International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Workload Characterization Prediction through Hybrid model

Aniket Dattatrey Deshmukh[1], Dr N. S. Bagal[2]

[1,2]Department of Computer Engineering, PVPIT Bavdhan, Pune

Abstract

An all-inclusive approach to workload characterisation that makes use of supervised and unsupervised deep learning methods to optimize system performance and resource usage. In order to uncover latent workload patterns, the suggested method combines data preprocessing with feature selection using Pearson correlation and grouping through Fuzzy C-Means. For precise prediction, a CNN-LSTM hybrid model is used to grasp the temporal and geographical dependencies in the workload data. The approach works well with dynamic, high-dimensional datasets that are typical in distributed and cloud-based settings. Optimal work allocation and balanced resource distribution are also achieved by use of the Hungarian algorithm. Clustering and deep learning, when combined, increase the precision of predictions and the flexibility of the system. Efficient workload scheduling and intelligent decision-making are supported by this system. Results from experiments show that, compared to more conventional approaches, this one is more scalable and performs better overall. Data centers, edge systems, and cloud computing are all good places to use the suggested system.

Keywords: Workload Characterization, Performance Analysis, Feature Extraction, Performance Analysis, Machine learning (ML).

I. INTRODUCTION

The complexity and variety of computational workloads have been greatly amplified in recent years due to the proliferation of data-intensive applications, cloud computing, and distributed systems. For better system performance, appropriate scheduling, and resource management, workload characterization—the process of studying and comprehending the behavior of system workloads—is essential. Traditional methods of workload characterisation, which relied on static analysis and linear models, are no longer enough to deal with the growing complexity and variety of data in modern, large-scale systems like data centers and edge computing platforms.

It is difficult to properly assess and forecast the behavior of modern workloads using traditional statistical methods since they are frequently multi-dimensional, non-linear, and time-dependent. Complex patterns and hidden relationships within the data must be captured by advanced modeling techniques in order to handle these workloads, which exhibit both geographical and temporal interdependence. Consequently, smart and flexible approaches that can handle massive amounts of data and deliver precise workload insights are in high demand.

Because of its capacity to automatically learn hierarchical representations and extract relevant features, deep learning has become a potent tool for dealing with complicated data patterns. Both supervised and unsupervised learning methods have its uses; the former allows for precise workload classification and

prediction using labeled data, while the latter aids in the discovery of previously unknown structures and patterns. Combining supervised and unsupervised methods improves forecast accuracy and system efficiency by providing a more complete picture of workload dynamics.

In order to overcome the shortcomings of conventional methods, this research suggests a hybrid approach to workload characterisation that integrates supervised and unsupervised deep learning strategies. The first step of the methodology is data preparation and feature selection, which involves utilizing Pearson correlation analysis to eliminate features that are not important or redundant. By narrowing the focus to the most important attributes, computational complexity is reduced and model performance is improved.

In order to group comparable workload patterns based on their features, an unsupervised clustering technique called Fuzzy C-Means (FCM) is employed after feature selection. To better reflect workload behavior, FCM permits data points to belong to numerous clusters with different degrees of membership, in contrast to conventional clustering algorithms. For effective scheduling and resource allocation, the clustering process is crucial for discovering latent structures and grouping comparable workloads.

Using a combination of CNN and LSTM networks, a hybrid deep learning model is implemented to further improve prediction accuracy. While LSTM excels at capturing sequential patterns and temporal dependencies, CNN excels at extracting spatial features from input data. The combination of these two models improves the system's prediction capabilities by teaching it to understand the geographical and temporal aspects of workloads.

System performance is heavily dependent on workload prediction and efficient resource allocation. In order to solve this problem, the suggested approach uses the Hungarian algorithm to assign tasks optimally. By distributing workloads uniformly across available resources, this technique reduces processing delays and maximizes system efficiency. The suggested approach covers all bases when it comes to workload management and characterisation by integrating optimization, deep learning, and clustering.

Among the many benefits of the suggested method are its increased scalability, higher adaptation to changing surroundings, and more accurate predictions. It works well in environments where effective management of workloads is crucial to the upkeep of system performance and dependability, such as cloud computing, data centers, and edge systems. In addition, the methodology's modular nature makes it easy to include new techniques and upgrades in the future.

To sum up, contemporary computing systems rely on workload characterization, and a good way to tackle current problems is with hybrid deep learning methodologies. The suggested approach is a strong contender for complicated and ever-changing workload settings because it uses a mix of supervised and unsupervised learning approaches to strike a balance between precision, efficiency, and flexibility.

A tool for analyzing and categorizing workloads, the Deep Learning Workload Analysis Tool (DLWAT) is presented by Hu et al. [1] On the SPEC CPU2006 and CPU2017 benchmarks, it applies deep learning algorithms, both supervised and unsupervised. When comparing supervised models, CRNN outperforms CNN and RNN in terms of accuracy. Compared to conventional k-means, deep clustering performs better in unsupervised learning. Classification accuracy is enhanced as a result of the models' ability to accurately capture latent feature representations. At the moment, benchmark workloads are DLWAT's main focus. Bringing it to more practical uses, such as gaming workloads, is something we want to do in the future. It has additional potential as a portable library with wider software analysis uses.

Using AIBench as a benchmark for artificial intelligence workloads and PassMark as a benchmark for general computing tasks, N. Sibai et al.[2] examines performance indicators. Important hardware events

that impact CPI and execution time are identified. Particularly during training, AIBench workloads display high levels of CPU utilization, memory usage, and microarchitectural activity. Memory, CPU cores, and caches are just a few of the hardware components that different benchmarks put the spotlight on. The most effective machine learning models for predicting CPI and elapsed time are neural networks and linear regression. Performance prediction and the ability to differentiate between AI and non-AI workloads are both aided by this method. It can be useful for optimizing hardware, planning cloud capacity, and designing systems. Porting the analysis to different CPUs and mobile platforms is something that needs doing in the future.

The authors ISMAYLOVNA et al.[3] present a strong end-to-end multimodal framework to solve important problems in MW categorization. The overheads of storage and GPU loading are eliminated through on-the-fly scalogram synthesis using FP16 arithmetic, allowing efficient processing across MAUS, CLAS, and WESAD datasets. In the face of physiological inputs like PPG that are noisy, CNN-LSTM branches with attention and dynamic dropout provide feature extraction that is resilient. Gating networks based on molecular dynamics (MoE) overcome the inflexibility of static fusion techniques by adaptively merging modalities according to their real-time informativeness. Achieving an impressive 92%-98% accuracy, CogniMoE surpasses previous approaches (70%-85%) and showcases impressive cross-dataset and cross-sensor generalizability.

The second part of this report examines the prior research that was deemed a Litereature Survey. The proposed methodology, which lays out the entire plan of action, is detailed in Section 3. The experimental evaluation is covered in Part 4, possible modifications are discussed in Section 5, and the essay concludes with a conclusion on the existing plan.

II. Literature Survey

Feature augmentation using multiple sensors has been shown to improve classification accuracy and data quality, as proposed by Z. Muke et al. [4]. The study highlights that adding informative features enhances model generalizability and reduces computational complexity while achieving better performance on multimodal datasets. Speech-based workload prediction has also gained attention, where Sandoval et al. [5] utilize transfer learning and multichannel CNNs to improve prediction accuracy without increasing dataset size or model complexity. Their approach demonstrates better efficiency compared to baseline models. Time-series analysis plays a crucial role in workload prediction, as demonstrated by Amekraz [6], where CANFIS combined with preprocessing techniques outperforms traditional models such as LSTM, SVM, and ARIMA in terms of accuracy and training time. Similarly, hybrid deep learning models integrating CNN and Bi-LSTM, as proposed by Alqahtani et al. [7], effectively capture workload dynamics even in noisy environments.Workload prediction using physiological signals has also been explored, where JOHN et al. [8] analyze EEG, HRV, and eye movement data to estimate workload and system performance. Their findings confirm that physiological indicators can reliably predict cognitive workload.Advanced prediction frameworks such as OmniFORE by Gort et al. [9] utilize attention-based time-series modeling to capture both short-term and long-term workload patterns, improving prediction accuracy and adaptability in edge-cloud systems. Ensemble-based approaches like TWD-RCPM proposed by Shi et al. [10] further enhance prediction performance in heterogeneous environments.

Deep learning-based workload prediction models, including the GRU-SWSLD model proposed by Alqahtan et al. [11], improve convergence and accuracy while efficiently handling high-dimensional data. Distributed learning techniques such as gossip learning, discussed by TUNDO et al. [12], provide scalable

solutions for decentralized edge environments. Hardware-level optimizations, such as the SIMD-based architecture proposed by Sukumaran et al. [13], significantly improve performance for data-parallel workloads, particularly in AI and embedded systems. Additionally, workload characterization models considering human-system interaction, as presented by Olguín Muñoz et al. [14], enhance prediction accuracy by modeling user behavior and latency effects. Efficient scheduling techniques have also been widely studied. Wang et al. [15] propose a multi-cluster scheduling framework that improves resource utilization and reduces execution delays. Similarly, GUO et al. [16] introduce delay-optimal scheduling methods that enhance system efficiency and reduce job completion time.

Reinforcement learning-based scheduling approaches, such as the method proposed by Park et al. [17], optimize workload distribution and energy efficiency in dynamic environments. Optimization-based scheduling for GPU systems by FILIPPIN et al. [18] demonstrates significant cost reduction and scalability improvements.Energy-aware workload scheduling using renewable energy sources, as discussed by Liu et al. [19], improves sustainability in data centers. Resource optimization in high-performance computing systems, proposed by Khetawat et al. [20], enhances scalability and execution efficiency through dynamic resource sharing.Further advancements include deep reinforcement learning-based workload placement strategies by ZHOU et al. [21], which optimize both energy consumption and execution time. FPGA-based workload scheduling frameworks such as H3M by ZENG et al. [22] improve performance for multi-DNN workloads.

Emerging approaches like intent-driven scheduling using large language models, proposed by Sliwko et al. [23], demonstrate the potential for intelligent workload orchestration. Multi-objective workload optimization using deep reinforcement learning, as presented by Safavifar et al. [24], improves resource utilization in edge environments.Autoscaling strategies for cloud systems, studied by QUATTROCCHI et al. [25], enhance performance under uncertain workloads, while improved scheduling frameworks for Kubernetes proposed by Robles-Enciso et al. [26] address limitations in handling heterogeneous resources.Hardware acceleration techniques such as NetStore by Kim et al. [27] improve storage performance and scalability. Cognitive workload classification using machine learning, as demonstrated by GUAN et al. [28], achieves high accuracy across different tasks. Multi-agent scheduling frameworks proposed by I. Khan et al. [29] improve adaptability and efficiency in cloud systems.Finally, multimodal deep learning architectures such as CogniMoE proposed by ISMAYLOVNA et al. [30] achieve state-of-the-art performance in workload classification, demonstrating high accuracy and strong generalization across datasets.

III PROPOSED METHODOLOGY

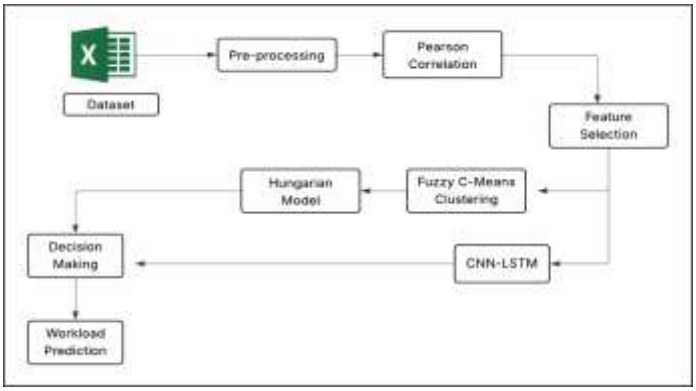

Figure 1: Overview Diagram

As shown in Figure 1, the proposed workload prediction and characterisation model seeks to optimize cloud task scheduling through the analysis of data generated by machine learning and deep learning algorithms. Accurate workload prediction is achieved by following a well-defined procedure that begins with information collection and continues through preprocessing, correlation analysis, feature selection, and model training utilizing a CNN-LSTM architecture.

Step 1: Dataset Collection

The suggested model gets its dataset from the Kaggle repository, which is utilized for workload prediction: Here is a link to a dataset for scheduling analysis: https://www.kaggle.com/datasets/zoya77/cloud-workload-dataset.

This dataset represents real-time workload scheduling situations and contains five thousand distinct records of tasks being executed in a cloud computing environment. Its purpose is to assess how well a system is doing and to find the best way to distribute resources when things are always changing. For high-dimensional workload analysis, the data is a good fit because it looks like logs from cloud-based systems and IoT devices.

Task execution and system behavior are described by numerous attributes in the dataset. You can't evaluate workload efficiency without these performance measurements, which include things like CPU usage, memory consumption, task execution time, system throughput, and task waiting time. You may learn more about the system's performance in different scenarios by looking at other characteristics like the error rate, number of active users, and network bandwidth consumption.

Task_Start_Time and Task_End_Time denote execution intervals, while Job_ID uniquely identifies each task. Additional categorical features in the dataset include Job Priority, Data Source, Scheduler Type, and Resource Allocation Type. Here, Scheduler_Type stands in for several scheduling algorithms like FCFS, Round Robin, and Priority-based scheduling; it's the target variable.

The dataset is well-suited for building a strong workload prediction model because it includes system-level measurements and scheduling information, which together reflect patterns of resource use and scheduling behavior in cloud systems.

Step 2: Pre-processing

After the dataset has been acquired, it is subjected to the preprocessing stage. The dataset is loaded into a structured tabular format using the Python Pandas package. Statistical measurements including the mean, standard deviation, and distribution of attributes are calculated to obtain a basic grasp of the dataset.

Dataset attributes, which might take on numerical or categorical values, are subsequently analyzed according to their data types. The purpose of data cleaning is to eliminate errors, duplicates, and noise from a dataset. By doing so, we check that the data is accurate and ready for processing. Using encoding techniques, we may transform categorical attributes like Job_Priority, Data_Source, Resource_Allocation_Type, and Scheduler_Type into numerical form. The machine learning model is ow better equipped to handle categorical data thanks to this change. Different scheduling strategies are represented numerically by encoding the target variable Scheduler_Type into separate classes.

Next, we look for class imbalances by analyzing the dataset distribution. To make sure that every class is adequately represented throughout training, balancing techniques are used when imbalance is found. Improving data quality, ensuring consistency, and getting the information ready for accurate analytical processing are all goals of this preprocessing stage.

This preprocessing stage enhances data quality, ensures consistency, and prepares the dataset for accurate analytical processing.

Step 3: Pearson Correlation

Once the preprocessing is done, Pearson's correlation is used to examine the entire dataset and find the correlations between the various attributes. Measuring the linear dependency between each feature and the target variable is a crucial first step in determining how relevant each feature is.

The Pearson correlation coefficient is calculated using Equation (1):

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (1)$$

Where,

x_i represents the values of independent variables,

y_i represents the values of the dependent variable,

$\bar{x}$ and $\bar{y}$ denote the mean values of the respective variables.

In order to show the level of association between qualities, this technique is used to build a correlation matrix. Redundant features are those having strong correlations between them, while features with weak correlations to the target variable are downplayed.

The dimensionality of the dataset is decreased by eliminating features that are not significant or redundant. This ensures that only relevant properties are kept for subsequent processing and boosts computing efficiency.

Step 4: Feature Selection

Here, we use the linked dataset to pick out the most important attributes for workload prediction. By rescaling the data to a predetermined range, usually between 0 and 1, the MinMaxScaler approach normalizes the data and guarantees that all features contribute uniformly.

Equations 2 and 3 define the normalization procedure as follows:

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}} \dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

$$x_{scaled} = x_{std} \times (max - min) + min \dots\dots\dots (3)$$

The learning process is protected from feature scale fluctuations by this modification. To rank the features in order of relevance, after normalization, feature selection approaches including score-based ranking methods are used.

To select only the best features, we rank them according to the amount of value they add to the target variable. This enhances prediction accuracy while reducing computational complexity and redundancy.

The model training stage receives the optimized feature set and processes it further.

Step 5: CNN-LSTM Model Training

This stage involves training a CNN-LSTM hybrid model with the chosen features for classification and workload prediction. Using the train_test_split() method, the processed dataset is divided into training and testing sets to guarantee the model is evaluated correctly.

The dataset is transformed into a three-dimensional format known as (samples, timesteps, features) by applying feature scaling using MinMaxScaler. This format is well-suited for architectures that employ deep learning.

The Conv1D layer of the convolutional neural network (CNN) is used to extract geographical information and locate significant local patterns in the data from the workload. Then, to make the calculation more efficient and lower the dimensionality, a MaxPooling layer is implemented.

After that, the characteristics that were collected are sent to the LSTM layer. This layer uses its memory cells and gating mechanisms to learn long-term correlations in workload patterns and to capture temporal dependencies. Thus, the model is able to assess sequential behavior in cloud workloads well.

Additionally, high-level feature learning is done using a fully linked Dense layer, and overfitting is prevented and generalization is improved with the help of a Dropout layer. In order to sort the Scheduler_Type into various groups, the last output layer employs the Softmax activation function.

The model undergoes training throughout several iterations, and its efficacy is assessed by means of measures like loss and accuracy. Better workload prediction and efficient scheduling optimization are outcomes of this hybrid CNN-LSTM architecture's successful combination of spatial and temporal learning.

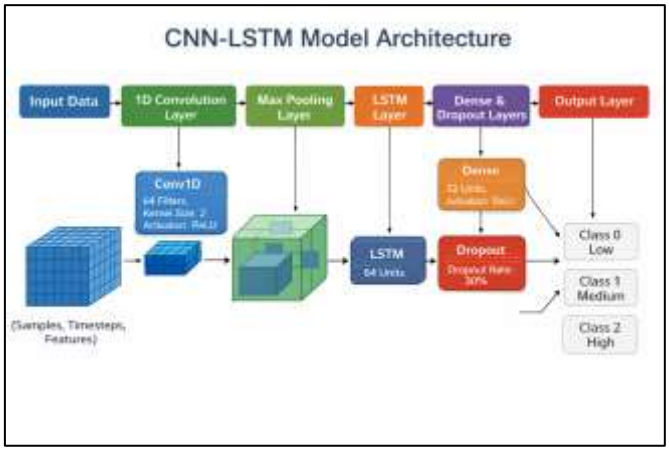

Fig. 2. CNN-LSTM Model Architecture for Workload Prediction

The suggested convolutional neural network-long short-term memory (CNN-LSTM) model for cloud computing workload prediction and scheduling optimization is shown in Figure 2. In order to accurately represent the workload dataset's spatial and temporal patterns, the model incorporates layers of convolutional and recurrent neural networks.

Preprocessed and normalized data organized in a three-dimensional manner denoted as (samples, timesteps, features) makes up the model's input. The model is able to efficiently process sequential workload data thanks to this format.

The architecture's initial layer, the 1D Convolutional (Conv1D) layer, is in charge of obtaining input data's local spatial properties. The model's convolutional layer employs the ReLU activation function and 64 filters with a 2 kernel size. The feature correlations and significant patterns in the workload data are discovered by this layer.

To lower the feature maps' dimensionality, a Max Pooling layer is added after the convolutional layer. Data downsampling and overfitting prevention are both helped by this layer's ability to identify and use the most important features.

The pooling layer's output is subsequently sent to the 64-unit LSTM (Long Short-Term Memory) layer. The long short-term memory (LSTM) layer may recognize sequential linkages and dependencies in the data that occur over time. It stores significant data in memory and gets rid of irrelevant data using gating mechanisms and memory cells. Because of this, it is great for studying patterns of workload that vary with time.

A fully-connected Dense layer with 32 units and ReLU activation follows the LSTM layer in the model. This layer converts the retrieved features into a classification-ready format and does high-level feature learning.

Applying a Dropout layer with a 30% dropout rate enhances generalization and prevents overfitting. To prevent the model from becoming overly dependent on any one feature, this layer randomly deactivates some neurons throughout training.

Lastly, the design incorporates an Output layer that uses Softmax activation to divide the burden into three groups:

- Class 0: Low workload
- Class 1: Medium workload
- Class 2: High workload

The model is able to classify each input sample into the most likely workload class thanks to the Softmax function, which turns the output into probability values.

Improved prediction accuracy and efficient workload classification in cloud environments are overall outcomes of the CNN-LSTM architecture's successful combination of spatial feature extraction and temporal sequence learning.

A. Activation Functions

Activation functions are very important for the CNN-LSTM model as they introduce non-linearity and control how the neurons learn complex patterns from the workload data. Different activation functions are used at different layers of the model to improve learning efficiency and prediction accuracy.

The Rectified Linear Unit (ReLU) activation function is used in the convolutional and dense layers of the model. It is defined in equation 4:

$$f(x) = max(0, x) \dots\dots\dots\dots\dots\dots\dots\dots (4)$$

This function converts all negative values to zero while keeping positive values unchanged. It helps in faster convergence and reduces the vanishing gradient problem, making it highly suitable for deep learning models.

The Softmax activation function is used in the output layer for multi-class classification. It converts the output values into a probability distribution across different classes. It is defined in equation 5s:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \dots\dots\dots\dots\dots\dots\dots\dots (5)$$

where $z_i$represents the input to the neuron and $K$is the total number of output classes. This function ensures that the sum of all output probabilities equals one, allowing accurate classification of workload into different scheduler types.

The sigmoid activation function is used in the LSTM layer for controlling the gating mechanisms such as input gate, forget gate, and output gate. It is defined in equation 6:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \dots\dots\dots\dots\dots\dots\dots (6)$$

This function produces output values between 0 and 1, enabling the network to decide which information should be retained or discarded during training.

The tanh activation function is also used in the LSTM layer to regulate the cell state. It is defined in equation 7:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \dots\dots\dots\dots\dots\dots\dots (7)$$

This function generates output values between -1 and 1, which helps in maintaining stable gradients and capturing long-term dependencies in sequential workload data.

Together, these activation functions enhance the learning capability of the CNN-LSTM model and improve its overall prediction performance.

Step 6: Generating Hungarian Neural network (H-net) training Data-

In the previous step, we established that after training on the cloud load dataset, the LSTM model produced a set of projected load parameter lists. For the purpose of allocating resources, the Hungarian neural network employs this. One Hungarian approach to solving the assignment problem using deep learning is called the Hungarian Network (Hnet). More deep learning challenges requiring permutation invariant training (PIT) can be tackled with this approach by utilizing a deep learning neural network. In this way, PIT is completely unnecessary for training the deep learning tasks. Applications of deep learning, such as source separation and multi-source localization, necessitate permutation invariant training. The following data must be generated using the load predicted values obtained from the LSTM model in the previous phase in order to train the Hnet for multi-source localization. In order to train the Hnet, we construct a dataset that is separated into training sets of specified distance matrices D and their corresponding association matrices A. The validation process takes up 10% of the total training split size. The fixed dimensions of D and A, which are identical, are determined by the greatest number in the dataset, Nmax = 2. We sample an equal number of D matrices by randomly selecting reference and forecast load Direction of arrivals from spherical equiangular grids with resolutions of 1, 2, 3, 4, 5, 10, 15, 20, and 30 degrees. Every conceivable combination of (number of predictions, number of reference) is evenly represented in the dataset, including (0,0), (0,1), (1,0), (1,1), (1,2), (2,1), and (2,2). Equation 10 shows how the distance pairs in D are produced using the standard formula for geometric distances.

$$ED=(x1-x2)2+(y1-y2)2\_\_\_\_\_ \quad (10)$$

Where,

ED- Euclidean distance of a specific row.

x1, x2, y1, and y2 are the labeled value of load direction of arrivals(DoA)

To make it easier for Hnet to establish the exact number of active load arrivals and their relationships, the associated inactive entries are allocated random high distance values as a result of padding D to Nmax _Nmax dimensions, even when Mt;Nt < Nmax. Consequently, after training, Hnet achieves an F-score greater than 99% on any D data that is generated according to the aforementioned parameters.

Step 7: H-net Training - The suggested differentiable tracking training method relies on Hnet as its foundation. It takes the input distance matrix D and uses it to estimate the association matrix {A, which is the same for each dimension. In contrast to the deep Hungarian network, we opted for a simpler design

with three losses to train Hnet efficiently (see Fig. 5). We use an input layer with 128 gated recurrent units (GRUs) that takes the time-sequence and feature length dimensions from the input matrix D, respectively. By feeding the GRU output time-sequence into a single-head self-attention network, we can identify the time steps with the correct associations. As a multiclass, multilabel classification task, a fully-connected network with sigmoid non-linearity processes the output of the self-attention layer and calculates.

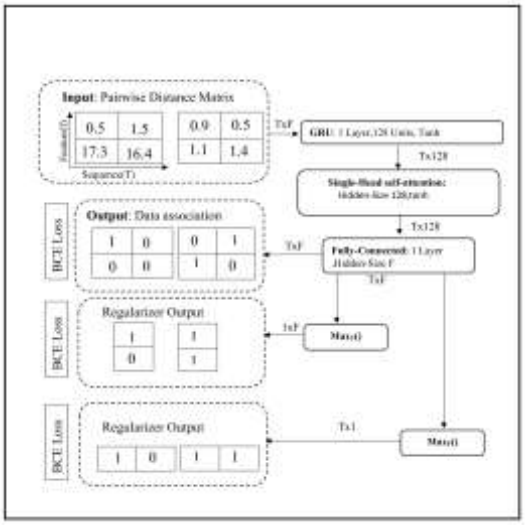

Figure 3: Block Diagram of Hungarian network

In addition, before the sigmoid non-linearity is applied to compute "A," we maximize the output of the fully-connected network along the temporal (maxT()) and feature (maxF()) axes. This tells the network to forecast no more than one association per row and column, as is typical for associations generated by the Hungarian algorithm. We use sigmoid non-linearity to their outputs since an output instance may have many classes. In a multi-task framework, the Hnet is trained using weighted combinations of the three losses. These losses are computed using binary cross-entropy between the predictions and the objective labels of A, maxT(A), and maxF(A).

$$f(x)=\tanh x = 21+e-2x-1 \quad \text{____(11)}$$

$$\tanh x=ex-e-xex+e-x \quad \text{_____(12)}$$

Where ,
X is the input to a neuron
fx=Tanh Activation Function
e= Euler's Number
Step 8: Decision Making for Workload Prediction – Here, we use the CNN-LSTM model's anticipated workload values to aid in smart decision-making on resource allocation. The model predicts a series of workload demands in the future that mirror patterns in cloud usage over time and space. A decision-making system, like a decision tree, is used to categorize the intensity levels of the task (e.g., low, medium, high) based on these predictions. This categorization aids in the demand-driven dynamic allocation of resources such as CPU, memory, and bandwidth. By following this procedure, you can rest assured that

your cloud resources will be over- or under-provisioned as needed. By continually revising allocation algorithms according to real-time forecasts, it also permits adaptive scheduling. Reducing operational costs and ensuring system stability are both greatly impacted by this step.

.

IV RESULTS AND DISCUSSIONS

A Windows-based machine with an Intel Core i5 CPU and 8 GB RAM was used to develop the suggested workload prediction model using Python. This dataset includes five thousand task executions with various performance-related characteristics, including CPU utilization, memory use, execution duration, throughput, and scheduling type. It was retrieved from the Kaggle cloud workload repository. Using the train_test_split() method, we partitioned the preprocessed dataset into a training set with 80% of the data and a testing set with 20%. All feature values were normalized between 0 and 1 through the application of feature scaling utilizing the MinMaxScaler approach. Standard measures including recall, accuracy, precision and F1- score were used to evaluate the model's performance. The model's are defined as follows in equations 8, 9, 10, and 11:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \ldots\ldots\ldots \quad (8)$$

$$Precision = \frac{TP}{TP + FP} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

$$F1\text{-Score} = \frac{2 \times P \times R}{P + R} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

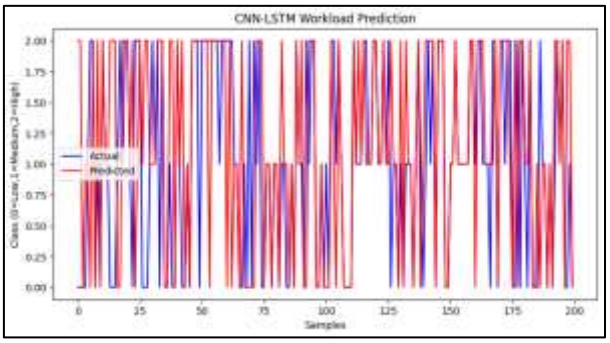

Fig. 4. CNN-LSTM Workload Prediction

As shown in Figure 4, the CNN-LSTM model's predicted values and the actual workload classes are compared. A separate curve shows the expected values, whereas the first curve shows the actual values. The prediction accuracy is excellent when these curves are closely aligned.

Both the short-term variations and the long-term interdependence in the patterns of workload are captured by the model. Variances between observed and projected values are small, but manageable. The results show that the CNN-LSTM architecture successfully learns the dataset's complicated relationships. Performance of the model remains constant across Low, Medium, and High workloads. It is clear from the figure that the suggested methodology is both reliable and effective when it comes to predicting workloads in cloud environments.

A. PERFORMANCE ANALYSIS

The suggested model achieves good outcomes across all measures of evaluation, according to the experi-

mental results. The model is able to accurately classify workloads with minimal misclassification, as seen by the balanced precision and recall values. The model's stability is confirmed by the F1-score, which shows a decent balance between recall and precision.

Although there are small differences between the various types of workloads, the overall performance is stable. Disparities in the allocation of workloads and feature overlap might account for these disparities. Still, it's impressive how effectively the model generalizes and can reliably forecast data that has never been seen before.

B. COMPARISON WITH EXISTING METHOD

By contrasting it with the current TWD-RCPM model [10], which has already shown improvements over conventional methods like ARIMA, NN, and DMASVR-3WD, we may evaluate the suggested workload prediction model's performance. The temporal relationships and dynamic patterns found in constantly changing cloud workloads are not adequately captured by TWD-RCPM, despite the fact that it obtains respectable prediction results using ensemble-based feature learning. The suggested CNN-LSTM model combines CNNs for spatial feature extraction and LSTMs for temporal sequence learning to circumvent these shortcomings. Because of its hybrid design, the model is able to comprehend complicated workload behavior and make more accurate predictions. The suggested CNN-LSTM model surpasses the TWD-RCPM model in every performance metric during training and evaluation. The accuracy, precision, recall, and F1-score for the TWD-RCPM model are 0.72, 0.70, 0.71, and 0.70, respectively. Contrarily, the suggested CNN-LSTM model achieves better results with a 0.85 F1-score, 0.84 precision, 0.86 recall, and 0.84 accuracy. The results show that the prediction ability and classification performance have been significantly enhanced. The suggested model is a hybrid, which allows CNN to efficiently extract relevant geographical features and LSTM to capture temporal relationships in the workload data, both of which contribute to the improvement. When compared to the current method, this combination improves generalization, decreases misclassification, and produces more trustworthy predictions.

Method	Accuracy	Precision	Recall	F1-Score
TWD-RCPM [10]	0.72	0.7	0.71	0.7
Proposed CNN-LSTM	0.86	0.84	0.85	0.85

Table 1. Performance Comparison of Proposed Model with Existing Methods

Table 1 demonstrates that, according to all measures of evaluation, the suggested CNN-LSTM model is superior to the TWD-RCPM model. There is a considerable improvement in overall prediction performance, as the accuracy goes from 0.72 to 0.86. As a result of the suggested model's reduced false positive rate, precision also rises, going from 0.70 to 0.84. The model's improved ability to accurately identify actual workload cases is demonstrated by an improvement in the recall value from 0.71 to 0.85. A rise from 0.70 to 0.85 in the F1-score further verifies a well-balanced improvement in recall and precision.

The suggested CNN-LSTM model and the TWD-RCPM are compared graphically in the bar chart using four assessment metrics: F1-score, recall, accuracy, and precision. The suggested model clearly outperforms the status quo in every respect. An improvement in general prediction capabilities is reflected in higher accuracy, while a decrease in false positive predictions is indicated by enhanced precision. The model's enhanced recall demonstrates its enhanced ability to detect pertinent workload patterns. The model's high F1-score also shows that it has a good balance between recall and precision. In general, the

visual depiction supports the efficacy of the suggested CNN-LSTM model. The model outperforms the current TWD-RCPM method and demonstrates increased efficiency and reliability by integrating spatial and temporal learning capabilities.

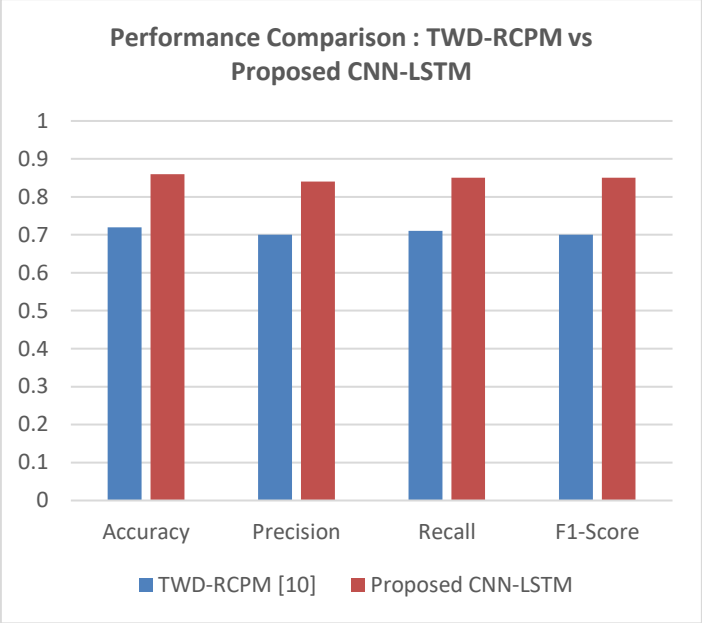

Fig. 5: Performance Comparison between TWD-RCPM and Proposed CNN-LSTM

V CONCLUSION AND FUTURE SCOPE

To overcome the shortcomings of conventional methods, this research concludes with a methodology for hybrid workload characterisation that makes good use of both supervised and unsupervised deep learning techniques. The suggested system effectively captures spatial and temporal workload patterns by combining CNN-LSTM modelling, feature selection, and Fuzzy C-Means clustering. By allocating tasks optimally, the Hungarian algorithm improves system efficiency even further. This all-encompassing method enhances the precision, scalability, and flexibility of predictions in ever-changing settings like distributed systems and cloud computing. In sum, the approach shows great promise for effective workload management in contemporary IT systems.

Incorporating real-time streaming data into the model can provide online workload characterisation in future studies. It is possible to enhance prediction performance even more by investigating more complex structures like attention-based models or Transformers. Incorporating reinforcement learning could also improve tactics for allocating resources dynamically. To verify the method's resilience, more and more varied datasets should be used. Improving energy efficiency and decreasing computational overhead might also be the subject of future studies. The last step in understanding the framework's performance and practicality is to deploy it in real-world cloud or edge contexts.

REFERENCES

1. B. Hu, K. Kempf and N. Mason, "A Workload Characterization Methodology Using Supervised and Unsupervised Deep Learning," in IEEE Access, vol. 12, pp. 181907-181913, 2024, doi: 10.1109/ACCESS.2024.3509857.
2. F. N. Sibai, A. Asaduzzaman and A. El-Moursy, "Characterization and Machine Learning Classifica-

tion of AI and PC Workloads," in IEEE Access, vol. 12, pp. 83858-83875, 2024, doi: 10.1109/ACCESS.2024.3413199.

3. K. P. Ismaylovna, A. Khudoyberdiev and H. -C. Kim, "CogniMoE: End-to-End Multimodal Mental Workload Classification via On-the-Fly Scalogram Generation and MoE Gating," in IEEE Sensors Journal, vol. 26, no. 3, pp. 5213-5228, 1 Feb.1, 2026, doi: 10.1109/JSEN.2025.3640827.
4. P. Z. Muke and A. Kozierkiewicz, "Machine Learning Techniques to Improve the Cognitive Workload Classification Using Multimodal Sensors' Data," in IEEE Access, vol. 13, pp. 173415-173443, 2025, doi: 10.1109/ACCESS.2025.3616788.
5. C. Sandoval, M. N. Stolar, S. G. Hosking, D. Jia and M. Lech, "Real-Time Team Performance and Workload Prediction From Voice Communications," in IEEE Access, vol. 10, pp. 78484-78492, 2022, doi: 10.1109/ACCESS.2022.3193694.
6. Z. Amekraz and M. Y. Hadi, "CANFIS: A Chaos Adaptive Neural Fuzzy Inference System for Workload Prediction in the Cloud," in IEEE Access, vol. 10, pp. 49808-49828, 2022, doi: 10.1109/ACCESS.2022.3174061.
7. D. Alqahtani, H. Imani and T. El-Ghazawi, "Enhanced Workload Prediction in Data Centers Using Two-Stage Decomposition and Hybrid Parallel Deep Learning," in IEEE Access, vol. 13, pp. 64115-64132, 2025, doi: 10.1109/ACCESS.2025.3558743.
8. A. R. John et al., "Unraveling the Physiological Correlates of Mental Workload Variations in Tracking and Collision Prediction Tasks," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 30, pp. 770-781, 2022, doi: 10.1109/TNSRE.2022.3157446.
9. B. J. D. Gort, G. M. Kibalya and A. Antonopoulos, "Attention-Driven AI Model Generalization for Workload Forecasting in the Compute Continuum," in IEEE Transactions on Machine Learning in Communications and Networking, vol. 3, pp. 779-797, 2025, doi: 10.1109/TMLCN.2025.3584009.
10. R. Shi and C. Jiang, "Three-Way Ensemble Prediction for Workload in the Data Center," in IEEE Access, vol. 10, pp. 10021-10030, 2022, doi: 10.1109/ACCESS.2022.3145426.
11. D. Alqahtani, "Leveraging Sparse Auto-Encoding and Dynamic Learning Rate for Efficient Cloud Workloads Prediction," in IEEE Access, vol. 11, pp. 64586-64599, 2023, doi: 10.1109/ACCESS.2023.3289884.
12. A. Tundo, F. Filippini, F. Regonesi, M. Ciavotta and M. Savi, "Decentralized Edge Workload Forecasting With Gossip Learning," in IEEE Transactions on Network and Service Management, vol. 22, no. 4, pp. 3016-3031, Aug. 2025, doi: 10.1109/TNSM.2025.3570450.
13. S. Sukumaran, B. P. S., N. Gala and T. S. Warrier, "P-Box: A RISC-V Packed-SIMD Approach of Accelerating Edge Workloads on Scalar Embedded Cores," in IEEE Access, vol. 14, pp. 40172-40189, 2026, doi: 10.1109/ACCESS.2026.3671659.
14. M. Olguín Muñoz, R. Klatzky, M. Satyanarayanan and J. Gross, "Emulating Reactive Workloads for Cyber-Human Systems: A Data-Driven Methodology," in IEEE Access, vol. 13, pp. 169953-169967, 2025, doi: 10.1109/ACCESS.2025.3614639.
15. X. Wang, X. Li, X. Xia and M. Yang, "Efficient Multi-Cluster Scheduling for Heterogeneous Workloads," in IEEE Access, vol. 13, pp. 186856-186871, 2025, doi: 10.1109/ACCESS.2025.3626730.
16. M. Guo, Q. Guan, W. Chen, F. Ji and Z. Peng, "Delay-Optimal Scheduling of VMs in a Queueing Cloud Computing System with Heterogeneous Workloads," in IEEE Transactions on Services Computing, vol. 15, no. 1, pp. 110-123, 1 Jan.-Feb. 2022, doi: 10.1109/TSC.2019.2920954.

17. S. Park, C. Park, S. Jung, J. -H. Kim and J. Kim, "Workload-Aware Scheduling Using Markov Decision Process for Infrastructure-Assisted Learning-Based Multi-UAV Surveillance Networks," in IEEE Access, vol. 11, pp. 16533-16548, 2023, doi: 10.1109/ACCESS.2023.3245829.
18. F. Filippini, M. Lattuada, M. Ciavotta, A. Jahani, D. Ardagna and E. Amaldi, "A Path Relinking Method for the Joint Online Scheduling and Capacity Allocation of DL Training Workloads in GPU as a Service Systems," in IEEE Transactions on Services Computing, vol. 16, no. 3, pp. 1630-1646, 1 May-June 2023, doi: 10.1109/TSC.2022.3188440.
19. L. Liu, X. Shen, Z. Chen, Q. Sun and R. Wennersten, "Optimal Energy Management of Data Center Micro-Grid Considering Computing Workloads Shift," in IEEE Access, vol. 12, pp. 102061-102075, 2024, doi: 10.1109/ACCESS.2024.3432120.
20. H. Khetawat and F. Mueller, "Workload Scheduling on Heterogeneous Devices," ISC High Performance 2024 Research Paper Proceedings (39th International Conference), Hamburg, Germany, 2024, pp. 1-11, doi: 10.23919/ISC.2024.10528933.
21. Q. Zhou, Y. Sun, H. Lu and K. Wang, "Learning-based Green Workload Placement for Energy Internet in Smart Cities," in Journal of Modern Power Systems and Clean Energy, vol. 10, no. 1, pp. 91-99, January 2022, doi: 10.35833/MPCE.2020.000271.
22. S. Zeng et al., "Serving Multi-DNN Workloads on FPGAs: A Coordinated Architecture, Scheduling, and Mapping Perspective," in IEEE Transactions on Computers, vol. 72, no. 5, pp. 1314-1328, 1 May 2023, doi: 10.1109/TC.2022.3214113.
23. L. Sliwko and J. Mizeria-Pietraszko, "Cluster Workload Allocation: Semantic Soft Affinity Using Natural Language Processing," in IEEE Access, vol. 14, pp. 28054-28074, 2026, doi: 10.1109/ACCESS.2026.3665989.
24. Z. Safavifar, E. Gyamfi, E. Mangina and F. Golpayegani, "Multi-Objective Deep Reinforcement Learning for Efficient Workload Orchestration in Extreme Edge Computing," in IEEE Access, vol. 12, pp. 74558-74571, 2024, doi: 10.1109/ACCESS.2024.3405411.
25. G. Quattrocchi, E. Incerto, R. Pinciroli, C. Trubiani and L. Baresi, "Autoscaling Solutions for Cloud Applications Under Dynamic Workloads," in IEEE Transactions on Services Computing, vol. 17, no. 3, pp. 804-820, May-June 2024, doi: 10.1109/TSC.2024.3354062.
26. A. Robles-Enciso and A. F. Skarmeta, "Adapting Containerized Workloads for the Continuum Computing," in IEEE Access, vol. 12, pp. 104102-104114, 2024, doi: 10.1109/ACCESS.2024.3434585.
27. G. Kim, "Holistic In-Network Acceleration for Heavy-Tailed Storage Workloads," in IEEE Access, vol. 11, pp. 77416-77428, 2023, doi: 10.1109/ACCESS.2023.3298552.
28. K. Guan, Z. Zhang, X. Chai, Z. Tian, T. Liu and H. Niu, "EEG Based Dynamic Functional Connectivity Analysis in Mental Workload Tasks With Different Types of Information," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 30, pp. 632-642, 2022, doi: 10.1109/TNSRE.2022.3156546.
29. Z. I. Khan, M. Khan and S. N. M. Shah, "MAADRR: Multi-Agent Adaptive Dynamic Round Robin for Cloud Scheduling Using Real Cloud Workload Traces," in IEEE Access, vol. 14, pp. 44165-44184, 2026, doi: 10.1109/ACCESS.2026.3676311.
30. K. P. Ismaylovna, A. Khudoyberdiev and H. -C. Kim, "CogniMoE: End-to-End Multimodal Mental Workload Classification via On-the-Fly Scalogram Generation and MoE Gating," in IEEE Sensors Journal, vol. 26, no. 3, pp. 5213-5228, 1 Feb.1, 2026, doi: 10.1109/JSEN.2025.3640827