE which are less performing for deepAR and TFT, to make it more readable. 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.

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. For completeness, we include the remaining plot in Appendix C.

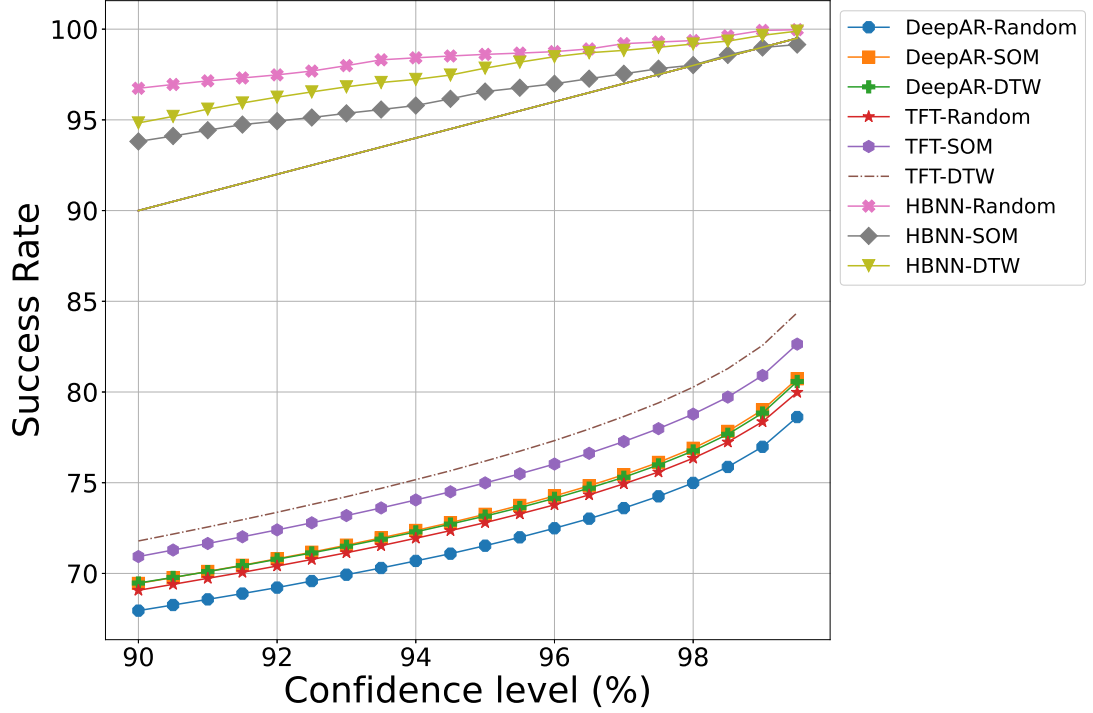


Figure 6.5: Memory SR comparison of DeepAR, TFT and HBNN as a function of the confidence level with models trained with 1000 datasets. The model is the most accurate when the curve is closest to the line $y = x$. DTW and SOM-based selections improve the SR compared to the random selection of datasets.

6.3.3 Runtime Performance Analysis

As discussed in previous chapter, 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. We are interested in measuring both the time necessary to train a model and the time to predict 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 Fig. 6.7. 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,

Table 6.2: CPU and Memory demand prediction statistics (Success Rate, Overprediction and Underprediction in percentage) of HBNN, DeepAR and TFT with 500, 1000 and 2000 training datasets at 95% confidence. In bold, for each model, the best clustering technique.

Training size	Model	Clustering	CPU			Memory		
			SR (%)	OP (%)	UP (%)	SR (%)	OP (%)	UP (%)
500	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
1000	HBNN	Random	98.35	139.93	10.30	99.49	80.99	2.57
		PCA	98.54	170.5	22.25	99.08	130.48	3.82
		VRAE	98.53	125.04	12.82	98.98	105.31	2.75
		SOM	97.64	141.81	37.20	98.33	90.05	6.41
		DTW	98.29	123.46	12.94	99.44	100.14	4.78
	DeepAR	Random	73.94	33.63	14.00	75.88	23.38	12.61
		PCA	73.23	42.07	17.43	75.65	28.26	14.19
		VRAE	73.81	33.55	14.64	75.94	24.58	12.85
		SOM	77.74	27.86	12.28	78.83	20.55	9.18
		DTW	78.28	32.59	12.75	79.05	22.33	9.17
	TFT	Random	73.85	33.73	14.20	76.77	23.92	12.60
		PCA	74.14	43.95	17.74	77.44	28.86	14.37
		VRAE	75.03	34.65	14.65	78.21	25.46	13.15
		SOM	79.84	29.34	12.43	82.23	22.13	9.09
		DTW	79.82	33.38	12.76	81.55	22.98	8.90
2000	HBNN	Random	99.2	418.12	0.25	99.99	248.18	0.0
		PCA	99.21	316.27	0.24	99.96	247.53	0.01
		VRAE	99.51	353.35	0.15	99.99	300.76	0.0
		SOM	98.66	427.27	2.07	99.37	273.71	0.07
		DTW	99.19	341.33	0.06	99.98	282.6	0.0
	DeepAR	Random	76.85	41.64	6.38	78.7	30.75	5.27
		PCA	76.31	39.56	6.29	78.84	28.78	4.48
		VRAE	76.75	39.45	6.37	78.81	30.9	5.03
		SOM	80.96	37.05	4.58	81.94	28.39	3.13
		DTW	81.45	41.53	4.38	82.21	29.39	2.93
	TFT	Random	76.76	41.94	6.49	79.64	31.92	5.08
		PCA	77.12	42.43	6.17	80.56	30.33	4.21
		VRAE	77.95	41.87	6.05	81.06	33.32	4.66
		SOM	82.91	40.75	4.15	85.38	32.46	2.57
		DTW	82.92	43.9	4.03	84.86	31.45	2.48

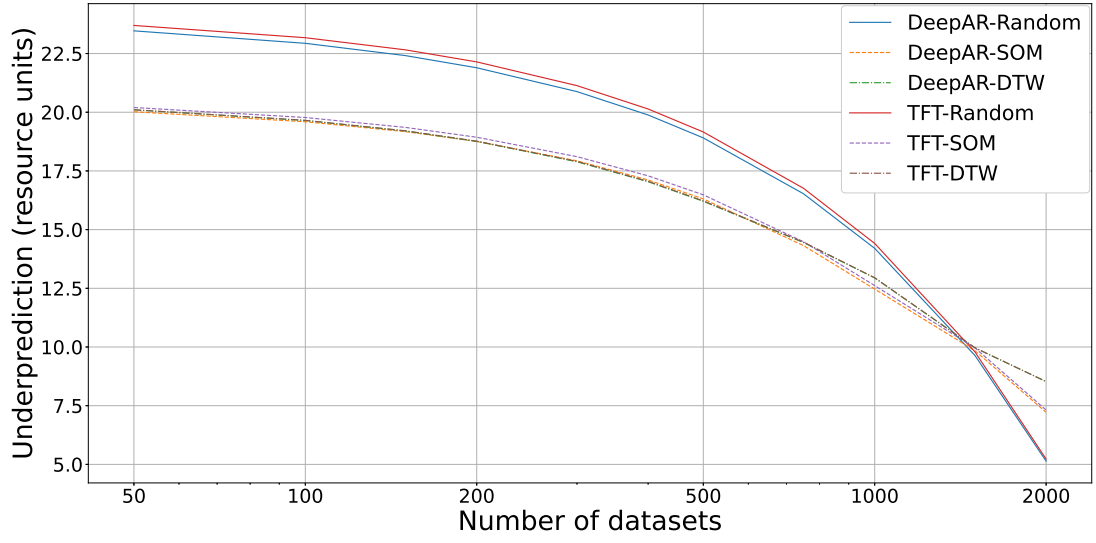


Figure 6.6: Memory OP comparison of DeepAR and TFT. If DTW and SOM are applied, a much lower number of datasets is required to get the same OP compared to a random selection of the datasets. The x-axis is in a logarithmic scale.

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.

The Bayesian layer is much slower to train because of the reparametrization trick applied in each epoch, and it scales poorly 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 differ, with Keras 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. As a result, to achieve the same MSE, a random selection requires many more datasets, compared to training based on clustering approaches, with reduced computational cost and, consequently, energy-saving. Table 6.3 lists the time-saving in percentage. This is computed as the difference in training time between the random selection of the datasets and the best clustering approach used to train the model, where the cluster-based model achieves at least the same MSE accuracy. For example, to achieve the same MSE, a random selection requires 1,000 datasets, compared to 200 or 300 when we apply

SOM (see Fig. 6.3), with a reduction of the computational cost, confirming the advantages of clustering methods compared to a random selection in terms of runtime performance.

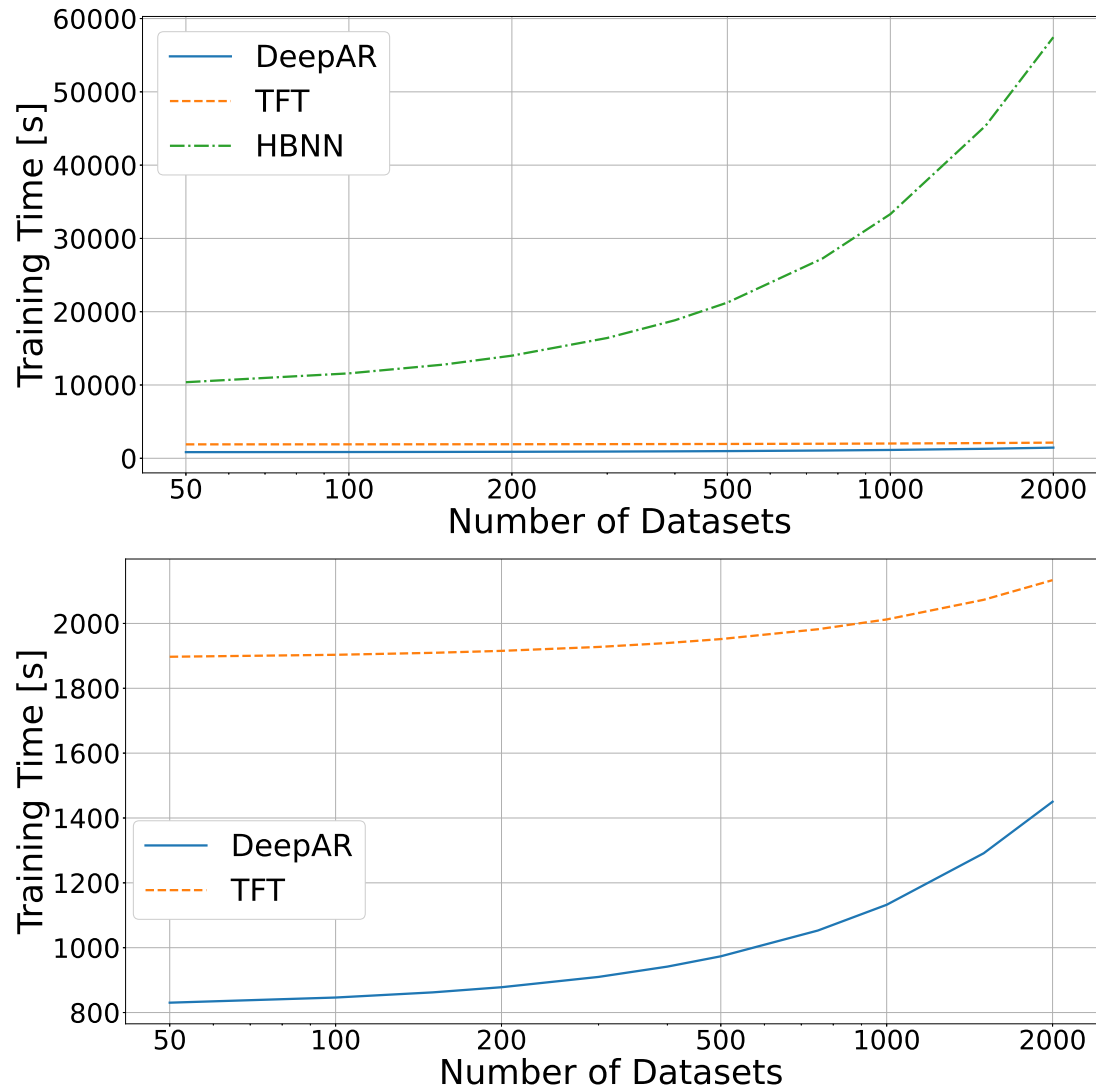


Figure 6.7: Training time curve as a function of the number of datasets. On the left, all three models are plotted. On the right, we plot only DeepAR and TFT to show the differences between the two models. The training time increases when we increase L , but TFT grows slower when the number of datasets increases, compared to the other methods. The x-axis is in a logarithmic scale.

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

Table 6.3: Time saving for all the models trained with the best clustering approach w.r.t. a random selection of the training datasets, with the baseline trained using 2000, 1500 and 1000 datasets. The training size of the clustering is selected as the minimum size required to achieve an MSE less or equal to the MSE achieved by the baseline.

Training Size Baseline	Model	CPU		Memory	
		Training Size Clustering	Time Saving (%)	Training Size Clustering	Time Saving (%)
2000	HBNN	200	75.63	400	67.23
	DeepAR	400	34.82	500	32.65
	TFT	500	8.48	500	8.49
1500	HBNN	200	69.15	400	58.51
	DeepAR	200	31.75	300	29.31
	TFT	400	6.41	500	5.83
1000	HBNN	100	65.22	300	50.73
	DeepAR	200	22.26	200	22.26
	TFT	200	4.80	500	3.00

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.

The HBNN model discussed in this chapter exhibits an inference time that is double that presented in Chapter 4 of the thesis. This increase in inference time can be attributed to the input size, which is double that of the previous chapter, expanding from 12 hours to 24 hours of historical data. This enlargement inherently requires more computational resources and processing time, thus elongating the inference duration. Additionally, architectural modifications have been made to the model. While Chapter 4 describes a model with a single convolutional layer and one dense layer, the current iteration employs three convolutional layers and two dense layers, matching the model used in Chapter 5 (M-B-HBNN model). Despite these changes, the complexity considerations for the layers remain comparable to those previously detailed in Section 4.1. The additional convolutional and dense layers, while enhancing the model's ability to capture intricate patterns and features, do not significantly alter the overall complexity framework, adhering to similar computational principles and constraints.

Therefore, 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

Table 6.4: Fitting and inference time of the clustering approaches. In bold, the best fitting and inference time.

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

training time and the inference time are lower.

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 6.4. For the clustering algorithms, SOM is the fastest method under both training and inference time, while the training phase for DTW is computationally expensive. We can conclude that, among the clustering techniques, SOM offers the best trade-off between accuracy and runtime performance.

6.4 Conclusion

This chapter analyses the effect of clustering for grouping up workload time series with similar pattern for reducing the training time of forecasting models, evaluating the accuracy and runtime performance of forecasting models that predict future machine-level CPU and memory demand distributions from GC19. We compared several clustering algorithms to a random selection of the training datasets.

DeepAR and TFT, two state-of-the-art probabilistic models are compared to the HBNN, evaluating also the prediction uncertainty given by the predicted confidence intervals, built on the top of the output distributions.

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 because are designed to deal with time series data. 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 that quantifies also the epistemic uncertainty, provides a trade-off between accuracy and SR but allocating more resources.

Overall, 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. However, the HBNN is the model that benefits most in terms of time saving, capable of achieving the same point accuracy when trained on only 10% of the original datasets.

Chapter 7

Conclusions

The contributions of this dissertation focus on modelling the uncertainty of workload forecasting in cloud computing with BNNs. Although BNNs have been used for uncertainty modelling in various fields, their application in cloud resource demand prediction is entirely new. Our proposed deep learning architecture is designed to address the computational cost of BNNs. By leveraging BNN principles only in the final network stage, we captured both inherent uncertainties (aleatoric) and uncertainties due to limited data (epistemic). We also emphasized the importance of uniformly preprocessed datasets for training this model: consistent data formatting ensures the BNN can effectively learn patterns from the data and reduces noise caused by inconsistencies. Furthermore, we addressed the question of how to leverage the model's robustness and ability to generalize to unseen cloud data centres and to improve runtime performance through data selection using clustering techniques. Our findings empower practitioners to design novel, uncertainty-aware prediction models for improved cloud operations. Incorporating confidence measures into cloud predictions leads to better-informed decisions, particularly when aiming for high service quality. Furthermore, this approach demonstrates effectiveness and introduces a generalizable methodology for uncertainty quantification applicable beyond the cloud workload domain.

Modelling the uncertainty in cloud workload prediction

In Chapter 2, after giving an overview of the types of predictive models used in cloud workload forecasting and a formulation of the problem, we surveyed solutions proposed in the literature for modelling the uncertainty of the predic-

tion, which are incorporated in the models and the most popular time series clustering methods in the literature

In Chapter 3, after surveying the most popular traces used in workload forecasting, we described the datasets chosen for the experiments (from Google Cloud and Alibaba Cloud platforms) and how they have been preprocessed. This provides a collection of uniformly preprocessed datasets that can be used to reproduce the experiments and future studies in this area.

In Chapter 4, we presented the HBNN, our solution to model the uncertainty in the cloud workload forecasts: from an existing architecture for workload forecasting, we integrated a Bayesian layer at the end of the network to take into account for the epistemic uncertainty inherent by the model, and we predicted the parameters of the future cluster-level workload distribution, incorporating aleatoric uncertainty aspects as well. Through an extensive set of experiments on the 8 cluster cells from GC19, we found that HBNN achieves efficient run-time performance for real-world applications deployment and the same point accuracy of the baselines while providing a more robust and efficient probabilistic forecast, which is superior to the model that accounts only for the aleatoric uncertainty, i.e. LSTMD or that provides only a point forecast, i.e. LSTM.

Generalisation of uncertainty-aware predictive models

In Chapter 5, we investigated whether the HBNN can be trained for bivariate forecasting, i.e. if it can jointly predict the future cluster-level processing and memory units, and compared to non-Bayesian architectures, i.e. LSTM and LSTMD, proving comparable performance to the univariate version when trained with more data. We also tested these bivariate prediction capabilities on out-of-distribution datasets not used in training. Different source and target domains, especially from other cloud providers, lead to poor TL performance, and finetuning does not always help. However, acceptable TL results are achievable when the training source domain is large enough to account for the target domain diversity, opening future research directions in this underinvestigated area.

Complexity reduction for cloud workload probabilistic forecasting

In Chapter 6, we gave an overview of the numerosity reduction techniques. Then, we applied the clustering approaches suitable for the cloud workload domain, i.e. PCA, VRAE, SOM and DTW, to group up machine-level workload datasets with similar patterns. We then sampled some datasets from each cluster or at random and used them to train bivariate forecasting models, i.e. HBNN, DeepAR and TFT. The analysis showed that clustering-based training data selection, especially clustering approaches designed for dealing with time series data, always allows for the same accuracy with a reduced time for training, which can impact the utilisation of electricity and computation resources.

In conclusion, this work has significantly contributed to cloud workload forecasting, from the definition of a dataset preprocessing methodology to a broader exploration of BNNs in this field. The proposed HBNN architecture effectively addresses both epistemic and aleatoric uncertainties, providing a more informative and robust forecast compared to existing methods that model only one type at a time, with the limitation of requiring more data and longer training time to converge. Additionally, our explorations into bivariate forecasting and clustering-based training data selection (to reduce training time) open the opportunity for further research and optimization in this domain. By addressing the limitations in transfer learning and exploring the full potential of clustering techniques, we can achieve even more efficient and accurate cloud workload forecasting in the future, allowing cloud providers to offer more sustainable and cost-effective services.

Despite the significant contributions, this work is not without its limitations. Firstly, the datasets used for the experiments, while comprehensive, are limited to the specific characteristics of workloads from Google Cloud and Alibaba Cloud platforms. This limitation may affect the generalizability of the findings to other cloud providers with different workload patterns. Secondly, the methodology adopted for preprocessing and modelling involves several assumptions and simplifications, such as the uniform preprocessing of datasets, which might not capture all real-world complexities and variations in cloud workloads.

Moreover, the possibility of randomness in the results and findings cannot be

entirely ruled out. The experimental results are based on a finite set of data and specific configurations of the models. While efforts have been made to ensure robustness, inherent randomness in the data and the stochastic nature of the training algorithms, including the selection of random seeds, may impact the reproducibility of the results to some extent. The use of different random seeds can lead to variations in the initialization of model parameters, training data shuffling, and other aspects of the model training process, which might introduce variability in the outcomes. Therefore, further validation with different datasets, cloud environments, and extended experimentation is necessary to strengthen the conclusions and enhance the reliability of the proposed models.

7.1 Future Work

In this dissertation, we shed light on uncertainty modelling for cloud workload forecasting. However, our research is open to further extensions. This section gives a brief overview of some of these opportunities.

Multistep-ahead prediction and resource optimisation

When we trained DL models and evaluated their performance, we limited the analysis to the forecasting task and only evaluating one prediction at a time, i.e. given the past input, the models forecast the future resource demand only of the 5-minute interval required in 10 minutes, and they do not provide subsequent forecasts. However, some cloud resource management applications, such as resource provisioning, must be evaluated over multi-step-ahead predictions as they aim to optimise the resources over time. Resource provisioning can be defined as an optimisation problem that maximises the efficiency of resource utilisation while minimising operational costs (e.g., the cost of using the machines) and it is subject to some domain constraints, such as QoS guarantees while limiting the energy consumption [LL12]. With this objective, the proposed forecasting models can be extended to support multi-step-ahead predictions and the output can be used in cloud optimisation problems, with the opportunity of further showing the advantages of uncertainty-aware forecasts.

New deep learning methods for cloud workload forecasting

In Chapter 2, numerous models have been presented to address the challenges of forecasting future workloads in cloud computing environments. As this dis-

sertation introduces the HBNN and explores other recent methods for workload probabilistic forecasting, a promising avenue for future research involves conducting a comprehensive comparative analysis with the myriad existing models. Leveraging the workload prediction literature and exploring diverse approaches, such as novel deep learning architectures yet underinvestigated, can provide valuable insights into the trend of cloud workload forecasting models. In recent years, the prevalence of transformer-based models is evident when looking at NLP and Computer Vision applications, dominated by Large Language Models and Generative AI. While transformers have demonstrated remarkable success, their primary drawback lies in the substantial training data requirements, which limits their applicability in cloud workload scenarios. To address this limitation, future work should focus first on collecting more data and preprocessing more available datasets, such as those presented in Chapter 3, enabling the effective utilisation of transformers for workload prediction in cloud computing. Furthermore, the probabilistic extensions of the transformer [TM21] present an exciting exploration for uncertainty quantification in cloud workload forecasting and a fair competitor for the proposed HBNN.

Concurrent with these developments, conformal prediction [GVV13] has become a powerful tool for uncertainty quantification, providing reliable prediction confidence intervals. Although initially challenging to apply to time series because it assumes the exchangeability of the input data, recent advancements have made it feasible to explore integrating conformal prediction methods into forecasting models [SMAvdS21, XX23]. This extension would contribute to a more robust and interpretable framework for uncertainty quantification in workload forecasting. Conformal prediction has been integrated into a very recent time series forecasting model called TimeGPT [GMC23], a transformer-based architecture trained with over 10 billion data points of different natures, including finance, demographics, healthcare, weather, energy and sales data. Such a model, considered the first foundational model for time series forecasting, is tailored for hourly, daily, weekly, and monthly prediction, which are the granularity of most time series forecasting problems. This poses interesting research questions about whether this model can be trained or finetuned for a more fine-grained prediction (for instance, predicting 5-minute intervals instead of hourly or daily forecasts). This approach could offer superior accuracy and adaptability to varying time scales, exploiting data from different domains and extending the TL concept, presented in Chapter 5, providing a comprehensive solution for cloud service providers aiming to optimise resource allocation

and enhance overall system performance.

Understanding the prediction: explainability for cloud workload forecasting

As AI continues to permeate various domains, the growing emphasis on explainability and trustworthiness is a significant trend [KURD22]. Future research in cloud workload forecasting should incorporate explainable Artificial Intelligence (XAI) techniques to enhance the transparency and interpretability of predictive models. The need for explainability can be particularly crucial in applications such as cloud computing, where decision-making processes have far-reaching consequences on resource allocation, cost optimisation, and overall system performance. Recently, XAI approaches like SHAP (SHapley Additive exPlanations) [LL17] that can help in generating human-understandable explanations for complex machine learning models, have been extended to time series data [ZSQ⁺23]. This opens the opportunity to incorporate XAI approaches into cloud workload forecasting models. Moreover, explainability can contribute to identifying and mitigating biases present in the training data or the underlying algorithms, ensuring fairness and preventing unintended consequences. By understanding how the model arrives at specific predictions, cloud computing providers can intervene to rectify biases or inaccuracies, promoting ethical and responsible AI practices. In this direction, by embracing XAI techniques and developing transparent forecasting models, future research can contribute to building more trustworthy, fair, and green AI systems for cloud computing, ultimately advancing the field and addressing the evolving needs of stakeholders in the cloud ecosystem and address the requirements posed by the newly approved EU AI Act, the world's first comprehensive AI law.

Towards new computing paradigms: fog and edge computing

While cloud computing provides scalable, on-demand services over the internet but can face challenges with latency and bandwidth utilisation, fog computing extends cloud computing to the edge of a network, enabling real-time data analysis and decision-making, but can face challenges with data processing at the edge of a decentralised network. On the other hand, edge computing processes data directly on devices at the source, ensuring high operational speed and efficiency and minimising processing time by removing the need to transfer data to a central processing system. Thus, the primary difference between

these three computing frameworks is their data processing location [Ati21]. As more Internet of Things devices and applications require quick processing, the drawbacks of centralised cloud structures are becoming clear. Current research aims to improve fog and edge computing, taking advantage of their closeness to data sources for better speed and response times. Decentralised paradigms are becoming dominant along with distributed learning approaches such as Federated Learning (FL), which allows multiple devices or servers to train a model without sharing their data [LFTL20]. Developing privacy-protecting methods, like FL, is important for secure and efficient decentralised computing. Future work can focus on adopting FL algorithms for forecasting the future demand of resources in fog and edge networks, extending the cloud computing paradigm.

Acronyms

AC18 Alibaba Cluster Trace 2018. iii, vi, 50, 53, 56, 59, 62, 64, 65, 92, 105, 107, A7

AC20 Alibaba Cluster Trace 2020. iii, vi, 50, 54, 58–60, 62, 64, 65, 92, 105, 107, 108, A7

ADF augmented Dickey-Fuller. 60, 61

AE Auto-Encoder. 117

AI artificial intelligence. 1, 58, 139, 140

AIC Akaike Information Criterion. 74

AR autoregressive. 17, 19

ARIMA AutoRegressive Integrated Moving Average. viii, 16–18, 21, 73–75, 79

BNN Bayesian Neural Network. 4, 5, 7, 15, 30, 31, 33, 34, 36, 46, 69, 89, 135, 137

CNN Convolutional Neural Network. 24, 67

DBA DTW Barycenter Averaging. 44, 120, 123

DBN Deep Belief Network. 22, 23

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 44, 45, 120

DC data centre. 1, 2, 5, 13, 49, 50

DL deep learning. iii, 2, 4, 5, 11, 12, 15, 16, 22, 25, 26, 28, 31, 33, 45, 50, 54, 67, 71, 73, 74, 79, 89, 92, 115, 116, 121, 125, 131, 138

- DTW** Dynamic Time Warping. v, 43, 44, 119, 120, 122, 123, 125, 127, 128, 133, 137
- ELBO** evidence lower bound. 35, 36
- ETS** exponential smoothing. 19
- FL** Federated Learning. 141
- FT** fine-tuning. 104, 106, 107, 109, 112
- GARCH** Generalized AutoRegressive Conditional Heteroskedasticity. viii, 18, 19, 73–75, 79
- GC11** Google Cloud Trace 2011. iii, vi, 49, 53–56, 62, 64, 65, 92, 105, 107, A7
- GC19** Google Cloud Trace 2019. ii–iv, vi, 49, 54–58, 62–64, 72, 73, 83–87, 89, 92, 105–109, 119, 120, 133, 136, A3
- HBNN** Hybrid Bayesian Neural Network. iii, iv, viii, ix, xvi, xvii, 7–9, 11, 12, 66, 67, 69–75, 78–82, 84, 86, 88–97, 99, 101, 102, 104, 111, 112, 115, 116, 120, 121, 124–127, 130–134, 136, 137, 139
- KL** Kullback-Leibler. 35, 36, 41
- KNN** K-nearest neighbours. 20, 22
- LSTM** Long Short-Term Memory. iii, viii, ix, 23, 24, 37, 67–71, 73–75, 78–81, 86, 88, 89, 92–94, 96, 97, 102, 136
- LSTMD** LSTM Distributional. iii, iv, viii, ix, 68–75, 78–82, 84, 86, 88, 89, 92–97, 99, 102, 105, 111, 136
- MA** moving average. 17
- MAE** mean absolute error. 14, 72, 75, 79, 93–95, 99, 108, 119, 120, 124
- ML** machine learning. 3, 14–16, 20–23, 25, 27, 34, 42, 45, 58, 67, 90, 103, 104
- MLE** maximum likelihood estimation. 33, 73, 74
- MSE** mean squared error. ix, 14, 72–75, 79, 93–95, 99, 108, 119, 120, 124, 125, 130, 132

- NLP** Natural Language Processing. 4, 38, 115, 139
- NN** neural network. 16, 22, 23, 30, 33, 34, 37–39, 69
- OP** Overprediction. iv, 76–78, 80–82, 85, 86, 89, 93, 120, 126, 127
- PCA** Principal Component Analysis. 41, 42, 117, 119, 122, 125, 127, 133, 137
- QoS** quality of service. 2–5, 13, 18, 72, 76, 80, 138
- QR** quantile regression. 30, 31, 36
- RF** Random forest. 20, 21
- RNN** Recurrent Neural Network. 23–25, 36, 39, 41
- SLA** service level agreement. xvi, 2, 13, 76
- SOM** Self-Organising Map. 42, 43, 119, 123, 125, 127, 128, 131, 133, 134, 137
- SR** Success Rate. iv, 76–78, 80–86, 89, 93, 94, 96, 99, 109, 120, 126, 127, 134
- SVR** Support vector regression. 21, 22
- TCN** Temporal Convolutional Network. 24
- TFT** Temporal Fusion Transformer. 37, 120–122, 124–127, 131–133, 137
- TL** Transfer learning. 8, 90, 91, 102–105, 107, 109, 110, 112, 113, 136, 139
- TPR** Total Predicted Resources. iv, 77, 78, 80–82, 84, 85, 89, 93, 94, 96, 126
- UB** upper bound. 75–80, 94, 121, 127
- UP** Underprediction. 76–78, 80, 81, 93, 120, 126, 127
- VI** variational inference. 31, 34–36, 62
- VM** virtual machine. 13, 18, 49, 50
- VRAE** Variational Recurrent Auto-Encoder. 41, 119, 122, 125, 127, 133, 137
- XAI** explainable Artificial Intelligence. 140
- ZS** zero-shot. 104, 108, 112

References

- [AAEBS23] Sundus Naji Alaziz, Bakr Albayati, Abd al-Aziz H El-Bagoury, and Wasswa Shafik. Clustering of covid-19 multi-time series-based k-means and pca with forecasting. *International Journal of Data Warehousing and Mining (IJDWM)*, 19(3):1–25, 2023.
- [AAS⁺21] Ziyad R Alashhab, Mohammed Anbar, Manmeet Mahinderjit Singh, Yu-Beng Leau, Zaher Ali Al-Sai, and Sami Abu Alhayja’a. Impact of coronavirus pandemic crisis on technologies and cloud computing applications. *Journal of Electronic Science and Technology*, 19(1):100059, 2021.
- [AB13] Samuel A Ajila and Akindele A Bankole. Cloud client prediction models using machine learning techniques. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 134–142. IEEE, 2013.
- [ABAS⁺23] Laith Alzubaidi, Jinshuai Bai, Aiman Al-Sabaawi, Jose Santamaría, AS Albahri, Bashar Sami Nayyef Al-dabbagh, Mohammed A Fadhel, Mohamed Manoufali, Jinglan Zhang, Ali H Al-Timemy, et al. A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, 10(1):46, 2023.
- [Abo22] Mahmoud Abouelyazid. Forecasting resource usage in cloud environments using temporal convolutional networks. *Applied Research in Artificial Intelligence and Cloud Computing*, 5(1):179–194, 2022.
- [ACC⁺14] Danilo Ardagna, Giuliano Casale, Michele Ciavotta, Juan F Pérez, and Weikun Wang. Quality-of-service in cloud computing: mod-

- eling techniques and their applications. *Journal of Internet Services and Applications*, 5(1):1–17, 2014.
- [ACOM23] Daniel Bouzon Nagem Assad, Javier Cara, and Miguel Ortega-Mier. Comparing short-term univariate and multivariate time-series forecasting models in infectious disease outbreak. *Bulletin of Mathematical Biology*, 85(1):9, 2023.
- [AHPV⁺05] Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52(1):1–30, 2005.
- [AJL⁺22] Shivani Arbat, Vinodh Kumaran Jayakumar, Jaewoo Lee, Wei Wang, and In Kee Kim. Wasserstein adversarial transformer for cloud workload prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12433–12439, 2022.
- [AK23a] Shallaw Mohammed Ali and Gabor Kecskemeti. Sequal: an unsupervised feature selection method for cloud workload traces. *The Journal of Supercomputing*, pages 1–19, 2023.
- [AK23b] Samah F Aziz and Manar Y Kashmoola. Workload forecasting methods in cloud environments: An overview. *AL-Rafidain Journal of Computer Sciences and Mathematics*, 17(2):29–37, 2023.
- [Aka98] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotogu akaike*, pages 199–213. Springer, 1998.
- [Ali24] Alibaba Cloud. Ahpa resilience forecasting best practices, 2024.
- [APH⁺21] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Reza-zadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.
- [ARA23] Aastha Acharya, Rebecca Russell, and Nisar R Ahmed. Learning to forecast aleatoric and epistemic uncertainties over long horizon trajectories. *arXiv preprint arXiv:2302.08669*, 2023.

- [AS22] Mustafa M Al-Sayed. Workload time series cumulative prediction mechanism for cloud resources using neural machine translation technique. *Journal of Grid Computing*, 20(2):16, 2022.
- [ASKO16] Mustafa M Al-Sayed, Sherif Khattab, and Fatma A Omara. Prediction mechanisms for monitoring state of cloud resources using markov chain model. *Journal of Parallel and Distributed Computing*, 96:163–171, 2016.
- [Ati21] Ali T Atieh. The next generation cloud technologies: a review on distributed cloud, fog and edge computing and their opportunities and challenges. *ResearchBerg Review of Science and Technology*, 1(1):1–15, 2021.
- [AV17] Héctor Allende and Carlos Valle. Ensemble methods for time series forecasting. *Claudio Moraga: A passion for multi-valued logic and soft computing*, pages 217–232, 2017.
- [AW96] Martin F Arlitt and Carey L Williamson. Web server workload characterization: The search for invariants. *ACM SIGMETRICS Performance Evaluation Review*, 24(1):126–137, 1996.
- [Bar07] Guilherme A Barreto. Time series prediction with the self-organizing map: A review. *Perspectives of neural-symbolic integration*, pages 135–158, 2007.
- [BCKW15] 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.
- [BDN⁺20] Asma Belhadi, Youcef Djenouri, Kjetil Nørvåg, Heri Ramampiaro, Florent Masseglia, and Jerry Chun-Wei Lin. Space–time series clustering: Algorithms, taxonomy, and case study on urban smart cities. *Engineering Applications of Artificial Intelligence*, 95:103857, 2020.
- [BE18] Lotfi Belkhir and Ahmed Elmeligi. Assessing ict global emissions footprint: Trends to 2040 & recommendations. *Journal of cleaner production*, 177:448–463, 2018.
- [Bel66] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

- [BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- [BLYZ21] 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.
- [BMR⁺20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [Bol86] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.
- [BP79] Trevor S Breusch and Adrian R Pagan. A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the econometric society*, pages 1287–1294, 1979.
- [Bre01] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
- [BRGB⁺16] 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.
- [BRK21] Sounak Banerjee, Sarbani Roy, and Sunirmal Khatua. Efficient resource utilization using multi-step-ahead workload prediction technique in cloud. *The Journal of Supercomputing*, 77:10636–10663, 2021.
- [Bro56] Robert G Brown. *Exponential smoothing for predicting demand*. Little, 1956.
- [BS15] Masoud Barati and Saeed Sharifian. A hybrid heuristic-based tuned support vector regression model for cloud load prediction. *The Journal of Supercomputing*, 71:4235–4259, 2015.
- [BT11] George EP Box and George C Tiao. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

- [BWL⁺23] Veronica Biolcati, James Woolley, Élodie Lévêque, Andrea Rossi, Anna Grace Hoffmann, Andrea Visentin, Pádraig Ó Macháin, and Daniela Iacopino. Establishing the original order of the poems in harward’s almanac using paleography, codicology, x-ray fluorescence spectroscopy, and statistical analysis. *Heritage Science*, 11(1):1–14, 2023.
- [BYXS22] Jiaang Bao, Chao Yang, Nu Xia, and Dian Shen. Long-term workload forecasting in grid cloud using deep ensemble model. In *2022 Tenth International Conference on Advanced Cloud and Big Data (CBD)*, pages 42–47. IEEE, 2022.
- [CAK12] Yanpei Chen, Sara Alspaugh, and Randy Katz. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *arXiv preprint arXiv:1208.4174*, 2012.
- [CBM⁺17] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.
- [CCB11] Aymen Cherif, Hubert Cardot, and Romuald Boné. Som time series clustering and prediction with recurrent neural networks. *Neurocomputing*, 74(11):1936–1944, 2011.
- [CCR⁺03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.
- [CGGK11] Yanpei Chen, Archana Ganapathi, Rean Griffith, and Randy Katz. The case for evaluating mapreduce performance using workload suites. In *2011 IEEE 19th annual international symposium on modelling, analysis, and simulation of computer and telecommunication systems*, pages 390–399. IEEE, 2011.
- [CH16] Masayoshi Chayama and Yoshito Hirata. When univariate model-free time series prediction is better than multivariate. *Physics Letters A*, 380(31-32):2359–2365, 2016.

- [Che19] M Yu Chesnokov. Time series anomaly searching based on db-scan ensembles. *Scientific and Technical Information Processing*, 46:299–305, 2019.
- [CHM⁺19] Zheyi Chen, Jia Hu, Geyong Min, Albert Y Zomaya, and Tarek El-Ghazawi. Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. *IEEE Transactions on Parallel and Distributed Systems*, 31(4):923–934, 2019.
- [CJBR21] Adam D Cobb, Brian Jalaian, Nathaniel D Bastian, and Stephen Russell. Toward safe decision-making via uncertainty quantification in machine learning. *Systems Engineering and Artificial Intelligence*, pages 379–399, 2021.
- [CLKK16] Fan Cai, Nhien-An Le-Khac, and Tahar Kechadi. Clustering approaches for financial data analysis: a survey. *arXiv preprint arXiv:1609.08520*, 2016.
- [CLY⁺21] Wenyan Chen, Chengzhi Lu, Kejiang Ye, Yang Wang, and Cheng-Zhong Xu. Rptcn: Resource prediction for high-dynamic workloads in clouds based on deep learning. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 59–69. IEEE, 2021.
- [CMA23] Andrea Cini, Danilo Mandic, and Cesare Alippi. Graph-based time series clustering for end-to-end hierarchical forecasting. *arXiv preprint arXiv:2305.19183*, 2023.
- [CMRB14] 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.
- [CN20] João Caldeira and Brian Nord. Deeply uncertain: comparing methods of uncertainty quantification in deep learning algorithms. *Machine Learning: Science and Technology*, 2(1):015002, 2020.
- [COO⁺23] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecast-

- ing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6989–6997, 2023.
- [CPD23] Georgia Christofidi, Konstantinos Papaioannou, and Thaleia Dimitra Doudali. Toward pattern-based model selection for cloud resource forecasting. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, pages 115–122, 2023.
- [CRB⁺11] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.
- [CRV⁺23] Diego Carraro, Andrea Rossi, Andrea Visentin, Steven Prestwich, and Kenneth N Brown. Performance and energy savings trade-off with uncertainty-aware cloud workload forecasting. In *2023 IEEE 31st International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2023.
- [CTS19] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Machine learning vs statistical methods for time series forecasting: Size matters. *arXiv preprint arXiv:1909.13316*, 2019.
- [CZ21] Lei Chen and Weiwen Zhang. A deep learning-based approach with pso for workload prediction of containers in the cloud. In *2021 13th International Conference on Advanced Infocomm Technology (ICAIT)*, pages 204–208. IEEE, 2021.
- [CZY22] Lei Chen, Weiwen Zhang, and Haiming Ye. Accurate workload prediction for edge data centers: Savitzky-golay filter, cnn and bilstm with attention mechanism. *Applied Intelligence*, 52(11):13027–13042, 2022.
- [DAMZ21] Mustafa Daraghmeh, Anjali Agarwal, Ricardo Manzano, and Marzia Zaman. Time series forecasting using facebook prophet for cloud resource management. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2021.
- [DBK⁺96] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression ma-

- chines. *Advances in neural information processing systems*, 9, 1996.
- [DCdCKG19] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in neural information processing systems*, 32, 2019.
- [DF79] David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.
- [DF16] Andrey Davydenko and Robert Fildes. Forecast error measures: critical review and practical recommendations. *Business forecasting: Practical problems and solutions*, 34, 2016.
- [DKC12] 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.
- [DKC14] Sheng Di, Derrick Kondo, and Walfredo Cirne. Google host-load prediction based on bayesian model with optimized feature combination. *Journal of Parallel and Distributed Computing*, 74(1):1820–1832, 2014.
- [DKD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.
- [DKMS23] Javad Dogani, Farshad Khunjush, Mohammad Reza Mahmoudi, and Mehdi Seydali. Multivariate workload and resource prediction in cloud computing using cnn and gru by attention mechanism. *The Journal of Supercomputing*, 79(3):3437–3470, 2023.
- [DLCÖ19] Thang Le Duc, Rafael García Leiva, Paolo Casari, and Per-Olov Östberg. Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. *ACM Computing Surveys (CSUR)*, 52(5):1–39, 2019.
- [DM02] Francis X Diebold and Robert S Mariano. Comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1):134–144, 2002.

- [DMD⁺17] Martin Duggan, Karl Mason, Jim Duggan, Enda Howley, and Enda Barrett. Predicting host cpu utilization in cloud computing using recurrent neural networks. In *2017 12th international conference for internet technology and secured transactions (ICITST)*, pages 67–72. IEEE, 2017.
- [DO00] Peter A Dinda and David R O’Hallaron. Host load prediction using linear models. *Cluster Computing*, 3:265–280, 2000.
- [DPW03] Johann Du Preez and Stephen F Witt. Univariate versus multivariate time series forecasting: an application to international tourism demand. *International Journal of Forecasting*, 19(3):435–451, 2003.
- [DQRT15] Diego Didona, Francesco Quaglia, Paolo Romano, and Ennio Torre. Enhancing performance prediction robustness by combining analytical modeling and machine learning. In *Proceedings of the 6th ACM/SPEC international conference on performance engineering*, pages 145–156, 2015.
- [DQY21] Nhat-Minh Dang-Quang and Myungsik Yoo. Multivariate deep learning model for workload prediction in cloud computing. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 858–862. IEEE, 2021.
- [DR23] Tom Djaaleb and Julien Romero. Evaluating and finetuning models for financial time series forecasting. 2023.
- [EPM20] Carson Eisenach, Yagna Patel, and Dhruv Madeka. Mqtransformer: Multi-horizon forecasts with context dependent and feedback-aware attention. *arXiv preprint arXiv:2009.14799*, 2020.
- [EVG15] Herman A Engelbrecht and M Van Greunen. Forecasting methods for cloud hosted resources, a comparison. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 29–35. IEEE, 2015.
- [FFW⁺18] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhasane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. In *2018 IEEE international conference on big data (Big Data)*, pages 1367–1376. IEEE, 2018.

- [FFW⁺19] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhasane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.
- [FPL⁺16] Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Nguyen Trung Hieu, and Hannu Tenhunen. Energy-aware vm consolidation in cloud data centers using utilization prediction model. *IEEE Transactions on Cloud Computing*, 7(2):524–536, 2016.
- [FPRA20] Abolfazl Farahani, Behrouz Pourshojae, Khaled Rasheed, and Hamid R Arabnia. A concise review of transfer learning. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 344–351. IEEE, 2020.
- [Fre09] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.
- [Fri01] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
- [FS18] Lingbing Feng and Yanlin Shi. Forecasting mortality rates: multivariate or univariate models? *Journal of Population Research*, 35:289–318, 2018.
- [FVA14] Otto Fabius and Joost R Van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.
- [FVT22] Levente Foldesi and Matias Valdenegro-Toro. Comparison of uncertainty quantification with deep learning in time series regression. In *NeurIPS 2022 Workshop on Robustness in Sequence Modeling*, 2022.
- [GG16] Yarín Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [GHO17] Roger Ghanem, David Higdon, and Houman Owhadi. *Handbook of uncertainty quantification*. Springer, 2017.

- [GLP10] Chonghui Guo, Hailin Li, and Donghua Pan. An improved piecewise aggregate approximation based on statistical features for time series mining. In *Knowledge Science, Engineering and Management: 4th International Conference, KSEM 2010, Belfast, Northern Ireland, UK, September 1-3, 2010. Proceedings 4*, pages 234–244. Springer, 2010.
- [GMC23] Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.
- [Goo24] Google Cloud. Manage traffic and load for your workloads in google cloud, 2024.
- [Gra11] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.
- [GSA21] Aastha Gupta, Himanshu Sharma, and Anas Akhtar. A comparative analysis of k-means and hierarchical clustering. *EPRA International Journal of Multidisciplinary Research (IJMR)*, 7(8), 2021.
- [GVV13] Alex Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. *arXiv preprint arXiv:1301.7375*, 2013.
- [GWS20] 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.
- [HA18] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.
- [HAA⁺17] Nikolas Herbst, Ayman Amin, Artur Andrzejak, Lars Grunske, Samuel Kounev, Ole J Mengshoel, and Priya Sundararajan. On-line workload forecasting. *Self-Aware Computing Systems*, pages 529–553, 2017.
- [HDM15] Ramzi A Haraty, Mohamad Dimishkieh, and Mehedi Masud. An enhanced k-means clustering algorithm for pattern discovery in healthcare data. *International Journal of distributed sensor networks*, 11(6):615740, 2015.
- [HDPW16] Yazhou Hu, Bo Deng, Fuyang Peng, and Dongxia Wang. Workload prediction for cloud computing elasticity mechanism. In

- 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 244–249. IEEE, 2016.
- [HHKA13] Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource provisioning. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 187–198, 2013.
- [HHKA14] 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.
- [HJL⁺14] Rongdong Hu, Jingfei Jiang, Guangming Liu, Lixin Wang, et al. Efficient resources provisioning based on load forecasting in cloud. *The Scientific World Journal*, 2014, 2014.
- [HLZ21] Ziyi Huang, Henry Lam, and Haofeng Zhang. Quantifying epistemic uncertainty in deep learning. *arXiv preprint arXiv:2110.12122*, 2021.
- [HPK11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*, pages 111–118. Elsevier, 2011.
- [HSV21] Lingxiao Huang, K Sudhir, and Nisheeth Vishnoi. Coresets for time series clustering. *Advances in Neural Information Processing Systems*, 34:22849–22862, 2021.
- [HSY⁺21] Qinghao Hu, Peng Sun, Shengen Yan, Yonggang Wen, and Tianwei Zhang. Characterization and prediction of deep learning workloads in large-scale gpu datacenters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2021.
- [HW21] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110:457–506, 2021.
- [HYZ⁺21] Jia Hao, Kun Yue, Binbin Zhang, Liang Duan, and Xiaodong Fu. Transfer learning of bayesian network for measuring qos of virtual machines. *Applied Intelligence*, 51(12):8641–8660, 2021.

- [HZ19] Yangdong He and Jiabao Zhao. Temporal convolutional networks for anomaly detection in time series. In *Journal of Physics: Conference Series*, volume 1213, page 042050. IOP Publishing, 2019.
- [HZH22] Zichao He, Chunna Zhao, and Yaqun Huang. Multivariate time series deep spatiotemporal forecasting with graph neural network. *Applied Sciences*, 12(11):5731, 2022.
- [IG20] Sai Inabattini and Priyanka Gayam. Workload volatility management in datacenters and cloud. In *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 82–86. IEEE, 2020.
- [IKLL12] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162, 2012.
- [IM16] Salam Ismaeel and Ali Miri. Multivariate time series elm for cloud data centre workload prediction. In *Human-Computer Interaction. Theory, Design, Development and Practice: 18th International Conference, HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings, Part I 18*, pages 565–576. Springer, 2016.
- [JB17] 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. IEEE, 2017.
- [JBP22] Praphula Kumar Jain, Mani Shankar Bajpai, and Rajendra Pampula. A modified dbscan algorithm for anomaly detection in time-series data with seasonality. *Int. Arab J. Inf. Technol.*, 19(1):23–28, 2022.
- [JLB⁺22] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

- [JLKW20] Vinodh Kumaran Jayakumar, Jaewoo Lee, In Kee Kim, and Wei Wang. A self-optimized generic workload prediction framework for cloud computing. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 779–788. IEEE, 2020.
- [JQS⁺20] Congfeng Jiang, Yitao Qiu, Weisong Shi, Zhefeng Ge, Jiwei Wang, Shenglei Chen, Christophe Cerin, Zujie Ren, Guoyao Xu, and Jiangbin Lin. Characterizing co-located workloads in alibaba cloud datacenters. *IEEE Transactions on Cloud Computing*, 2020.
- [JS15] Brendan Jennings and Rolf Stadler. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, 23:567–619, 2015.
- [JSLH22] Weikuan Jia, Meili Sun, Jian Lian, and Sujuan Hou. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693, 2022.
- [JVP⁺19] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Wencong Xiao, and Fan Yang. Analysis of {Large-Scale}{Multi-Tenant}{GPU} clusters for {DNN} training workloads. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 947–960, 2019.
- [Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.
- [KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [KGR⁺24] Syed Karimunnisa, Arunkumar Gopu, T Prabhakara Rao, M Ayyadurai, and ESaravana Kumar. A novel workload forecasting model for cloud computing using alaa-dnn algorithm. *Multimedia Tools and Applications*, pages 1–25, 2024.
- [KHH20] 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.
- [KJS24] Praveen Kumar Kollu, Tejaswini Sambraajyam Janjanam, and Kavya Sharmila Siram. Comparative analysis of cloud resources

- forecasting using deep learning techniques based on vm workload traces. *Transactions on Emerging Telecommunications Technologies*, 35(1):e4933, 2024.
- [KKHN18] HM Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE access*, 6:36218–36234, 2018.
- [KKM⁺21] HM Dipu Kabir, Abbas Khosravi, Subrota K Mondal, Mustaneer Rahman, Saeid Nahavandi, and Rajkumar Buyya. Uncertainty-aware decisions in cloud computing: Foundations and future directions. *ACM Computing Surveys (CSUR)*, 54(4):1–30, 2021.
- [KMDA21] Md Ebtidaul Karim, Mirza Mohd Shahriar Maswood, Sunanda Das, and Abdullah G Alharbi. Bhyprec: a novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine. *IEEE Access*, 9:131476–131495, 2021.
- [Koh90] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [KP24] Syed Karimunnisa and Yellamma Pachipala. Deep learning approach for workload prediction and balancing in cloud computing. *International Journal of Advanced Computer Science & Applications*, 15(4), 2024.
- [Kra91] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.
- [KRA⁺14] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, pages 232–238. IEEE, 2014.
- [KS18] Jitendra Kumar and Ashutosh Kumar Singh. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41–52, 2018.
- [KS19a] Jitendra Kumar and Ashutosh Kumar Singh. Cloud resource demand prediction using differential evolution based learning. In

- 2019 7th International Conference on Smart Computing & Communications (ICSCC), pages 1–5. IEEE, 2019.
- [KS19b] Jitendra Kumar and Ashutosh Kumar Singh. An efficient machine learning approach for virtual machine resource demand prediction. *International Journal of Advanced Science and Technology*, 123:21–30, 2019.
- [KS21] Jitendra Kumar and Ashutosh Kumar Singh. Performance assessment of time series forecasting models for cloud datacenter networks’ workload prediction. *Wireless Personal Communications*, 116:1949–1969, 2021.
- [KS23] Shobhana Kashyap and Avtar Singh. Prediction-based scheduling techniques for cloud data center’s workload: a systematic review. *Cluster Computing*, 26(5):3209–3235, 2023.
- [KSB21] Jitendra Kumar, Ashutosh Kumar Singh, and Rajkumar Buyya. Self directed learning based workload forecasting model for cloud resource management. *Information Sciences*, 543:345–366, 2021.
- [KSK⁺24] Yeong-Min Kim, Seunghwan Song, Byoung-Mo Koo, Jeena Son, Yeseul Lee, and Jun-Geol Baek. Enhancing long-term cloud workload forecasting framework: Anomaly handling and ensemble learning in multivariate time series. *IEEE Transactions on Cloud Computing*, 2024.
- [KSPA21] Diamantis Koutsandreas, Evangelos Spiliotis, Fotios Petropoulos, and Vassilios Assimakopoulos. On the selection of forecasting accuracy measures. *Journal of the Operational Research Society*, pages 1–18, 2021.
- [KSZ20] Lingkai Kong, Jimeng Sun, and Chao Zhang. Sde-net: Equipping deep neural networks with uncertainty estimates. *arXiv preprint arXiv:2008.10546*, 2020.
- [KURD22] Davinder Kaur, Suleyman Uslu, Kaley J Rittichier, and Arjan Durresi. Trustworthy artificial intelligence: a review. *ACM Computing Surveys (CSUR)*, 55(2):1–38, 2022.

- [KW⁺19] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.
- [KW21] Martijn Koot and Fons Wijnhoven. Usage impact on data center electricity needs: A system dynamic forecasting model. *Applied Energy*, 291:116798, 2021.
- [KWQH20] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Forecasting cloud application workloads with cloudinsight for predictive resource management. *IEEE Transactions on Cloud Computing*, 10(3):1848–1863, 2020.
- [LALP21] Bryan Lim, Sercan Ö Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.
- [LCD04] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM computer communication review*, 34(4):219–230, 2004.
- [LFK⁺21] Habte Lejebo Leka, Zhang Fengli, Ayantu Tesfaye Kenea, Abebe Tamrat Tegene, Peter Atandoh, and Negalign Wake Hundera. 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)*, pages 474–478. IEEE, 2021.
- [LFTL20] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.
- [LFV⁺17] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.
- [Li19] Hailin Li. Multivariate time series clustering based on common principal component analysis. *Neurocomputing*, 349:239–247, 2019.

- [Lia05] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.
- [LJL⁺22] Chunhong Liu, Jie Jiao, Weili Li, Jingxiong Wang, and Junna Zhang. Tr-predictor: An ensemble transfer learning model for small-sample cloud workload prediction. *Entropy*, 24(12):1770, 2022.
- [LL12] Chunlin Li and La Yuan Li. Optimal resource provisioning for cloud computing environment. *The Journal of Supercomputing*, 62:989–1022, 2012.
- [LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [LLC16] Bingwei Liu, Yinan Lin, and Yu Chen. Quantitative workload analysis and prediction using google cluster traces. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 935–940. IEEE, 2016.
- [LLS⁺17] Chunhong Liu, Chuanchang Liu, Yanlei Shang, Shiping Chen, Bo Cheng, and Junliang Chen. An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications*, 80:35–44, 2017.
- [LMC20] Yan Li, Junming Ma, and Donggang Cao. Cross-domain workloads performance prediction via runtime metrics transferring. In *2020 IEEE International Conference on Joint Cloud Computing*, pages 38–42. IEEE, 2020.
- [LSM⁺23] ChangHoon Lee, MinJae Song, KyungHa Min, EunGyeom Ha, JunHa Lee, and Wooju Kim. Optimization of cloud computing workload prediction model with domain-based feature selection method. In *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, pages 868–871. IEEE, 2023.
- [M⁺67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

- [MAH18] Haithem Mezni, Sabeur Aridhi, and Allel Hadjali. The uncertain cloud: State of the art and research challenges. *International Journal of Approximate Reasoning*, 103:139–151, 2018.
- [MBK⁺23] Priyanka Mary Mammen, Noman Bashir, Ramachandra Rao Kolluri, Eun Kung Lee, and Prashant Shenoy. Cuff: A configurable uncertainty-driven forecasting framework for green ai clusters. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*, pages 266–270, 2023.
- [MDB⁺18] Karl Mason, Martin Duggan, Enda Barrett, Jim Duggan, and Enda Howley. Predicting host cpu utilization in the cloud using evolutionary neural networks. *Future Generation Computer Systems*, 86:162–173, 2018.
- [Men02] Daniel A Menascé. Tpc-w: A benchmark for e-commerce. *IEEE Internet Computing*, 6(3):83–87, 2002.
- [MF14] Dorian Minarolli and Bernd Freisleben. Cross-correlation prediction of resource demand for virtual machine resource allocation in clouds. In *2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks*, pages 119–124. IEEE, 2014.
- [MHS15] Sune Dueholm Müller, Stefan Rubæk Holm, and Jens Søndergaard. Benefits of cloud computing: literature review in a maturity model perspective. *Communications of the Association for Information Systems*, 37(1):42, 2015.
- [Mic22] Umberto Michelucci. An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*, 2022.
- [Mit97] Tom M Mitchell. Machine learning, 1997.
- [MK20] Mohammad Masdari and Afsane Khoshnevis. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–2424, 2020.
- [MLH10] 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.

- [MM22] Markets and Markets. Cloud computing market by service model, by deployment model, organization size, vertical and region - global forecast to 2027, 2022.
- [MMF17] 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*, 6:1–18, 2017.
- [MNH⁺15] Saad Mustafa, Babar Nazir, Amir Hayat, Sajjad A Madani, et al. Resource management in cloud computing: Taxonomy, prospects, and challenges. *Computers & Electrical Engineering*, 47:186–203, 2015.
- [MPV21] José Mena, Oriol Pujol, and Jordi Vitria. A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective. *ACM Computing Surveys (CSUR)*, 54(9):1–35, 2021.
- [MRCV23] Kate Murray, Andrea Rossi, Diego Carraro, and Andrea Visentin. On forecasting cryptocurrency prices: A comparison of machine learning, deep learning, and ensembles. *Forecasting*, 5(1):196–209, 2023.
- [MRKV21] 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.
- [MRR⁺53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.
- [MTSA22] Sania Malik, Muhammad Tahir, Muhammad Sardaraz, and Abdullah Alourani. A resource utilization prediction model for cloud data centers using evolutionary algorithms and machine learning techniques. *Applied Sciences*, 12(4):2160, 2022.
- [NME12] Hoang H Nguyen, Ben Matthews, and Irfan Elahi. Deep domain adaptation for runtime prediction in dynamic workload sched-

- uler. In *neural information processing systems*, pages 1097–1105, 2012.
- [NNSK22] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.
- [OCCB19] Boris N Oreshkin, Dmitri Carпов, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.
- [OCCB21] Boris N Oreshkin, Dmitri Carпов, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9242–9250, 2021.
- [OHU21] Soukaina Ouhamme, Youssef Hadi, and Arif Ullah. An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model. *Neural Computing and Applications*, 33:10043–10055, 2021.
- [PAL⁺22] Farhad Pourpanah, Moloud Abdar, Yuxuan Luo, Xinlei Zhou, Ran Wang, Chee Peng Lim, Xi-Zhao Wang, and QM Jonathan Wu. A review of generalized zero-shot learning methods. *IEEE transactions on pattern analysis and machine intelligence*, 2022.
- [PAM23] Jaime Pérez, Patricia Arroba, and José M Moya. Data augmentation through multivariate scenario forecasting in data centers using generative adversarial networks. *Applied Intelligence*, 53(2):1469–1486, 2023.
- [Par14] Van L Parsons. Stratified sampling. *Wiley StatsRef: Statistics Reference Online*, pages 1–11, 2014.
- [PB23] Yashwant Singh Patel and Jatin Bedi. Mag-d: A multivariate attention network based approach for cloud workload forecasting. *Future Generation Computer Systems*, 142:376–392, 2023.
- [Pea01] 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.

- [PFW⁺14] 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.
- [PG15] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1855–1870, 2015.
- [PHB18] Fotios Petropoulos, Rob J Hyndman, and Christoph Bergmeir. Exploring the sources of uncertainty: Why does bagging for time series forecasting work? *European Journal of Operational Research*, 268(2):545–554, 2018.
- [PK22] Eva Patel and Dharmender Singh Kushwaha. A hybrid cnn-lstm model for predicting server load in cloud computing. *The Journal of Supercomputing*, 78(8):1–30, 2022.
- [PKG11] 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.
- [PLEF20] John Paparrizos, Chunwei Liu, Aaron J Elmore, and Michael J Franklin. Debunking four long-standing misconceptions of time-series distance measures. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pages 1887–1905, 2020.
- [PNKJ11] John J Prevost, KranthiManoj Nagothu, Brian Kelley, and Mo Jamshidi. Prediction of cloud data center networks loads using stochastic and neural models. In *2011 6th International Conference on System of Systems Engineering*, pages 276–281. IEEE, 2011.
- [PP⁺20] Neena Pandey, Abhipsa Pal, et al. Impact of digital surge during covid-19 pandemic: A viewpoint on research and practice. *International journal of information management*, 55:102171, 2020.
- [PR21] R Pushpalatha and B Ramesh. Amalgamation of neural network and genetic algorithm for efficient workload prediction in data

- center. In *Advances in VLSI, Signal Processing, Power Electronics, IoT, Communication and Embedded Systems: Select Proceedings of VSPICE 2020*, pages 69–84. Springer, 2021.
- [PS18] 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.
- [PSA21] 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.
- [QYLC21] Shengpan Qian, Yang Yu, Ling Li, and Yanbo Chang. An attention-based gru encoder decoder for hostload prediction in a data center. In *2021 International Conference on Computer Communication and Artificial Intelligence (CCAI)*, pages 121–125. IEEE, 2021.
- [QZG16] Feng Qiu, Bin Zhang, and Jun Guo. A deep learning approach for vm workload prediction in the cloud. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 319–324. IEEE, 2016.
- [RBL⁺22] Li Ruan, Yu Bai, Shaoning Li, Jiaxun Lv, Tianyuan Zhang, Limin Xiao, Haiguang Fang, Chunhao Wang, and Yunzhi Xue. Cloud workload turning points prediction via cloud feature-enhanced deep learning. *IEEE Transactions on Cloud Computing*, 2022.
- [RGP08] Pedro Pereira Rodrigues, Joao Gama, and Joao Pedroso. Hierarchical clustering of time-series data streams. *IEEE transactions on knowledge and data engineering*, 20(5):615–627, 2008.
- [RKKK22] Ridhima Rani, Meenu Khurana, Ajay Kumar, and Neeraj Kumar. Big data dimensionality reduction techniques in iot: Review,

- applications and open research challenges. *Cluster Computing*, 25(6):4027–4049, 2022.
- [RLJ⁺21] Valentin Resseguier, Long Li, Gabriel Jouan, Pierre Dérian, Etienne Mémin, and Bertrand Chapron. New trends in ensemble forecast strategy: uncertainty quantification for coarse-grid computational fluid dynamics. *Archives of Computational Methods in Engineering*, 28:215–261, 2021.
- [RLM⁺20] Ana Radovanović, Junshi Li, Jovica V Milanović, Nina Milosavljević, and Riccardo Storchi. Application of agglomerative hierarchical clustering for clustering of time series data. In *2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, pages 640–644. IEEE, 2020.
- [Rou87] 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.
- [RPF⁺21] 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.
- [RT84] Trond Riise and Dag Tjozstheim. Theory and practice of multivariate arma forecasting. *Journal of Forecasting*, 3(3):309–317, 1984.
- [RTBSD13] Bahman Rostami-Tabar, M Zied Babai, Aris Syntetos, and Yves Ducq. Demand forecasting by temporal aggregation. *Naval Research Logistics (NRL)*, 60(6):479–498, 2013.
- [RVC⁺23] Andrea Rossi, Andrea Visentin, Diego Carraro, Steven Prestwich, and Kenneth N. Brown. Forecasting workload in cloud computing: Towards uncertainty-aware predictions and transfer learning, 2023.
- [RVPB22] 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.

- [RVPB23] Andrea Rossi, Andrea Visentin, Steven Prestwich, and Kenneth N Brown. Clustering-based numerosity reduction for cloud workload forecasting. In *International Symposium on Algorithmic Aspects of Cloud Computing*, pages 115–132. Springer, 2023.
- [RWH11] Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, 1:1–14, 2011.
- [SBSC⁺19] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32, 2019.
- [SBY⁺17] Galit Shmueli, Peter C Bruce, Inbal Yahav, Nitin R Patel, and Kenneth C Lichtendahl Jr. *Data mining for business analytics: concepts, techniques, and applications in R*. John Wiley & Sons, 2017.
- [SE11] 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.
- [Set] Neha Seth. Beginners Guide to Anomaly Detection Using Self-Organizing Maps — analyticsvidhya.com. <https://www.analyticsvidhya.com/blog/2021/09/beginners-guide-to-anomaly-detection-using-self-organizing-maps/>.
- [SFG⁺20] Mohammad Shahradd, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pages 205–218, 2020.
- [SFGJ20] 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.
- [SFSHTM20] Roohollah Shirani Faradonbeh, Sina Shaffiee Haghshenas, Abbas Taheri, and Reza Mikaeil. Application of self-organizing map and

- fuzzy c-mean techniques for rockburst clustering in deep underground projects. *Neural Computing and Applications*, 32:8545–8559, 2020.
- [SH20] Hengheng Shen and Xuehai Hong. Host load prediction with bi-directional long short-term memory in cloud computing. *arXiv preprint arXiv:2007.15582*, 2020.
- [SK21a] SR Shishira and A Kandasamy. Beem-nn: An efficient workload optimization using bee mutation neural network in federated cloud environment. *Journal of Ambient Intelligence and Humanized Computing*, 12:3151–3167, 2021.
- [SK21b] SR Shishira and A Kandasamy. A novel feature extraction model for large-scale workload prediction in cloud environment. *SN Computer Science*, 2(5):354, 2021.
- [SKPS20] Anshul Sharma, Abhinav Kumar, Anil Kumar Pandey, and Rishav Singh. Time series data representation and dimensionality reduction techniques. *Applications of Machine Learning*, pages 267–284, 2020.
- [SKW15] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International conference on machine learning*, pages 1218–1226. PMLR, 2015.
- [SMAvdS21] Kamile Stankeviciute, Ahmed M Alaa, and Mihaela van der Schaar. Conformal time-series forecasting. *Advances in neural information processing systems*, 34:6216–6228, 2021.
- [SS20] Ali Sunyaev and Ali Sunyaev. Cloud computing. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, pages 195–236, 2020.
- [SS21] Deepika Saxena and Ashutosh Kumar Singh. Workload forecasting and resource management models based on machine learning for cloud computing environments. *arXiv preprint arXiv:2106.15112*, 2021.
- [SSMB24] Gurjot Singh, Prajit Sengupta, Anant Mehta, and Jatin Bedi. A feature extraction and time warping based neural expansion ar-

- chitecture for cloud resource usage forecasting. *Cluster Computing*, pages 1–20, 2024.
- [SVBI15] Siqi Shen, Vincent Van Beek, and Alexandru Iosup. Statistical characterization of business-critical workloads hosted in cloud datacenters. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 465–474. IEEE, 2015.
- [SYZ⁺18] 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*, 74:6554–6568, 2018.
- [SZL⁺21] Wenjuan Shu, Fanping Zeng, Zhen Ling, Junyi Liu, Tingting Lu, and Guozhu Chen. Resource demand prediction of cloud workloads using an attention-based gru model. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pages 428–437. IEEE, 2021.
- [TBD⁺20] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *Proceedings of the fifteenth European conference on computer systems*, pages 1–14, 2020.
- [THHSS14] Jun-jie Tong, E Hai-Hong, Mei-na Song, and Jun-de Song. Host load prediction in cloud based on classification methods. *The Journal of China Universities of Posts and Telecommunications*, 21(4):40–46, 2014.
- [Tho53] Robert L Thorndike. Who belongs in the family. In *Psychometrika*. Citeseer, 1953.
- [TL18] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.
- [TM21] Binh Tang and David S Matteson. Probabilistic transformer for time series analysis. *Advances in Neural Information Processing Systems*, 34:23592–23608, 2021.
- [UBY23] Farman Ullah, Muhammad Bilal, and Su-Kyung Yoon. Intelligent time-series forecasting framework for non-linear dynamic workload and resource prediction in cloud. *Computer Networks*, 225:109653, 2023.

- [UPVS09] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 53(11):1830–1845, 2009.
- [Vaz15] Carlos Vazquez. *Time series forecasting of cloud data center workloads for dynamic resource provisioning*. The University of Texas at San Antonio, 2015.
- [VF05] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pages 758–770. Springer, 2005.
- [VKJ15] C. Vazquez, R. Krishnan, and E. John. Time series forecasting of cloud data center workloads for dynamic resource provisioning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 6:87–110, 2015.
- [VKSR21] K Valarmathi and S Kanaga Suba Raja. Resource utilization prediction technique in cloud using knowledge based ensemble random forest with lstm model. *Concurrent Engineering*, 29(4):396–404, 2021.
- [VMPS18] 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.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [WBB⁺22] Can Wang, Mitra Baratchi, Thomas Bäck, Holger H Hoos, Steffen Limmer, and Markus Olhofer. Towards time-series feature engineering in automated machine learning for multi-step-ahead forecasting. *Engineering Proceedings*, 18(1):17, 2022.
- [WBM⁺20] Peng Wang, Nidhal C Bouaynaya, Lyudmila Mihaylova, Jikai Wang, Qibin Zhang, and Renke He. Bayesian neural networks uncertainty quantification with cubature rules. In *2020 Interna-*

- tional Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.
- [Węg18] Stanisław Węglarczyk. Kernel density estimation and its application. In *ITM web of conferences*, volume 23, page 00037. EDP Sciences, 2018.
- [WFC21] Thomas Wang, Simone Ferlin, and Marco Chiesa. Predicting cpu usage for proactive autoscaling. In *Proceedings of the 1st Workshop on Machine Learning and Systems*, pages 31–38, 2021.
- [WFFWM18] 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.
- [WHL⁺22] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.
- [Wil20] 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>.
- [WKW16] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.
- [WLL⁺22] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.
- [WSS23] Bingchao Wang, Xiaoyu Shi, and Mingsheng Shang. A self-decoupled interpretable prediction framework for highly-variable cloud workloads. In *International Conference on Database Systems for Advanced Applications*, pages 588–603. Springer, 2023.
- [WTBR22] 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.

- [WXY⁺22] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. 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)*, pages 945–960, 2022.
- [XKS⁺06] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040, 2006.
- [XLSA18] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.
- [XSW⁺22] Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. esdnn: deep neural network based multivariate workload prediction in cloud computing environments. *ACM Transactions on Internet Technology (TOIT)*, 22(3):1–24, 2022.
- [XX23] Chen Xu and Yao Xie. Sequential predictive conformal inference for time series. In *International Conference on Machine Learning*, pages 38707–38727. PMLR, 2023.
- [XYB⁺15] Ji Xue, Feng Yan, Robert Birke, Lydia Y Chen, Thomas Scherer, and Evgenia Smirni. Practise: Robust prediction of data center time series. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 126–134. IEEE, 2015.
- [XYLX21] Huidan Xi, Cheng Yan, Huixi Li, and Yinhao Xiao. An attention-based recurrent neural network for resource usage prediction in cloud data center. In *Journal of Physics: Conference Series*, volume 2006, page 012007. IOP Publishing, 2021.
- [YD18] Rui Ye and Qun Dai. A novel transfer learning framework for time series forecasting. *Knowledge-Based Systems*, 156:74–99, 2018.
- [YGL⁺22] Jixiang Yu, Ming Gao, Yuchan Li, Zehui Zhang, Wai Hung Ip, and Kai Leung Yung. Workflow performance prediction based on

- graph structure aware deep attention neural network. *Journal of Industrial Information Integration*, 27:100337, 2022.
- [YJYB16] 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.
- [Yoa13] Daniel W Yoas. *Using forecasting to predict long-term resource utilization for web services*. Nova Southeastern University, 2013.
- [YPZ⁺14] Qiangpeng Yang, Chenglei Peng, He Zhao, Yao Yu, Yu Zhou, Ziqiang Wang, and Sidan Du. A new method based on psr and ea-gmdh for host load prediction in cloud computing system. *The Journal of Supercomputing*, 68:1402–1417, 2014.
- [YS21] Peyman Yazdanian and Saeed Sharifian. E2lg: a multiscale ensemble of lstm/gan deep learning architecture for multistep-ahead cloud workload prediction. *The Journal of Supercomputing*, 77:11052–11082, 2021.
- [YZY⁺15] Qiangpeng Yang, Yu Zhou, Yao Yu, Jie Yuan, Xianglei Xing, and Sidan Du. Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing. *The Journal of Supercomputing*, 71:3037–3053, 2015.
- [ZDY⁺17] Weishan Zhang, Pengcheng Duan, Laurence T Yang, Feng Xia, Zhongwei Li, Qinghua Lu, Wenjuan Gong, and Su Yang. Resource requests prediction in the cloud computing environment with a deep belief network. *Software: Practice and Experience*, 47(3):473–488, 2017.
- [ZLP⁺22] Xinlei Zhou, Han Liu, Farhad Pourpanah, Tiejong Zeng, and Xizhao Wang. A survey on epistemic (model) uncertainty in supervised learning: Recent advances and applications. *Neurocomputing*, 489:449–465, 2022.
- [ZLW⁺22] Yongde Zhang, Fagui Liu, Bin Wang, Weiwei Lin, Guoxiang Zhong, Minxian Xu, and Keqin Li. A multi-output prediction model for physical machine resource usage in cloud data centers. *Future Generation Computer Systems*, 130:292–306, 2022.

- [ZLZ⁺16] Weishan Zhang, Bo Li, Dehai Zhao, Faming Gong, and Qinghua Lu. Workload prediction for cloud cluster using a recurrent neural network. In *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, pages 104–109. IEEE, 2016.
- [ZLZ⁺20] Shaojing Zhou, Jinguo Li, Kai Zhang, Mi Wen, and Qijie Guan. An accurate ensemble forecasting approach for highly dynamic cloud workload with vmd and r-transformer. *IEEE Access*, 8:115992–116003, 2020.
- [ZSQ⁺23] Yuyi Zhang, Qiushi Sun, Dongfang Qi, Jing Liu, Ruimin Ma, and Ovanes Petrosian. Shaptime: A general xai approach for explainable time series forecasting. In *Intelligent Systems Conference*, pages 659–673. Springer, 2023.
- [ZUHK⁺17] Qazi Zia Ullah, Shahzad Hassan, Gul Muhammad Khan, et al. Adaptive resource utilization prediction system for infrastructure as a service cloud. *Computational intelligence and neuroscience*, 2017, 2017.
- [ZWYZ21] 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.
- [ZYW20] Juan Zhang, Junping Yin, and Ruili Wang. Basic framework and main methods of uncertainty quantification. *Mathematical Problems in Engineering*, 2020:1–18, 2020.
- [ZYY⁺18] Qingchen Zhang, Laurence T Yang, Zheng Yan, Zhikui Chen, and Peng Li. An efficient deep learning model to predict cloud workload for industry informatics. *IEEE transactions on industrial informatics*, 14(7):3170–3178, 2018.
- [ZZCG19] Yonghua Zhu, Weilin Zhang, Yihai Chen, and Honghao Gao. A novel approach to workload prediction using attention-based lstm encoder-decoder network in cloud environment. *EURASIP Journal on Wireless Communications and Networking*, 2019:1–18, 2019.

- [ZZL⁺21] Liang Zhang, Wenli Zheng, Chao Li, Yao Shen, and Minyi Guo. Autrascale: An automated and transfer learning solution for streaming system auto-scaling. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 912–921. IEEE, 2021.
- [ZZP⁺21] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.
- [ZZSG18] Wei Zhong, Yi Zhuang, Jian Sun, and Jingjing Gu. A load prediction model for cloud computing using pso-based weighted wavelet support vector machine. *Applied Intelligence*, 48:4072–4083, 2018.
- [ZZT⁺22] Xiaohu Zheng, Jun Zhang, Guijian Tang, Tingsong Jiang, Wei Peng, and Wen Yao. A reliability analysis method based on quantile regression and feedforward neural network. In *12th International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE 2022)*, volume 2022, pages 294–300. IET, 2022.

Appendix A

Preprocessed Datasets

A.1 Cluster-level Datasets from Google Cloud Trace 2019

In this section, we include the plots of all 8 preprocessed cluster-level datasets from GC19.

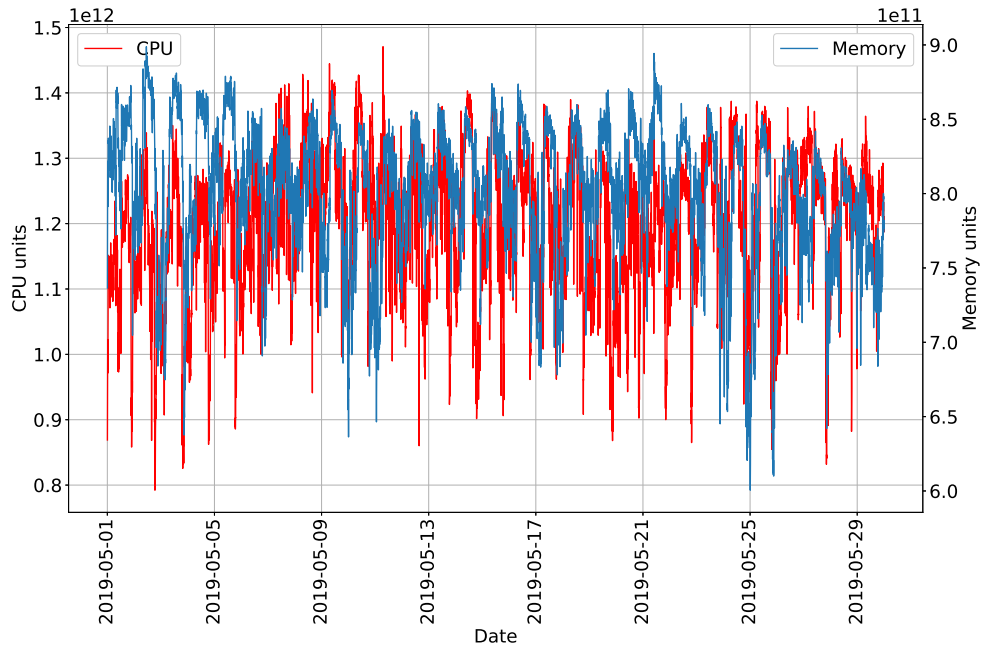


Figure A.1: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell A.

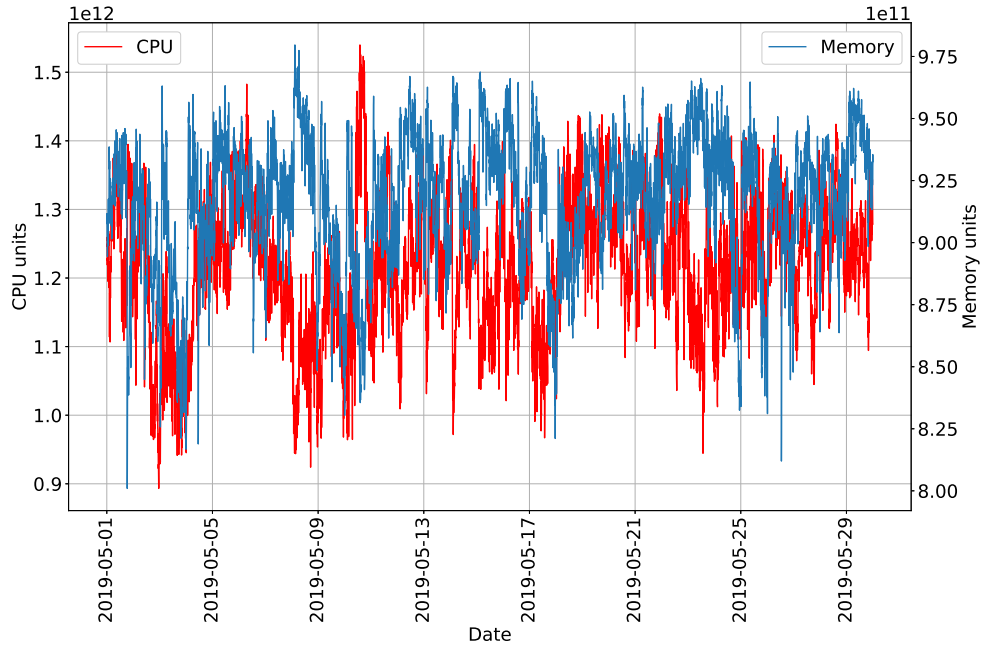


Figure A.2: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell B.

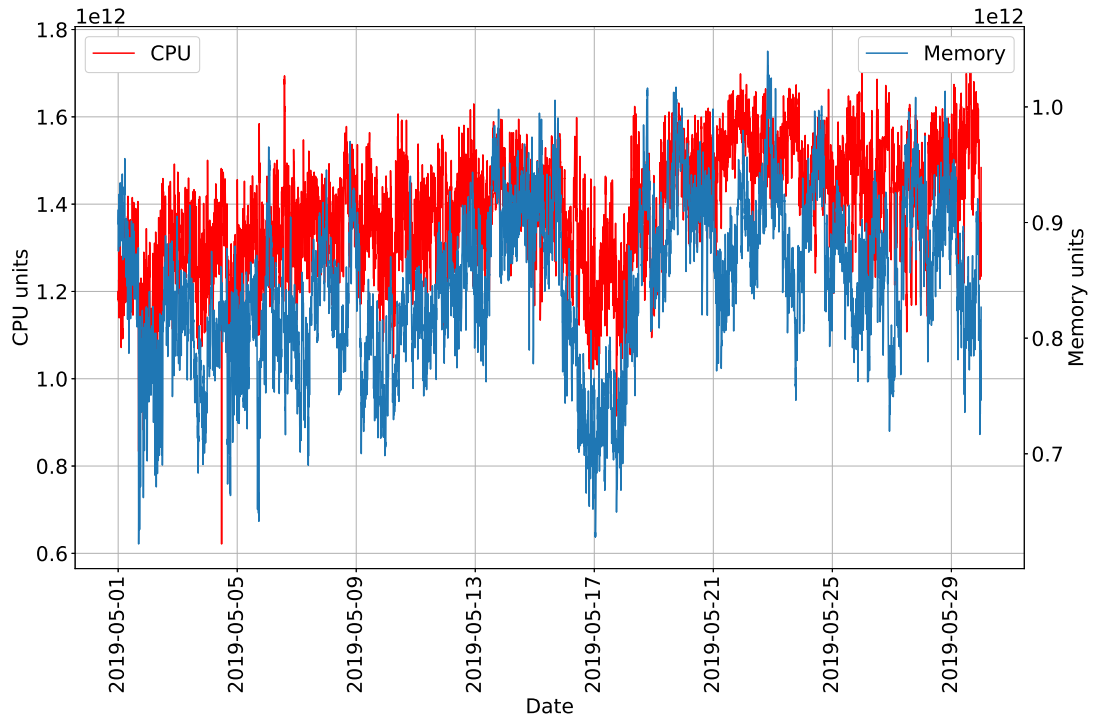


Figure A.3: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell C.

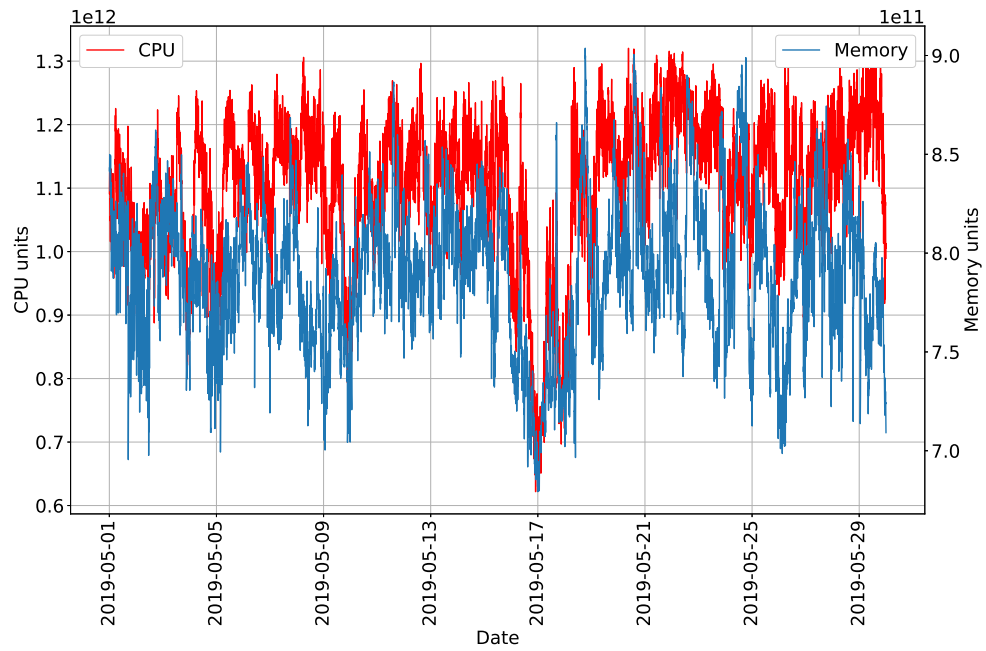


Figure A.4: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell D.

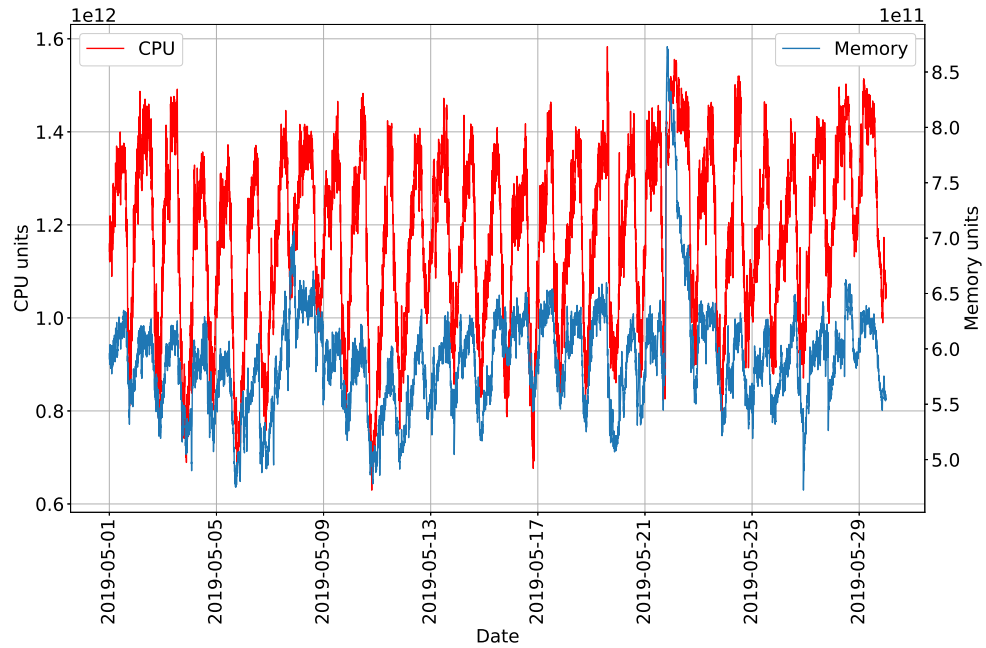


Figure A.5: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell E.

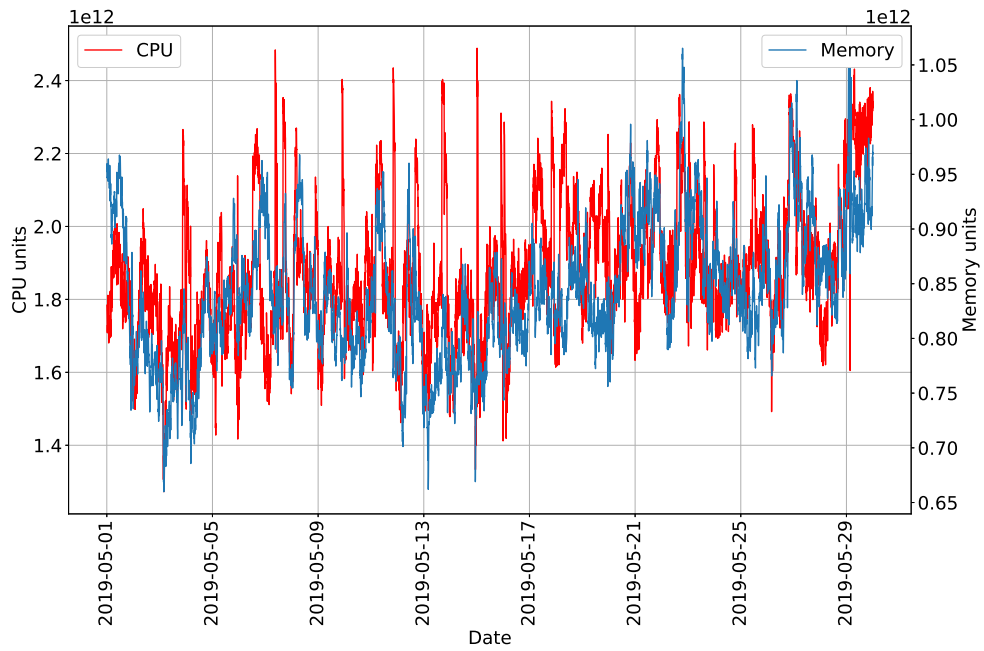


Figure A.6: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell F.

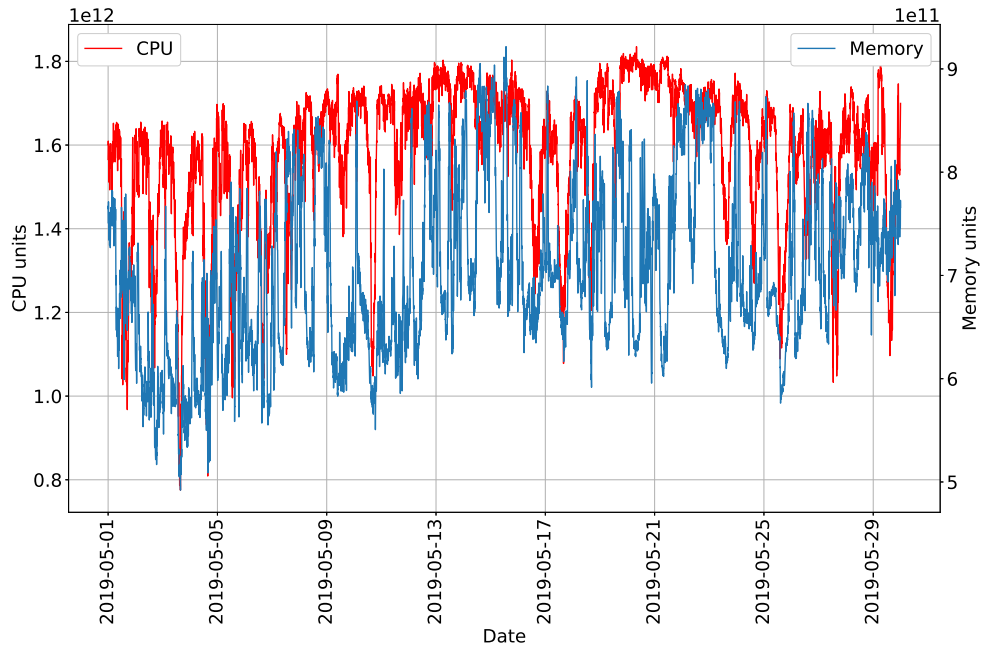


Figure A.7: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell G.

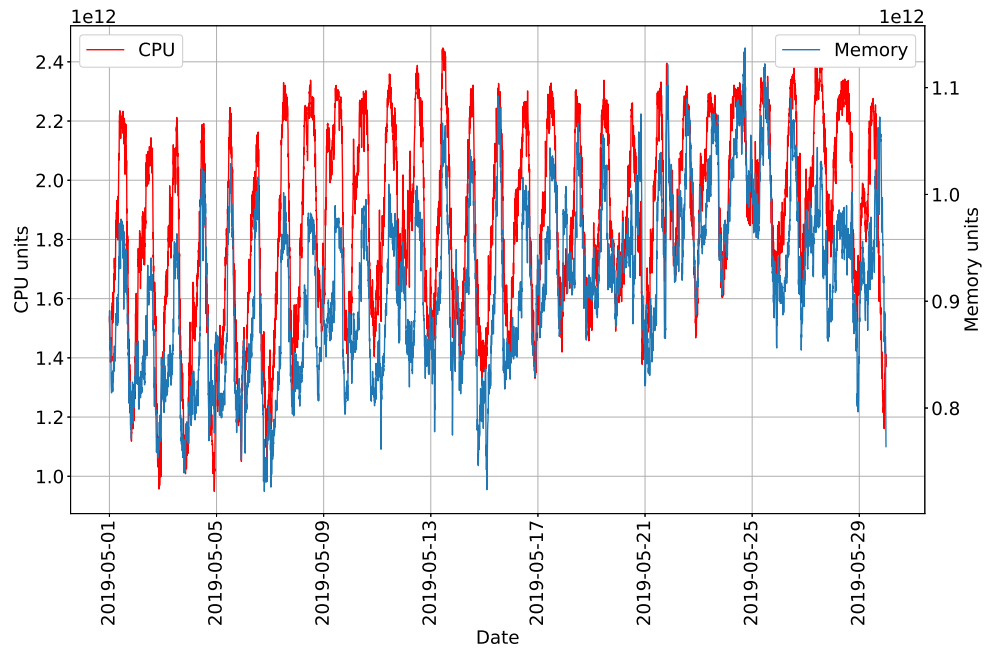


Figure A.8: Average CPU and memory (RAM) workload of the preprocessed Google Cloud Trace 2019 for the cluster cell H.

A.2 Memory distribution of cluster-level preprocessed datasets

In this section, we include the plots of the memory unit distribution of preprocessed cluster-level datasets.

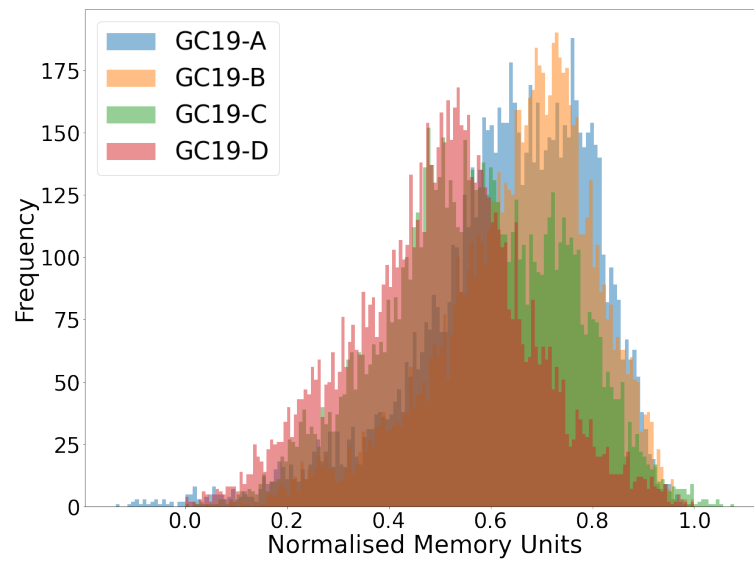


Figure A.9: Histogram of the normalised memory units of the compute cells A to D from the preprocessed GC19.

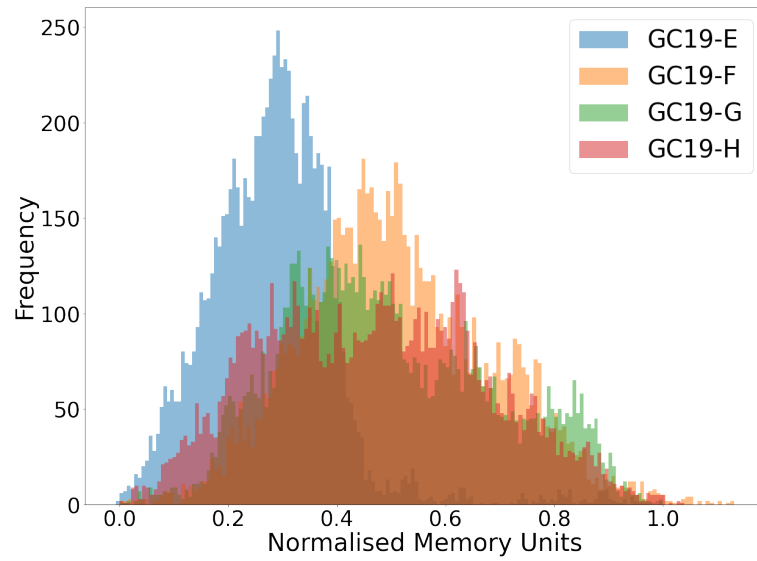


Figure A.10: Histogram of the normalised memory units of the compute cells E to H from the preprocessed GC19.

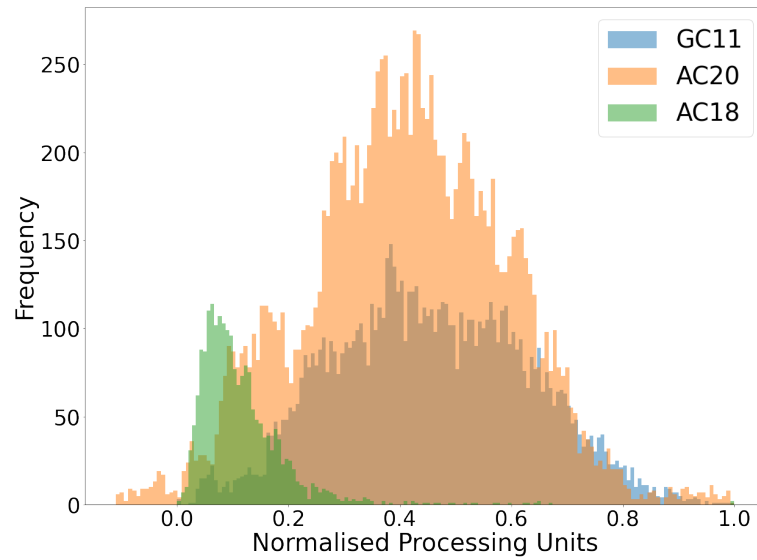


Figure A.11: Histogram of the normalised memory units of the compute cells from the preprocessed GC11, AC20 and AC18.

Appendix B

Bivariate HBNN and Transfer Learning Results

B.1 Additional Results on Bivariate HBNN

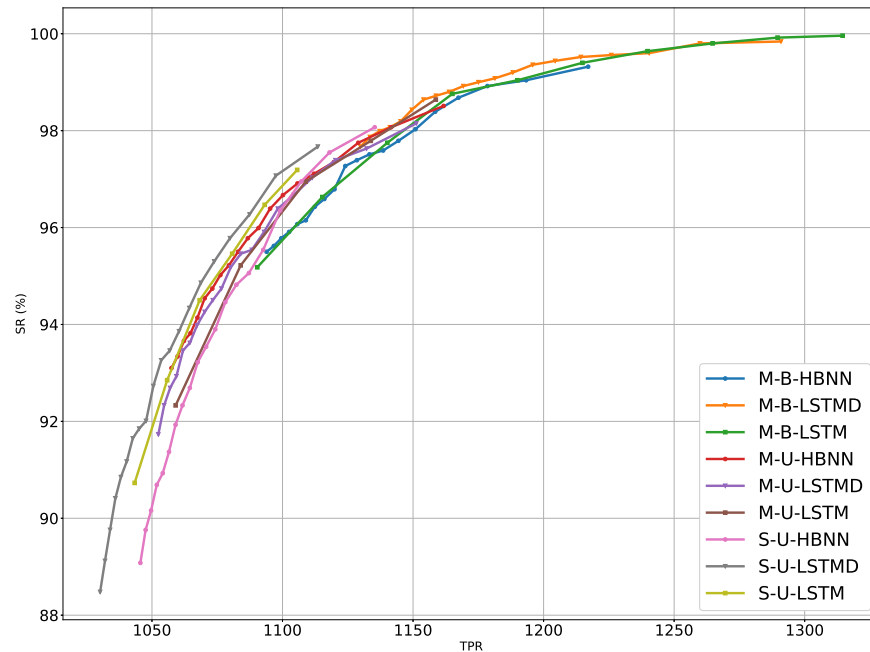


Figure B.1: Total predicted memory units for S-U, M-U and M-B models. The closer the curve to the top left corner, the better the model.

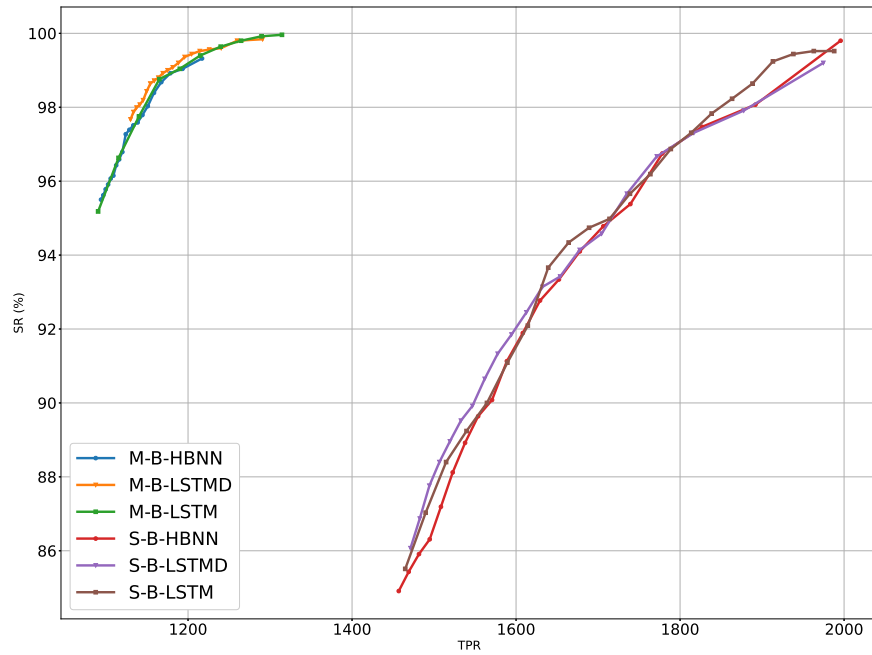


Figure B.2: Total predicted memory units for bivariate models. The ones trained with a single dataset perform very poorly, with a much higher amount of TPR w.r.t. than the models trained with multiple datasets.

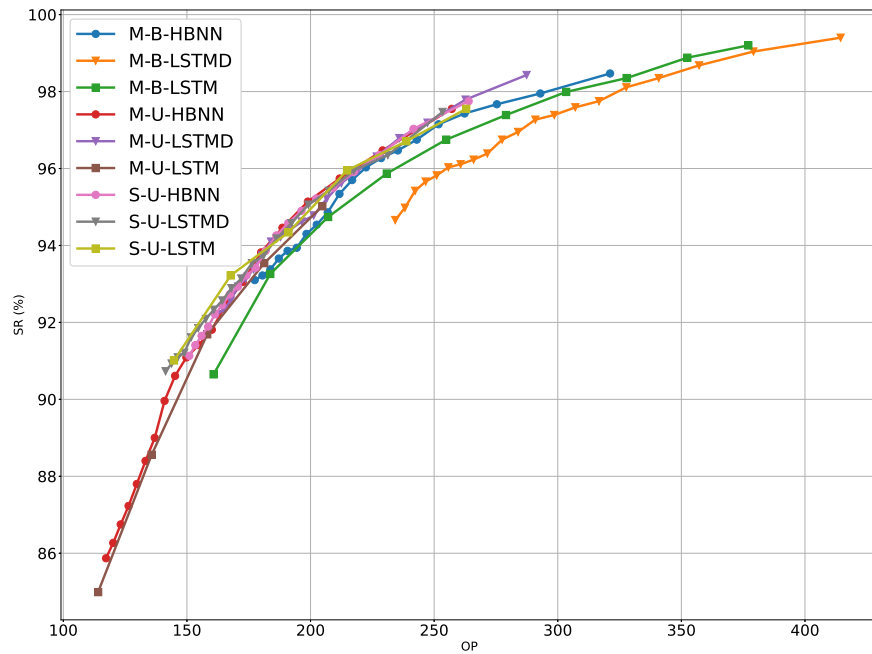


Figure B.3: Total overprediction for processing units for S-U, M-U and M-B models. The closer the curve to the top left corner, the better the model.

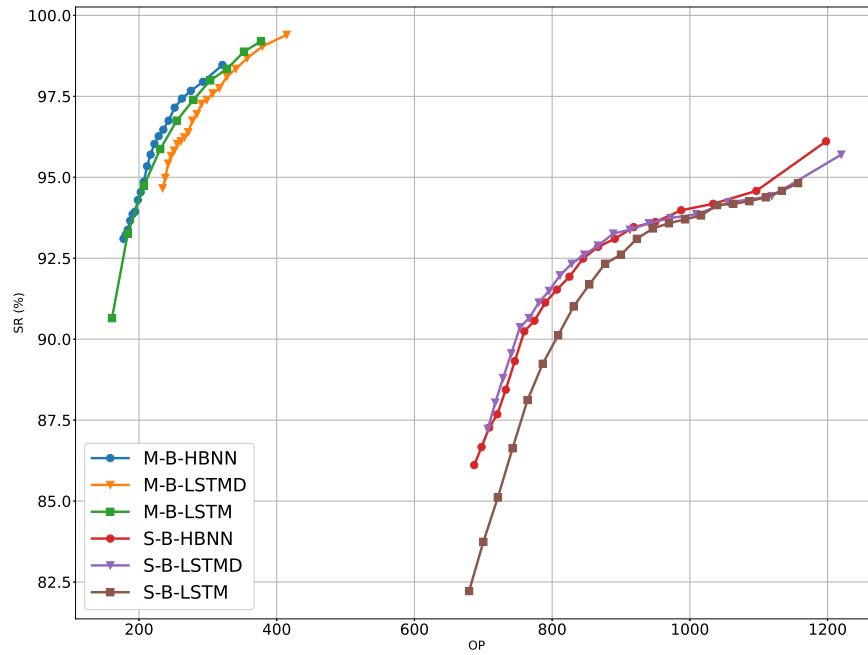


Figure B.4: Total overprediction for processing units for bivariate models. The ones trained with a single dataset perform very poorly, with a much higher amount of TPR w.r.t. than the models trained with multiple datasets.

B.2 Additional Results on Transfer Learning for HBNN

To complete the analysis, in this section, we evaluate scenarios *GC19*, *GC19 FT*, *GC19-but-one* and *GC19-but-one FT* by including all the remaining results regarding the point estimate accuracy and confidence interval evaluation.

Confidence Interval Evaluation

Table B.1: Processing Units and Memory confidence interval statistics of the models for the transfer learning experiments on Google Cloud 2019 traces and tested on Google Cloud 2011 and Alibaba. In bold, the model with the best SR for each confidence level.

Confidence Level	Model	Processing Units				Memory			
		SR (%)	OP (%)	UP (%)	TPR (%)	SR (%)	OP (%)	UP (%)	TPR (%)
95%	GC19	93.18	167.03	3.43	263.6	76.31	52.72	5.6	147.12
	GC19 FT	76.31	54.45	5.89	148.55	95.58	119.98	1.22	218.76
97%	GC19	93.42	167.86	3.19	264.68	76.8	53.43	5.29	148.14
	GC19 FT	84.75	65.69	3.43	162.26	96.19	131.88	0.84	231.04
99%	GC19	94.18	169.47	2.75	266.71	78.08	54.8	4.74	150.06
	GC19 FT	93.86	89.51	1.36	188.14	97.55	154.51	0.29	254.22

Confidence Interval Accuracy

Table B.2: Average MSE and MAE for scenarios *All*, *All FT*, *All-but-one* and *All-but-one FT*. The MSE and MAE are computed between the SR achieved by the models and the confidence level. In bold, the model overall.

Scenario	Processing units		Memory	
	MSE	MAE	MSE	MAE
All	1.22	0.93	2.76	1.27
All FT	48.83	6.67	99.44	8.88
All-but-one	1.97	1.11	8.66	2.39
All-but-one FT	1.19	0.94	2.40	1.27

Table B.3: Average MSE and MAE comparison for scenario *GC19*. The MSE and MAE are computed between the SR achieved by the models and the confidence level. In bold, the model overall.

Scenario	Processing units		Memory	
	MSE	MAE	MSE	MAE
GC19	10.21	2.92	6.44	2.07
GC19 FT	327.91	16.54	33.72	5.32
GC19-but-one	656.59	23.10	507.39	20.03
GC19-but-one FT	1628.71	39.36	870.12	28.76

Table B.4: Average MSE and MAE comparison for resource prediction accuracy for the transfer learning experiments on Google Cloud 2019 traces and tested on Google Cloud 2011 and Alibaba. The MSE and MAE computed between the SR achieved by the models and the confidence level. In bold, the model overall.

Model	Processing Units		Memory	
	MSE	MAE	MSE	MAE
GC19	7.81	2.30	319.97	17.34
GC19 FT	319.91	16.19	4.54	1.69

Plots

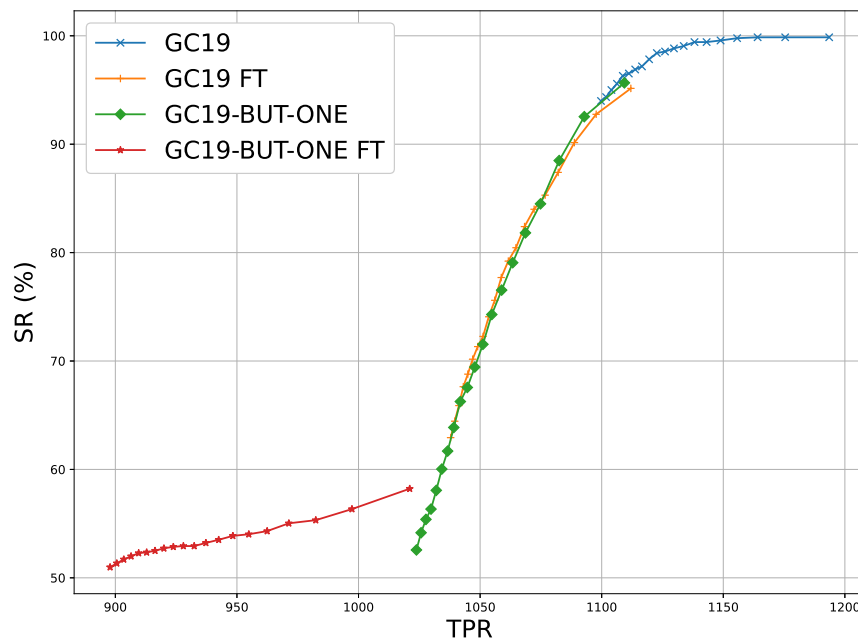


Figure B.5: Total predicted processing units for *GC19*, *GC19 FT*, *GC19-but-one* and *GC19-but-one FT*. The closer the curve to the top left corner, the better the model. *GC19-but-one FT* performs very poorly.

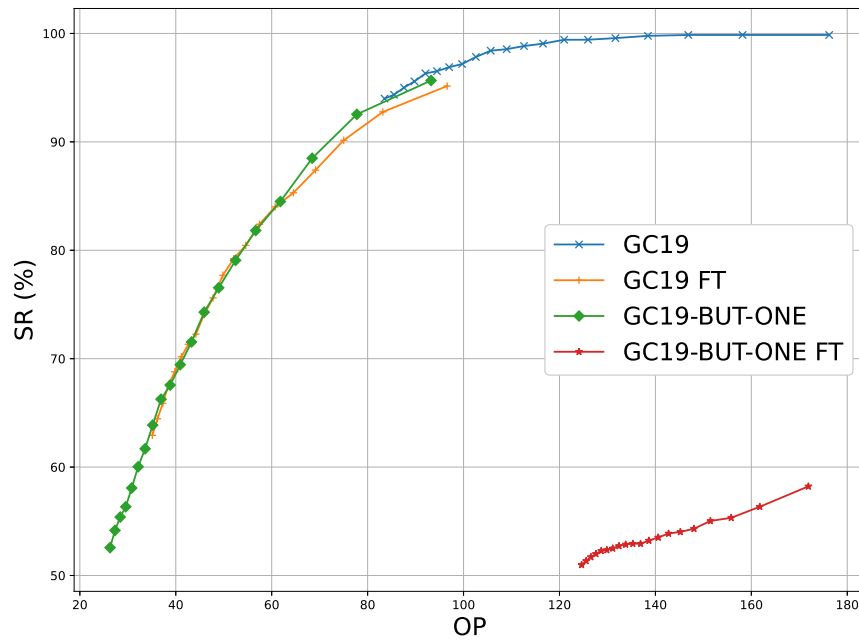


Figure B.6: Total overprediction of processing units for *GC19*, *GC19 FT*, *GC19-but-one* and *GC19-but-one FT*. The closer the curve to the top left corner, the better the model. *GC19-but-one FT* performs very poorly.

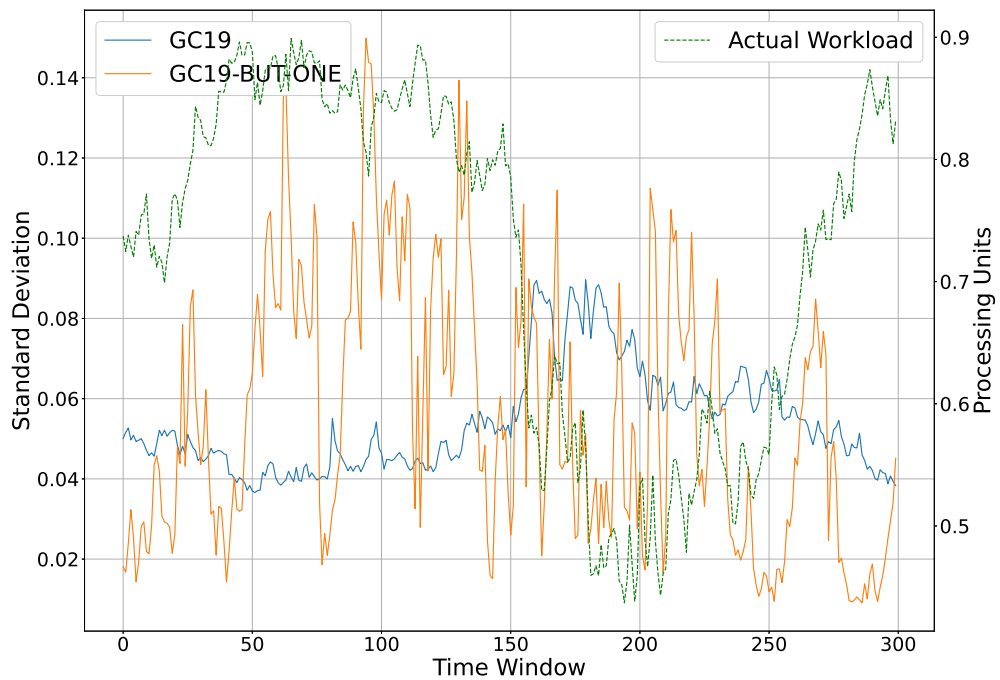


Figure B.7: Standard deviation variability for across inference on the test set *GC19* and *GC19-but-one*. Uncertainty for *GC19-but-one* is higher for predictions on unseen data.

Appendix C

Clustering-based Numerosity Reduction Results

In this section, we present the remaining results of Chapter 6 on the use of clustering to reduce the training size.

C.1 Point Estimate Accuracy

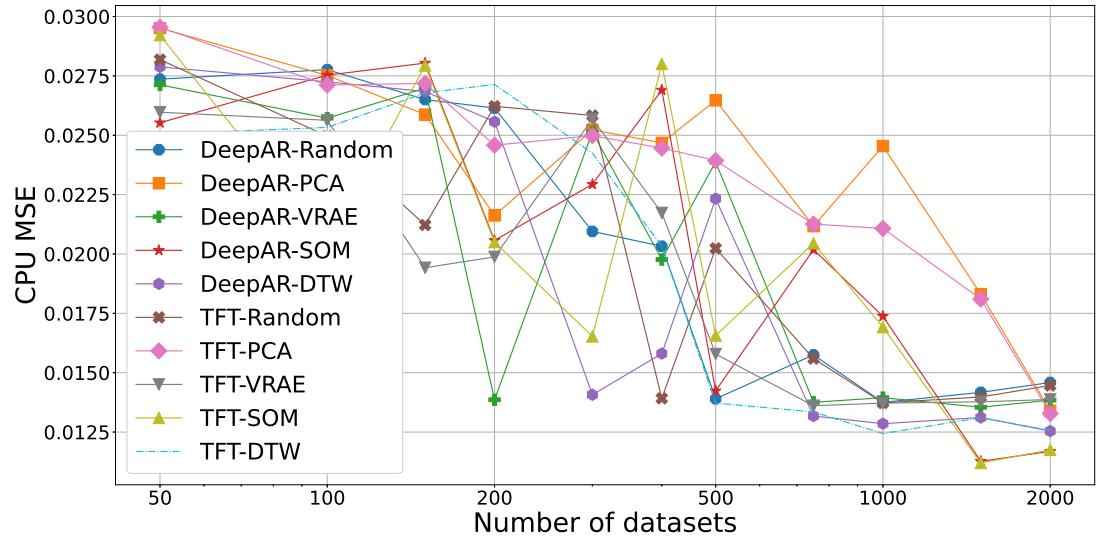


Figure C.1: MSE comparison of DeepAR and TFT for the processing units demand. DTW and SOM are the best clustering-based selection approaches and allow the models to achieve the same MSE with fewer training datasets compared to a random selection. When the number of datasets is high, the MSE of all the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

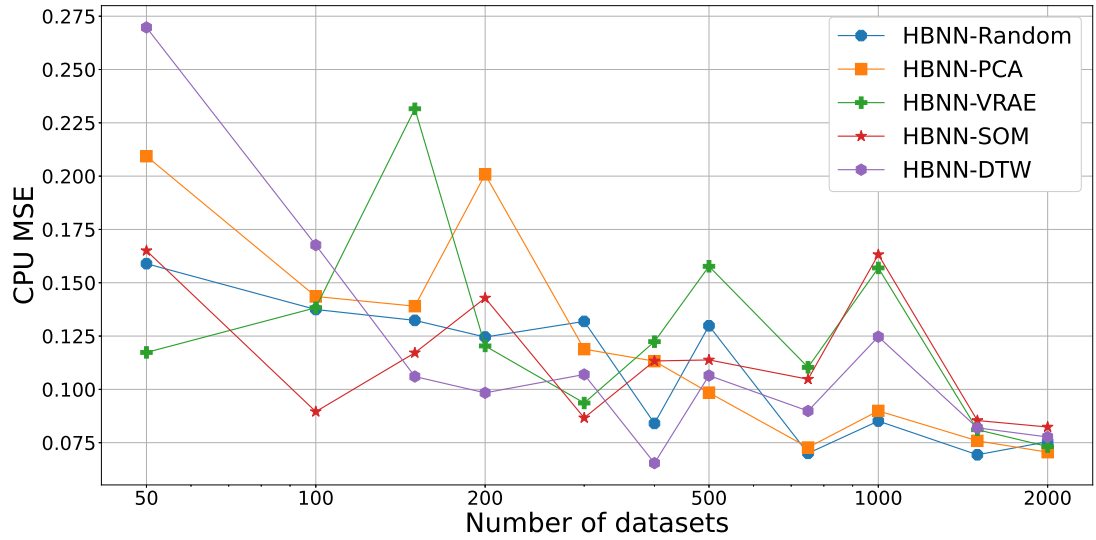


Figure C.2: MSE comparison of HBNN for the processing units demand. DTW and SOM are the best clustering-based selection approaches and allow the models to achieve the same MSE with fewer training datasets compared to a random selection. When the number of datasets is high, the MSE of all the approaches converge. The x-axis is in a logarithmic scale.

C.2 Confidende Interval Evaluation

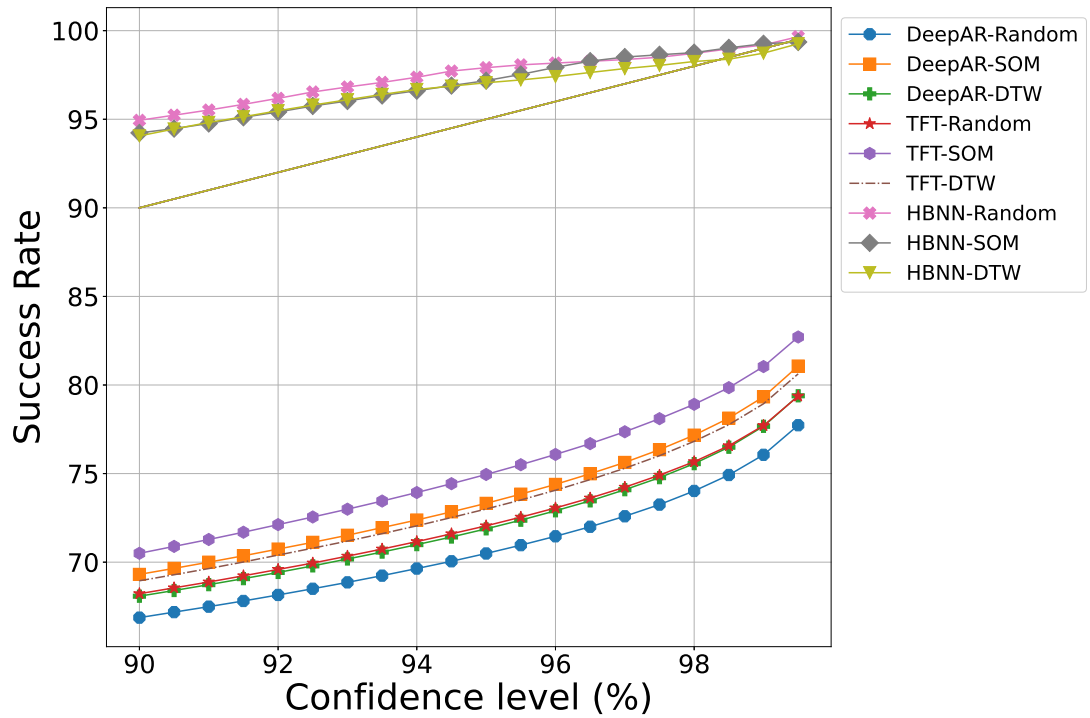


Figure C.3: Processing unit SR comparison of DeepAR, TFT and HBNN as a function of the confidence level with models trained with 1000 datasets. The model is the most accurate when the curve is closest to the line $y = x$. DTW and SOM-based selections improve the SR compared to the random selection of datasets.

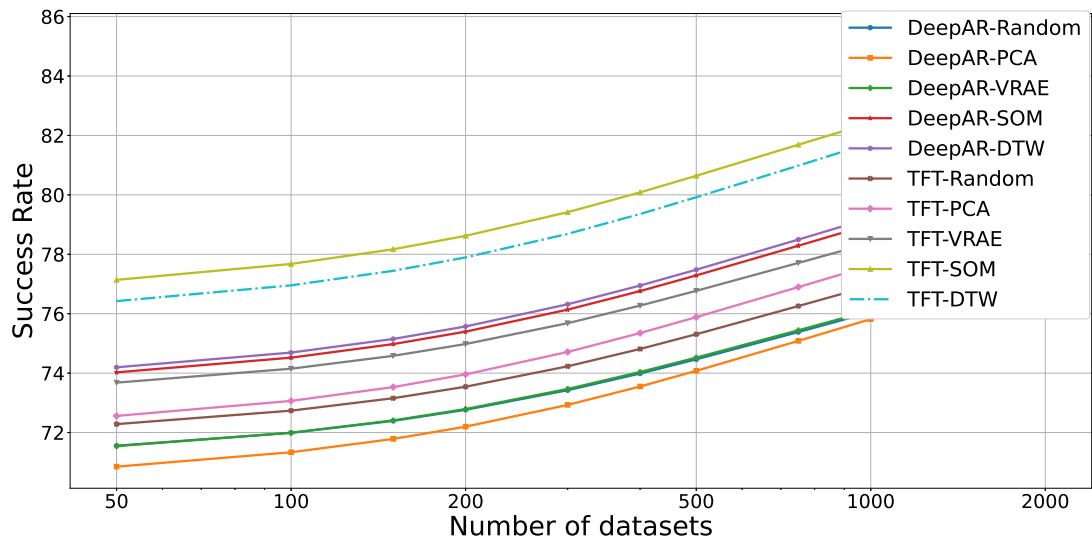


Figure C.4: Memory SR comparison of DeepAR and TFT as a function of the confidence level with models trained with 1000 datasets. The model is the most accurate when the curve is closest to the line $y = x$. DTW and SOM-based selections improve the SR compared to the random selection of datasets.

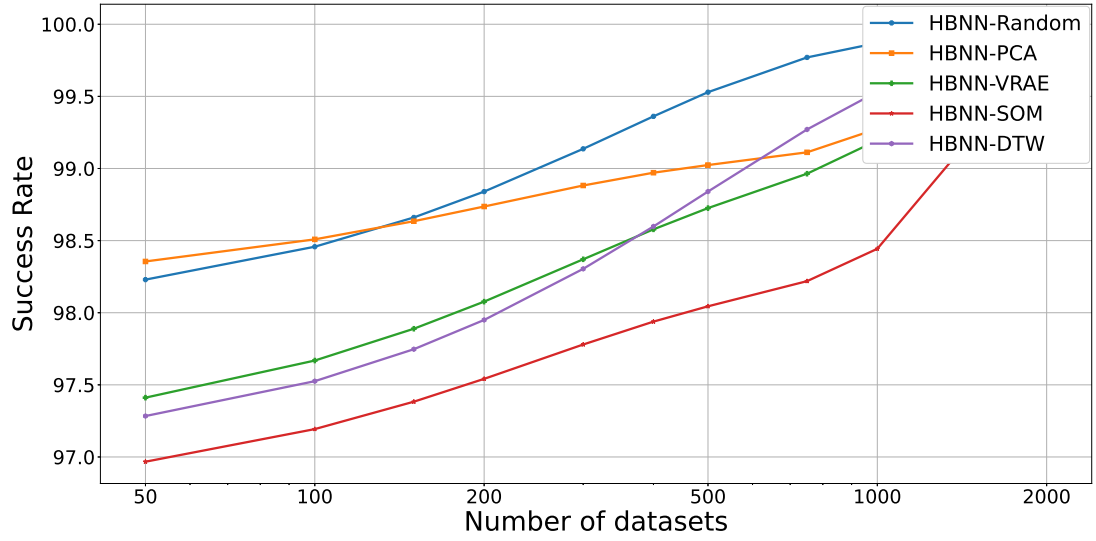


Figure C.5: Memory SR comparison of HBNN as a function of the confidence level with models trained with 1000 datasets. The model is the most accurate when the curve is closest to the line $y = x$. DTW and SOM-based selections improve the SR compared to the random selection of datasets.

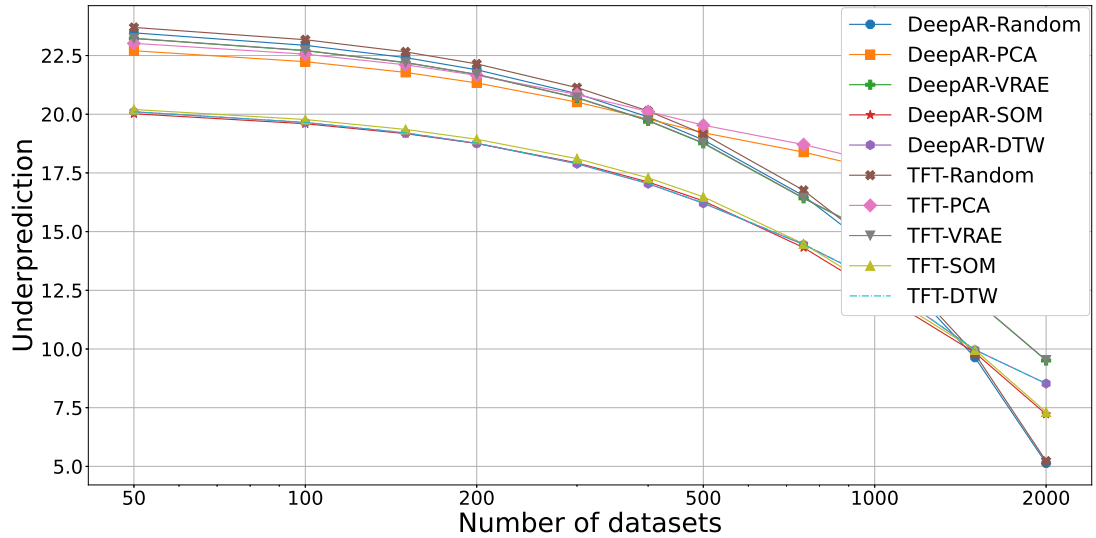


Figure C.6: Processing unit OP comparison of DeepAR and TFT. If DTW and SOM are applied, a much lower number of datasets is required to get the same OP compared to a random selection of the datasets. The x-axis is in a logarithmic scale.

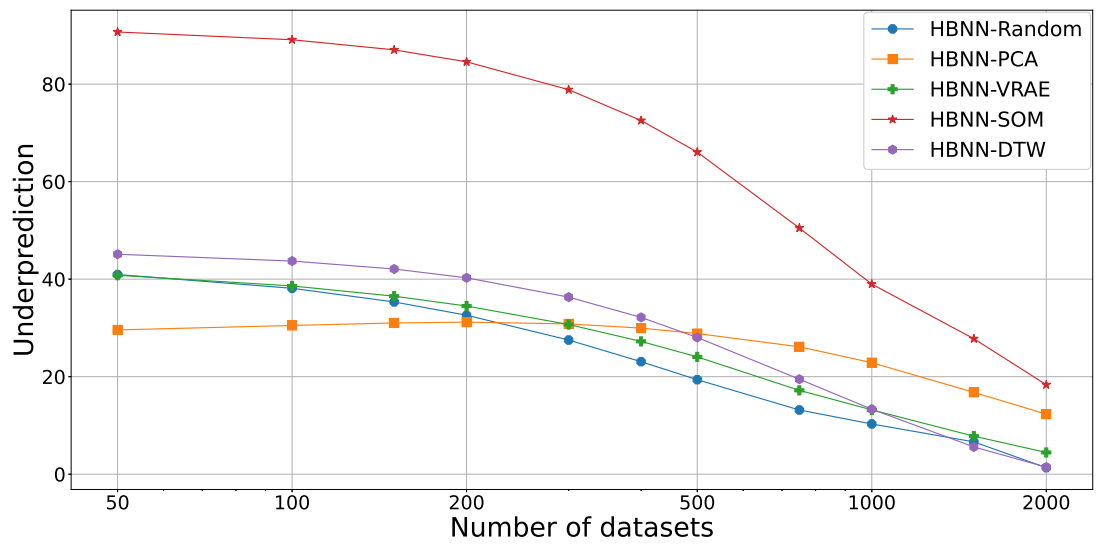


Figure C.7: Processing unit OP comparison of HBNN. If DTW and SOM are applied, a much lower number of datasets is required to get the same OP compared to a random selection of the datasets. The x-axis is in a logarithmic scale.