Future Workload and Cloud Resource Usage: Insights from an Interpretable Forecasting Model

Amadou Ba
IBM Research Europe
Dublin, Ireland
amadouba@ie.ibm.com

***Abstract*—The emergence of proactive autoscaling aims to guarantee the Quality of Service (QoS) in accordance with the Service Level Agreement (SLA) between cloud providers and users, while promoting efficient resource utilization and minimizing operational costs. Despite its benefits, proactive autoscaling remains complex due to challenges in accurately forecasting resource needs amid workload fluctuations and implementing effective actions based on these forecasts. To address these issues, we propose a mechanism for forecasting workload and cloud resources using a variant of the Transformer. This multivariate, multi–horizon forecasting approach provides both forecasts and insights into the significance of the features associated with the forecasting results, enabling time–granular autoscaling. Through experiments with real–world data, we demonstrate that, rather than first forecasting workload and then estimating resource usage, we can directly forecast resource usage. This method yields the same conclusions regarding the feature importance in workload and resource forecasting, thereby simplifying the existing autoscaling approaches.**

***Index Terms*—Cloud Computing, Feature Importance, Forecasting, Transformer.**

I. Introduction

Application developers are often attracted to cloud systems for their expansive infrastructure and automation capabilities, which offer scalability, flexibility, and cost–efficiency, among other benefits. Cloud providers aim to deliver high Quality of Service (QoS) by balancing resource allocation with QoS needs. However, the challenge lies in provisioning resources that meet performance requirements while minimizing waste and controlling costs. [1]. This challenge is further amplified by the rise of the cloud–native tools that promote the microservices architecture, reducing the software development effort needed to create software applications. In such circumstances, multiple microservices work in conjunction to address the trade–off between the resource usage and QoS. A microservice architecture allows for the development of software applications as a collection of loosely coupled, independently deployable, monitorable, and scalable services. This approach enables each microservice to be deployed as a separate container through orchestration engines such as the Kubernetes platform, making it a unit of deployment as well as a unit of resource allocation, facilitating elasticity in resource scaling. Prior to deploying cloud applications, cloud providers determine the number of pod replicas and the associated resources, such as CPU, memory, and disk I/O, required for each pod in the Kubernetes cluster to execute a specific workload [2]. To ensure high QoS in a production environment, cloud providers often resort to overprovisioning the aforementionned resources [3], which generally results in low resource utilization and high costs. Various approaches have been developed to address autoscaling challenges. These approaches include reactive and proactive methods [4], [5]. Reactive autoscaling solutions make scaling decisions by analyzing current system metrics such as CPU utilization and memory. The autoscaling mechanism is triggered when current system metrics, such as CPU utilization and memory, show abnormalities, indicating the need for autoscaling. However, reactive autoscaling is limited because it does not act in advance to prevent QoS degradation. Reactive autoscaling acts based on momentaneous metrics. To mitigate potential QoS degradation, reactive autoscalers tend to run microservices with overprovisioned resources. However, this approach does not provide the optimal balance between QoS and resource utilization. Proactive autoscaling approaches enable autoscaling based on the forecasted workloads [6], [7]. They analyze historical workload, forecast the needs of each microservice, and make scaling decisions ahead of a possible QoS degradation. However, current proactive orchestration methods have several shortcomings, specifically their reliance on non–interpretable machine learning (ML) or deep learning (DL) techniques. To address this limitation, we propose a new approach in this paper that provides interpretable and efficient forecasting of workload and cloud resources, enabling the implementation of an informed, time–granular autoscaling mechanism. Our approach is based on the Temporal Fusion Transformer (TFT), an attention–based architecture that offers the possibility to use multivariate input data for forecasting and arrive at interpretable insights by highlighting the most relevant features in the forecasted workload and cloud resources. The TFT utilizes recurrent layers for local processing to learn temporal relationships at different scales and interpretable self–attention layers for capturing long–term dependencies. This dual–layered structure enables our approach to provide high–quality forecasts and interpretable results simultaneously. The contribution of our paper lies in the development of a TFT [8] tailored for workload and cloud resources forecasting. To the best of our knowledge, this is the first paper proposing the use of TFT for interpretable, multivariable workload and cloud resources forecasting, which can lead to improved, time–granular proactive autoscaling. The remainder of the paper is organized as follows. Section II focuses on the development of

our TFT for workload and cloud resources forecasting. Next, in Section III, we present the results of our experiments and discuss the benefits of TFT for workload and cloud resources forecasting. Finally, the concluding remarks and future work are presented in Section IV.

II. Temporal Fusion Transformer for Workload and Resource Usage Forecasting

The Temporal Fusion Transformer (TFT) [8] is an ML model designed for time series forecasting. It combines the Transformer architecture with temporal fusion mechanisms to capture temporal patterns in sequential data. It is composed of the multi–head attention mechanism from the Transformer [9] with Recurrent Neural Networks (RNNs). The TFT architecture includes an LSTM encoder–decoder and multi–head attention components, primarily composed of gating mechanisms, variable selection networks, and static covariate encoding.

A. Multivariate, multi-horizon forecasting with quantile regression

Let's consider $x_{p,t} \in \mathbb{R}^{M \times N}$, where p represents the features, and $y_t \in \mathbb{R}^N$ is the response variable to forecast. We use quantile regression [8] to determine the quantile forecast at each time step

$$\hat{y}\left(q,t,\tau\right) = f_q\left(\tau, y_{t-k:t}, x_{p,t-k:t}\right). \tag{1}$$

Here, $\hat{y}\left(q,t,\tau\right)$ is the forecasted quantile at the q^{th} percentile of the τ-step ahead at time t from the forecasting model $f_q\left(\cdot\right)$. The forecast is obtained using all past information within a finite look–back window k, including the target $y_{t-k:t} = \{y_{t-k}, \cdots, y_t\}$ and the features $x_{p,t-k:t} = \{x_{p,t-k}, \cdots, x_{p,t}\}$ till and including the forecast start time at t.

B. Gating for the workload and resource usage forecasting

The fundamental part of the TFT architecture is the Gated Residual Networks (GRN). The information flow through the GRN can be represented as follows

$$\mathrm{GRN}\left(a\right) = \mathrm{LayerNorm}\left(a + \mathrm{GLU}(\eta_1)\right) \tag{2}$$

$$\mathrm{GLU}\left(\eta_1\right) = \sigma\left(W_1\eta_1 + b_1\right) \odot \left(W_2\eta_1 + b_2\right) \tag{3}$$

$$\eta_1 = W_3\eta_2 + b_3 \tag{4}$$

$$\eta_2 = \mathrm{ELU}\left(W_4 a + b_4\right) \tag{5}$$

a is the input to the GRN. W and b are weights and biases optimized during the training phase. a is passed through a dense layer with Exponential Linear Unit (ELU) activation function, followed by a linear layer with dropout. The output from this layer is then fed to the Gated Linear Unit (GLU), where $\odot$ denotes the element–wise product and σ is the sigmoid activation function. The output from the GRN is determined using standard layer normalization of the sum of input a and the GLU output (residual connection). The GRN can identify the relevant components needed for further processing and eliminate any unused components, allowing the TFT to flexibly learn the relationship between inputs and targets. Additionally, the GRN has the advantage of facilitating learning even on noisy or small datasets.

C. Variable selection for the features

In the TFT, the variable selection networks play a crucial role in identifying the relevant inputs for forecasting the workload and resource usage at each time step. These networks pinpoint the features that most impact the forecasting, enhancing the interpretability of the forecasting results. The following equations illustrate how the variable selection networks operate:

$$\xi^j_{t,vs} = \mathrm{GRN}\left(\xi^j_t\right) \tag{6}$$

$$\nu_{\chi t} = \mathrm{softmax}\left(\mathrm{GRN}\left(\Xi_t\right)\right) \tag{7}$$

$$\xi_{t,vs} = \sum_{j=1}^{m_\chi} \nu^{(j)}_{\chi t}\, \xi^j_{t,vs} \tag{8}$$

The first step in the variable selection is to linearly transform all variables at time t. The transformed input for the j^{th} variable is represented by ξ^j_t, where each transformed input is fed to a GRN block, which considers the non–linearities. The output of these blocks is $\xi^j_{t,vs}$. Another GRN block, which takes as input Ξ_t, the flattened ξ^j_t produces a vector $\nu_{\chi t} \in \mathbb{R}^{m_\chi}$ after passing through a softmax activation function to get the variable selection weights. The variable weights $\nu_{\chi t}$ are multiplied with the corresponding processed inputs to get the final output of the variable selection networks. This approach highlights important variables for prediction and filters out noisy inputs that could degrade performance.

D. LSTM encoders and decoders

LSTM is used to leverage local context through a sequence–to–sequence operation, where past inputs are fed to the encoder and future values are fed to the decoder. The LSTM is followed by a gating mechanism. The LSTM encoder facilitates learning short–term interactions effectively.

E. Multi–Head Attention

Multi–Head Attention (MHA) is used to capture long–term dependencies across different time steps. MHA processes temporal features through multiple parallel scaled dot–product attention operations, followed by the GRN for non–linear processing. The attention weights provided by the MHA offer compelling representational capability, enabling the interpretation of how each variable contributes to the output of the model.

F. Quantile forecasts

Prediction intervals are generated from the output of MHA for each future time step. This process provides a range of probable values for future forecasts.

G. Loss Functions

Our TFT is trained by minimizing the sum of all quantile losses

$$\mathcal{L}(\Omega, W) = \sum_{y_t \in \Omega} \sum_{q \in \mathcal{Q}} \sum_{\tau=1}^{\tau_{max}} \frac{QLoss\left(y_t, \hat{y}\left(q, t-\tau, \tau\right), q\right)}{N \tau_{max}} \quad (9)$$

where

$$QLoss\left(y, \hat{y}, q\right) = q\left(y - \hat{y}\right)_+ + (1-q)\left(\hat{y} - y\right)_+ \quad (10)$$

and Ω represents the training datasets with N the size of the training samples and W denotes the weights of TFT. Q is the set of output quantiles and $(\cdot)_+ = \max(0, \cdot)$.

III. Experiments

A. Adopted architecture for our experiment

We adopt the TFT which is composed of the variable selection networks, the LSTM encoder–decoder, the GRN, and the multi–head attention. These blocks receive inputs including past features, the target variable, and future known features. They produce quantile forecasts of the workload and resource usage. We utilize real workload and cloud resources datasets obtained from the Materna data center [10]. The network received throughput is considered as the desired workload, while CPU and memory are the resources we forecast.

B. Datasets

We evaluated the TFT model using the GWA–T–13 dataset from the Materna data center [10]. This dataset comprises three traces of performance metrics from 520, 527, and 547 Virtual Machines (VMs) in the Materna data center, collected over a three–month period. Each trace represents one month of data and includes critical business applications running on VMware ESX environments with 49 Hosts, 69 CPU cores, and 6780 GB RAM. The dataset consists of 11 metrics, including CPU usage, memory usage, disk write throughput, network received and transmitted throughput. We specifically focus on the network received throughput as the workload to forecast, and CPU and memory as the resources to forecast. We show in this experiment that with the classical features present in any time series data representing the workload and cloud resources such as the day of the week, the hour of the day, and the minute of the hour, we are able to exploit them in an efficient way through the interpretability of the TFT and perform informed and time–granular autoscaling. This can easily be utilized for various applications running on cloud. Our experiments do not rely on any autoregressive features, but only on the seasonal features. We present the network received throughput and the CPU and memory usage in Fig. 1. These metrics are collected at a sampling rate of 5 minutes. The close patterns observed between the network received throughput and CPU usage (Fig. 1) indicate efficient processing by the CPU, reflecting a balanced handling of workload processing demands. Furthermore, the seasonal behavior observed in both metrics, with higher processing during the day and lower processing in the evening, aligns with typical user behavior for certain applications. These patterns reflect usage intensity peaking during the day and decreasing in the evenings.

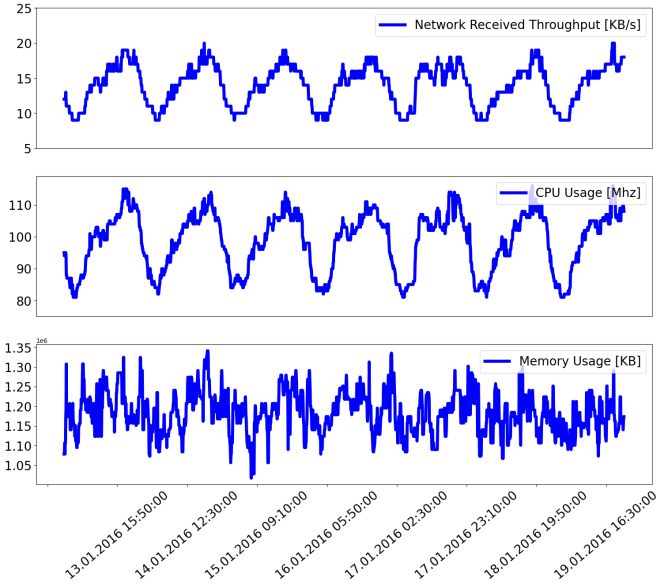


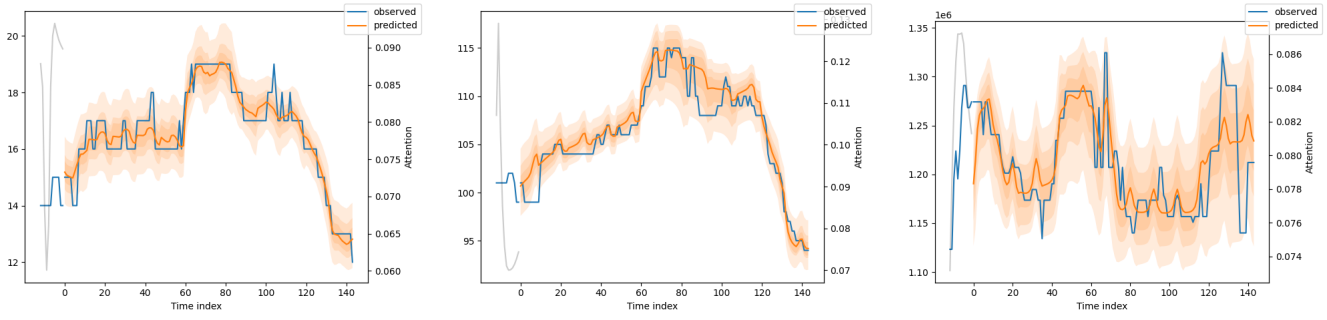
Fig. 1: Data composed of the network receiver throughput, which we consider as our workload, CPU and memory.

C. Training procedures

To build our forecasting model, we use features such as the hour of the day, the minute of the hour, and the day of the week, extracted from the date and time records in the previously presented Materna dataset. Additionally, for the TFT model, we include a time index that increments by one at each time step. Each time series is standardized separately to ensure that the values remain positive. This is achieved with the EncoderNormalizer, which dynamically scales each encoder sequence during model training. We train our model using PyTorch Lightning and evaluate its performance using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) as metrics, the best performances are given by the low values of these metrics. We tune parameters such as a batch size of 32, a learning rate of 0.03, and 100 epochs. Our model architecture includes a hidden size of 8, an attention head size of 1, and a dropout rate of 0.1. Furthermore, we set a maximum prediction length of 144, allowing us to forecast 12 hours ahead, and a maximum encoder length of 12. Given the sampling rate of 5 minutes, this corresponds to one hour. To evaluate the performance of our model, we utilize a quantile loss function. Across the results reported in this paper, we keep these tuning parameters intact.

D. Workload and cloud resources forecasting

Traditionally, the classical approach to forecast the required resources and determine the need for autoscaling involves finding a mapping function between workload and resources. This approach uses the forecasted workload data to estimate future resource requirements. The rationale behind this method



(a) An example of the workload forecasting. (b) An example of the CPU forecasting. (c) An example of the memory forecasting.

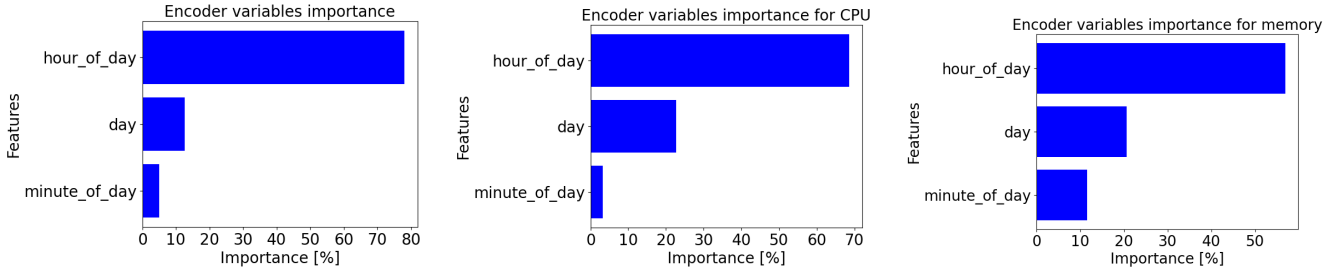
Fig. 2: Example of multi–horizon forecasting of the workload and the resource usage.

is that workload, being reflective of user behavior, can be directly forecasted, with CPU and/or memory serving as proxies for workload. In this experiment, we demonstrate that in certain scenarios, such as this application, directly forecasting CPU and/or memory can provide the same insights as forecasting the workload. This approach offers the advantage of eliminating the computational step involved in estimating future resource needs, thereby reducing the cost associated with running autoscaling algorithms. Forecasting resources enables to determine if the resources required at a future time will be sufficient to prevent SLA violations. For this purpose, we apply the TFT to our data. A distinctive characteristic of the TFT model is its attention mechanism, which assigns varying levels of importance to different time points during forecasting, providing interpretability to the forecasted results. The TFT is designed for multi–horizon forecasting, enabling it to forecast future values across multiple time horizons simultaneously. The model incorporates output layers that forecast values for each time horizon of interest, allowing it to generate forecasts for different future points in the workload and cloud resources data. We forecast the workload, CPU and memory using the features described in the dataset section, which include the minute of the hour, the hour of the day, and the day of the week. Fig. 2 presents the forecasting results of the workload, CPU and memory covering a 12–hour period with 144 samples. The figure also shows the quantiles associated with the forecasts and the attention allocated to the encoder length of 1 hour (12 samples) (grey).

Fig. 3 and Fig. 4 illustrate the significance of the features used during the training and the forecasting of the workload, CPU and memory. We observe that the most influential feature in the predicted results is the hour of the day, followed by the day of the week, and then the minute of the hour of the day. However, the importance of these features differs between the encoder and decoder. While the hour of the day remains highly important for both, the encoder places more emphasis on it compared to the decoder. On the other hand, the importance of the day of the week and minute of the day varies between the encoder and decoder. This analysis suggests that for informed and time–granular autoscaling based on workload, CPU and memory forecasting, better performance can be achieved if autoscaling occurs hourly. Our results demonstrate good accuracy provided by the TFT, which is achieved alongside the interpretability offered by the model. The high values of the RMSE and MAE associated to the memory are due to the fact that the memory presents high values in [KB], compared to the scale of the workload and CPU. For the workload, CPU, and memory, the TFT provides RMSE and MAE values of 0.68 and 0.5, 1.75 and 1.32, and 35,097.32 and 24,536.86, respectively. These results are competitive with the most established regression methods, while the proposed approach also offers interpretability through feature importance.

E. Discussions

In the experiments conducted for the workload, CPU, and memory forecasting, we tested the robustness of the TFT by varying the sizes of the training data, forecasting horizons, and tuning parameters. We observed that the hour of the day feature remained the most influential in the forecasted results across different configurations. This paper highlights two key findings. Firstly, we demonstrate that for forecasting cloud computing resource utilization, it is not necessary to first forecast the workload and then use the forecasted workload to predict resource usage, as suggested in [6] and [7]. This is due to the emergence of advanced approaches like the TFT, a neural network capable of universal approximation and directly learning mapping functions from input to output data. The second finding emphasizes the advantage of interpretability provided by the significance of the features in the forecasting results. Across the three experiments, we consistently observed the importance of various features, with the hour of the day consistently emerging as the most significant. This insight is crucial for designing time–granular autoscaling strategies. Recognizing the hour of the day as the most influential feature allows for time–granular and informed decisions about autoscaling frequency, balancing the trade–off between lower granularity, which may lead to resource wastage, and higher granularity, which risks SLA violations. Simplified autoscaling mechanisms are particularly important for large-scale applications, where autoscaling must accommodate millions of users.

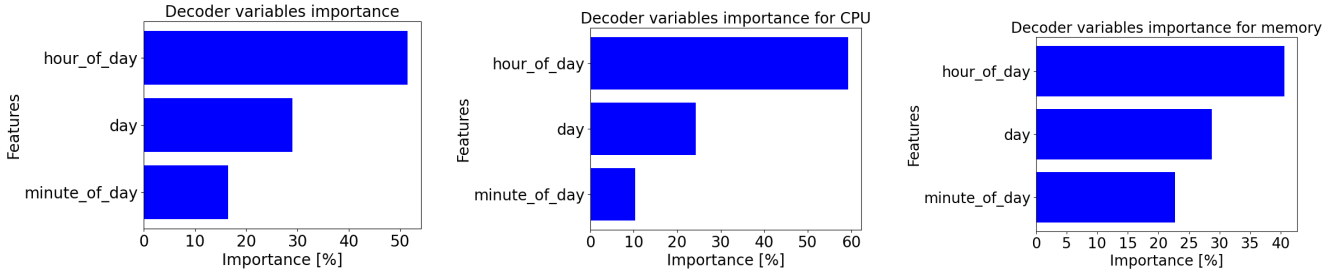


(a) Feature importance associated to the workload provided by the encoder (training).

(b) Feature importance associated to the CPU provided by the encoder (training).

(c) Feature importance associated to the memory provided by the encoder (training).

Fig. 3: Feature importance associated to the workload and the resource usage during the training phase.



(a) Feature importance associated to the workload provided by the decoder (forecasting).

(b) Feature importance associated to the CPU provided by the decoder (forecasting).

(c) Feature importance associated to the memory provided by the decoder (forecasting).

Fig. 4: Feature importance associated to the workload and the resource usage during the forecasting.

Informed, time–granular autoscaling thus optimizes resource utilization while ensuring SLA compliance.

IV. Conclusions and Future Work

This paper demonstrates the efficient use of the TFT for workload and cloud resources forecasting, yielding interpretable results regarding feature influence. This interpretability is valuable for time–granular autoscaling. The consistency of these findings across all experiments highlights the hour of day as the most influential feature in our study. Additionally, the performance metrics demonstrate the high performance of the TFT for the three experimented data. In future work, we envisage the combination of this multivariate, multi–horizon, and interpretable forecasting approach with a reinforcement learning method for deploying and scaling the proposed time–granular autoscaling mechanism in various applications.

Acknowledgments

This work was supported by the European Union's Horizon Research and Innovation program under AC^3 project and grant agreement No 101093129.

References

[1] J. Lee, "A view of cloud computing," *Int. J. Networked Distributed Comput.*, vol. 1, pp. 2–8, 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:267829120

[2] M. Mekki, B. Brik, A. Ksentini, and C. Verikoukis, "Xai-enabled fine granular vertical resources autoscaler," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, 2023, pp. 161–169.

[3] H. Shen and L. Chen, "Resource demand misalignment: An important factor to consider for reducing resource over-provisioning in cloud datacenters," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1207–1221, 2018.

[4] M. C. Calzarossa, L. Massari, M. I. M. Tabash, and D. Tessera, "Cloud autoscaling for http/2 workloads," in *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*, 2017, pp. 1–6.

[5] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 500–507.

[6] Z. Wang, S. Zhu, J. Li, W. Jiang, K. K. Ramakrishnan, Y. Zheng, M. Yan, X. Zhang, and A. X. Liu, "Deepscaling: microservices autoscaling for stable cpu utilization in large scale cloud systems," in *Proceedings of the 13th Symposium on Cloud Computing*, ser. SoCC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 16–30. [Online]. Available: https://doi.org/10.1145/3542929.3563469

[7] S. Xue, C. Qu, X. Shi, C. Liao, S. Zhu, X. Tan, L. Ma, S. Wang, S. Wang, Y. Hu, L. Lei, Y. Zheng, J. Li, and J. Zhang, "A meta reinforcement learning approach for predictive autoscaling in the cloud," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 4290–4299. [Online]. Available: https://doi.org/10.1145/3534678.3539063

[8] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207021000637

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[10] Materna, "GWA-T-13 Materna," http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna, 2023.