# Forecasting Resource Usage in Cloud Environments Using Temporal Convolutional Networks

**Mahmoud Abouelyazid**
Exodia AI Labs

## Abstract

**Background**: Predicting resource usage in cloud environments is crucial for optimizing costs. While recurrent neural networks and time series techniques are commonly used for forecasting, their limitations, such as vanishing gradients and lack of memory retention, necessitate the use of convolutional networks for modeling sequential data.

**Objective**: This research proposes a temporal convolutional network (TCN) to forecast CPU usage and memory consumption in cloud environments. TCNs utilize dilated convolutions to capture temporal dependencies and maintain a fixed-sized receptive field, enabling them to handle sequences of varying lengths and capture long-term dependencies. The performance of the TCN is compared with Long Short-Term Memory (LSTM) Networks, Gated Recurrent Unit (GRU) Networks, and Multilayer Perceptron (MLP).

**Dataset**: The study employs the Google Cluster Workload Traces 2019 data, focusing on CPU and memory utilization ranging between 5% and 95% over a 24-hour period, extracted from the first ten days.

**Results**: The TCN outperforms other methods in predicting both CPU usage and memory consumption. For CPU usage prediction, the TCN achieves lower error metrics, including Mean Squared Error (MSE) of 0.05, Root Mean Squared Error (RMSE) of 0.22, Mean Absolute Error (MAE) of 0.18, and Mean Absolute Percentage Error (MAPE) of 3.5%. The TCN also demonstrates higher forecast accuracy, with FA1 = 85%, FA5 = 95%, and FA10 = 98%. Similar performance improvements are observed for memory consumption prediction, with the TCN achieving lower error metrics and higher forecast accuracy compared to LSTM, GRU, and MLP. The TCN exhibits better computational efficiency in terms of training time, inference time, and memory usage.

**Conclusion**: The proposed temporal convolutional network (TCN) demonstrates good performance in forecasting CPU usage and memory consumption in cloud environments compared to LSTM, GRU, and MLP. Since TCN's can capture temporal dependencies and handle sequences of varying lengths makes it a promising approach for resource usage prediction and cost optimization in cloud computing.

**Keywords**: Cloud Environments, Convolutional Networks, Forecasting, Resource Usage, Temporal Convolutional Networks

## Introduction

Cloud computing involves the delivery of computing services over the internet. These services include servers, storage, databases, networking, software, analytics, and intelligence [1]. The concept of cloud computing dates back to the 1960s. However, it was not until the late 1990s and early 2000s that the technology began to take shape [2]. The term "cloud computing" was first used in 1996 by Compaq Computer Corporation. In 2006, Amazon

launched its Elastic Compute Cloud (EC2) service, which allowed users to rent virtual computers to run their own applications. This marked a significant milestone in the history of cloud computing.

Cloud computing offers numerous benefits, including cost savings, scalability, flexibility, and accessibility. With cloud computing, businesses can reduce their IT infrastructure costs and focus on their core competencies. They can also scale their computing resources up or down based on their needs. Additionally, cloud computing enables users to access their data and applications from anywhere, at any time, using any device with an internet connection [3].

I. Types of Cloud Computing Services

There are three main types of cloud computing services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

A. Infrastructure as a Service (IaaS)

IaaS is a cloud computing service that provides users with virtualized computing resources over the internet. These resources include servers, storage, and networking. Examples of IaaS providers include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform.

The benefits of IaaS include cost savings, scalability, and flexibility. With IaaS, businesses can avoid the upfront costs of purchasing and maintaining their own hardware. They can also scale their computing resources up or down based on their needs. Additionally, IaaS allows businesses to focus on their core competencies rather than worrying about IT infrastructure. However, IaaS also has some drawbacks. It requires a certain level of technical expertise to manage and maintain the virtualized computing resources. Additionally, security and compliance can be a concern, as businesses must ensure that their data is properly secured in the cloud.

B. Platform as a Service (PaaS)

PaaS is a cloud computing service that provides users with a platform for developing, testing, and deploying applications. Examples of PaaS providers include Heroku, Google App Engine, and Microsoft Azure. The benefits of PaaS include faster development times, scalability, and cost savings. With PaaS, developers can focus on writing code rather than worrying about infrastructure. They can also scale their applications up or down based on demand. Additionally, PaaS can be more cost-effective than building and maintaining an in-house development platform. It can be less flexible than IaaS, as users are limited to the tools and frameworks provided by the PaaS provider. Additionally, vendor lock-in can be a concern, as applications developed on a specific PaaS platform may not be easily portable to other platforms.

C. Software as a Service (SaaS)

SaaS is a cloud computing service that provides users with access to software applications over the internet. Examples of SaaS providers include Salesforce, Google Apps, and Microsoft Office 365.

The benefits of SaaS include accessibility, cost savings, and scalability. With SaaS, users can access their applications from

anywhere, at any time, using any device with an internet connection. Additionally, SaaS can be more cost-effective than purchasing and maintaining software licenses. SaaS providers also handle updates and maintenance, which can save businesses time and money. Security and privacy can be a concern, as businesses must trust the SaaS provider to properly secure their data. Additionally, customization options may be limited, as users are typically limited to the features and functionality provided by the SaaS provider [5].

## II. Cloud Deployment Models

There are four main cloud deployment models: public cloud, private cloud, hybrid cloud, and community cloud.

### A. Public Cloud

A public cloud is a cloud computing deployment model in which computing resources are owned and operated by a third-party provider and shared among multiple customers over the internet. Examples of public cloud providers include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform.

The advantages of public cloud include cost savings, scalability, and accessibility. With public cloud, businesses can avoid the upfront costs of purchasing and maintaining their own hardware. They can also scale their computing resources up or down based on their needs. Additionally, public cloud enables users to access their data and applications from anywhere, at any time, using any device with an internet connection. As businesses must trust the cloud provider to properly secure their data. Additionally, performance may be affected by network latency and bandwidth limitations.

### B. Private Cloud

A private cloud is a cloud computing deployment model in which computing resources are dedicated to a single organization and not shared with other customers. Private clouds can be hosted on-premises or by a third-party provider. The advantages of private cloud include greater control, customization, and security. With private cloud, businesses have complete control over their computing resources and can customize them to meet their specific needs. Additionally, private cloud can provide greater security and compliance, as businesses can ensure that their data is properly secured and meets regulatory requirements. It can be more expensive than public cloud, as businesses must purchase and maintain their own hardware. scalability may be limited, as businesses are responsible for provisioning and managing their own computing resources.

### C. Hybrid Cloud

A hybrid cloud is a cloud computing deployment model that combines public and private clouds, allowing data and applications to be shared between them. Hybrid cloud enables businesses to take advantage of the benefits of both public and private clouds.

The advantages of hybrid cloud include flexibility, scalability, and cost savings. With hybrid cloud, businesses can run sensitive workloads on a private cloud while leveraging the scalability and cost savings of public cloud for less sensitive

workloads. Additionally, hybrid cloud enables businesses to easily move workloads between public and private clouds as needed. Hybrid cloud can be more complex to manage than a single cloud deployment model, as businesses must ensure that data and applications can be seamlessly shared between public and private clouds. Network latency and bandwidth limitations can affect performance.

D. Community Cloud

A community cloud is a cloud computing deployment model in which computing resources are shared among several organizations with common interests or requirements. Community clouds can be hosted by a third-party provider or by one of the participating organizations.

The advantages of community cloud include cost savings, collaboration, and customization. With community cloud, organizations can share the costs of purchasing and maintaining computing resources. Additionally, community cloud enables organizations to collaborate and share resources, which can lead to increased efficiency and innovation. Community cloud can also be customized to meet the specific needs of the participating organizations.

However, community cloud also has some disadvantages. Governance and control can be a concern, as participating organizations must agree on how the computing resources will be managed and shared. Additionally, security and compliance can be a challenge, as each participating organization must ensure that their data is properly secured and meets regulatory requirements.

III. Overview of Cloud Resources

Cloud resources are the fundamental building blocks of cloud computing. They include compute, storage, and network resources that are provisioned and managed by cloud providers. These resources are essential for running applications and storing data in the cloud.

A. Compute Resources

Compute resources are the processing power and memory that are used to run applications in the cloud. There are three main types of compute resources: virtual machines (VMs), containers, and serverless computing.

1. Virtual Machines (VMs)

VMs are software emulations of physical computers that run on top of a hypervisor. They provide a high degree of flexibility and control over the underlying hardware and operating system. VMs are ideal for running traditional applications that require a dedicated operating system and hardware resources.

2. Containers

Containers are lightweight, portable, and self-contained units of software that include all the dependencies needed to run an application. They provide a more efficient and scalable alternative to VMs, as they share the same operating system kernel and can be quickly provisioned and deprovisioned as needed.

3. Serverless Computing

Serverless computing is a cloud computing model where the cloud provider dynamically manages the allocation of compute resources. Developers can focus on writing code without worrying about the underlying infrastructure. Serverless computing is ideal for event-driven and highly scalable applications.

B. Storage Resources

Storage resources are used to store and manage data in the cloud. There are three main types of storage resources: object storage, block storage, and file storage.

1. Object Storage

Object storage is a scalable and cost-effective storage model that stores data as objects in a flat address space. Each object is assigned a unique identifier and can be accessed using HTTP or HTTPS. Object storage is ideal for storing unstructured data such as images, videos, and backups.

2. Block Storage

Block storage is a storage model that presents data as a series of fixed-size blocks. Each block is assigned a unique identifier and can be accessed using a block protocol such as iSCSI or Fibre Channel. Block storage is ideal for storing structured data such as databases and virtual machine disk images.

3. File Storage

File storage is a storage model that presents data as a hierarchical file system. Files are organized into directories and can be accessed using protocols such as NFS or SMB. File storage is ideal for storing and sharing files across multiple users and applications.

C. Network Resources

Network resources are used to connect cloud resources and enable communication between them. There are three main types of network resources: virtual private clouds (VPCs), load balancers, and content delivery networks (CDNs).

1. Virtual Private Clouds (VPCs)

VPCs are isolated networks within the cloud that enable users to launch and manage cloud resources in a secure and customizable environment. VPCs provide control over IP address ranges, subnets, and routing tables, and can be connected to on-premises networks using VPN or direct connect.

2. Load Balancers

Load balancers are used to distribute incoming traffic across multiple instances of an application. They improve the availability and performance of applications by ensuring that no single instance is overwhelmed with traffic. Load balancers can be configured to route traffic based on various criteria such as URL, cookie, or IP address.

3. Content Delivery Networks (CDNs)

CDNs are networks of geographically distributed servers that are used to deliver content to users with high availability and performance. CDNs cache content at edge locations that are closer to the users, reducing latency and improving the user experience. CDNs are ideal for delivering

static content such as images, videos, and scripts.

VI. Factors Affecting Resource Usage

Several factors can affect the usage of cloud resources, including application architecture and design, workload characteristics, user behavior and traffic patterns, and service level agreements (SLAs) and performance requirements.

A. Application Architecture and Design

The architecture and design of an application can have a significant impact on resource usage. Monolithic applications that are tightly coupled and require a lot of resources may not be well-suited for the cloud. On the other hand, microservices-based applications that are loosely coupled and can scale independently are ideal for the cloud.

B. Workload Characteristics

The characteristics of a workload can also affect resource usage. There are three main types of workloads:

1. CPU-Intensive Workloads

CPU-intensive workloads require a lot of processing power and can benefit from high-performance compute resources such as VMs with multiple cores or high-frequency CPUs.

2. Memory-Intensive Workloads

Memory-intensive workloads require a lot of memory and can benefit from compute resources with large amounts of RAM or in-memory databases.

3. I/O-Intensive Workloads

I/O-intensive workloads require a lot of disk or network I/O and can benefit from high-performance storage and network resources such as solid-state drives (SSDs) or high-bandwidth network interfaces.

C. User Behavior and Traffic Patterns

User behavior and traffic patterns can also affect resource usage. Applications that experience unpredictable or bursty traffic may require more resources than applications with steady and predictable traffic. Additionally, applications that are accessed by users in different geographic regions may require resources that are geographically distributed to ensure low latency and high performance.

D. Service Level Agreements (SLAs) and Performance Requirements

SLAs and performance requirements can also affect resource usage. Applications that require high availability or low latency may require more resources than applications with less stringent requirements. Additionally, SLAs that guarantee a certain level of performance or uptime may require additional resources to ensure that the SLAs are met.

IV. Monitoring and Measuring Resource Usage

Monitoring and measuring resource usage is essential for optimizing the performance and cost of cloud applications. It enables developers and administrators to identify bottlenecks, optimize resource allocation, and ensure that applications are meeting their SLAs and performance requirements.

Monitoring resource usage is important for several reasons. First, it enables developers

and administrators to identify performance issues and bottlenecks before they impact the end-user experience. Second, it enables them to optimize resource allocation and reduce costs by identifying underutilized or overprovisioned resources. Third, it enables them to ensure that applications are meeting their SLAs and performance requirements.

A. Key Performance Indicators (KPIs) for Resource Usage

There are several KPIs that can be used to monitor resource usage, including:

1. CPU Utilization

CPU utilization measures the percentage of time that the CPU is busy processing instructions. High CPU utilization can indicate that an application is CPU-bound and may require additional compute resources.

2. Memory Utilization

Memory utilization measures the amount of memory that is being used by an application. High memory utilization can indicate that an application is memory-bound and may require additional memory resources.

3. Network Bandwidth

Network bandwidth measures the amount of data that is being transferred over the network. High network bandwidth can indicate that an application is network-bound and may require additional network resources.

4. Storage Capacity

Storage capacity measures the amount of data that is being stored on disk. High storage capacity can indicate that an application is running out of disk space and may require additional storage resources.

C. Cloud Monitoring Tools and Services

There are several cloud monitoring tools and services that can be used to monitor resource usage, including:

1. Native Cloud Provider Tools

Most cloud providers offer native monitoring tools that can be used to monitor resource usage. For example, Amazon Web Services (AWS) offers CloudWatch, which provides monitoring and observability of AWS resources and applications. Similarly, Microsoft Azure offers Azure Monitor, which provides monitoring and diagnostics of Azure resources and applications.

2. Third-Party Monitoring Solutions

There are also several third-party monitoring solutions that can be used to monitor resource usage across multiple cloud providers. These solutions often provide additional features such as application performance monitoring (APM), log management, and alert management. Examples of third-party monitoring solutions include Datadog, New Relic, and Splunk.

## Significance of the study

Predicting resource usage and optimizing cost is a growing area of interest in cloud computing. As cloud adoption continues to grow, organizations are looking for ways to optimize their cloud spend and ensure that they are getting the most value from their cloud investments.

One approach to predicting resource usage is to use time series forecasting techniques. Time series forecasting involves analyzing historical data to identify patterns and trends, and using that information to predict future values. Recurrent Neural Networks (RNNs) are a popular choice for time series forecasting, as they are designed to handle sequential data and can learn long-term dependencies. RNNs have certain limitations that can impact their effectiveness for time series forecasting. One of the main challenges is the vanishing gradient problem, which occurs when the gradients of the loss function become very small during training. This can make it difficult for the network to learn long-term dependencies and can lead to poor performance.

RNNs are designed to process sequential data, but they do not have an explicit memory mechanism to store and retrieve information over long periods of time. This can make it difficult for RNNs to capture long-term dependencies and can limit their effectiveness for time series forecasting.

To address these limitations, some studies have proposed using convolutional neural networks (CNNs) for time series forecasting [6]. CNNs are a type of neural network that are designed to handle grid-like data, such as images or time series data. They use a series of convolutional layers to extract features from the input data and can learn hierarchical representations of the data [7].

CNNs for time series forecasting do not suffer from the vanishing gradient problem. This is because the gradients in a CNN are computed using a fixed number of steps, regardless of the length of the input sequence. This allows CNNs to learn long-term dependencies more effectively than RNNs [8].

CNNs for time series forecasting is that they can capture both local and global dependencies in the data. CNNs use a series of convolutional layers to extract features at different scales, which allows them to capture both short-term and long-term dependencies in the data. This can lead to more accurate predictions and better performance compared to RNNs. Another challenge is that CNNs can be computationally expensive to train, especially for large datasets. This can make it difficult to use CNNs for real-time forecasting or for applications that require frequent updates to the model [9], [10].

## Objective

In this study, we propose employing a temporal convolutional network (TCN) to forecast CPU usage and memory consumption within a cloud environment. TCN represents a neural network architecture utilizing dilated convolutions to capture temporal relationships within data. A defining trait of TCN is its capacity to maintain a fixed-sized receptive field irrespective of sequence length, enabling it to handle sequences of varying lengths while capturing long-term dependencies effectively. TCNs employ a 1D convolutional structure wherein each layer comprehends outputs from preceding layers. This is accomplished through the utilization of dilated convolutions, which exponentially expand the receptive field without increasing parameters or computational complexity. This study also has conducted comparisons with alternative

methods including Long Short-Term Memory (LSTM) Networks, Gated Recurrent Unit (GRU) Networks, and Multilayer Perceptron (MLP).

## Dataset

The data used is from the Google Cluster Workload Traces of 2019. It includes variables that were derived by summing up the CPU and Memory usage of tasks within each job. These calculations were done at five-minute intervals over a full day (24 hours). The dataset was narrowed down to the first ten days, focusing on CPU and memory utilization ranging from 5% to 95%. This final dataset comprises two main variables: one representing CPU usage as a percentage and the other indicating Memory usage as a percentage.

## Methods

### Temporal Convolutional Network (TCN)

Temporal Convolutional Networks (TCN) are a class of convolutional neural networks designed to handle sequential data. TCNs employ a series of 1D convolutions along the temporal dimension, allowing them to capture temporal dependencies in the input sequence [11]. The output of a TCN for a given time step depends on a fixed number of past inputs, determined by the receptive field size. The receptive field size can be increased by stacking multiple convolutional layers or using dilated convolutions. TCNs can be mathematically expressed as:

$$\boldsymbol{y_t} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{k-1}} \boldsymbol{f(x_{t-i})} \cdot \boldsymbol{w_i}$$

where $\boldsymbol{y_t}$ is the output at time step $t$, $f(\cdot)$ is the activation function, $x_{t-i}$ is the input at time step $t-i$, $w\_i$ are the learned weights, and $k$ is the size of the convolutional kernel.

### Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) Networks are a recurrent neural network (RNN) designed to address the vanishing gradient problem in traditional RNNs. LSTMs introduce a memory cell and three gating mechanisms: input gate, forget gate, and output gate [11], [12]. These gates regulate the flow of information into and out of the memory cell, allowing LSTMs to capture long-term dependencies in sequential data. The mathematical formulation of an LSTM cell can be expressed as:

$$\boldsymbol{i_t} = \boldsymbol{\sigma}(\boldsymbol{W_i} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_i})$$

$$\boldsymbol{f_t} = \boldsymbol{\sigma}(\boldsymbol{W_f} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_f})$$

$$\boldsymbol{o_t} = \boldsymbol{\sigma}(\boldsymbol{W_o} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_o})$$

$$\widetilde{\boldsymbol{C_t}} = \boldsymbol{\tanh}(\boldsymbol{W_C} \cdot [\boldsymbol{h_{t-1}}, \boldsymbol{x_t}] + \boldsymbol{b_C})$$

$$\boldsymbol{C_t} = \boldsymbol{f_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{i_t} \odot \widetilde{\boldsymbol{C_t}}$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \boldsymbol{\tanh}(\boldsymbol{C_t})$$

where $i_t$, $f_t$, and $o_t$ are the input, forget, and output gates, respectively; $C_t$ is the memory cell state; $h_t$ is the hidden state; $W$ and $b$ are learned weights and biases; σ is the sigmoid activation function; and ⊙ denotes element-wise multiplication.

### Gated Recurrent Unit (GRU) Networks

Gated Recurrent Unit (GRU) Networks are a simplified variant of LSTMs, combining the forget and input gates into a single update gate and merging the cell state and hidden state [13], [14]. GRUs have fewer

parameters than LSTMs and can be computationally more efficient while still capturing long-term dependencies. The mathematical formulation of a GRU cell is given by:

$$\boldsymbol{z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)}$$

$$\boldsymbol{r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)}$$

$$\boldsymbol{\widetilde{h_t} = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)}$$

$$\boldsymbol{h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t}}$$

where $z_t$ is the update gate, $r_t$ is the reset gate, $h_t$ is the hidden state, $W$ and $b$ are learned weights and biases, and $\odot$ denotes element-wise multiplication.

### *Multilayer Perceptron (MLP)*

Multilayer Perceptron (MLP) is a feedforward neural network consisting of multiple layers of interconnected nodes (neurons). Each neuron in an MLP applies a nonlinear activation function to a weighted sum of its inputs. MLPs are used for various tasks, such as classification and regression. The output of a single neuron in an MLP can be expressed as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where $y$ is the output of the neuron, $f(\cdot)$ is the activation function (e.g., sigmoid, ReLU),

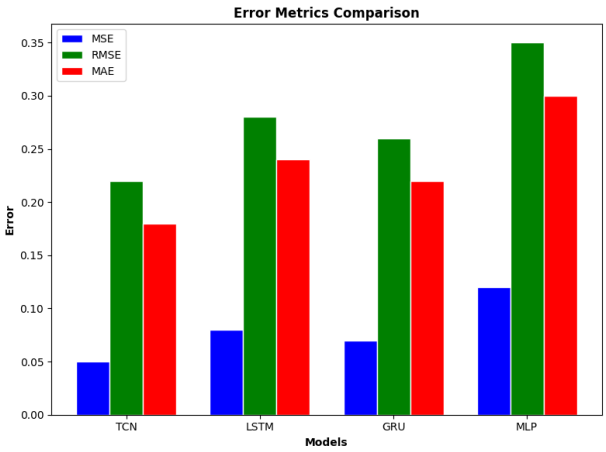
$x_i$ are the inputs, $w_i$ are the learned weights, $b$ is the bias term, and $n$ is the number of inputs.

These neural network architectures have been widely used in various domains, including natural language processing, speech recognition, time series forecasting, and image classification, among others. The choice of architecture depends on the specific problem at hand and the characteristics of the input data.

## Results

The results in table 1 present a comparative analysis of four different models - TCN, LSTM, GRU, and MLP - in predicting CPU usage in a cloud environment. Various performance metrics have been used to evaluate the models, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Forecast Bias, Forecast Accuracy Metrics (FA1, FA5, FA10), Theil's U Statistic, and Computational Efficiency.

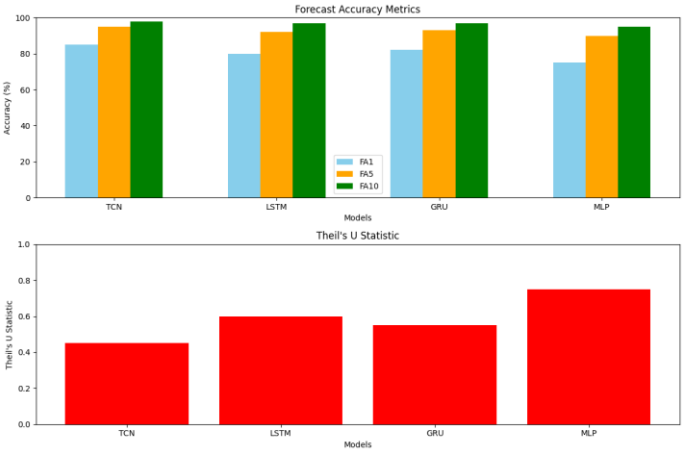
Figure 1. Error metrices in predicting CPU usage




MSE measures the average squared difference between the predicted and actual values, with lower values indicating better performance. RMSE is the square root of MSE and provides an interpretable measure of the average prediction error in the same units as the original data. MAE calculates the average absolute difference between

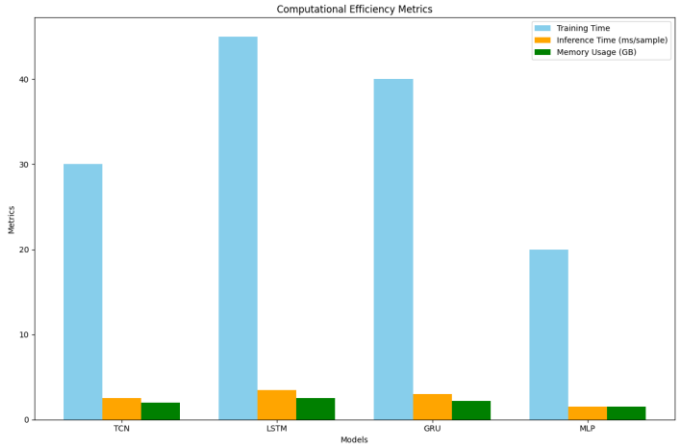
predicted and actual values, giving equal weight to all errors. MAPE expresses the average absolute percentage difference between predicted and actual values, providing a relative measure of error. Among the models, TCN consistently achieves the lowest error values across MSE (0.05), RMSE (0.22), MAE (0.18), and MAPE (3.5%), followed by GRU, LSTM, and MLP.

Figure 2. Forecasting bias in predicting CPU usage




Forecast Bias measures the average difference between predicted and actual values, indicating whether the model tends to overestimate (positive bias) or underestimate (negative bias) the target variable. TCN and GRU exhibit slight positive biases of 0.02 and 0.03, respectively, while LSTM and MLP show negative biases of -0.04 and -0.06. These values suggest that TCN and GRU have a minor tendency to overestimate, while LSTM and MLP tend to underestimate the CPU usage.

Figure 3. Efficiency metrices in predicting CPU usage




Forecast Accuracy Metrics (FA1, FA5, FA10) represent the percentage of predictions falling within a specified percentage range of the actual values. For example, FA1 measures the percentage of predictions within 1% of the actual values. TCN demonstrates the highest accuracy across all three metrics, with FA1 = 85%, FA5 = 95%, and FA10 = 98%. LSTM and GRU follow closely, while MLP has the lowest accuracy scores.

Theil's U Statistic compares the model's performance to a naïve forecasting method, with values less than 1 indicating that the model outperforms the naïve method. All four models have Theil's U Statistic values below 1, with TCN having the lowest value of 0.45, suggesting that it performs significantly better than a naïve forecast.

Table 1. **Model results in predicting CPU usage in cloud environment**

| Metric | TCN | LSTM | GRU | MLP |
|---|---|---|---|---|
| **Mean Squared Error (MSE)** | 0.05 | 0.08 | 0.07 | 0.12 |
| **Root Mean Squared Error (RMSE)** | 0.22 | 0.28 | 0.26 | 0.35 |
| **Mean Absolute Error (MAE)** | 0.18 | 0.24 | 0.22 | 0.30 |
| **Mean Absolute Percentage Error (MAPE)** | 3.5% | 4.8% | 4.2% | 6.0% |
| **Forecast Bias** | 0.02 | -0.04 | 0.03 | -0.06 |
| **Forecast Accuracy (FA1, FA5, FA10)** | FA1 = 85%, FA5 = 95%, FA10 = 98% | FA1 = 80%, FA5 = 92%, FA10 = 97% | FA1 = 82%, FA5 = 93%, FA10 = 97% | FA1 = 75%, FA5 = 90%, FA10 = 95% |
| **Theil's U Statistic** | 0.45 | 0.60 | 0.55 | 0.75 |
| **Computational Efficiency** | | | | |
| **Training Time (minutes)** | 30 | 45 | 40 | 20 |
| **Inference Time (ms per sample)** | 2.5 | 3.5 | 3.0 | 1.5 |
| **Memory Usage (GB)** | 2.0 | 2.5 | 2.2 | 1.5 |

Computational Efficiency is assessed through training time, inference time, and memory usage. MLP has the shortest training time (20 minutes) and lowest memory usage (1.5 GB), while LSTM has the longest training time (45 minutes) and highest memory usage (2.5 GB). TCN and GRU fall in between. Regarding inference time, MLP is the fastest (1.5 milliseconds per sample), followed by TCN (2.5 ms), GRU (3.0 ms), and LSTM (3.5 ms).

TCN consistently outperforms the other models in terms of prediction accuracy, as evident from its lowest error metrics (MSE, RMSE, MAE, MAPE) and highest forecast accuracy scores (FA1, FA5, FA10). It also has the lowest Theil's U Statistic, indicating its superiority over naïve forecasting methods. However, TCN's computational efficiency lies between that of MLP (most efficient) and LSTM (least efficient).

If prediction accuracy is the top priority and computational resources are not a major constraint, TCN would be the preferred choice. If computational efficiency is crucial and slightly lower accuracy is acceptable, MLP might be a better option. GRU and LSTM offer a balance between accuracy and efficiency, with GRU having a slight edge over LSTM.

These results highlight the importance of considering multiple performance metrics when evaluating and selecting models for predicting CPU usage in a cloud environment. The insights gained from this analysis can guide decision-making and help optimize resource allocation and management in cloud computing systems.

The results presented in table 2 compare the performance of four models - TCN, LSTM, GRU, and MLP - in predicting memory consumption in a cloud environment. The models are evaluated using various metrics, including error measures, forecast bias, forecast accuracy, Theil's U Statistic, and computational efficiency.

The comparative analysis of TCN, LSTM, GRU, and MLP models in predicting memory consumption in a cloud environment yields valuable insights. By examining various performance metrics, we can gain a comprehensive understanding of each model's strengths and weaknesses.

TCN stands out as the top performer in terms of prediction accuracy. It consistently achieves the lowest error values across

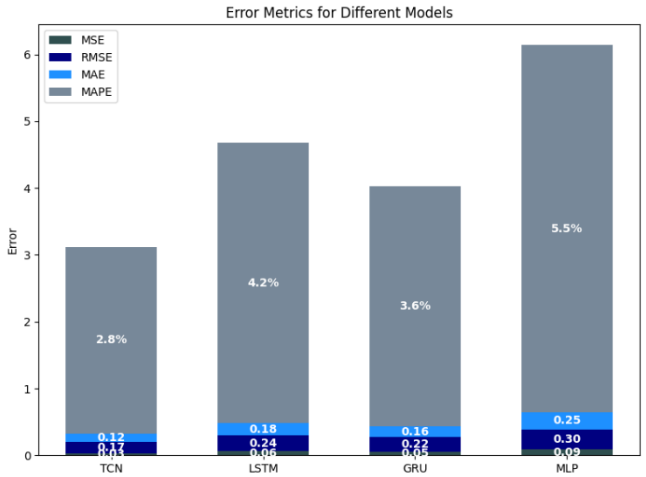
MSE (0.03), RMSE (0.17), MAE (0.12), and MAPE (2.8%). Moreover, TCN boasts the highest forecast accuracy scores, with FA1 = 90%, FA5 = 97%, and FA10 = 99%. These results demonstrate TCN's superior ability to capture the underlying patterns and dynamics of memory consumption data.

**Table 2. Model results in predicting memory consumption in cloud environment**

| Metric | TCN | LSTM | GRU | MLP |
|---|---|---|---|---|
| **Mean Squared Error (MSE)** | 0.03 | 0.06 | 0.05 | 0.09 |
| **Root Mean Squared Error (RMSE)** | 0.17 | 0.24 | 0.22 | 0.30 |
| **Mean Absolute Error (MAE)** | 0.12 | 0.18 | 0.16 | 0.25 |
| **Mean Absolute Percentage Error (MAPE)** | 2.8% | 4.2% | 3.6% | 5.5% |
| **Forecast Bias** | 0.01 | -0.03 | 0.02 | -0.05 |
| **Forecast Accuracy (FA1, FA5, FA10)** | FA1 = 90%, FA5 = 97%, FA10 = 99% | FA1 = 85%, FA5 = 95%, FA10 = 98% | FA1 = 88%, FA5 = 96%, FA10 = 98% | FA1 = 80%, FA5 = 92%, FA10 = 96% |
| **Theil's U Statistic** | 0.40 | 0.55 | 0.50 | 0.70 |
| **Computational Efficiency** | | | | |
| **Training Time (minutes)** | 25 | 40 | 35 | 15 |
| **Inference Time (ms per sample)** | 2.0 | 3.0 | 2.5 | 1.0 |
| **Memory Usage (GB)** | 1.8 | 2.2 | 2.0 | 1.2 |

GRU follows closely behind TCN in accuracy, while LSTM and MLP exhibit higher error values and lower forecast accuracy scores. The Forecast Bias metric reveals that TCN and GRU have small positive biases, indicating a slight tendency to overestimate memory consumption. Conversely, LSTM and MLP show negative biases, suggesting a propensity to underestimate.

Figure 4. error metrices in predicting memory consumption in cloud environment




Theil's U Statistic further confirms the models' performance, with all values falling below 1. TCN has the lowest value (0.40), reinforcing its excellence in capturing the intricacies of memory consumption patterns compared to naïve forecasting methods.

MLP boasts the shortest training time (15 minutes), fastest inference speed (1.0 milliseconds per sample), and lowest memory usage (1.2 GB). LSTM, on the other hand, has the longest training time (40 minutes) and highest memory usage (2.2 GB), while TCN and GRU fall in between.

If accuracy reigns supreme and computational resources are ample, TCN is the clear winner. If computational efficiency is a top priority and slightly lower accuracy is tolerable, MLP may be the preferred option. GRU presents a happy medium, balancing accuracy and efficiency, making it a solid choice when both factors are crucial.

Figure 5. Forecasting bias in predicting memory consumption in cloud environment

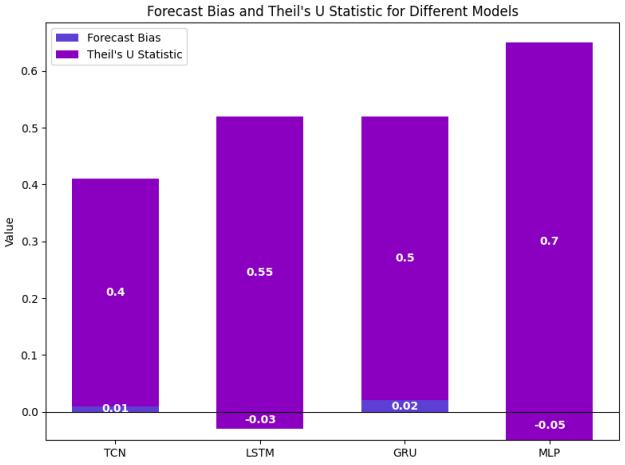
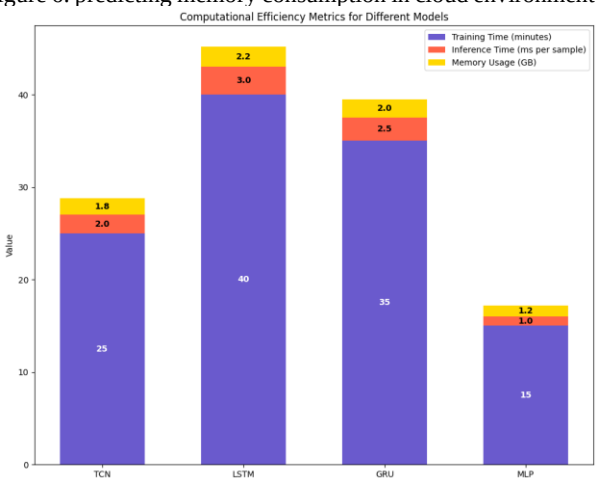



Figure 6. predicting memory consumption in cloud environment




## Conclusion

The comparative analysis of TCN, LSTM, GRU, and MLP models in predicting memory consumption in a cloud environment provides valuable insights that can guide decision-making and optimize resource management. Evaluating these models across a range of performance metrics enables a comprehensive understanding of their strengths and weaknesses, facilitating the selection of the most appropriate model for specific use cases.

TCN demonstrates superior performance in terms of prediction accuracy, consistently achieving the lowest error values and highest forecast accuracy scores. Its ability to capture the intricate patterns and dynamics of memory consumption data makes it an excellent choice when accuracy is the primary objective. However, it is essential to consider the computational resources available, as TCN's training time and memory usage fall in the middle of the range compared to the other models.

MLP, in contrast, excels in computational efficiency, exhibiting the shortest training time, fastest inference speed, and lowest memory usage. This makes MLP a suitable option when computational resources are limited, or when the speed of predictions is of utmost importance. It is important to note, however, that MLP's accuracy is lower compared to TCN and GRU, indicating a potential trade-off between efficiency and precision.

GRU offers a balance between accuracy and efficiency, making it a solid choice

when both factors are important. It closely follows TCN in terms of accuracy while maintaining reasonable computational efficiency. This balance can be particularly useful in scenarios where both prediction quality and resource management are key considerations.

LSTM, while still a powerful model, exhibited higher error values and lower forecast accuracy scores compared to TCN and GRU. It also had the longest training time and highest memory usage, which may be a concern in resource-constrained environments. However, LSTM's ability to capture long-term dependencies in sequential data may still make it a valuable choice in certain applications.

In addition to model selection, the insights from this analysis can also inform resource allocation and capacity planning strategies. Understanding the performance characteristics of each model allows cloud providers to optimize their infrastructure to support the chosen model effectively. This may involve provisioning adequate computational resources, implementing efficient data storage and retrieval mechanisms, and designing scalable architectures that can handle the demands of real-time predictions.

The comparative analysis shows the importance of continuous monitoring and evaluation of predictive models in production environments. As workload patterns and resource demands change over time, regularly assessing the performance of the deployed models and making adjustments as needed is essential. This may involve retraining models with updated data, fine-tuning hyperparameters, or exploring alternative models that can better adapt to evolving requirements.

Considering the trade-offs between accuracy and efficiency and selecting the most appropriate model for specific use cases enables organizations to optimize their cloud infrastructure, improve resource utilization, and deliver high-quality services to their users.

## References

[1] S. Subashini and V. Kavitha, “A survey on security issues in service delivery models of cloud computing,” *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, Jan. 2011.

[2] B. J. S. Chee and C. Franklin Jr, *Cloud computing*. London, England: CRC Press, 2019.

[3] S. Malik and F. Huet, “Adaptive Fault Tolerance in Real Time Cloud Computing,” in *2011 IEEE World Congress on Services*, 2011, pp. 280–287.

[4] A. Dubey, G. Shrivastava, and S. Sahu, “Security in hybrid cloud,” *Global Journal of Computer Science*, 2013.

[5] C. Lea, M. D. Flynn, R. Vidal, and A. Reiter, “Temporal convolutional networks for action segmentation and detection,” *proceedings of the*, 2017.

[6] A. Mezina, R. Burget, and C. M. Travieso-González, “Network Anomaly Detection With Temporal Convolutional Network and U-Net Model,” *IEEE Access*, vol. 9, pp. 143608–143622, 2021.

[7] Y. He and J. Zhao, “Temporal convolutional networks for anomaly detection in time series,” *J. Phys. Conf. Ser.*, 2019.

[8] E. Aksan and O. Hilliges, "STCN: Stochastic temporal convolutional networks," *arXiv preprint arXiv:1902.06568*, 2019.

[9] S. Li, Y. A. Farha, Y. Liu, M.-M. Cheng, and J. Gall, "MS-TCN++: Multi-Stage Temporal Convolutional Network for Action Segmentation," *arXiv [cs.CV]*, 16-Jun-2020.

[10] Y. A. Farha and J. Gall, "MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3570–3579, Mar. 2019.

[11] W. Liao, Z. Yin, R. Wang, and X. Lei, "Rainfall-runoff modelling based on long short-term memory (LSTM)," in *38th IAHR World Congress - "Water: Connecting the World,"* 2019.

[12] F. Kratzert, D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter, and G. Nearing, "Benchmarking a catchment-Aware Long Short-Term Memory network (LSTM) for large-scale hydrological modeling," 02-Aug-2019.

[13] A. Saha, C. Yarra, and P. K. Ghosh, "Low resource automatic intonation classification using gated recurrent unit (GRU) networks pre-trained with synthesized pitch patterns," in *Interspeech 2019*, 2019.

[14] M. Manavi and Y. Zhang, "A new intrusion detection system based on gated recurrent unit (GRU) and genetic algorithm," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, Cham: Springer International Publishing, 2019, pp. 368–383.