Deep Learning Techniques for Anomaly Detection in Cloud Computing

MSc Research Project
Cloud Computing

Serena Santosh
Student ID: x23246642

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Serena Santosh
Student ID:	x23246642
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	12/12/2024
Project Title:	Deep Learning Techniques for Anomaly Detection in Cloud Computing
Word Count:	6733
Page Count:	20

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:	Serena Santosh
Date:	11th December 2024

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):	

Deep Learning Techniques for Anomaly Detection in Cloud Computing

Serena Santosh
x23246642

Abstract

Cloud computing, which enables on-demand access of computing resources over the internet, is a paradigm that is undergoing a lot of changes. With a lot of organisations shifting to cloud, there is a vast amount of data and resources in cloud, which needs to be protected. This research proposes the usage of GRU-BERT (Gated Recurrent Unit - Bidirectional Encoder Representations from Transformers) model with self-attention mechanism to detect anomalous behaviour in cloud which is essential to increase the resiliency of the cloud. The possibility of using autoencoders with self-healing mechanism is also discussed in this research. The aim of this project is to implement a solution that can increase the security in cloud with increased performance in terms of cost, computational complexity, execution time, and energy consumption. GRU-BERT model was evaluated based on performance metrics and it outperformed LSTM (Long Short-Term Memory) in terms of cost by 46.55%. Optimisation of NAB (Numenta Anomaly Benchmark) by making use of standard performance metrics for evaluation generated better results with regards to reducing the false negatives. Bayesian optimisation technique was utilised to optimise the hyperparameters in GRU-BERT. This technique was implemented using Hyperopt, producing great results with increased overall performance in terms of validation loss.

1 Introduction

Cloud infrastructure is prone to different types of security issues that can be challenging to the users and cloud providers. A security breach and the resulting downtime would adversely affect cost, data, energy and labour and lead to huge losses. In this context, it is of utmost importance to detect any anomalies in the cloud as and when they occur, take appropriate measures to rectify them and also take the required steps to prevent them. Deep learning techniques could be used to detect and resolve unusual behaviours in the cloud efficiently and rapidly.

The computing resources are provided by the cloud service providers remotely, while computing is done over the cloud. Disruption of services and loss of data are the major issues here and hence strict measures have to be taken to strengthen the security of the data (Singh and Chatterjee; 2017, pp. 88-115). End users and cloud providers are both concerned about the security of their system. Virtualisation is a feature in the cloud that is subject to security threats. This is because the VMs (Virtual Machines) that are not properly isolated can be attacked easily. Multi-tenancy is another feature in the cloud that is susceptible to security issues since multiple users share the same cloud platform. Hence, maintaining high security in cloud computing is extremely important.

1.1 Background and Motivation

Many organisations are transitioning to the cloud and there is an increased demand for secured cloud services and solutions. There is great need for enhanced security for the data and transactions in the cloud and this is the motivation behind this research. Better processing power and security are provided by edge computing, but it is not cost effective while deploying at a big scale. Cloud computing is relatively inexpensive while considering the infrastructure requirement. Hence there is a huge amount of data on the cloud which is susceptible to cyber threats and attacks (Khan and Al-Yasiri; 2016, pp. 485-490). Since security is vulnerable on the cloud in connection with storage, virtualisation, and network, there is a great need for a resilient system that is secure enough to withstand such attacks.

Similar works in this direction were studied thoroughly and analysed critically. Though deep learning techniques have been used to enhance the security in the cloud, there is scope for improvement of the performance of the models used. It was identified that an ensemble model with self-attention mechanism and self-healing mechanism would enhance the ability to detect anomalies in the cloud. The review of the literature also led to the identification of the following research gap. Reduction of future security issues in the cloud by taking preventive steps and corrective actions have not been addressed adequately in the previous works.

1.2 Research Question

The above research problem motivates the following research question:

- How well can anomalies in public cloud infrastructure be detected by using the GRU-BERT model with self-attention mechanism to improve cloud security and protect end users?
- To what extent can autoencoders be used to predict patterns and reduce future security issues in the cloud?
- What are the contributions made to enhance the overall resiliency and market value by utilising self-healing mechanisms to automate processes in cloud security?

Identifying how well anomalies in public cloud infrastructure can be detected by using the GRU-BERT model with self-attention mechanism is the primary aim of this research. GRU, a variant of RNN (Recurrent Neural Network), is capable of handling large datasets and can be used to study cloud behavioural patterns. BERT, a transformer-based model, can effectively process and understand the meaning of data from multiple sources. For non-textual data, a simulated BERT model can be applied. By using a self-attention mechanism, the model can identify the most significant parts of the logs during analysis. This can be further extended by using autoencoders with self-healing mechanism. This would help in detecting any variations from the usual behaviour and also take necessary action to prevent any loss to the users or the cloud providers.

The rest of this research proposal is organised into 7 sections with subsections for all the sections. Related work is discussed in section 2, which is a critical analysis of similar works. Section 3 describes the methodology utilised in this study, where GRU-BERT model with self-attention mechanism and autoencoder with self-healing mechanism to detect anomalies in cloud (proposed methods) are discussed. Section 4 discusses the design specifications, while section 5 presents the implementation of the proposed methods. Section 6 discusses the evaluation of the proposed methods, while conclusion and future work are presented in section 7.

2 Related Work

This section provides a review of recent research within the past 10 years. The review is structured into 4 subsections, with the first 3 focusing on specific research themes. Subsection 4 discusses research gaps and compares the critically analysed papers. This section ends with a conclusion that summarises the review and defines the research niche.

2.1 Methods

Many researchers have worked on identifying the best approaches for cloud anomaly detection. Some have used self-attention mechanisms to effectively pre-process and classify cloud logs (Hu, Cao, Ruan and Wu; 2023). Though this approach yields good results, exploring combinations with other models can further improve classification accuracy and reduce security risks.

CNN (Convolutional Neural Network), Bi-LSTM (Bidirectional- LSTM), attention mechanisms and decision trees were successfully utilised to detect cloud anomalies (Gao; 2022, pp. 1-11). However, converting traffic data into images can impact performance especially when large datasets or new anomaly types are encountered.

A modified LSTM model to detect internal attacks on cloud systems was also proposed (Nathezhtha and Yaidehi; 2018, pp. 60-65). This approach effectively reduces false positives. But the model's complexity and memory requirements limit its scalability. Transformer models like BERT could be used for a more efficient and scalable solution for cloud anomaly detection.

Combining BERT with LSTM while considering both global and temporal cloud traffic features effectively classifies anomalies with good accuracy (Shi et al.; 2023, p. 821). This research suggests that a combined approach outperforms the use of BERT or CNN alone. However, the computational complexity of LSTM, especially for large datasets can lead to significant training time and cost. Hence, simpler and more efficient models like GRU could be a better option for moderately sized datasets.

The 4 research papers reviewed various approaches to cloud anomaly detection. While CNN, LSTM and Bi-LSTM models are commonly used, they may not be suitable for large datasets due to computational limitations. Combination of transformer models like BERT with efficient models like GRU provides a better and scalable solution for anomaly detection. This approach can improve cloud security, protect end users, and increase the market value of cloud-based products.

2.2 Solutions

A study on the classification of documents using a GRU model with more than 5000 citations has proven to be a good solution to classify textual data accurately and efficiently (Yang et al.; 2016, pp. 1480-1489). An attention mechanism based on the hierarchical format of the documents is the backbone of this research. The GRU model is quite efficient for classification without the requirement of much memory. This could be further refined and used to detect and classify cloud anomalies. Combining GRU with BERT can significantly enhance performance especially in the case of large datasets. Such a combination would improve the cloud security through the insights received on the detected anomalies.

An improved GRU model has been proposed for trajectory based anomaly detection. Combining this model with a random forest algorithm gives better results than traditional GRU and other RNNs in cloud anomaly detection (Guohao et al.; 2024, pp. 1-17). Integrating an improved GRU with BERT can further enhance the performance. This leads to a more efficient and quicker anomaly detection while improving cloud security and user protection. In addition to this, the use of autoencoders helps in the identification of anomaly patterns so as to provide valuable insights in preventing future security issues in the cloud.

Use of autoencoders resulted in the generation of classified anomalies with good accuracy (Torabi et al.; 2023, pp. 1-13). Here errors are calculated based on a reconstruction mechanism. The reconstruction error, which is a metric used for classification has led to an improved performance when tested on CIDDS-001 (Coburg Intrusion Detection Datasets), but issues still persist in many practical scenarios. Hence, instead of depending on autoencoders alone for the detection of anomalies, the combination of other models like GRU, along with autoencoders would lead to better results and accurate classification of anomalies in the cloud. The combination of autoencoder with self-healing mechanism that automatically fixes the detected anomalies in the cloud is a better option that would enhance the overall resiliency and market value.

The PIACERE project suggests a self-healing mechanism for cloud continuum applications (Alonso et al.; 2021, p. 308). This focuses on IaC ((Infrastructure as Code) optimisation, self-learning, and self-healing using anomaly detection to identify issues at the early stage itself. When anomalies are identified, corrective actions to restore it to the original state without disrupting ongoing processes are initiated. While this research focuses on cloud applications, the basic principles can be extended to identify and rectify anomalies in cloud networks.

In the preceding section, the discussions based on 4 research papers focused on various solutions to detect anomalies in the cloud. Though GRU and autoencoder models have been effective for cloud anomaly detection, combination of multiple models would further improve performance and efficiency. A self-attention mechanism can be integrated to identify critical features and a self-healing mechanism for the automation of corrective actions. This approach significantly improves the resilience and market value of cloud based solutions.

2.3 Design and Architectures

Combination of sentence BERT and Bi-LSTM for cloud anomaly detection was explored in a recent study (Hu, Sun, Dai, Zhang and Liu; 2023). While this model gives good results, further improvements are required to handle logs from diverse sources and to optimise pre-processing. Also, future research should focus on customising the design to meet specific user requirements.

A GRU-BERT model to summarise multiple documents was also proposed recently (Sana and Akhtar; 2023, pp. 503-507). This model can be modified to detect anomalies in cloud logs from various sources. The BERT model is used to identify semantic meaning from the logs while the GRU model is used to analyse sequential patterns. Anomalies can be detected by identifying deviations from the normal behaviour. Autoencoders can be used to improve the system so as to provide deeper insights into the nature of detected anomalies.

The 2 research papers discussed above focused on a critical analysis of the similar works that relates to the design aspects of the research proposal. It was concluded that for proper implementation of the proposed research, a very good cloud-based design is required. All the necessary implementation details, as well as the requirements of the end user and the specifications by the cloud provider are to be considered by this design. Although the preceding designs provide useful insights and serve as a starting point for the proposed research, they require modification to accommodate factors like efficiency, dataset scale and composition, training duration, cost, and performance.

2.4 Gaps and Comparisons

Table 1 summarises the comparisons of the top 5 papers that were critically analysed. The comparison was based on the methods used, the results obtained, identified gaps and cloud-based design.

Table 1: Gaps & Comparisons

Methods	**Results**	**Gaps**	**Cloud-based Design**	**Authors**
LSTM with self-attention mechanism	Binary classification as abnormal and normal with great accuracy	Combination of models for multi-class classification of detected anomalies	Only the architecture of the network is specified	Hu, Cao, Ruan and Wu (2023)
LSTM and BERT	Classification of malicious traffic in the cloud with great accuracy	Consideration of models based on computational complexity, training time and cost	Only the architecture of the model is illustrated	Shi et al. (2023)
Optimisation, self-learning and self-healing	Anomaly detection and corrective actions in cloud-based applications	Usage of transformer based models with self-attention mechanism for effective anomaly detection	A 3 layered design that focuses on optimisation, self-learning and self-healing on IaC	Alonso et al. (2021)
Autoencoders with reconstruction of error mechanism	Detection and classification of anomalies with high accuracy	Combination of autoencoders with self-healing mechanism and transformer based models	Only the model architecture is specified	Torabi et al. (2023)
GRU-BERT	Accurate summarisation of results from multiple sources	Application of autoencoders to further analyse the summarised results	Architecture of the GRU-BERT model is illustrated, which would be improvised for the proposed research	Sana and Akhtar (2023)

The literature review revealed research gaps in cloud anomaly detection, particularly concerning methodologies, results, and cloud-based design. The analysis indicated that a hybrid approach combining GRU and BERT models with self-attention mechanism offers a robust solution for detecting and classifying anomalies. Furthermore, using autoencoders with self-healing mechanisms can provide valuable insights into the detected anomalies, improving system resilience and market value. This research aims to evaluate the performance of the GRU-BERT model with self-attention mechanism to enhance cloud security and user protection. Another aspect being investigated is the potential of autoencoders with self-healing mechanism to predict patterns in cloud logs and tide over future security risks, thereby increasing market value.

The preceding section provides a critical analysis of existing research, examining methodologies, solutions, designs, and architectures. Key contributions, limitations, and research gaps were identified. Based on the literature review, the research niche was identified and the proposed contributions have also been discussed.

3 Methodology

There are various steps in the research methodology which contribute to the problem being solved. These steps are explain under this section.

3.1 Problem Formulation

The first step is to identify the problem that is being solved. Based on thorough research and critical analysis of previous works, it was found that ensuring the security in cloud is extremely necessary. It was also concluded from previous research that the existing deep learning models are not robust enough to withstand attacks and that there could be other models that can improve the performance and accuracy of anomaly detection. Based on further research, it was concluded that the GRU-BERT model with self-attention mechanism can be used for anomaly detection in cloud, with high accuracy and performance. The observation that the usage of autoencoders with self-healing mechanism can be used to take corrective actions and avoid the occurrence of such anomalies is something that this research tries to implement as well.

3.2 Data Collection

The research makes use of the dataset that is collected from the official NAB GitHub repository[1]. It consists of 58 time-series data files that help in anomaly detection in cloud. The NAB dataset is publicly available to be downloaded and used freely for research and academic purposes. This dataset contains logs from multiple sources including the clicking rates of online advertisements, data with anomalies having known causes, traffic data in real-time of a particular area, Twitter data of two large companies, artificially generated data with and without anomalies, and AWS CloudWatch logs. The AWS CloudWatch logs provide information on the network bytes coming in, CPU utilisation rates and, disk read bytes. For the purpose of this research, only the AWS CloudWatch logs that give information on CPU utilisation were used. This dataset has two labels, namely timestamp and value, that show the value of CPU utilisation at a particular timestamp. The size of the dataset was increased which resulted in lesser validation loss. Further pre-processing steps were performed on the dataset before training it with the model, which is explained in the later sections.

[1]https://github.com/numenta/NAB

3.3 Tools Used

Google Colab made use of to implement this research because of the processing power that it offers on cloud for free. Since Python was the programming language used, Google Colab is a good choice of development environment. Versioning control, easy sharing, and access are also some other reasons to choose Google Colab. Hyperopt, which is a Python library, was made use of to perform hyperparameter optimisation. TensorFlow, which is an AI (Artificial Intelligence) tool was used to implement the AI models. NumPy was used to perform certain mathematical functions with the dataset. Pandas was used to work with the dataset, mainly in cleaning and optimising the data. Various libraries in Scikit-learn were made use of to produce better results. Matplotlib was used to generate insightful results visually which can be further interpreted.

3.4 Algorithm Implementation and Integration

The algorithms used in this project involve the implementation of the GRU-BERT model with self-attention mechanism and autoencoder with self-healing mechanism for anomaly detection.

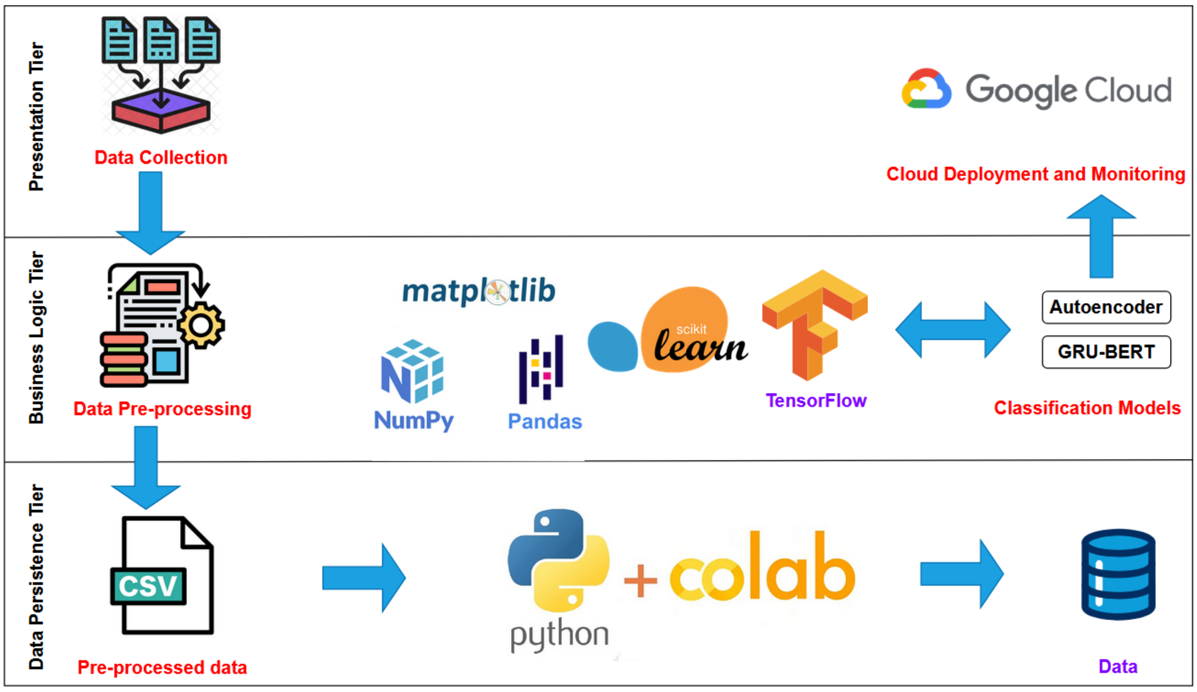

Figure 1: Anomaly Detection Using GRU-BERT and Autoencoder

As shown in Figure 1, the implementation and integration of the GRU-BERT algorithm, along with autoencoder span across various steps. These are explained in detail with the help of the block diagram given below.

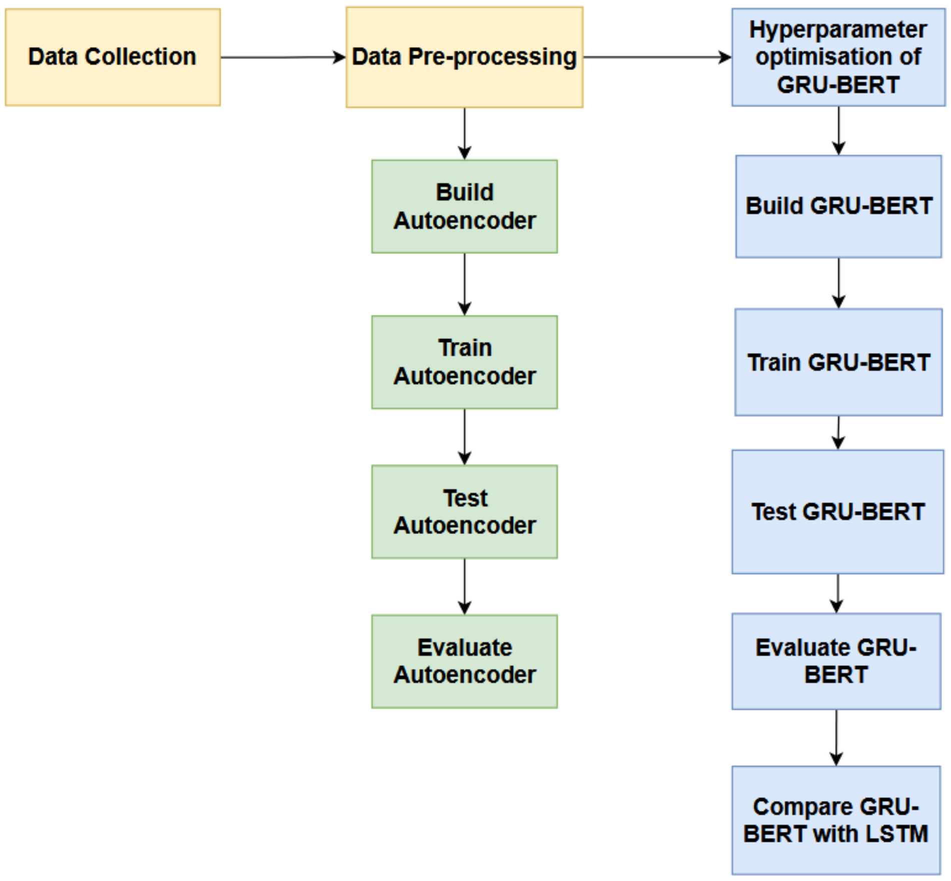

Figure 2: GRU-BERT and Autoencoder Implementation

As shown in Figure 2, the first step is to utilise the CPU utilisation data from the NAB dataset and, optimise the dataset by increasing its size. Apply data pre-processing techniques such as removing the missing values, removing the outliers using IQR (Inter Quartile Range) technique, and splitting the dataset for training and testing purposes. Perform hyperparameter optimisation using Hyperopt and use the optimised hyperparameters to build the GRU-BERT model with self-attention mechanism. Train GRU-BERT on the training set and then test it on the test set. Evaluate GRU-BERT based on performance metrics mentioned in section 3.5 and using standard performance metrics, while also making sure to compare it with LSTM.

Build the autoencoder model with self-healing mechanism. Train autoencoder the training data and test it on the test data. Evaluate autoencoder based on standard performance metrics. Include visualisations for better analysis and interpretation of results.

The NAB makes use of a sigmoidal scoring function to calculate the weights of the anomalies detected (Singh and Olinsky; 2017, pp.1570-1577). This is dependent on the anomaly window and application profile. This approach has some limitations such as determining the size of the anomaly window and gaps in the scoring function. Hence, NAB is optimised to make use of other standard performance metrics to evaluate the model's performance.

3.5 Performance Metrics

The performance of GRU-BERT is calculated using the following parameters:

- Cost: Calculated using the value of the validation loss function (Binary Cross Entropy) that measures how well the prediction of the model matches with the actual target.
- Computational Complexity: Measures the efficiency of the algorithm based on the number of parameters used in the model, thus calculating the computational complexity of the model.
- Execution Time: Refers to the amount of time taken to train the model.
- Energy Consumption: Measured in terms of the power consumption based on CPU usage and execution time.

3.6 Comparison with existing algorithms

The performance of the GRU-BERT model was compared with the LSTM model and the results were analysed. LSTM, which is a type of RNN, is capable of learning and retaining information over time. This feature helps in detecting anomalies in cloud using LSTM efficiently. While both algorithms provide results with great accuracy, LSTM is more complex, and prone to overfitting, while GRU is faster, less complex, and less prone to overfitting.

4 Design Specification

The system requirements, techniques, architecture, and description of the GRU-BERT and autoencoder model are presented in this section.

4.1 System Specifications

Table 2 shows the system specifications that was made use of to carry out this research. Connection to the Python 3 GCE (Google Compute Engine) was made from Google Colab to execute the code.

Table 2: System Specifications

Parameter	**Specification**
System Architecture	x64
Operating System	Windows 11
System RAM (GCE)	12.7
Disk (GCE)	107.7

4.2 Techniques

This section describes the major techniques used in this research project

- Optimise the hyperparameters: Automate hyperparameter optimisation using Hyperopt to choose the best parameters for training the models. The parameters chosen are number of dropout rate, batch size, learning rate and, number of GRU units.
- Build GRU model, simulated-BERT model and integrate: Build the GRU model with hidden states. Implement a self-attention mechanism to pay attention to the relevant sections of the output from GRU. Build the simulated BERT model, which is less complex and more suitable for non-textual data and integrate it with GRU model. Add dense layers for optimisation purposes.
- Build autoencoder model: Build the autoencoder model that has an encoder that decompresses the data and a decoder that reconstructs the data back to its original form. The self-attention mechanism helps in paying attention to the relevant sections of the input data.
- Train and test the models: Train the GRU-BERT model using the optimised hyperparameters and analyse the results. Also train the autoencoder model. Test both models on the test data and observe the output.
- Evaluate the models: Evaluate the GRU-BERT model by comparing it with LSTM. Compare both algorithms based on performance metrics such as cost, computational complexity, execution time and energy consumption. Also compare the algorithms based on standard performance parameters such as F1 score, precision, accuracy, and recall.

4.3 Architecture

- Bayesian optimisation: Bayesian Optimisation was performed using HyperOpt library in Python. This is an automated way of finding out the best hyperparameters that can be used to build and train the model. Finding the best hyperparameters is essential to increase the performance of the algorithm. This will also help to find a trade-off between F1 score and validation loss. Initially, more than 20 hyperparameter combinations were searched. Each combination was evaluated on a validation set, thus optimising the validation loss. The best parameters were extracted and used to train the model for 20 epochs. The model was evaluated again based on the performance metrics.

  4 hyperparameters were considered for the GRU-BERT model: Units, which describe the number of GRU units, also known as hidden states. Learning rate measures the size of the steps that the model takes to update its weights. Dropout rate which determines the number of neurons to be dropped while training, is a technique to decrease overfitting. Batch size defines the total number of training samples that are processed before the model updates its weights.

- GRU-BERT: The GRU model consists of various layers with each having a specific purpose. The input layer was used to define the shape of the data given as input, which is a single feature, scalar in nature. GRU model has units which refers to the number of GRU units, also known as hidden states. The GRU layer processes the sequential data, while also making sure to not just return the output from the final state, but also to save the hidden states. Lambda layer was made use of to make the input dimension compatible with the GRU layer. The output from the GRU layer consists of sequential features. The attention layer was utilised to focus on the relevant parts of the output from the GRU. The reshape layer made sure to flatten the output from the attention layer, making it suitable to be used in the dense layers.

Simulated BERT model was implemented with dense layers and by utilising ReLU (Rectified Linear Unit) activation function. Simulated BERT model was used instead of an actual BERT model, because the actual BERT model is more suitable for textual data. Since CPU utilisation data is non-textual, and less complex, a simulated BERT model would produce better results than an actual BERT model. The simulated BERT model learned and generated BERT embeddings (simulated) from the input (single feature).

The GRU model and simulated BERT model were combined and dense layers with ReLU activation function were added for dimensionality reduction. Dropout layer was added to decrease overfitting. The output layer was added along with sigmoid activation function, which is suitable for binary classification, thus returning a probability between 0 and 1. The input layer and output layer was combined, thus defining the model, and producing a binary classification output. Adam optimiser was used for optimisation in this stage and BCE (Binary Cross Entropy) was used as the loss function, which is suitable for binary classification.

- Autoencoder: An autoencoder model was utilised with self-healing mechanism to reconstruct the error and fix it. The encoder decompresses the input data into a lower dimension. The decoder reconstructs the original input from the encoder. The input from the input layer was made use of by the encoder. Initially, the dimension was reduced to 64 dimensions and later on to 32 dimensions. ReLU activation function was used in this stage. An attention layer was utilised to pay attention to the relevant sections of the input. The output from the encoder was reshaped to make it compatible with the attention layer, and further reshaped back to its original form to be suitable to be used in the further dense layers.

  The decoder initially converted the input of 32 dimensions to 64 dimensions. ReLU activation function was used in this stage. Then the decoder reconstructed the original input, while making sure to use sigmoid activation function. Finally all these layers were combined and the autoencoder was returned. MSE (Mean Squared Error) was used as the loss function and adam optimiser was used for optimisation. Self-healing mechanism was applied to fix the errors. The autoencoder tried to reconstruct the original data, with the aim of producing error-free data. The predictions made by the trained GRU-BERT model and a threshold were made use of to apply the self-healing mechanism. Thus, the errors were replaced by the healed data produced by the autoencoder.

4.4 Anomaly detection in cloud using GRU-BERT and Autoencoder

The pre-processed input data (CPU utilisation rates) was the input for the GRU-BERT model. The input varies from 0 to 100, and values greater than 80 and less than 10 were considered to be anomalous since values that are too high (above 80) and too low (below 10) are classified as anomalies in cloud. 80% of the data was used for training and 20% for testing. Based on hyperparameter optimisation results, the GRU-BERT model was created and optimised with 64 GRU units, and a batch size of 94. The dropout rate was 0.1946, while the Learning rate was 0.0173. Self-attention mechanism was used to pay attention to the relevant sections of the output from GRU. After training and testing the model on the data, the results were evaluated based on the standard performance metrics which can be calculated by equations (1)(2)(3)(4) respectively for Accuracy, Precision, Recall and F1 score.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

The performance of the GRU-BERT model was also evaluated based on the BCE loss function used, as shown in equation (5).

$$\text{BCE}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} - \left[y_t \times \log(y_p) + (1 - y_t) \times \log(1 - y_p) \right]$$

where N is the total number of samples, y_t is the true label for each sample i and y_p is the predicted probability that the sample is part of the positive class (5)

The autoencoder was used to reconstruct the input without anomalies and self-healing mechanism was applied to fix anomalies detected using GRU-BERT with the reconstructed input. The autoencoder model was created using different layers such as input, encoder, decoder, attention, dense and so on. The model was trained, tested, and evaluated based on the MSE loss function as shown in equation (6).

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{6}$$

where

y_i is the actual value for each sample i,

$\hat{y}_i$ is the predicted value for each sample i,

N is the total number of samples.

5 Implementation

The final stage of the implemented solution is presented here, along with the outputs produced, tools and languages used, and key outcomes.

5.1 Development of GRU-BERT model with self-attention mechanism and Autoencoder with self-healing mechanism

The solution was implemented on Google Colab with Python as the programming language used. The data used for implementation contains values of the CPU utilisation rates, in between the range of 0 and 100. Pre-processing was performed on the data by removing missing values and removing outliers using the IQR technique. The data was split into training and testing sets, with 80% of the data for training and 20% for testing.

This data after reshaping, was input to the GRU and simulated BERT model. Hyperparameter optimisation was done using the Hyperopt library in Python to find trade-offs between hyperparameters and use the ideal ones while training the model. GRU-BERT was trained on the training set and tested on the test set, resulting in the classification of logs as anomalous or non-anomalous. Execution time, CPU usage, energy consumption and so on were measured to evaluate the model based on performance metrics. GRU-BERT was also compared with LSTM model and the performance was analysed based on performance metrics. The results were also visualised using bar graphs for better analysis and interpretation of the results. An autoencoder model with self-healing mechanism was implemented to fix the anomalies detected using GRU-BERT. The model was implemented with different layers, trained and tested. The training loss and validation loss were observed. Self-healing mechanism was implemented and the original data and self-healed data were observed. The autoencoder was evaluated based on various performance metrics, and the results were analysed.

5.2 Outputs Produced

The classification of logs as anomalous or non-anomalous by GRU-BERT and the generation of healed data by autoencoder are the two main outputs. This is described in terms of the values of performance metrics, hyperparameters, and loss function.

5.3 Tools and Languages Used

- Google Colab: Google Colab was used as the IDE due to various reasons such as free processing power up to a certain limit, easy access and sharing, compatibility with Python and so on.

- Python Programming Language: Python was used as the programming language due to the various libraries it offers, flexibility and ease of coding.

- TensorFlow: TensorFlow was used to implement and optimise the models, since it makes it easier to implement the models and train them.

- Scikit-learn: Scikit-learn was made use of because of the various libraries that it offers to split the dataset, measure performance and so on.

- NumPy: NumPy was utilised because of the features it offers to compute mathematical operations on the dataset.
- Pandas: Pandas was made use of to load the dataset, analyse and optimise it.
- Matplotlib: Matplotlib was made use of to create visualisations for better analysis and interpretation.

5.4 Key Outcomes

- Optimisation of NAB: The NAB makes use of a sigmoidal scoring function to calculate the weights of the anomalies detected. This is dependent on the anomaly window and application profile. This approach has some limitations such as determining the size of the anomaly window and gaps in the scoring function. Hence this study optimised the NAB and made use of other standard performance metrics to evaluate the model's performance.
- Performance of GRU-BERT in terms of validation loss: GRU-BERT outperformed LSTM in terms of the validation loss (BCE) indicating the robustness of the model to detect anomalies in cloud. Hence making use of GRU-BERT would help in binary classification with high efficiency.
- Optimisation of hyperparameters using Hyperopt: Bayesian optimisation by making use of the Hyperopt library in Python, resulted in finding trade-offs between hyperparameters, which was then utilised to train the model. This approach resulted in achieving better results.

6 Evaluation

GRU-BERT model and LSTM were compared and evaluated based on performance metrics such as cost, computational complexity, execution time, and energy consumption. Additionally, the two algorithms were also compared based on other parameters and hyperparameters mentioned in section 4.2.

6.1 Peformance Metrics 1: Cost

Cost is an important criterion to determine the performance of an algorithm. The validation loss (cost) must be minimal to achieve high performance. If the validation loss is high, the error in detecting anomalies is high, thus resulting in wrong predictions. This will in turn lead to compromising the security in cloud, resulting in great loss for the end users. The validation loss generated by GRU-BERT (0.0031) was lower than that of LSTM (0.0058) as shown in Figure 3. This signifies the enhanced performance of GRU-BERT when compared to LSTM in detecting anomalies in cloud.

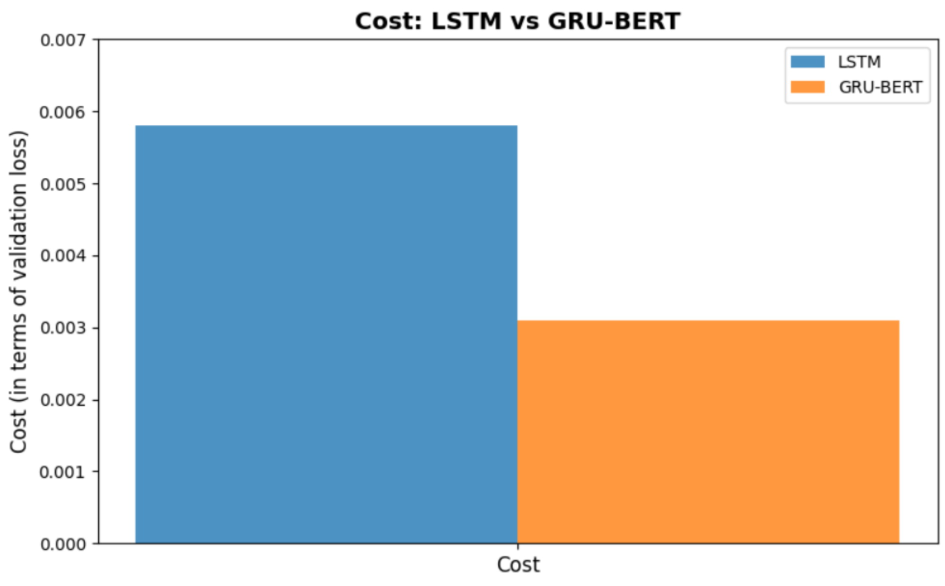

Figure 3: Cost (Validation loss) obtained from GRU-BERT and LSTM models

6.2 Peformance Metrics 2: Computational Complexity

Computational complexity refers to the complexity of the model that is based on the number of parameters used. While, a complex model can be powerful in generating better results, it can also lead to increased consumption of resources, and overfitting. This would also affect the performance of the algorithm. The computational complexity of GRU-BERT was 17153, while that of LSTM was 19009, which is higher than GRU-BERT, as shown in Figure 4. It is highly essential to find a trade-off between complexity and efficiency, in order to achieve the best performance.

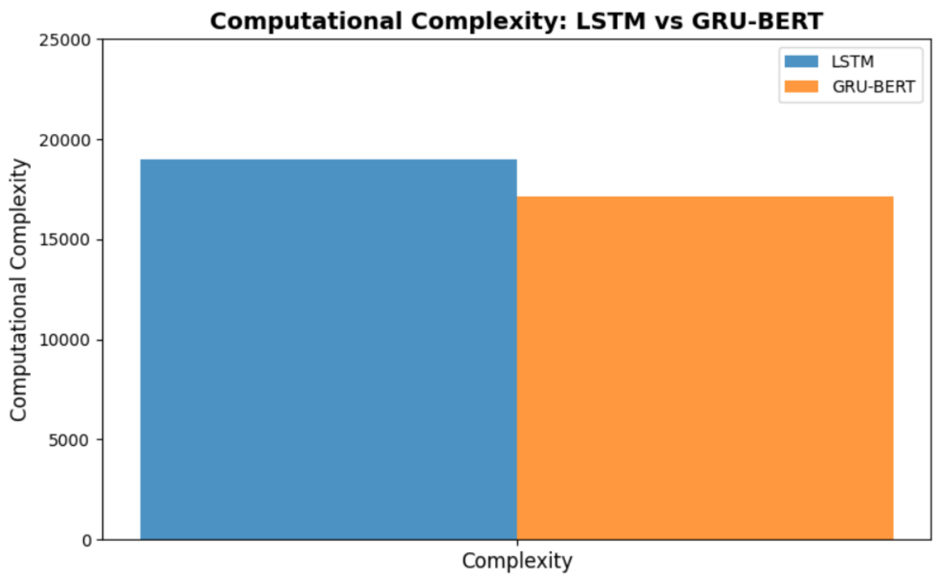

Figure 4: Computational Complexity obtained from GRU-BERT and LSTM models

6.3 Peformance Metrics 3: Execution Time

Execution time, which is an important measure of performance, refers to the time taken to train the model. Complex models would require more execution time when compared to less complex models. The execution time of GRU-BERT was 16.23 seconds while that of LSTM was 16.94 seconds, as shown in Figure 5. Depending on the problem being solved, importance has to be given to faster results efficiently or more powerful results with a longer time duration.

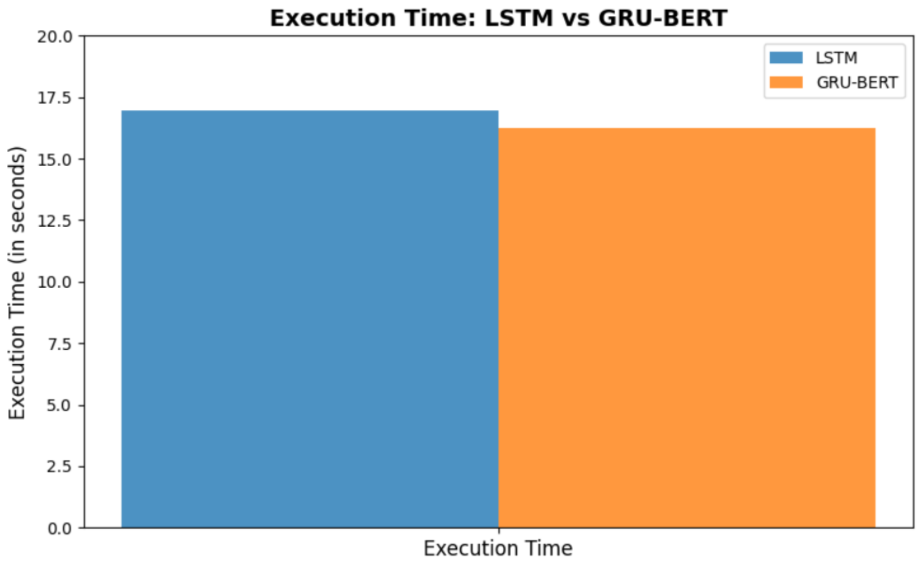

Figure 5: Execution Time (in seconds) obtained from GRU-BERT and LSTM models

6.4 Peformance Metrics 3: Energy Consumption

Energy consumption, measured as a function of CPU usage and execution time gives information on the resources consumed by the algorithms. While algorithms consume significant energy, a high value can drain the resources and cause system failures, affecting the end users. The energy consumption (in Joules) of GRU-BERT was 1186.42 while that of LSTM was 1155.65, as shown in Figure 6. Even though GRU-BERT and LSTM performed well in terms of energy consumption, the energy consumption of LSTM is lower than that of GRU-BERT.

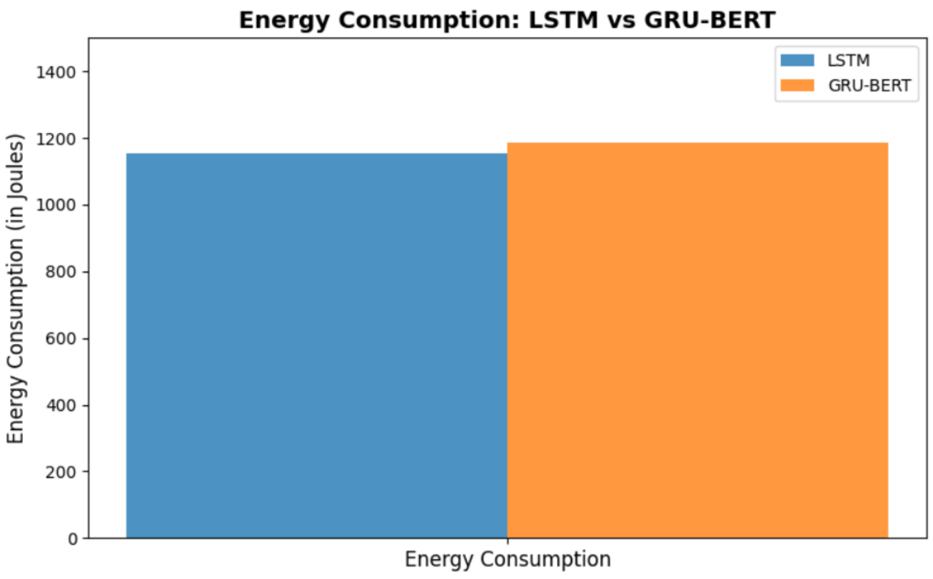

Figure 6: Energy Consumption (in Joules) obtained from GRU-BERT and LSTM models

6.5 Discussion

Comparison between GRU-BERT and LSTM was done based on the parameters, as shown in Table 3. Both GRU-BERT and LSTM performed well for almost all the parameters. The recall of GRU-BERT is slightly higher, indicating that it has more robustness against false negatives, which is highly crucial in anomaly detection in cloud.

Table 3: Parameters: GRU-BERT & LSTM

Parameters	**GRU-BERT**	**LSTM**
Accuracy	0.99	0.99
Precision	0.99	0.99
Recall	1.00	0.99
F1-score	0.99	0.99

Comparison between GRU-BERT and LSTM was done based on the hyperparameters, as shown in Table 4. The batch size of both the models are moderate (96). The dropout rate of GRU-BERT and LSTM are within the ideal range (0.1 - 0.5). The learning rate of GRU-BERT is higher than that of LSTM, which is also one of the reasons for the higher execution time in case of LSTM. This is because, as learning rate decreases, the time taken to train the model also increases.

Table 4: Parameters: GRU-BERT & LSTM

Hyperparameters	**GRU-BERT**	**LSTM**
Dropout Rate	0.194	0.172
Learning Rate	0.017	0.009
Batch Size	96.000	96.000

Based on the results obtained, GRU-BERT model with self-attention mechanism has proven to be an efficient algorithm to detect anomalies in cloud, enhancing cloud security. Ensuring security in cloud is highly crucial, especially considering the amount of data and resources present there. Hence, GRU-BERT serves as an efficient algorithm to detect anomalies. The significant reduction in the validation loss helps in making accurate predictions. The good performance of the model with regards to cost, computational complexity, execution time and energy consumption makes it simpler to use, with faster training times. Optimising the parameters using Hyperopt library resulted in generation of ideal hyperparameters, further increasing the performance of the algorithm. Even though further research needs to be done to improve the energy efficiency and computational complexity of GRU-BERT, its accuracy has improved well, overcoming the limitations. The usage of autoencoders with self-healing mechanism was intended to fix the detected anomalies. More work needs to be done on autoencoders to reduce the training and validation loss, and it could be made use of to automatically fix the detected anomalies. This would also help to increase the security in cloud.

7 Conclusion and Future Work

In this study, GRU-BERT with self-attention mechanism was utilised to detect anomalies in cloud. Based on the findings from this study, it can be concluded that anomalies in public cloud infrastructure can be detected using the GRU-BERT model efficiently, with a low validation loss, enhancing the performance of the algorithm, thus improving cloud security and protecting end users. The autoencoder model with self-healing mechanism was made use of to fix the detected anomalies. Based on the results obtained, it was observed that the extent to which autoencoders can be used to predict patterns and reduce future security issues in cloud needs further research. It was observed that even though significant contributions have been made to enhance the overall resiliency and market value by utilising self-healing mechanisms to automate processes in cloud security, the performance of these techniques needs to be improved.

Through this study, it was concluded that optimisation of NAB by making use of standard performance metrics to evaluate the performance is better because of the limitations of NAB. Also, it was found that GRU-BERT outperformed LSTM in terms of validation loss (BCE), clearly showing the robustness of the model to detect anomalies in cloud. Bayesian optimisation of hyperparameters by making use of Hyperopt resulted in finding trade-offs between them, which was then utilised to build the models, train and test them, producing better results.

7.1 Future Work

GRU-BERT with self-attention mechanism can be made use of to implement predictive maintenance in cloud. This would help in the prediction of failures before they occur, reducing the cost of maintenance, downtime, and so on. Also, this algorithm can be used in practice to detect anomalous behaviour in cloud. Further research could be done to improve the energy efficiency and computational complexity of GRU-BERT. The performance of GRU-BERT could be evaluated on datasets from different industries (e.g., healthcare, e-commerce) to ensure robustness and generalisability. Once further research is done to reduce training and validation loss of autoencoder by techniques such as hyperparameter optimisation and so on, autoencoder with self-healing mechanism can be utilised to reduce false alarms in cloud logs caused due to noise, maintaining the integrity of the data in cloud, its recovery and so on.

This research has resulted in gaining insights on the usage of GRU-BERT, autoencoders and other deep learning techniques to enhance cloud security. It has also resulted in a deeper understanding of future improvements to the algorithm, making it suitable to implement in actual cloud computing scenarios.

References

Alonso, J., Orue-Echevarria, L., Osaba, E., Lobo, J. L., Martinez, I., de Arcaya, J. D. and Etxaniz, I. (2021). Optimization and Prediction Techniques for Self-Healing and Self-Learning Applications in a Trustworthy Cloud Continuum, *Information (2078-2489)* **12**(8): 308.

Gao, J. (2022). Network Intrusion Detection Method Combining CNN and BiLSTM in Cloud Computing Environment, *Computational Intelligence & Neuroscience* pp. 1–11.

Guohao, T., Huaying, Z. and Baohua, Y. (2024). Low-cost and high-performance abnormal trajectory detection based on the GRU model with deep spatiotemporal sequence analysis in cloud computing, *Journal of Cloud Computing: Advances, Systems and Applications* **13**(1): 1–17.

Hu, C., Sun, X., Dai, H., Zhang, H. and Liu, H. (2023). Research on Log Anomaly Detection Based on Sentence-BERT, *Electronics (Basel)* **12**(17).

Hu, W., Cao, L., Ruan, Q. and Wu, Q. (2023). Research on Anomaly Network Detection Based on Self-Attention Mechanism, *Sensors* **23**(11).

Khan, N. and Al-Yasiri, A. (2016). Identifying cloud security threats to strengthen cloud computing adoption framework, *Procedia Computer Science* **94**: 485–490.

Nathezhtha, T. and Yaidehi, V. (2018). Cloud Insider Attack Detection Using Machine Learning, *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*, pp. 60–65.

Sana, E. and Akhtar, N. (2023). Improving Multi-Document Summarization with GRU-BERT Network, *2023 International Conference on Recent Advances in Electrical, Electronics Digital Healthcare Technologies (REEDCON)*, pp. 503–507.

Shi, Z., Luktarhan, N., Song, Y. and Yin, H. (2023). TSFN: A Novel Malicious Traffic Classification Method Using BERT and LSTM, *Entropy* **25**(5): 821.

Singh, A. and Chatterjee, K. (2017). Cloud security issues and challenges: A survey, *Journal of Network and Computer Applications* **79**: 88–115.

Singh, N. and Olinsky, C. (2017). Demystifying Numenta anomaly benchmark, *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1570–1577.

Torabi, H., Mirtaheri, S. L. and Gerco, S. (2023). Practical autoencoder based anomaly detection by using vector reconstruction error, *Cybersecurity (2523-3246)* **6**(1): 1–13.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. and Hovy, E. (2016). Hierarchical Attention Networks for Document Classification, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489.