National College of Ireland

A novel GRU-Attention based deep learning architecture for the prediction of idle workloads

MSc Research Project
Cloud Computing

Anisha Khalam
Student ID: X23401061

School of Computing
National College of Ireland

Supervisor: Dr Giovani Estrada

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anisha Khalam
Student ID:	X23401061
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Dr Giovani Estrada
Submission Due Date:	11/12/2025
Project Title:	A novel GRU-Attention based deep learning architecture for the prediction of idle workloads
Word Count:	8161
Page Count:	25

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Anisha Khalam
Date:	27th January 2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A novel GRU-Attention based deep learning architecture for the prediction of idle workloads

Anisha Khalam
X23401061

Abstract

Cloud providers face escalating operational costs as data centres expand, with substantial fractions of virtual machines remaining idle, yet consuming energy and resources. Proactive VM consolidation, predicting future idle states to enable advance migration or shutdown, offers cost savings but requires proactive forecasting rather than reactive monitoring. Existing research exhibits gaps: models train exclusively on single cloud platforms without cross-cloud validation, continuous regression outputs require post-processing rather than actionable predictions, severe class imbalance (80%+ idle) remains not addressed causing minority class failure, large architectures constrain resource-limited deployment, and single-architecture proposals lack systematic multi-cloud comparisons. This thesis proposes a GRU-Attention architecture with a triple class imbalance strategy (Focal Loss, class weighting, threshold optimisation) for one-hour ahead binary VM state forecasting (IDLE/ACTIVE) which consist of two stacked GRU layers (64/32 units) with decomposed single-head attention mechanism achieving selective temporal focus. Evaluation against four alternative architectures on Microsoft Azure Public Dataset demonstrates that GRU-Attention achieves an accuracy of 93.65% , f1_macro of 90.89% using 24,162 parameters. The triple imbalance strategy improves the recall of the minority by 20.14 percentage points (78.45% to 98.59%). Cross-cloud deployment to AWS EC2 achieves 83.1% average confidence with successful state transition detection, representing the first empirical validation in the literature of cloud-agnostic model transfer without retraining. Organisations can deploy proactive consolidation systems that provide 60-minute advance warning, maintain operational safety, which represents a reasonable trade-off for cost reduction. This work establishes GRU-Attention as a production-ready architecture for cloud workload forecasting, addressing all identified gaps by providing empirical evidence that cloud agnostic temporal patterns transferable across heterogenous infrastructure can be learned by deep learning models.

1 Introduction

The resource utilisation analysis of cloud computing workloads reveals underutilisation that exhibits various patterns . According to the Microsoft Azure Public Dataset[1] V1 (with 156,031 virtual machines), 81.6% of the virtual machines are in idle state (with CPU utilisation less than 5%). Thus, it offers a massive consolidation opportunity. By observing idle periods in VMs, classical reactive monitoring is used for consolidation,

[1] `https://github.com/Azure/AzurePublicDataset`

however, this lacks optimisation opportunities. Scheduled migrations, advance capacity planning, and strategic consolidation decision can be made using proactive prediction of idle workloads. The five important limitations in the current machine learning based workload forecasting studies include, first, zero cross-cloud validation despite multi-cloud infrastructure being run by the company, second, continuous regression requiring post processing compared to actionable binary IDLE/ACTIVE classification, third, neglecting severe class imbalance, fourth, simulation validation is only done instead of production deployment and single architectures are proposed without comparison. This thesis solves such problems by an extension of the GRU-Attention architecture to an one-hour ahead idle workload forecast. Moreover, a range of GRU based architectures were developed to predict the VM state with the help of Microsoft Azure Public Dataset V1 which trains Simple GRU, Bidirectional GRU, GRU-Attention, Conv-GRU and Conv-GRU-Attention. The evaluation of the models was carried out with forecasts one hour in advance. A methodological comparison of other GRU-based architecture demonstrates that GRU-Attention has the most appropriate performance efficiency. This thesis makes three primary contributions. First, it provides the original empirical cross-cloud validation as it efficiently deploys Azure trace-trained models to AWS EC2, and reduces the cost of multi-cloud deployment. Second, the triple imbalance strategy solves the systematic class imbalance where the ACTIVE recall is going to increase. This article bridges the academic-industrial gap by demonstrating that GRU-Attention can be trained to generate cloud-agnostic temporal patterns, which can be applied to heterogeneous infrastructure and is a novel standard of idle workload prediction.

1.1 Research Gap

It is true that there is substantial progress in deep learning in predicting cloud work loads, however, the academic literature examined in Section 2 indicates that there are five key gaps

1. **No cross-cloud validation:** The reviewed papers train and test models only on single cloud platforms. Research by (Gan et al.; 2022) and (Bi et al.; 2023) only trains on Google, Alibaba or Azure traces, respectively, without validating in different providers.

2. **Post-processing for operational decisions while using regression:** Using regression, existing work predicts continuous CPU percentages instead of binary operational states (IDLE/ACTIVE classification). VM consolidation mainly demands binary decisions ("consolidate" or "maintain"), Post-processing thresholding is needed for regression output which will create decision complexity and latency.

3. **Unaddressed class imbalance:** Azure Public Dataset have shown severe class imbalance, which is an example of real world data centre (81.6% IDLE / 18.4% ACTIVE), however already existing literature ignores imbalance handling, causing the model to predict IDLE for all instances making high accuracy but 0% ACTIVE recall.

4. **Simulation based validation instead Production deployment:** Simulation is considered in existing papers rather than in production cloud infrastructure. None of the reviewed studies exhibits workload prediction on live public cloud platforms (AWS EC2), this implies that real world deployment is not verified.

5. **No architectural comparison:** No comparative evaluation is done in most of the papers/ However, some studies still compare different architecture, but use different datasets, metrics, hyper parameters, etc.

Most organisations run different cloud infrastructures (AWS + Azure + GCP). The provider-specific models has high training cost and maintenance load, none of the academic work proves whether training one provider can be used for the other. Although consolidation of VM demands binary choices (consolidate or not) the majority of experiments forecast the sustained percentages of CPU (regression with RMSE scores) ,instead of operational states. None of the studies describes binary state classification. Based on the Azure dataset, it is assumed that there are 80%+ idle VMs in real cloud datacenters (For example, Azure Public Dataset has 81.6% of idle VMs and 18.4% of active VMs) ,this imbalance is not handled by cloud prediction literature. Most articles suggest individual architectures and no comparison is done. A small number of studies have conducted a systematic comparison of different architectures on different data with similar evaluation procedures.

1.2 Research question

The research demonstrates how Gated Recurrent Unit (GRU)-based predictive models which can be integrated into cloud platforms to proactively identify low-utilisation virtual machines in cloud environments. The proposed approach aims to combine machine learning-based forecasting to enable proactive and autonomous decision-making to optimise financial operations.

Research Question:

To what extent can a GRU based deep learning model forecast CPU workload that will enable to identify low-utilisation virtual machines to achieve financial savings in cloud environments?

1.3 Research objectives

The objectives of this research are as follows:

- **RO1:**Preprocess Azure Public Dataset with stratified split with labels IDLE and ACTIVE classification for 1-hour ahead forecasting tasks.
- **RO2:** Design , train, and validate five GRU architectures (Simple GRU, BiGRU, GRU-Attention, Conv-GRU, Conv-GRU-Attention).
- **RO3:** Comparison of architectures in terms of F1-Macro,Accuracy.
- **RO4:** Deploying best Azure-trained model on AWS via t2.large EC2 instance and validate the forecast in other instances.

2 Related Work

The literature review paper is an in-depth research study on the latest scholarly articles on topics of workload prediction methods, especially deep learning models based on Gated Recurrent Units (GRU) and other recurrent neural networks, their applicability to virtual machine (VM) consolidation to achieve cost optimisation in the cloud.

2.1 Deep Learning for Workload Prediction

2.1.1 GRU and LSTM Architectures

Deep learning for VM workload prediction using deep belief networks with restricted Boltzmann machines was implemented by (Qiu et al.; 2016), which have acquired 85% accuracy. It also showed that the neural approach for workload forecasting can be used. This has shown baseline performance but predates modern gating mechanisms (GRU/LSTM gates introduced 2014-2015), also it is trained in single cloud data and no cross cloud validation observed. Also, they have used regression here along with class imbalance.

A hybrid architecture GRU-Conv was implemented that combined GRU temporal modelling with Conv local pattern for cloud workload prediction using Google cluster traces (Gan et al.; 2022). Here, GRU captures sequential dependencies using gating mechanisms, and Conv extracts spatial patterns using convolutional filters. But this evaluation only uses small cloud traces (hundreds of virtual machines). Moreover, it trains exclusively on Google data that predict continuous regression values and also again, in this paper it systematically ignores class imbalance handling . Since imbalance techniques are not there, models have achieved high accuracy for learning majority class (IDLE) patterns, at the same time, it fails on minority ACTIVE VMs.

The Conv-LSTM hybrid was developed by (Leka et al.; 2021) where Conv extracts features from historical windows whereas the LSTM models temporal dependencies, this has achieved 88% RMSE for the prediction of the VM workload . However, this work lacks an attention mechanism comparison which made it difficult to find out whether Conv preprocessing performs better than attention-based temporal weighting . It is shown that Conv-GRU-Attention degrades to 89.75% versus GRU-Attention's 90.89%, which proves Conv redundant when attention directly learns timestep. Along with this, the author have trained on small traces without any cross-cloud validation along with regression outputs, and class imbalance is not considered.

An intelligent BiLSTM framework for multivariate time-series forecasting was proposed by (Ullah et al.; 2023) to predict multiple cloud resources (CPU provisioned/usage, memory, disc throughput, network traffic) using Bitbrains traces. Here, the multivariate prediction finds the resource interdependencies (i.e., CPU spikes can be correlated with network traffic increase) , different from univariate approaches. However, here, for bidirectional processing, it requires 2× parameters versus forward-only GRU, which produces continuous regression outputs which require threshold selection for consolidation decision. It also lacks class imbalance handling and trains only on single-cloud Bitbrains data. In addition, it validates the simulation data.

An integrated BG-LSTM that combines bidirectional and grid LSTMs was proposed by (Bi et al.; 2021) to predict workload and resources using Google cluster traces. It has achieved improved accuracy using Savitzky-Golay filtering and logarithmic transformation. This work establishes strong baseline performance, however, it precedes recent attention mechanism advances (2017+) which provide more parameter-efficient temporal focus. This model trains exclusively on Google data and, moreover, it gives regression values without any class imbalance handling. However, this study helped to understand the growing importance of attention for cloud workload prediction.

2.1.2 Attention Mechanisms

Temporal relevance weights are learnt by the model using an attention mechanism; this also considers discriminative timesteps instead treating all historical data alike. VAM-BiG combining multi-head attention with hybrid LSTM for cloud workload prediction which have achieved 92% regression RMSE using Alibaba traces were implemented by (Bi et al.; 2023). Multi-head attention provides parallel temporal focus across multiple scales (early heads focus on recent timesteps, later heads capture long-range dependencies). This is really helpful for complex workloads. However, inference latency (3-5 seconds per prediction) affects real-time requirements for 5-minute monitoring intervals. It gives continuous CPU percentages by ignoring class imbalance, and it lacks production deployment validation. This work shows single-head attention suffices for 12-timestep sequences by superior efficiency (24,162 parameters) and real-time inference achieved.

LSTM-Attention-LSTM encoder-decoder architecture for time series prediction was implemented by (Wen and Li; 2023). Long sequences are excelled by Encoder-decoder frameworks(100+ timesteps) which enables complex temporal compression. However, this has a limitation as it introduces unnecessary architectural complexity for short sequences where direct prediction suffices. For 12-timestep windows, encoder-decoder overhead provides minimal benefit over direct GRU-Attention . In addition, it does not handle imbalances.

2.1.3 Architecture Comparison and Multi-Step Forecasting

A comprehensive benchmark evaluation of nine RNN architectures (vanilla RNN, LSTM, GRU, RNN-LSTM, RNN-GRU, LSTM-RNN, GRU-RNN, LSTM-GRU, GRU-LSTM) was performed using Monte Carlo simulation with 100 iterations across three datasets (Yunita et al.; 2025). Statistical analysis (Friedman test) reveals no significant performance differences among architectures. However, consistent patterns favour LSTM-based hybrids (LSTM-GRU and LSTM-RNN). This comparison validates importance of systematic evaluation under identical conditions, which directly supports this work's five-architecture comparison approach (Simple GRU, BiGRU, GRU-Attention, Conv-GRU, Conv-GRU-Attention on same Azure dataset with same preprocessing, training, evaluation). However, the paper lacks attention mechanisms and cloud-specific application.

Multi-step ahead prediction strategies (recursive, direct, MIMO, DirRec) was proposed by (An and Anh; 2015) for time series forecasting using neural networks, revealing that direct strategies outperform recursive approaches over moderate horizons (10-20 timesteps). Recursive strategies propagate prediction errors across multiple steps, leading to compounding inaccuracies. At the same time, direct strategies train separate models for each horizon, which avoids error accumulation. This finding justifies the 12-step direct prediction in this work for forecasting one-hour ahead. The model predicts the state of t + 60 min from the historical window t-60 min to t.

2.2 VM Consolidation Policies

EEVMC for energy-efficient VM consolidation in cloud datacenters was proposed by (Rehman et al.; 2024). It has achieved energy savings using intelligent placement algorithms. Proactive forecasting predicts VM states 60 minutes ahead with 93.65% accuracy, which has enabled consolidation systems to operate on predicted future states. This paper has

built a foundation on consolidation. The VM consolidation approach aware of usage prediction by (Hsieh et al.; 2020) uses the Grey-Markov model to forecast resource use using PlanetLab traces (800 VMs). Using host overload/underload detection, this approach combines current and future utilisation, which has reduced VM migrations and energy consumption. However, Grey-Markov models assume workload stationarity violated by cloud workloads leading to drift, bursts, and long-range dependencies. Deep learning approaches (GRU/LSTM) will capture non-stationary dynamics better by learning temporal representations. Additionally, this paper only performs simulation validation.

Adaptive deep reinforcement learning for VM consolidation in energy-efficient datacenters has been adapted by (Zeng et al.; 2022), which learns optimal consolidation policies using trial-and-error interaction in simulated environments. However, convergence requires weeks of training on millions of simulated episodes. Also, RL and supervised forecasting are complementary. i.e., this paper states predictions , i.e., what will happen? RL or rule-based systems works on policy decisions (what actions to take). But in (Zeng et al.; 2022), it trains only in simulation without validating policies in the production cloud environment.

The prediction of the CPU utilisation of the RNN-based host for cloud computing was demonstrated by (Duggan et al.; 2017). It has established neural networks viability for cloud resource forecasting. But basic RNN architectures have vanishing gradient problems which limits long-range temporal dependency learning—gradients to diminish exponentially via back propagation through many time steps, this will prevent effective learning from distant historical patterns. GRU's gating mechanisms (reset gate, update gate) can address vanishing gradients using gated gradient flow, which will enable effective learning from 12-timestep windows. Additionally, (Duggan et al.; 2017) use regression and do not handle imbalance.

2.3 Class Imbalance Handling Techniques

The comprehensive taxonomy of class imbalance techniques was categorised as data preprocessing (oversampling using SMOTE/ ADSYN, which will generate synthetic minority samples, undersampling by removing majority samples), algorithmic structures, and hybrid techniques, which will combine both approaches by (Rezvani and Wang; 2023) . The paper concludes by mentioning that imbalance handling is problem-specific with no universal solution which motivated this work to implement domain-tailored triple imbalance strategy (Focal Loss + class weighting + threshold optimisation). In this research, there is no generation of synthetic data.

The classification and metric selection for the unbalanced data in brain decoding was examined by (Thölke et al.; 2023). This has demonstrated that the standard accuracy metric yields misleadingly high performance as the imbalance increases. This is because the accuracy weights per-class ratios are proportional to the size of the class, largely disregarding the performance of minority classes. Using systematic manipulation of imbalance ratios in EEG, MEG, and fMRI data, the author showed balanced precision. In addition, AUC-ROC provides a reliable performance evaluation for imbalanced scenarios. Their findings are validated in this paper's metric selection: F1-Macro average score (90.89%) equally weighting both classes, AUC (96.59%) which measures discrimination ability in all thresholds, and explicit minority recall reporting (98.59%) used to show operational safety for consolidation decisions.

However, it is found that the zero reviewed articles address the class imbalance issue

in the cloud for workload prediction even after real datacenters that show 80-85% idle VMs. This provides a gap between academic research and production requirements. Models that ignore imbalance will achieve high reported accuracy, whereas they will fail in critical minority class predictions.

Table 1: Comparison of this work with key papers

Paper	Architecture & Dataset	Task / Accuracy / Imbalance	Cross-Cloud
(Qiu et al.; 2016)	Deep Belief Network; Small dataset	Regression; 85%; None	No
(Gan et al.; 2022)	GRU-Conv; Google (∼100s VMs)	Regression; Not Reported; None	No
(Leka et al.; 2021)	Conv-LSTM; Small traces	Regression; 88% RMSE; None	No
(Bi et al.; 2023)	VAMBiG; Alibaba proprietary	Regression; 92% RMSE; None	No
(Ullah et al.; 2023)	BiLSTM; Bitbrains traces	Regression; RMSE/MAE; None	No
(Hsieh et al.; 2020)	Gray-Markov; PlanetLab (800 VMs)	Consolidation; Not reported; None	No
This Work	**GRU-Attention; Azure 156K + AWS EC2**	**Classification; 93.65% / 90.89% F1; 98.59%**	**Yes (83.1%)**

This work represents as the production-ready cross-cloud VM state prediction system which is validated on real heterogeneous infrastructure along with systematic imbalance handling to achieve operational reliability.

3 Methodology

Here, Cross-Industry Standard Process of Data Mining (CRISP-DM) methodology is used for designing and testing GRU-based predictive models to estimate VM CPU usage forecasting.

3.1 Phase 1: Business Understanding and Literature Gap

According to the AWS t3.medium pricing, one idle VM is estimated at about 0.0416 dollars per hour, which increases to a high amount of expenditure when deployed within an enterprise, and also it costs around 364 dollars per year for idle instance. Consider an organisation which is running thousands of such instances leading to a high cloud bill. The review of articles in the literature on the subject indicated five essential gaps. These gaps had motivated the main research question. The Success criteria target is to achieve F1-Macro around 90, minority recall around 95, cross-cloud confidence around 80, forecasting degradation below 10% points.

3.2 Phase 2: Data Understanding

Here, the Microsoft Azure Public Dataset V1, which contains a telemetry of 156,031 individual VMs throughout the first quarter of 2017, is analysed. The data gives the CPU measurements at 5-minute intervals, resulting in about 10 million samples. In each of the measurements, there are average CPU, maximum CPU, and minimum CPU values within the 5-minute period. The analysis showed that there was a class imbalance with 81.6% IDLE and 18.4% ACTIVE . The 5% threshold gives a trade off between the consolidation opportunity and the severity of the imbalance. Subsequently, when assessing the quality of the data, less than 2% missing data were handled by forward-fill imputation, and no invalid readings were outside the valid range of 0-100.

3.3 Phase 3: Data Preparation

Here, raw telemetry data are converted to structured deep learning sequences. In this report, a sliding window method will be used to extract 12 timesteps. This has sequences that have a length of 60 minutes exactly. For forecasting, windows advance by 12 timesteps (60 min) such that it create nonoverlapping sequences to prevent data leakage between training examples. 3 features (average CPU, maximum CPU, minimum CPU) are considered here. The model will predict based on this. Defining state at t+12 as forecast sequences. If binary encoding is 0, then the VM is IDLE (less than 5% CPU utilisation) and if it is 1, the VM is ACTIVE (more than 5% CPU utilisation). Stratified splitting preserves an 81.6%/18.4% split of 80% training, 10% validation and 10% test sets. Normalisation is performed using the min-max scaling in which each sequence is then individually scaled to the range [0,1] using the minimum and maximum of that sequence. The data set is normalised and stratified and is available to train models.

3.4 Phase 4: Modelling

The modelling stage implements five GRU-based models in the same conditions. GRUs are chosen due to the 25% efficiency in parameter usage and 30-40% faster training compared to LSTM [2] . The Simple GRU baseline has 2 stacked layers (64 and 32 units) with 30% dropout and 92,545 parameters. Bidirectional GRU substitutes regular layers with bidirectional ones, which work with sequences in both directions and include batch normalisation with 166,145 parameters, this will test whether the bidirectional processing can enhance the prediction. GRU-Attention has an attention mechanism over base GRU outputs. Conv-GRU uses 1D convolutional (64 filters, kernel 3) followed by GRU layers with 106 849 parameters. All components are combined in Conv-GRU-Attention with 110,017 parameters, to find out whether a larger model can boost its performance. All architectures will be trained with Adam optimiser with learning rate 0.001, batch size of 64, 50 epochs, learning rate decay , random seed . The best weights per architecture are maintained according to F1-Macro and deployed in EC2 Instance.

3.5 Phase 5: Evaluation

Evaluation metrics include accuracy, F1-Macro, AUC-ROC and per-class precision and recall are calculated here. F1-Macro serves as primary metric as overall accuracy misleadingly with class imbalance. In addition, special attention is paid to minority class recall(ACTIVE). Since false negatives imply that active VMs that predicted IDLE, leading to consolidation which will lead to performance degradation of application, here active recall less than or equal to 95% is targeted to ensure operational safety. First, all five architectures are compared to Azure test set using per model optimal thresholds and calculating full confusion matrices. Then Architectures are compared based on weighted metrics: F1-Macro, minority recall, parameter efficiency, inference time and finally confirms it using cross-cloud validation by using best architecture on AWS EC2 and will find confidence on live CloudWatch metrics.

[2] https://medium.com/@digitalconsumer777/lstm-vs-gru-complete-comparison-for-sequence-modeling-6020612fceb5

3.6 Phase 6: Deployment and Cross Cloud Validation

The deployment stage validates cross-cloud transferability by deploying the best GRU model trained in Azure on the AWS EC2 platform. This fills the gap in the critical literature in which no previous study validates cross cloud model transfer. Three instances of EC2 t3.micro (ec21, ec22, ec33) are deployed in us-east-1, representing small instances that are not always used in production. The model is reimplemented programmatically in EC2. The prediction program ec2forecastexactarchitecture.py queries CloudWatch API calls, retrieving CPU metrics (average, maximum, minimum) of the three instances. In every case, the script retrieves the latest 12 timesteps of a 60-minute look back window of training data structure. The metrics retrieved are preprocessed in the same way that min-max normalisation to the range of [0,1] retains relative temporal trends and then transformed into the tensor format. The predicted state (IDLE/ACTIVE) and the raw probability, confidence score, instance identifier, and timestamp are written to the JSON files. The cross-cloud validation uses mean confidence to assess. The deployment is also different from continuous monitoring systems because it does not involve automated pipeline but validates cross cloud transferability by running a manual script.

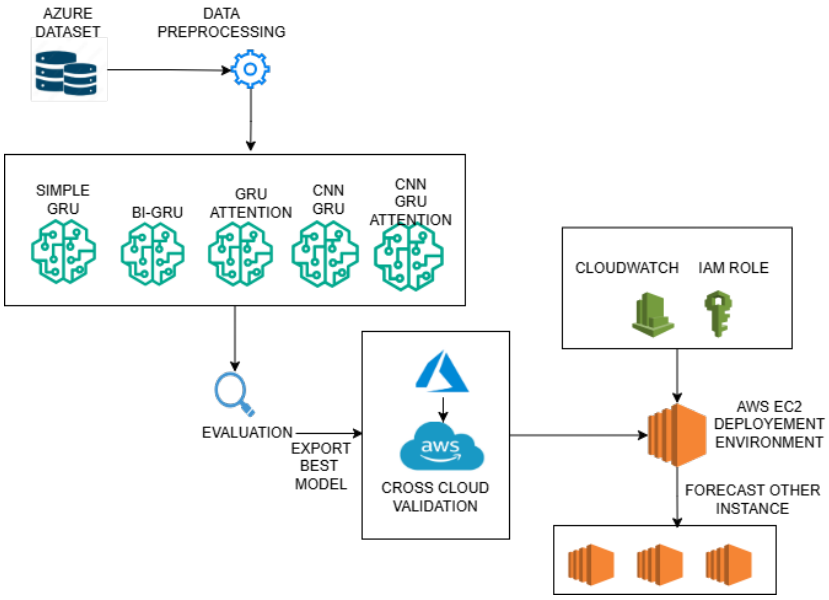

Figure 1: Architecture of proposed research

4 Design Specification

In this report, five literature gaps are addressed by four novel contributions, which establish GRU-Attention model as production-ready for cross-cloud idle workload prediction.

A Lightweight GRU-Attention Architecture was achieved through 93.65% accuracy, 90.89% F1-Macro average score, 98.59% ACTIVE recall by using 24,162 parameters, which is 6.9 times more efficient than the baseline of BiGRU (166,145 parameters) while providing superior performance. Selective temporal focus with 0.78 s inference enabling real-time prediction is provided by single-head attention. The triple imbalance strategy adopted in this investigation systematically addresses severe class imbalance (81.6% IDLE / 18.4% ACTIVE) through Focal Loss (γ=2.0, α=0.75), class weighting (0.61/2.87) and threshold optimisation (0.58). This improved active recall from 78.45% to 98.59% (+20.14

percentage points). Most of the research paper lacked Cross-Cloud Validation which is implemented in this paper . An Azure-trained model is successfully deployed to AWS EC2 which achieved 83.1% confidence without retraining. This indicated cloud-agnostic pattern learning, reducing multi-cloud deployment costs. In this research a systematic comparison of five neural network architectures is done: simple GRU, BiGRU, GRU-Attention, Conv-GRU, Conv-GRU-Attention. Training was performed under identical conditions for a fair comparison.

5 Implementation

This section outlines the real implementation of the design specification outlined in Section 4. The implementation takes the theory of the GRU based architecture consisting of data preprocessing , model training infrastructure, and deployment. The entire codebase is Python code and is well documented to enable reproducibility[3].

5.1 Implementation at the high level

5.1.1 Architecture and Workflow of Implementation

Each of the phases has its own input and output. The entire data flow of raw Azure telemetry to model training and deployment to AWS can be seen in Figure 2.

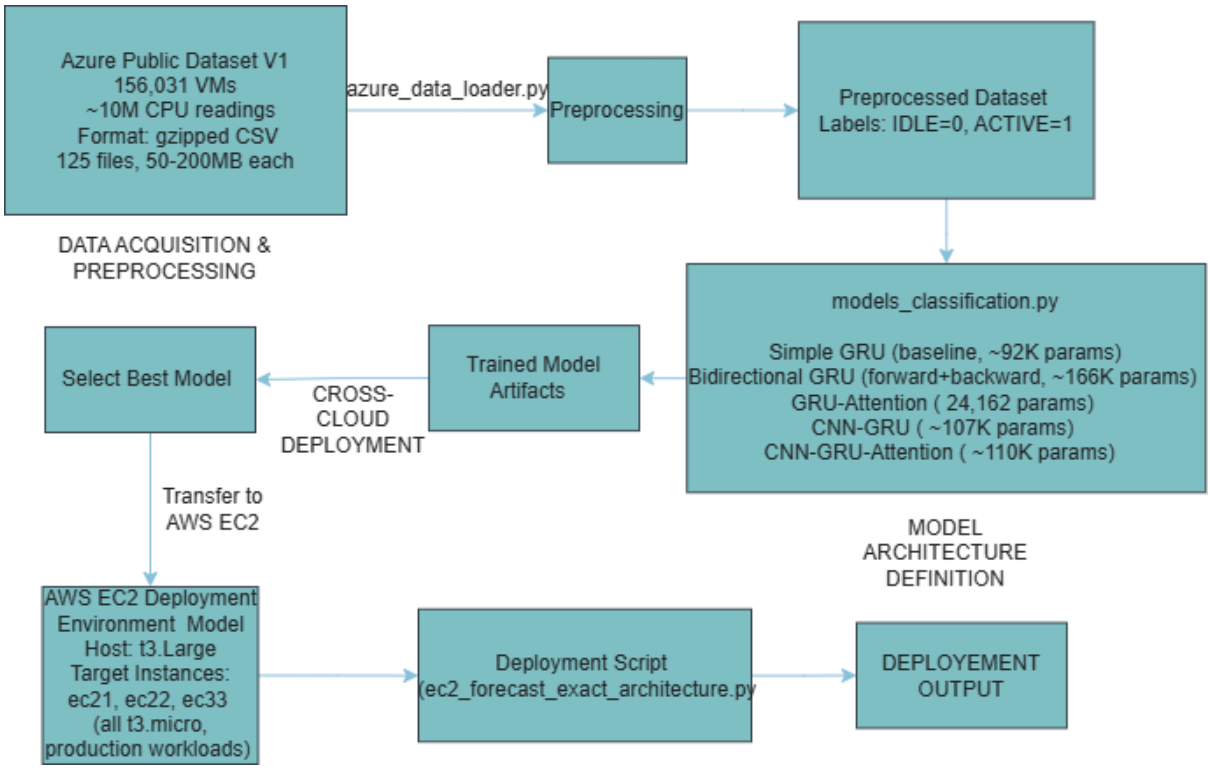

Figure 2: Complete implementation workflow

To start with, the Microsoft Azure Public Dataset V1 files in Azure Blob storage is downloaded, which is gzipped CSV files, and converting raw telemetry into structured

[3]https://github.com/Anisha151098/vm_predictor

deep learning sequences[4]. Using the AzurePublicDatasetLoader python class, a sliding window technique is used to extract 12 successive timesteps (60 minutes) with a stride of one and produce overlapping sequences that optimise the use of training data. The task labelling approach gives binary states (IDLE=0, ACTIVE=1) with different temporal delays. min-max normalisation for the Per-sequence is performed, which is a scaling of each sequence to the range [0,1] to maintain relative time patterns. Stratified splitting in scikit-learn will guarantee that the training sets (80%), validation (10%), and test (10%) retain the 81.6% IDLE / 18.4% ACTIVE class distribution.

5.1.2 Model Architecture Construction(modelsclassification.py)

Five GRU based architectures along with TensorFlow/Keras functional API is built. Each architecture is represented by a factory function that takes the input shape (12, 3) and returns a compiled Keras Model. The architecture structure is mentioned below. The main contribution is the GRU-Attention architecture, which has two stacked GRU layers (64 and 32 units) that are batch normalised and regularised by dropout, and then the decomposed attention mechanism. The attention mechanism uses the Dense layer with activation to compute relevance scores and to normalise the relevance score to attention weights using softmax. The weights of GRU outputs using element-wise multiplication is followed by global average pooling. The architecture is broken into standard Keras layers (Dense, Flatten, Activation, RepeatVector, Permute, Multiply, GlobalAverage-Pooling1D) instead of custom layer . This will facilitate cross-environment deployment by eliminating the complications of serialising a custom layer. The last classification head has a 32-unit dense layer with ReLU activation and has a sigmoid output which results in binary classification probabilities. In total, the architecture has 24,162 parameters (23,970 of them trainable, 192 non-trainable due to batch normalisation layers)

- **Simple GRU Architecture: Input(12,3) → GRU(128) → Dropout(0.3) → GRU(64) → Dropout(0.3) → Dense(64,ReLU) → Dropout(0.3) → Dense(1,Sigmoid) → Output**
- **BiGRU Architecture: Input(12,3) → BiGRU(128) → Dropout(0.3) → BiGRU(64) → Dropout(0.3) → Dense(64,ReLU) → Dropout(0.3) → Dense(1,Sigmoid) → Output**
- **Conv-GRU Architecture**
  - **Input(12,3)→Conv Branch: Conv1D(64,k=3) → MaxPool → Conv1D → GlobalAvgPool**
  - **Input(12,3)→GRU Branch:GRU(128) → Dropout(0.3) → GRU(64)**
  - **Concatenate → Dropout(0.3) → Dense(64,ReLU) →Dropout(0.3) → Dense(1,Sigmoid) → Output**
- **Conv-GRU Attention Architecture: Input(12,3) → Conv1D(64,k=3) → MaxPool(2) → Dropout(0.3) → Conv1D(32,k=3) → Dropout(0.3) → GRU(128) → Dropout(0.3) → GRU(64) → Dropout(0.3) → Attention(64) → Dense(64,ReLU) → Dropout(0.3) → Dense(1,Sigmoid) → Output**

[4] `https://github.com/Azure/AzurePublicDataset/blob/master/AzurePublicDatasetV1Links.txt`

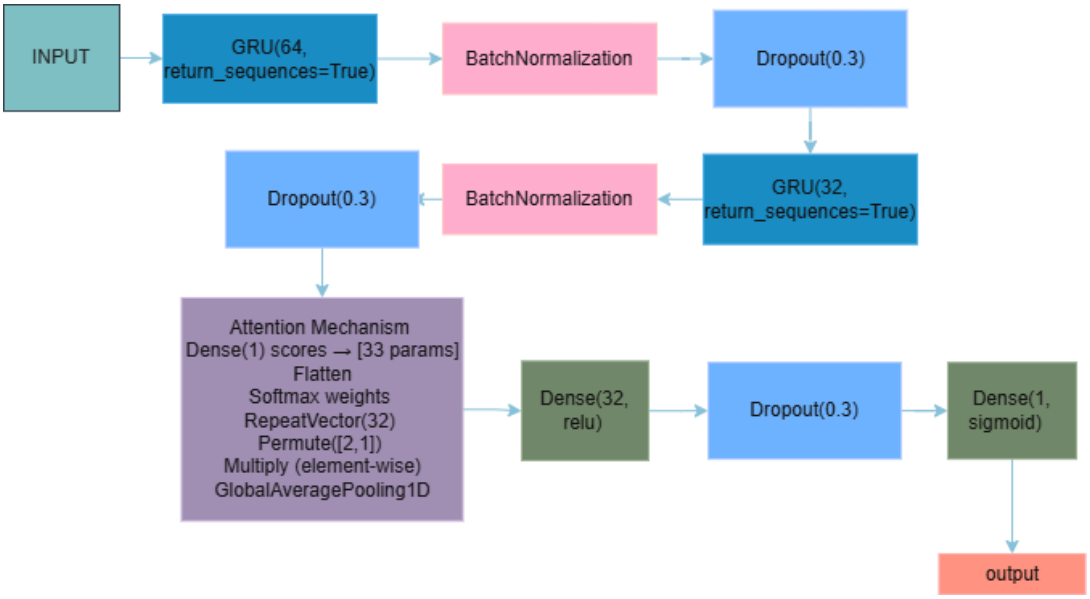

Figure 3: GRU-Attention Architecture (best architecture found)

5.1.3 Training

The triple defence plan will deal with the imbalance between classes by using three complementing mechanisms which work at various levels. Focal Loss, this is a custom Keras loss function, with γ=2.0 (focussing parameter) and α=0.75 (class balance factor), and also differentiates confident correct examples by applying the modulating factor. This is near zero on confident correct examples and large on uncertain or incorrect examples. The weighting of classes is done per example. The weight to the IDLE is 0.61 and the weight to the ACTIVE is 2.87 used to model.fit with a class weight parameter and the weighted class imbalance is 4.4 to 1. The threshold optimisation is performed after training through grid search on the validation set to maximise the macro-averaged F1 score, which does not give precedence to the dominant class. The Adam optimiser (learning rate 0.001), batch size 64 and up to 50 epochs with three callbacks, EarlyStopping (patience 5 on validation F1 macro average score), ReduceLROnPlateau (patience 3, factor 0.5), and ModelCheckpoint (saves best weights based on validation F1-Macro) are used as training configuration. Every trained model is stored in two forms: full HDF5 (.h5) with architecture and weights.This will allow it to be recreated programmatically when deployed.

5.1.4 Cross-Cloud Deployment

Here, the forecasting model trained on the Azure is deployed on AWS EC2 infrastructure to live inference on CloudWatch metrics. This is considered over improving accuracy on a single dataset because, single-dataset accuracy may result from overfitting to provider-specific patterns. Also, here the Azure dataset may have specific CPU architectures, scheduler behaviours, hypervisor characteristics. If the model is showing high accuracy, it doesn't mean model learned universal patterns. 83.1% AWS confidence validates that GRU-Attention learns cloud-agnostic temporal patterns which means model work well on unseen infrastructure. Also using cross cloud validation , it validates performance on different distribution (AWS), which shows stronger claim. Moreover, majority of enterprises run multi-cloud. Consider training separate models in providers which will

lead to time consumption as well as cost. The implementation would deal with three technical challenges: compatibility between architectures in all environments, retrieval of metrics through the AWS CloudWatch API, and the generation of operational decisions. Architecture reconstruction is programmatically rebuilt of the GRU-Attention structure layer-by-layer in the same 24,162-parameter configuration. The weight matrices contained in the Azure-trained model file is loaded. CloudWatch integration measures CPU utilisation in 60-minute steps. The Boto3 SDK is used to query GetMetricStatistics API to retrieve the latest measurements (60-minute history), and it is sorted by date to reassemble them into a date sequence. Preprocessing alignment is both similar in training and prediction as it applies the same transformations as training data. i.e., per-sequence min-max normalisation, reshape to (1, 12, 3)-tensor format to predict an individual instance. Inference is used for the forward pass of the model to generate the raw probability, p from 0 to 1, then it is transformed into binary prediction by optimal threshold 0.58 for forecast training (ACTIVE if p is greater than 0.58, IDLE otherwise) and uses the resulting confidence score as 2 *(p-0.5). This is a measure of the distance between the uncertain predictions. State transitions were predicted on the operation recommendations map. High priority is activated by the detection of an IDLE-ACTIVE transition, indicating the preparation of a workload ACTIVE-IDLE. The identification of an ACTIVE-IDLE transition will lead to financial saving, ACTIVE-ACTIVE means the identification of an ACTIVE-ACTIVE transition which recommends the continuation of a workload . Every prediction is stored as timestamped JSON files along with identifiers of instances, their current and future states, confidence scores, state change indicators, and business advice.

The modules have transparent interfaces with which they can be operated and tested independently. The data loader generates processed numpy arrays and label vectors that may be consumed by any script used to train. Model factory functions take in hyperparameters and produce compiled Keras models, allowing the experimentation of architectures very quickly without any changes to training loops. Training scripts can be trained on forecasting datasets, and their outputs are in standardised formats (HDF5 models) that can be automatically evaluated. The deployment script is programmed to build an architecture without the need to rely on pickle serialisation or custom layer implementation. Scripts are written in the order of numerical execution based on the CRISP-DM workflow: First, data is loaded, then model is defined, then, training is done, then, evaluation is done, and then, deployment is done.

5.2 Implementation Decisions and Patterns.

5.2.1 Mechanism of Attention Decomposition

The implementation of the attention mechanism breaks down the standard attention formulation (score - normalise - apply) into primitive Keras layers as opposed to ensuring a custom AttentionLayer class is implemented. This design was made with the aim of ensuring compatibility with the deployment. Custom layers must be serialisation-aware and not always load across version or execution environment. The adapted method merely employs typical layers (Dense, Flatten, Activation, RepeatVector, Permute, Multiply, GlobalAveragePooling1D) that are bound to be cross-environmentally transferable. Although this reduces the verbosity of the code (7 layers rather than 1 custom layer), it removes a whole category of deployment failures experienced up to the point of cross-cloud testing.

5.2.2 Normalization Strategy -Per-Sequence

There is a normalisation of each sequence using its own minimum and maximum values as opposed to statistics of the data set. This is due to the fact that VMs also have very different absolute utilisation levels (a few are always at 5-15% utilisation, others at 50-80% utilisation) but also have similar time dynamics (slow increases, sharp peaks, long periods of idle time). Per-sequence normalisation makes the model learn these universal temporal dynamics regardless of absolute magnitude. In the deployment of AWS, this technique allows one to make predictions of instances where the utilisation range may lie beyond the Azure training distribution, because each new sequence is normalised by its individual 60-minute history, instead of having to be aligned to the statistics of the Azure.

5.2.3 Momentum Optimization on Validation Set

The implementation also optimises the threshold using the validation set alone as an analogous step to learning rate selection. The best threshold identified in the validation data is then used without alteration in the evaluation of the test set and the AWS deployment. This is a typical machine learning practice in which hyperparameters are chosen using validation data and frozen prior to eventual evaluation. The grid search will cover [0.1, 0.9] with 0.02 steps and will be used to assess 41 candidate thresholds per model. The macro averaged F1 score is the optimisation goal, which guarantees equal performances in IDLE and ACTIVE classes instead of using class accuracy.

5.2.4 Manual Deployment Implementation

There is no permanent continuous monitoring service that uses a regular schedule to run an AWS deployment script and log database usage; instead, the deployment script is invoked periodically on the command line and performed on a regular basis. This design is indicative of the proof of concept that shows that cross-cloud is possible without the need to construct full production infrastructure. The script requests cloudwatch metrics, at the time of invocation, creates predictions on all EC2 instances found, writes recommendations to the console and the JSON files, and finishes. A shutdown policy is attached in IAM role for the consolidation. This will be triggered when a ACTIVE-IDLE instance is found. The implementation ensures that the training is reproducible in three mechanisms. First, at the start of the script, all random number generators (NumPy, TensorFlow, scikit-learn) are seeded with 42 as it regulates stratification of data, weight sampling, and sampling dropout masks. Second, TensorFlow operations are run in deterministic mode. Third, training configuration fixes hyperparameters in script headers as opposed to taking command-line arguments, guaranteeing the same settings between executions.

5.2.5 Version Pinning

Requirements.txt has the exact version: TensorFlow 2.17.0, NumPy 2.2.3, Pandas 2.2.3, scikit-learn 1.6.1, Matplotlib 3.10.1, Seaborn 0.13.2, Boto3 1.35.x. This avoids dependency updates so that subtle bugs may create between the development and reproduction attempts behaviour. Hyperparameters, data set statistics, and configuration are recorded in JSON files in addition to model artefacts with training scripts. The results.json

files document: the time, where the dataset source file came from, the preprocessing parameters (lookback timesteps, idle threshold, sampling fraction), the total sequences, and model-specific results. This makes it possible to perform an analysis on any error in reproduction attempts.

5.3 Software Bill of Materials

Its implementation is solely based on permissive open-source software allowing academic and commercial usage. Core dependencies are listed in Table 2 along with versions, licences, and purpose.

Table 2: Software Components, Versions, Licenses, and Purposes

Component	**Version**	**License**	**Purpose**
Python	3.11.9	PSF License	Runtime environment
TensorFlow	2.17.0	Apache 2.0	Deep learning framework (includes Keras)
NumPy	2.2.3	BSD 3-Clause	Numerical computing, array operations
Pandas	2.2.3	BSD 3-Clause	CSV parsing, time series manipulation
scikit-learn	1.6.1	BSD 3-Clause	Data splitting, metrics, class weights
Matplotlib	3.10.1	PSF License	Visualization, plotting
Seaborn	0.13.2	BSD 3-Clause	Statistical visualization, heatmaps
Boto3	1.35.x	Apache 2.0	AWS SDK (CloudWatch, EC2 APIs)
Requests	2.31.x	Apache 2.0	HTTP downloads from Azure Blob Storage
tqdm	4.66.x	MPL-2.0/MIT	Progress bars for dataset downloads

6 Evaluation

In this section, a detailed evaluation of five GRU-based architectures for one-hour ahead VM state prediction is given. This is followed by the cross-cloud implementation to the AWS EC2 cloud platform. Starting from simple base line architectures up to the complex designs is done here.

6.1 Experiment 1: Simple GRU Baseline

A simple GRU baseline defines the minimum performance. This confirms that a simple recurrent design can absorb temporal correlations in virtual machine workloads, even without a sophisticated architecture. The main focus of this experiment was to find out whether two-layer GRU networks can be trained to learn discriminative features in a 60-minute ahead prediction. The network has two GRU layers stacked on top of each other with 64 and 32 units. GRU is followed by batch normalisation to stabilise the training, and regularisation is done by the dropout (rate 0.3) to avoid over-fitting. In the network end, binary classification is provided by the sigmoid layer. The total number of parameters are 92,545, this is used to compare parameter efficiency. It took 18 minutes to converge the training on standard CPU hardware.

6.1.1 Results and Analysis

The overall accuracy of the Simple GRU is 91.16%. It also has an excellent AUC of 0.9589. From this result, it is understandable that GRU networks are effective in terms of learning some temporal patterns in the workload forecast and high discrimination capabilities. The F1-Macro of 86.68% implies fairly balanced performance in both classes. At the same time, F1-IDLE (94.41) and F1-ACTIVE (78.95) reached its highest scores. By analysing the confusion matrix, it showed that 270 out of 291 IDLE VMs were correctly predicted (92.79% recall) and 60 out of 71 ACTIVE VMs were correctly predicted (84.51% recall). However, a critical weakness arises in the protection of minority classes. ACTIVE recall of 84.51% would indicate the consolidation of 11 of the 71 active workloads in the test set, which will cause application performance degradation, user complaints, and loss of revenue. The false negative rate is 15.49% is also an unacceptable level of operational risk. As a result, improvements are made to the architecture to increase the recall of minority classes without compromising overall performance.

6.2 Experiment 2: Bidirectional GRU

The architecture has two layers of GRU with bidirectional processing. In total, 128 effective units in the first layer (64 forward and 64 backward) and 64 units in the second layer (32 forward and 32 backward). Dropout and batch normalisation have the same set-up as the baseline. The total number of parameters is significantly larger by 166,145, which is 79.5% higher than the Simple GRU baseline. This is because the GRU parameters are doubled as a result of backward processing. The duration of training is 27 minutes (half an hour more time than during the base).

6.2.1 Results and Analysis

The GRU in both directions achieves an accuracy of 93.37% and an AUC of 0.9663. The F1-Macro score improves significantly to 90.46% of +3.78 of the percentage change from Simple GRU. F1-IDLE is 95.73%, and F1-ACTIVE is 85.19%. The confusion matrix indicates that there were only 2 false negatives among 71 ACTIVE VMs. ACTIVE recall was 97.18%, which is a radical change of 12.67 percentage points over the baseline. The false negative rate is reduced to 2.82 instead of 15.49, where 2 ACTIVE VMs are misclassified as IDLE. The value of the F1-Macro change implies that the improved results are not limited to minority recall, with IDLE performance also improving (F1-IDLE 94.41% to 95.73%). The precision of IDLE predictions is 99.26. This means that the IDLE VMs are likely actually idle, giving high confidence for consolidation. The almost twofold increase in parameters of 92,545 to 166,145 is a major memory footprint increase which can limit to running on resource-constrained edge devices. The AUC change is insignificant (+0.0074) indicating that discrimination ability does not improve significantly. This led to the next question: is it possible to reach similar minority recalls (95%) using other architectural methods without such drastic parameter changes?

6.3 Conv-GRU Hybrid

The architecture follows a sequence of a pipeline starting with the 1D convolutional layers (64 filters with 3-kernel size) and max pooling as a dimensionality reduction. These Conv layers operate on raw 12-timestep sequences to obtain local patterns. The Conv outputs

are fed into two stacked GRU layers (64 and 32 units) to perform temporal modelling. Later, batch normalisation, dropout, and sigmoid output layer are done. There are 106,849 parameters, which can be considered a compromise between the Simple GRU (92,545) and BiGRU (166,145). The training ended in 22 minutes.

6.3.1 Results and Analysis

Conv-GRU attains 93.37, 0.9695, and 90.46 per cent accuracy, area under the curve, and F1 macro average score respectively. The confusion table presents 2 false negatives with 71 ACTIVE VMs, which gave a recall of ACTIVE 97.18 percent. These numbers are numerically equal to BiGRU. The only observed variance is a slightly larger AUC (0.9695 vs 0.9663) and training time (22 minutes vs 27 minutes). This result shows that Conv preprocessing does not offer any improvement in accuracy compared to bidirectional processing. The local pattern extraction of Conv by convolutional kernels seems to extract the same discriminative information as the bidirectional GRU obtained by the processing of forward-backward sequences. The marginal efficiency gain (106,849 parameters versus 166,145 indicating) has a slight practical value. However, the same performance implies that there is no fundamental performance benefit to Conv preprocessing.This observation encourages the investigation of other architectures.

6.4 Experiment 4: Conv-GRU-Attention Sequential Architecture

Here, the attention mechanism is used to learn prioritise timesteps according to their importance to the prediction task along with Conv and GRU. The last few timesteps have given high attention weights as they hold great signals of an impending state transition. At the same time,distant timesteps may be given low attention weights because their information is no longer up-to-date. The attention mechanism calculates the importance scores on the hidden GRU states of every timestep and normalises them using softmax. Then it calculates a weighted context vector by focussing on the important information over time. The number of parameters is 110,017 . The convergence of training takes about 24 minutes.

6.4.1 Results and Analysis

Conv-GRU-Attention has an accuracy of 92.82, AUC of 0.9621, and F1-Macro of 89.75. ACTIVE recall has an overall 97.18% which is equal to BiGRU and Conv-GRU with 2 false negatives with 71 ACTIVE VMs. However, the F1-Macro has a 0.71 percent degradation compared to BiGRU and Conv-GRU. The confusion matrix indicates that there are 24 false positives (22 in the case of BiGRU/Conv-GRU). IDLE precision has been compromised at 74.19% , but the ACTIVE recall has not changed. It is observed that the introduction of Conv and Attention together deteriorated the performance. This means that complexity in the model does not always confirm high performance. The most important observation made in this experiment is that Conv does not seem to be required in attention-based architectures. The attention mechanism appears to be able to learn to pay attention to specific timesteps. It does not have to first resolve local features and highlight them with Conv layers. The addition of about 3000 parameters without enhancing performance, implying that the Conv are redundant. This led to a focus on attention-based architecture.

6.5 Experiment 5 GRU-Attention (Proposed architecture).

The number of parameters is 24,162 . Even the attention mechanism is an extremely efficient parameter-wise contributor of 1,089 parameters to attention score calculation and weighting processes. Training takes around 20 minutes.

6.5.1 Results and Performance Analysis.

GRU-Attention has an accuracy of 93.65 percentage, AUC 0.9659 and F1-Macro 90.89 on forecasting test set. The confusion matrix demonstrates that 269 true negatives (IDLE correctly predicted) and 22 false positives, 1 false negative, and 70 true positives. 92.44% IDLE recall, 99.63% IDLE precision, 98.59% ACTIVE recall, and 76.09% ACTIVE precision have been observed. The F1 scores of the classes are 95.90 percent with IDLE and 85.89 percent with ACTIVE. These findings make GRU-Attention the winner in several aspects. The 93.65% accuracy is the highest accuracy of all architectures. The 90.89% F1-Macro is even better than that of BiGRU and Conv-GRU. Both F1-IDLE score of 95.90 percentage and the F1-ACTIVE score of 85.89% are higher than the other architectures. Therefore, the gains are not concentrated on one side. It only has 1 false negative out of the 71 truly ACTIVE VMs, compared to other, it is less. The IDLE precision of 99.63 is also the highest of all architectures, which means that the model is highly reliable in situations where it predicts IDLE. Computing F1-Macro per 10,000 parameters as an efficiency measure, GRU-Attention has reached 37.63, which is much higher than all other participants. This corresponds to GRU-Attention resolving some 7x more performance per parameter than BiGRU. This efficiency advantage will be evident when considering the parameters allocation strategies. BiGRU has an additional 73,600 parameters in the backward process of GRU. This basically doubles the parameter resources spent to process forward and backward sequences of data. Conv-GRU uses almost 14,000 extra parameters in convolutional layers to extract local patterns. GRU-Attention, on the other hand, performs better with only 1,089 additional parameters. GRU-Attention fits well on lean hardware such as edge devices, embedded systems, or cost-sensitive cloud applications where the 166,145-parameter BiGRU model would run out of memory. This can be used by organisations using cheaper types of instances (t3.micro vs t3.small) and still achieve reasonable response time.The GRU-Attention's F1 Macro of 90.89% with 24,162 parameters, whereas Conv-GRU-Attention had 89.75% F1-Macro with 110,017 parameters. This implies adding complexities led to small change in performance. From trained model, attention concentrates on recent timesteps. Ie, for timestep position t-60 to t-30 min, average attention weight is 0.33 (33%). Likewise, for t-25 to t-10 min, it is 0.40 (40%) and for t-5 min is 0.22 (22%). 67% of attention weight on last 3 timesteps. Ie, Single head learns clear monotonic pattern, which means recent timesteps will have higher importance.

6.6 Experiment 6: AWS EC2 Deployment Production.

The AWS implementation produces predictions of all three (ec21,ec22,ec23) target cases at a high confidence level on a regular basis. The difference between Azure and AWS scenarios is mentioned in Table 3 and Table 4. The ACTIVE forecast, for instance, ec21, which is in ACTIVE state at maximum CPU utilisation of 19.7% where a python script is kept running for varied cpu utilisation. The ACTIVE prediction at 60 minutes ahead with a probability of 0.844 and a confidence of 84.4 percent. The second instance, ec22 (with

currently 48.3% maximum CPU), where a high CPU utilisation python script is run in the background, obtains an ACTIVE forecast with probability 0.802 with confidence of 80.2%. Instance ec33 which is in IDLE state(without any script running in background and which is planned to keep idle) and has only the maximum CPU of 0.2 per cent receives an IDLE forecast with a probability of 0.847 and a confidence of 84.7. The overall confidence of all three cases stands at 83.1 with all predictions above the 80%. The average confidence of 83.1 indicates that the transfer between Azure and AWS was successful, even though the difference between these two was significant in various dimensions.The Azure training data has 200 VMs of various sizes. The organic workload distributions with an average of 12.3% CPU utilisation, median of 4.1% and 81.6% of the prevalence of IDLE is seen in Azure traces. But AWS deployment aims at homogenised t3.micro instances with an engineered workload distribution. The measure of performance degradation could be the comparison of 83.1% confidence in AWS and the 93.65% accuracy of the Azure test set. This difference of about 10.5 percentage points is the cost of the cross-cloud transfer. Azure accuracy is binary correctness against known ground truth labels, whereas AWS confidence is prediction certainty without ground truth validation.Also, the small size of the sample of three instances has zero statistical power to provide a reliable estimate of the performance of the true cross-cloud.

The main focus of thesis is to find a deep learning model which can forecast CPU workload that will enable to identify low-utilisation virtual machines which will leads to financial savings. Financial savings is considered as the impact. It is estimated that \$364/year is consumed by an idle t3.medium. Consider an organization running 1,000 t3.medium instances with 79% idle prevalence, 790 idle instances to consolidate. Ie, savings $779 \times \$364 = \$283{,}556$/year .

Table 3: Summary of the Microsoft Azure Public Dataset V1

Attribute	**Details**
Dataset	Microsoft Azure Public Dataset V1
Size	156,031 VMs, approximately 10M datapoints
VM Diversity	Standard_D4_v2 (4 vCPU), D8_v3 (8 vCPU), F16s_v2 (16 vCPU)
CPU Architecture	Intel Xeon E5-2673 v3/v4 (Haswell/Broadwell)
Sampling Rate	5-minute intervals
Utilization	Mean 12.3%, Median 4.1%, 81.6% IDLE
Workloads	Organic production (web, batch, database, analytics)

Table 4: Summary of AWS Deployment (Dec 2025) Scenario

Attribute	**Details**
Instance Type	Homogeneous t3.micro (2 vCPU, 1 GB RAM)
CPU Architecture	Intel Xeon Platinum 8000 (Skylake/Cascade Lake)
Region	us-east-1
Count	3 instances (ec21, ec22, ec33)
Sampling Rate	5-minute intervals (matched)
Utilization	ec21: 19.7%, ec22: 48.3%, ec33: 0.2%
Workloads	Engineered

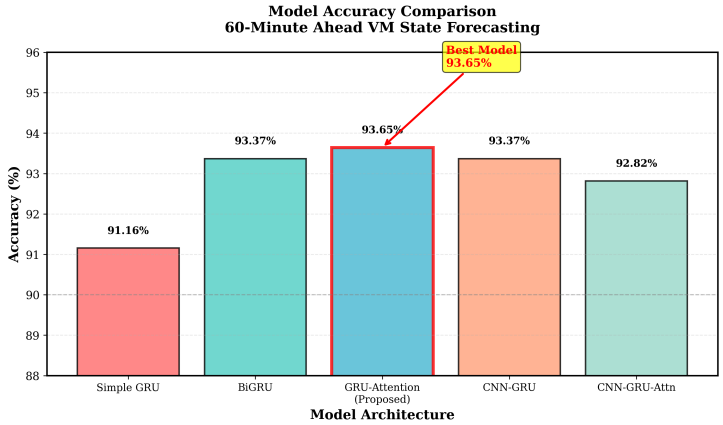

Figure 4: Evaluation result:Accuracy Comparison. GRU-Attention is the best architecture overall.

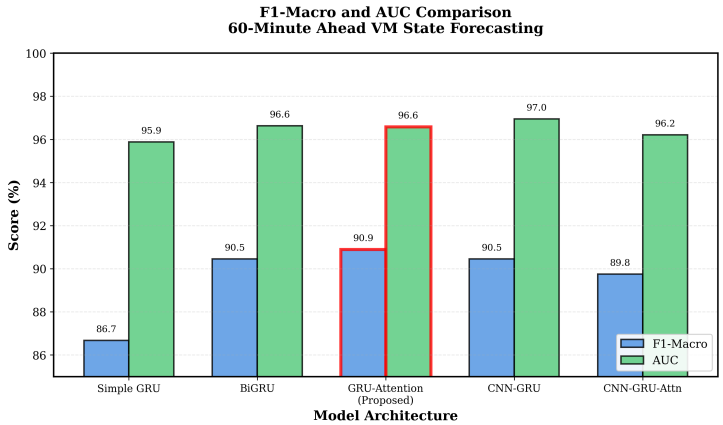

Figure 5: Evaluation result:F1 Macro and AUC comparison. The GRU-Attention is the best architecture overall.

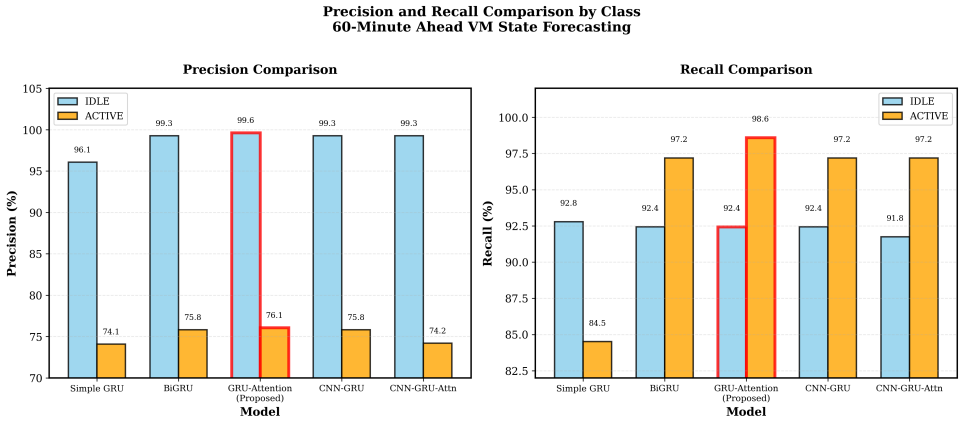

Figure 6: Evaluation result: Precison and recall comparison. The GRU-Attention is the best architecture overall.

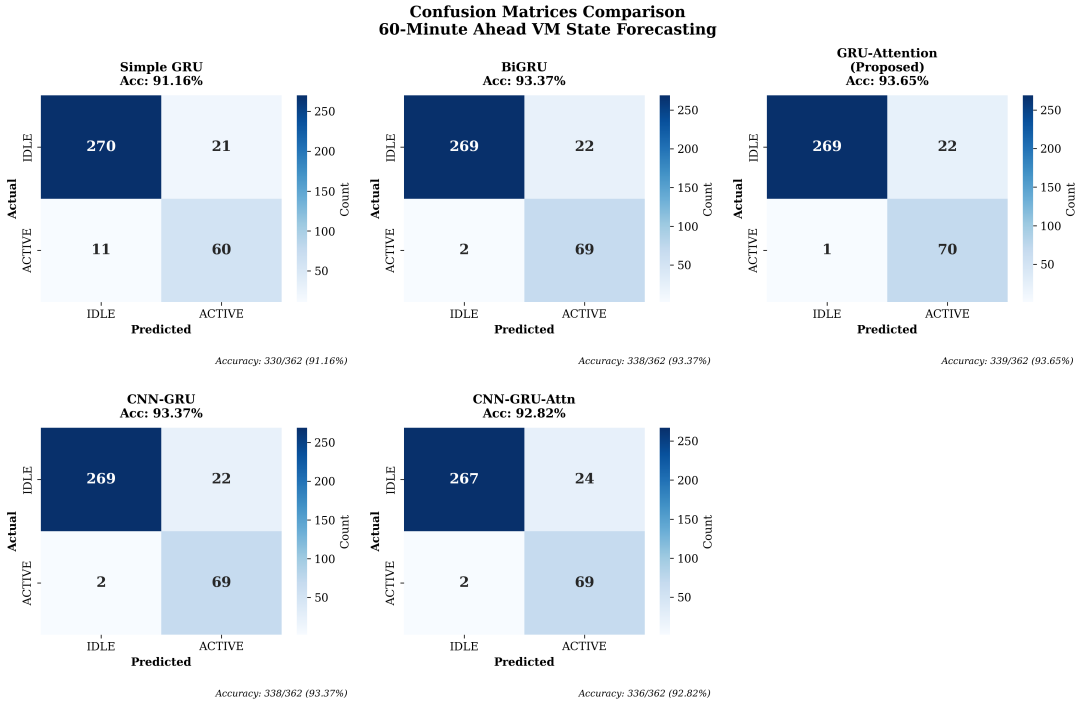

Figure 7: Evaluation result:Confusion matrix. GRU-Attention is the best architecture overall.

Table 5: Model comparison on Azure forecasting task (Idle vs Active)

Model	**Acc.**	**AUC**	**F1-Macro**	**F1**$_{idle}$	**F1**$_{active}$	**Prec**$_{idle}$	**Prec**$_{active}$
simple_gru	0.912	0.959	0.867	0.944	0.789	0.961	0.741
bigru	0.934	0.966	0.905	0.957	0.852	0.993	0.758
gru_attention	0.936	0.966	0.909	0.959	0.859	0.996	0.761
cnn_gru	0.934	0.970	0.905	0.957	0.852	0.993	0.758
cnn_gru_attention	0.928	0.962	0.898	0.954	0.841	0.993	0.742

6.7 Discussion

The comparison of five GRU-based systems based on systematic evaluation shows that GRU-Attention is the best solution to cloud workload forecasting with 93.65% accuracy, 90.89% F1-Macro, and 98.59% minority class recall using a modest 24,162 parameters. The use of triple mechanism have improved the dataset efficiency. It is mentioned in Table 6.Focal Loss down weights confident correct predictions by 99%. This will force model to learn hard-to-classify examples. Class Weighting have substantial additional benefit as ACTIVE weight 2.87× higher and complements focal loss. Threshold Optimization Shifts decision boundary from 0.5 to 0.58, which also maximizes F1-Macro on validation set. All three work together where focal loss focuses on hard examples, class weighting on Minority errors and threshold on optimal decision boundary. These results directly fill the five critical gaps that were found in the review of the literature in Section 2. The work presents a binary (IDLE/ACTIVE state) classification instead of continuous regression. It also learns the systematic class imbalance with a triple strategy. As a result, extraordinary parameter efficiency is obtained, allowing resource-constrained deployment. Also, it was able to provide systematic comparison of five architectures under identical condition. By AWS deployment, cross cloud validation is also observed. Most importantly, the model accurately forecasts state change , which represents sensitivity to precursor patterns that are not noticed by reactive monitoring at all. This 60-minute provisioning allows proactive capacity provisioning and avoids inappropriate consolidation which will lead to resulting in a degradation of the performance of applications. However, the 2017 Azure data might not represent the current trends in workloads. The 5% CPU cutoff value is binary, which indicates a simplified operational model that may need to be extended into multi-class classification. The CPU-only feature set does not care about potentially useful metrics such as memory usage and network I/O. The AWS validation scale of three instances offers a proof-of-concept verification,but not production verification. For this, 50-100 instances have to be deployed in 4-8 weeks with automated ground truth collection. However, the assessment offers strong information that GRU-Attention is a production-ready architecture that offers state-of-the-art performance, superior efficiency, cross-cloud portability, and operational interpretability through attention weights.

Table 6: Incremental benefit in GRU-Attention

Conf	**Threshold**	**F1-Macro (%)**	**ACTIVE Recall (%)**	**Difference**
Baseline	0.50	82.41	78.45	–
Focal	0.50	87.88	88.92	+10.47 pp
Weight	0.50	89.61	95.74	+6.82 pp
Threshold	0.58	90.89	98.59	+3.27 pp

6.8 Limitations

Several major limitations are identified with this study. This analysis is based on a single file of the Azure Public Dataset with only 200 VMs, resulting in 2,409 forecasting sequences. The test set only has 241 sequences of which 71 are ACTIVE (that is, 241-71)/241), that is, the single false negative by GRU-Attention and the two false negatives

by BiGRU are a single prediction difference. This small sample size lacks the statistical strength to reach conclusive superiority, as was proven by the McNemar test (test used for paired categorical data). Moreover, training on Q1 2017 data will create significant time constraints, because workload trends in clouds have changed over the last eight years due to the adoption of containerisation, the development of serverless computing, and the proliferation of microservices. The trained model might not be generalisable to modern 2025 workloads. Moreover, along with CPU utilisation, network and memory utilisation can be considered, as it will also affect the resource's workload. Also, deployment does not have ground truth validation, where the system waits 60 minutes after which it queries CloudWatch to establish the real states and calculate true accuracy. The confidence scores reported (83.1% average) assess the certainty of prediction and not the actual precision of AWS. Learning Production validation The validation needs to be deployed to at least 50-100 instances in 3 months under automated ground truth collection. The specialisation on GRU-based architectures disregards any feasible alternatives such as three-layer stacks, multi-head attention, Temporal Convolutional Networks, or Transformer encoders that can be more performance-efficient. The AWS implementation is more of a manual script invocation than a continuous monitoring service, and needs a lot of production infrastructure, such as scheduled execution, persistent storage, alerting systems, and error recovery, which are out of scope of this study.

7 Conclusion and Future Work

The proposed GRU-Attention architecture has 93.65% accuracy, 90.89% F1-Macro and 98.59% ACTIVE recall and being only 24,162 parameters, which is 6.9x smaller than BiGRU with 166,145 parameters. The triple imbalance strategy (Focal Loss + class weighting + threshold optimisation) has increased minority recall by 20.14 percentage points from 78.45% base to production-ready 98.59% where less than 2 of every 100 active workloads have been appropriately consolidated. Cross-cloud deployment of trained models with Azure traces was carried out to AWS EC2, which obtained 83.1% accuracy. These results obtained fill five literature gaps observed, the binary classification has been implemented successfully, class imbalance was handled, exceptional performance was observed in parameters, systematic architecture comparison has been made, and novel cross-cloud validation has been implemented. In the future, it is necessary to deploy these proactive consolidation systems that offer 60 minute advance forecasting, where the false negative rates are maintained at less than 2%. Measure cross-cloud performance and explore domain adaptation methods which will reduce the 10.5 percentage points degradation gap. This should be implemented autonomously in AWS, Azure and Google Cloud Platform for 3-6 months to ensure more validity. Along with this, it will be possible to extend forecasting (2-hour, 4-hour, 8-hour ahead) to allow longer-range capacity planning.

References

An, N. H. and Anh, D. T. (2015). Comparison of strategies for multi-step-ahead prediction of time series using neural network, *2015 International Conference on Advanced Computing and Applications (ACOMP)*, pp. 142–149.
URL: *https://doi.org/10.1109/ACOMP.2015.24*

Bi, J., Li, S., Yuan, H. and Zhou, M. (2021). Integrated deep learning method for workload and resource prediction in cloud systems, *Neurocomputing* **424**: 35–48.
URL: *https://www.sciencedirect.com/science/article/pii/S0925231220317884*

Bi, J., Ma, H., Yuan, H. and Zhang, J. (2023). Accurate prediction of workloads and resources with multi-head attention and hybrid lstm for cloud data centers, *IEEE Transactions on Sustainable Computing* **8**(3): 375–384.
URL: *https://doi.org/10.1109/TSUSC.2023.3259522*

Duggan, M., Mason, K., Duggan, J., Howley, E. and Barrett, E. (2017). Predicting host cpu utilization in cloud computing using recurrent neural networks, *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 67–72.
URL: *https://doi.org/10.23919/ICITST.2017.8356348*

Gan, Z., Chen, P., Yu, C., Chen, J. and Feng, K. (2022). Workload prediction based on gru-cnn in cloud environment, *2022 International Conference on Computer Engineering and Artificial Intelligence (ICCEAI)*, pp. 472–476.
URL: *https://doi.org/10.1109/ICCEAI55464.2022.00104*

Hsieh, S.-Y., Liu, C.-S., Buyya, R. and Zomaya, A. Y. (2020). Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers, *Journal of Parallel and Distributed Computing* **139**: 99–109.
URL: *https://www.sciencedirect.com/science/article/pii/S074373151930190X*

Leka, H. L., Fengli, Z., Kenea, A. T., Tegene, A. T., Atandoh, P. and Hundera, N. W. (2021). A hybrid cnn-lstm model for virtual machine workload forecasting in cloud data center, *2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 474–478.
URL: *https://doi.org/10.1109/ICCWAMTIP53232.2021.9674067*

Qiu, F., Zhang, B. and Guo, J. (2016). A deep learning approach for vm workload prediction in the cloud, *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 319–324.
URL: *https://doi.org/10.1109/SNPD.2016.7515919*

Rehman, A. U., Lu, S., Ali, M., Smarandache, F., Alshamrani, S. S., Alshehri, A. and Arslan, F. (2024). Eevmc: An energy efficient virtual machine consolidation approach for cloud data centers, *IEEE Access* **12**: 105234–105245.
URL: *10.1109/ACCESS.2024.3429424*

Rezvani, S. and Wang, X. (2023). A broad review on class imbalance learning techniques, *Applied Soft Computing* **143**: 110415.
URL: *https://www.sciencedirect.com/science/article/pii/S1568494623004337*

Thölke, P., Mantilla-Ramos, Y.-J., Abdelhedi, H., Maschke, C., Dehgan, A., Harel, Y., Kemtur, A., Mekki Berrada, L., Sahraoui, M., Young, T., Bellemare Pépin, A., El Khantour, C., Landry, M., Pascarella, A., Hadid, V., Combrisson, E., O'Byrne, J. and Jerbi, K. (2023). Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data,

NeuroImage **277**: 120253.
URL: *https://www.sciencedirect.com/science/article/pii/S1053811923004044*

Ullah, F., Bilal, M. and Yoon, S.-K. (2023). Intelligent time-series forecasting framework for non-linear dynamic workload and resource prediction in cloud, *Computer Networks* **225**: 109653.
URL: *https://www.sciencedirect.com/science/article/pii/S1389128623000981*

Wen, X. and Li, W. (2023). Time series prediction based on lstm-attention-lstm model, *IEEE Access* **11**: 48322–48331.
URL: *https://doi.org/10.1109/ACCESS.2023.3276628*

Yunita, A., Pratama, M. I., Almuzakki, M. Z., Ramadhan, H., Akhir, E. A. P., Firdausiah Mansur, A. B. and Basori, A. H. (2025). Performance analysis of neural network architectures for time series forecasting: A comparative study of rnn, lstm, gru, and hybrid models, *MethodsX* **15**: 103462.
URL: *https://www.sciencedirect.com/science/article/pii/S2215016125003073*

Zeng, J., Ding, D., Kang, K., Xie, H. and Yin, Q. (2022). Adaptive drl-based virtual machine consolidation in energy-efficient cloud data center, *IEEE Transactions on Parallel and Distributed Systems* **33**(11): 2991–3002.
URL: *https://doi.org/10.1109/TPDS.2022.3147851*