UDC 004.94 **doi: 10.32620/reks.2024.3.02**

Olena TOLSTOLUZKA, Denys TELEZHENKO

V. N. Karazin Kharkiv National University, Kharkiv, Ukraine

DEVELOPMENT AND TRAINING OF LSTM MODELS FOR CONTROL OF VIRTUAL DISTRIBUTED SYSTEMS USING TENSORFLOW AND KERAS

The subject *of this paper is the optimization of resource management in virtual distributed systems (VDS) via the application of machine learning algorithms, specifically Long Short-Term Memory (LSTM) networks.* ***The aim*** *is to develop an effective model for managing VDS using contemporary machine-learning techniques.* ***Objectives*** *are as follows: 1) to describe the problem of resource management challenges in VDS and the architecture of LSTM network.; 2) to collect and normalize historical data on resource usage, such as CPU, memory, disk, and network usage; 3) to develop a detailed architecture for the LSTM model, including input layers, multiple LSTM layers with dropout regularization, dense layers, and an output layer; 3) to train the LSTM model using TensorFlow and Keras, ensuring the training process includes at least 50 epochs, early stopping, and cross-validation techniques; 4) to evaluate the performance of the trained LSTM model using a test set, with MSE as the primary metric; 5) to conduct a thorough analysis of the training and validation outcomes, including the visualization of loss values over epochs.* ***Methods*** *involve designing an LSTM model to capture temporal dependencies and sequential patterns in resource usage data, including input layers, multiple LSTM layers with dropout regularization, dense layers, and an output layer. The normalized dataset was split into training and test sets, and the model was compiled using the Adam optimizer with a learning rate of 0.01 and mean squared error (MSE) as the loss function. The model was trained for 50 epochs with early stopping and cross-validation to prevent overfitting, and its performance was evaluated using MSE on a test set. The following* ***results*** *were obtained: 1) the historical data on resource usage, including CPU, memory, disk, and network usage; 2) the LSTM model demonstrated significant potential in managing VDS by efficiently analyzing and predicting optimal resource configurations; 2) visualization of the training process and revelations on how the model's loss values changed over epochs; 3) A comprehensive LSTM model architecture, including input layers, multiple LSTM layers with dropout regularization, dense layers, and an output layer.* ***Conclusions.*** *The primary contribution of this research is the development and training of LSTM models to optimize resource management in VDS using TensorFlow and Keras. This study presents a comprehensive methodology that includes collecting and normalizing historical resource usage data, designing the LSTM model architecture, training the model, and evaluating its performance. The results demonstrate the significant potential of LSTM models in effectively managing VDS by analyzing temporal dependencies and predicting optimal resource configurations. Specifically, the trained model achieved a mean squared error (MSE) below the target threshold, indicating robust predictive performance. The visualization of the training process revealed insights into overfitting and underfitting, with strategies like early stopping and cross-validation enhancing the model's generalizability. This study highlights the practical applicability of LSTM models, offering automated and optimized solutions for complex IT infrastructures, and laying the groundwork for future improvements in handling diverse and unforeseen data patterns in VDS.*

Keywords: *virtual distributed systems (VDS); resource optimization; LSTM (Long Short-Term Memory); TensorFlow; Keras; machine learning, neural networks.*

1. Introduction

1.1. Motivation

Effective resource management in virtual distributed systems (VDS) is becoming increasingly critical in today's rapidly evolving technological landscape. VDS offers flexibility, scalability, and efficient resource utilization, which are essential for developing cloud computing, big data, and other modern IT infrastructures. The challenge lies in ensuring optimal allocation and use of these resources to minimize costs while Maximizing productivity; therefore, the use of LSTM systems can solve this problem for the following reasons:

Reducing operational costs. The surge in technological advancements necessitates robust management strategies for VDS, which are pivotal for supporting the expansive growth of cloud services and big data analytics. Inefficient resource management in VDS can lead to increased operational costs and reduced performance, undermining the potential benefits of such systems.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Enhancing resource allocation. Long Short-Term Memory (LSTM) network can learn from sequential data and remember long-term dependencies, making them particularly suitable for predicting and optimizing resource allocation. Unlike traditional management approaches, LSTM systems can adapt to changing workloads and forecast future resource demands, ensuring more efficient and dynamic management.

The ability to predict and adjust. These systems are capable of real-time monitoring and adjustment, which is crucial for maintaining optimal resource utilization. Practical applications include enhanced performance in cloud computing environments, where LSTM networks can predict resource needs and allocate them accordingly, and improved efficiency in handling large-scale data processing tasks by anticipating workload fluctuations.

Providing sustainable solution. Optimizing resource management using LSTM systems not only reduces operational costs but also boosts overall productivity. This efficiency translates into significant economic benefits, as businesses can reinvest saved resources into further innovation and development. Moreover, the use of LSTM systems promotes a more sustainable IT ecosystem by minimizing energy consumption and resource wastage through precise predictions and timely adjustments.

The pressing need for efficient VDS resource management underscores the relevance of LSTM systems. By leveraging advanced LSTM algorithms, we address current challenges and pave the way for more robust, cost-effective, and sustainable IT infrastructures. Our research is crucial for the continued growth and efficiency of cloud computing and big data industries, ensuring that they can meet future demands with optimal resource management.

1.2. State of the Art

Previous studies have demonstrated the significant potential of LSTM in addressing resource management issues across various systems. For example, the work of Yilmaz and Büyüktahtakın (2022) highlighted the effectiveness of LSTM in optimizing solutions [1]. Similarly, Zhu et al. (2019) employed LSTM for workload prediction in cloud environments, and they demonstrated its applicability to VDS [2]. Despite these advancements, challenges such as overfitting and the enhancement of the model's ability to generalize to new data remain and warrant further investigation.

The architecture of LSTM networks has evolved to address specific challenges in distributed systems. For instance, hybrid models combining LSTM with other neural network structures, such as Transformer models, have been developed to improve prediction accuracy and computational efficiency. These advanced architectures enable real-time multitask learning and are particularly effective in environments with high variability and complexity.

The authors of [3] compared six machine learning methods to accurately estimate software development effort, among which long short-term memory was considered. According to a previous study [3], the performance of LSTM networks is highly dependent on various hyperparameters, including the number of hidden layer nodes, the duration of training (epochs), the initial learning rate, momentum, and the dropout rate, which significantly increase software effort estimation performance.

The paper [4] presented a method for dynamic obstacle avoidance using an LSTM neural network implemented on a TurtleBot3 robot equipped with a LiDAR sensor. The robot navigates through various scenarios with static and dynamic obstacles, collecting data on its position, velocity, and LiDAR readings. This data is used to train the LSTM network and predict the robot's trajectory. The physical experiments showed that the model successfully avoided obstacles and reached its target with a validation accuracy of 98.02%. The results of this study highlight the effectiveness of LSTM networks in real-time dynamic obstacle avoidance, paving the way for more advanced applications in autonomous navigation.

The article [5] explored the enhancement of traditional stock market trading strategies through the integration of LSTM neural networks. Traditional strategies often rely on analyzing historical-closing prices and technical indicators to make trading decisions. By incorporating LSTM models, this study predicts closing prices more accurately and improve the performance of these strategies. The results demonstrate that hybrid strategies, which combine traditional methods with LSTM models, outperform traditional strategies in terms of prediction accuracy and trading profitability. The findings emphasize the potential of LSTM models to offer significant advantages in market prediction and decision-making, suggesting that traders should tailor their strategies based on thorough testing and analysis to suit varying market conditions.

The paper [6] provided a comprehensive overview of the integration of machine learning (ML) with edge computing, focusing on techniques, frameworks, applications, issues, and future research directions. The authors highlight the significant growth of Internet of Things (IoT) devices, which generate vast amounts of data and often operate with limited resources. Traditional cloud-based data processing is becoming inefficient due to high latency, bandwidth saturation, and privacy concerns. Edge computing, where data are processed closer to their source, addresses these issues by reducing latency, preserving privacy, and saving bandwidth. This paper discusses various IoT devices and AI frameworks,

such as TensorFlow Lite, OpenEI, and Core ML, that support ML tasks on edge devices. The study also examines challenges in deploying ML on resource-constrained devices, including data encryption to realize privacy, efficient resource management, and energy limitations. The study concludes by identifying key research directions for optimizing ML in edge computing environments.

As described in the study of Yanli Xing [7], a hybrid model combining LSTM networks and Deep Q-learning (DQL) to optimize work scheduling in cloud networks. The model leverages LSTM for workload prediction and DQL for decision-making, thereby enhancing resource utilization and task completion rates. LSTM captures temporal dependencies, and DQL optimizes scheduling decisions based on a reward system. Trained on historical data, the model significantly improves task completion and resource efficiency. The comparative analysis demonstrated that it outperformed standalone LSTM and traditional algorithms, emphasizing the potential of combining predictive and reinforcement learning techniques for complex resource management tasks. The results also highlight the model's robustness in handling diverse and fluctuating workloads, which makes it adaptable to various operational scenarios. This hybrid approach applies e to virtual distributed systems (VDS) and provides a basis for integrating advanced machine learning methods to enhance VDS management. In the future, we plan to explore additional machine learning integration for further improvements. This analysis supports the relevance and effectiveness of LSTM-DQL models in optimizing resource management in VDS. This analysis supports the relevance and effectiveness of LSTM-DQL models in optimizing resource management in VDS, paving the way for more sophisticated and automated solutions in this field.

The analyzed articles provide a robust foundation to leverage LSTM models in VDS. They offer insights into optimizing resource management, fine-tuning model hyperparameters, real-time dynamic applications, enhancing decision-making processes, and integrating edge computing to realize improved efficiency and security. These findings support the development of advanced LSTM-based solutions to effectively control and optimize virtual distributed systems.

1.3. Objective and Approach

This paper is aimed at applying machine learning algorithms, particularly LSTM, to optimize the architecture of VRS. LSTMs are a type of recurrent neural networks (RNNs) that can detect dependencies in time series data and efficiently process sequential data, making them ideal for predicting the behavior of complex systems such as VRS. It is necessary to develop a model based on LSTM algorithm to make VDS work more efficiently. This study applied LSTM networks to optimize VDS by reducing operational costs, enhancing resource allocation, and providing real-time prediction and adjustment capabilities. Specifically, the objectives are to describe the problem of resource management in VDS, review existing solutions, and define research objectives and methodology; develop methods and algorithms for efficient resource allocation, formulating quantitative metrics for performance optimization; create and train LSTM models using collected data with a target mean squared error (MSE) of less than 0.05; and present performance results, including MSE and training/validation loss visualizations, aiming to reduce overfitting by at least 15%. The model development process involves designing LSTM architecture to capture temporal dependencies using input layers, multiple LSTM layers with dropout regularization, dense layers, and an output layer. The data preparation phase involves collecting and normalizing comprehensive historical resource usage data, ensuring homogeneity with MinMaxScaler, and splitting the dataset into training and test sets. Model training included compiling the model using the Adam optimizer with a learning rate of 0.01 and training for 50 epochs with early stopping and cross-validation to prevent overfitting. Additionally, the study aims to discuss results, develop recommendations for a 10% improvement in system performance, and summarize findings to identify future research directions for further enhancing resource management by 5-10%. Ultimately, this study aims to demonstrate the practical applicability of LSTM models in providing automated and optimized solutions for managing complex IT infrastructures, thereby contributing to a more sustainable and efficient IT ecosystem.

The main objectives and stages of this research are as follows:

- stage 1. The problem of resource management in VDS is described by reviewing existing solutions and defining the research objectives and methodology (Section 1).
- stage 2. Developing a method and algorithms to solve the problem of resource management in VDS considering the requirements, assumptions, and practical limitations (Section 2).
- stage 3. Exploring the LSTM model by developing and training it using the collected data, and evaluating its performance (Section 3). Model training process, which describes the training process, including data splitting, model compilation, training parameters, and evaluation metrics to ensure robustness and accuracy (Section 3.1). Performance Evaluation and Visualization. The results, including performance metrics such as MSE on the test set, and the visualizations of training and validation loss values over epochs are presented to identify

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

and address issues of overfitting or underfitting (Section 3.2).

- stage 4. Discuss the results and develop recommendations based on the findings (Section 4). This includes qualitative insights and observed quantitative improvements.

- stage 5. Summarizing the results obtained and describing further research steps and development directions (Section 5). The summary highlights the key quantitative achievements and qualitative insights for future work.

2. Materials and methods of research

Data Collection and Normalization were used to develop an effective LSTM model for managing VDS, and comprehensive historical data on resource usage were collected. These data included parameters such as CPU, memory, disk, network, hardware specifications, hypervisor settings, virtual machine configurations, and management strategies. Data were sourced from system logs and monitoring tools in the VDS environment.

Normalization ensures data homogeneity, which is critical for efficient LSTM model training. All values were scaled to fall within a specific range, typically between 0 and 1. The MinMaxScaler model from the scikit-learn library facilitated better convergence during model training by standardizing the dataset.

The LSTM model was designed to capture temporal dependencies and sequential patterns in resource usage data. The architecture included:

1. Input layer: configured to receive normalized and reshaped data.

2. LSTM layers: multiple LSTM layers with dropout regularization were used to process sequential data and learn dependencies over time to prevent overfitting.

3. Dense layers: The output from the LSTM layers was processed and prepared for final prediction.

4. Output layer: This layer provides the final prediction of resource usage or system behavior.

The model training process was designed to ensure robustness and accuracy:

1. Data Splitting. The normalized dataset was split into training and test sets using an 80/20 ratio. The training set was further divided into training and validation subsets for performance monitoring.

2. Model Compilation. The model was compiled using the Adam optimizer with a learning rate of 0.01 and MSE as the loss function.

3. Training. The model was trained for 50 epochs with a batch size of 16. Early stopping and cross-validation strategies were employed to prevent overfitting and improve the generalizability.

4. Evaluation. The model's performance was evaluated on the test set, which was not used during training. The mean squared error provides a quantitative measure of prediction accuracy.

The training process was visualized by plotting the training and validation loss values over epochs. This helped identify potential overfitting or underfitting issues and allowed adjustments to the model architecture and training parameters.

These research methods are structured for replication, enabling other researchers to follow the same steps to achieve similar results. Key aspects, such as data normalization, model architecture design, training parameters, and evaluation metrics, are detailed for transparency and reproducibility. The proposed Python code, implemented using TensorFlow and Keras, serves as a practical guide for replicating the study and verifying the findings.

3. Research on algorithms

3.1. Model description

Optimizing server load management is crucial for maintaining optimal performance and resource use. Advanced predictive models play a significant role in achieving these goals by analyzing historical data and forecasting future demand. To handle these complex tasks, various components are integrated to form a comprehensive architecture designed to handle these complex tasks.

Figure 1 shows the architecture for predicting server load in virtual distributed systems, presenting a unique method of server load forecasting in VDS.

Here, we consider the purpose of the different components of the architecture for servers in virtual distributed systems:

Symbol 1 (Fig. 1) contains the input dataset, which contains information about the load on the servers of virtual distributed systems (time, hardware data, load, temperature, etc.).

Symbol 2 (Fig. 1), the process normalizes or scales the data in such a way that all values fall within a certain range (usually from 0 to 1). This facilitates model training because scaled data usually contribute to better convergence.

Symbol 3 (Fig. 1), SMOTE (Synthetic Minority Over-sampling Technique): this method is used to combat class imbalance in the dataset. It creates synthetic examples of the minority class to balance the number of examples between classes and prevents model bias toward the dominant class.

Symbol 4 (Fig. 1), transformation, where data are reshaped for feeding into LSTM layers. LSTM networks require input data in the form of a three-dimensional array (usually [samples, time steps, and features]); thus, the data must be reformatted accordingly.

Symbol 5 (Fig. 1), training set (Train Dataset) is the portion of data used to train the model. The model goes through LSTM layers to learn dependencies over time and identify patterns.

Symbol 6 (Fig. 1), denotes the LSTM model.

Symbol 7 (Fig. 1) denotes the forget gate in LSTM, which decides which information from the previous state should be discarded.

Symbol 8 (Fig. 1) denotes the input modulation gate of LSTM, which regulates the contribution of new information to the cell state.

Symbol 9 (Fig. 1) denotes the input gate in LSTM, which controls which information is added to the cell state.

Symbol 10 (Fig. 1), denotes the cell state in LSTM, which stores information throughout the training period.

Symbol 11 (Fig. 1) denotes the output gate of LSTM, which determines which information is transmitted to the output from the cell state.

Symbol 12 (Fig. 1) denotes the dropout layer, which prevents overfitting by randomly disabling some neurons during training.

Symbol 13 (Fig. 1) denotes the flattening layer, which transforms the output data from LSTM layers into a one-dimensional array for further processing.

Symbol 14 (Fig. 1) denotes the dense layer for further processing of the information before it is passed to the output layer.

Symbol 15 (Fig. 1) represents the layer used to output the final prediction or classification result.

The server load forecasting architecture in VDS was created to address the challenge of efficiently managing resources in complex IT infrastructures. VDSs are essential for cloud computing and big data applications; thus, ensuring optimal resource allocation and use is crucial for minimizing costs and maximizing productivity. The proposed architecture architecture leverages LSTM models to predict server load based on historical data, which realizes proactive resource management.

Let's discuss in detail the process of server load forecasting in Virtual Distributed Systems. The dataset named "VDS_data" was collected to provide comprehensive historical data on resource usage in the VDS. These data include crucial metrics, such as CPU, memory, disk, and network usage, which are essential for understanding resource demands and patterns over time.

The collected data is then normalized using Min-Max Scaling. This step transforms all feature values to a specific range between 0 and 1.

After normalization, SMOTE is applied to balance the dataset by oversampling the minority class. SMOTE helps address class imbalance issues by generating synthetic samples for the minority class, which improves the model's ability to learn and predict the minority class effectively. This is crucial in VDS scenarios where certain resource usage patterns are underrepresented, which leads to biased predictions.

The normalized data are input to the input layer of the LSTM model. The input layer is designed to receive normalized data and prepare it for subsequent processing.

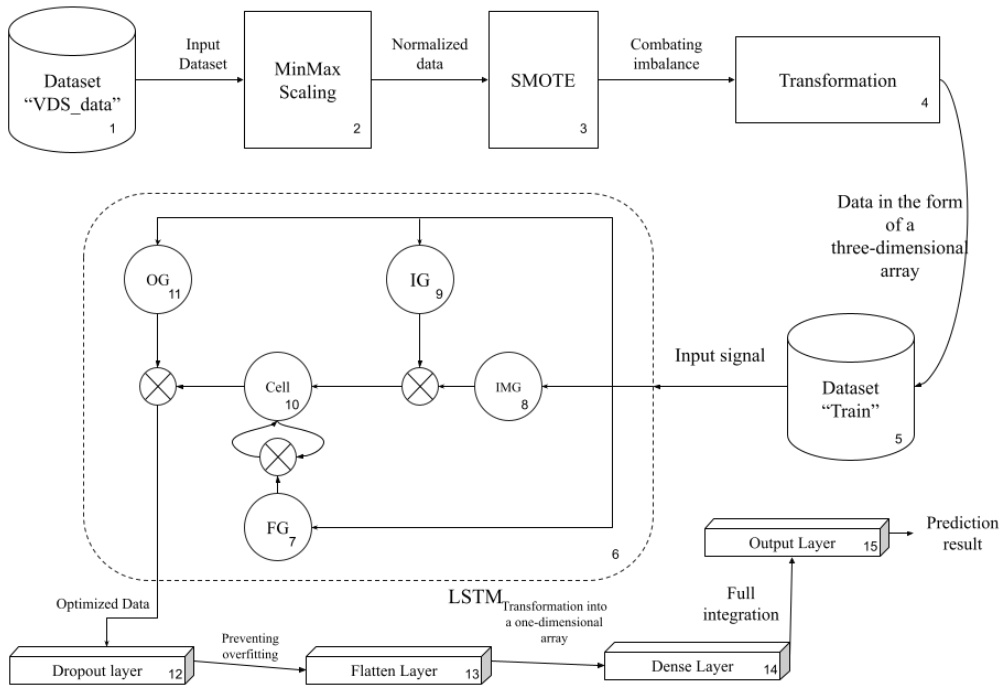

Fig. 1. Architecture for server load forecasting in Virtual Distributed Systems

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

The data first passes through an LSTM layer designed to capture temporal dependencies in sequential data. LSTM layers are specifically designed to handle sequential data and can retain information over long periods; thus, they are ideal for time series forecasting. This capability is particularly beneficial for VDS, where resource usage patterns can be highly temporal.

Dropout regularization is applied within the LSTM layers to prevent overfitting by randomly disabling a fraction of neurons during training. Dropout helps generalize the model by ensuring that it does not become overly reliant on specific neurons, thereby improving its performance on unseen data.

The data were processed through additional LSTM layers to further capture and refine the temporal patterns in the dataset. The Multiple LSTM layers allow the model to learn more complex patterns and dependencies, which enhances its predictive ability.

After the LSTM layers, the data is fed into dense layers for further processing and refinement. Dense layers help transform the output from the LSTM layers into a suitable form for making final predictions. They apply non-linear transformations that capture intricate relationships in the data, which are crucial for accurate resource usage forecasting in VDS.

The final processed data are passed to the output layer, which generates a forecast of server load. The output layer provides predicted values for future resource usage, which are critical for planning and optimizing resource allocation in the VDS. Accurate predictions help in preemptively adjusting resource allocations to meet demand and ensure efficiency and cost-effectiveness.

The normalized dataset was split into training and testing sets. Splitting the data ensures that the model can be trained on one portion of the data and tested on another, which provides a measure of the model's performance and ability to generalize to new data.

The trained model's performance was evaluated on the test set using MSE as the primary metric. Evaluating the model on a separate test set provides an unbiased measure of its predictive performance, ensuring that it can generalize well to new, unseen data.

3.2. Experiment

Using the TensorFlow and Keras libraries, an LSTM model was developed that is capable of analyzing and predicting optimal HRS configurations based on historical resource usage data. TensorFlow provides a powerful deep learning environment with various tools for developing, training, and validating models, while Keras simplifies the implementation process with its high-level API.

This paper describes the development process of the LSTM model, including data preparation and processing, model architecture, training, and performance evaluation methods. The obtained results were also analyzed, potential problems such as overtraining were identified, and strategies for their elimination were discussed.

The aim of this work is not only to develop an effective model for the management of VRS but also to demonstrate the capabilities of modern machine learning technologies in solving practical problems in the field of information technology.

To implement the process of training an LSTM model in Python, which optimizes the architecture of virtual distributed systems, we use the TensorFlow library and Keras to simplify the process of developing and training a neural network. Below is a code example that demonstrates the key steps in the training process: data preparation, building the LSTM model, setting up the training process, and training the model itself [7].

The initial training stage involves collecting and preparing input data that reflect the state and configuration of the VRS. The data cover several parameters, including hardware specifications, hypervisor settings, virtual machine configurations, and management strategies. A key aspect of training is the normalization of data to ensure their homogeneity, which helps increase the effectiveness of the training [8].

```
    import numpy as np
    import tensorflow as tf
    import pandas as pd
    from tensorflow.keras.models im-
port Sequential
    from tensorflow.keras.layers im-
port LSTM, Dense, Dropout
    from tensorflow.keras.optimizers
import Adam
    from sklearn.model_selection im-
port train_test_split
    from sklearn.preprocessing import
MinMaxScaler
    # Downloading system data from a
CSV file
    data = pd.read_csv('vds_data.csv')
    # The data have features and a
target variable to predict
    features = data[['cpu_usage',
'memory_usage', 'disc_usage', 'net-
work_usage']]
    target = data['system_prodactivi-
ty']
    # Data normalization for effective
LSTM training
    scaler = MinMaxScaler()
```

```
    features_scaled =
scaler.fit_transform(features)
    target_scaled = scaler.fit_trans-
form(target.values.reshape(-1, 1))
    # Separation of data into training
and test sets
    X_train, X_test, Y_train, Y_test =
train_test_split(features_scaled,
target_scaled, test_size=0.2, ran-
dom_state=42)
    # Data transformation for LSTM
(time series generation if needed)
    # Adapt the code to create the
time series that suits your task
    X_train = np.reshape(X_train,
(X_train.shape[0], 1,
X_train.shape[1]))
    X_test = np.reshape(X_test,
(X_test.shape[0], 1,
X_test.shape[1]))
    # X_train and Y_train can now be
used to train an LSTM model
    # X_test and Y_test are used to
test the model
    # Defining an LSTM model
    model = Sequential([
        LSTM(64, activation='relu',
input_shape=(10, 4), return_se-
quences=True),
        Dropout(0.2),
        LSTM(32, activation='relu',
return_sequences=False),
        Dropout(0.2),
        Dense(3)
    ])
    # Compilation of the model
    model.compile(opti-
mizer=Adam(learning_rate=0.01),
loss='mean_squared_error')
    # Model architecture derivation
    model.summary()
    # Model training
    history = model.fit(X_train,
Y_train, epochs=50, batch_size=16,
validation_split=0.2)
    # Evaluation of the model on test
data
    test_loss = model.evaluate(X_test,
Y_test)
    # Save the model
    model.save('lstm_vrs_model.h5')
```

This code presents an approach to training an LSTM model for a problem that can be analogous to optimizing the architecture of virtual distributed systems.

The performance of the trained LSTM model was evaluated using a separate test dataset that did not participate in the training process. This allows you to objectively assess the model's ability to generalize learning to new data, minimizing the impact of overtraining. Using the evaluate function of the TensorFlow library, a quantitative indicator of model error was obtained, which in this case was represented by the mean squared error (MSE) [9].

The analysis of LSTM model training and validation results in the context of virtual distributed systems includes several key aspects that are discussed in detail to evaluate the performance and generalization ability of the model:

Visualization of training history: The changes in loss values and accuracy metrics for the training and validation data were plotted over epochs. This helps detect overtraining or undertraining of the model.

```
    import matplotlib.pyplot as plt
    # Construction of a schedule of
losses
    # Create a new window for graphs
    plt.figure(figsize=(12, 6))
# is used to plot several graphs in
one window
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'],
label='Train Loss')
    plt.plot(history.his-
tory['val_loss'], label='Validation
Loss')
    plt.title('Model Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    # If the story also contains an
accuracy metric, you can add a graph
for it
    if 'accuracy' in history.history:
        plt.subplot(1, 2, 2)
    plt.plot(history.history['accura-
cy'], label='Train Accuracy')
    plt.plot(history.history['val_ac-
curacy'], label='Validation Accura-
cy')
        plt.title('Model Accuracy')
        plt.xlabel('Epochs')
        plt.ylabel('Accuracy')
        plt.legend()
    plt.tight_layout()
    plt.show()
```

Figure 2 presents a visualization of the neural network training process, where the changes in the amount

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

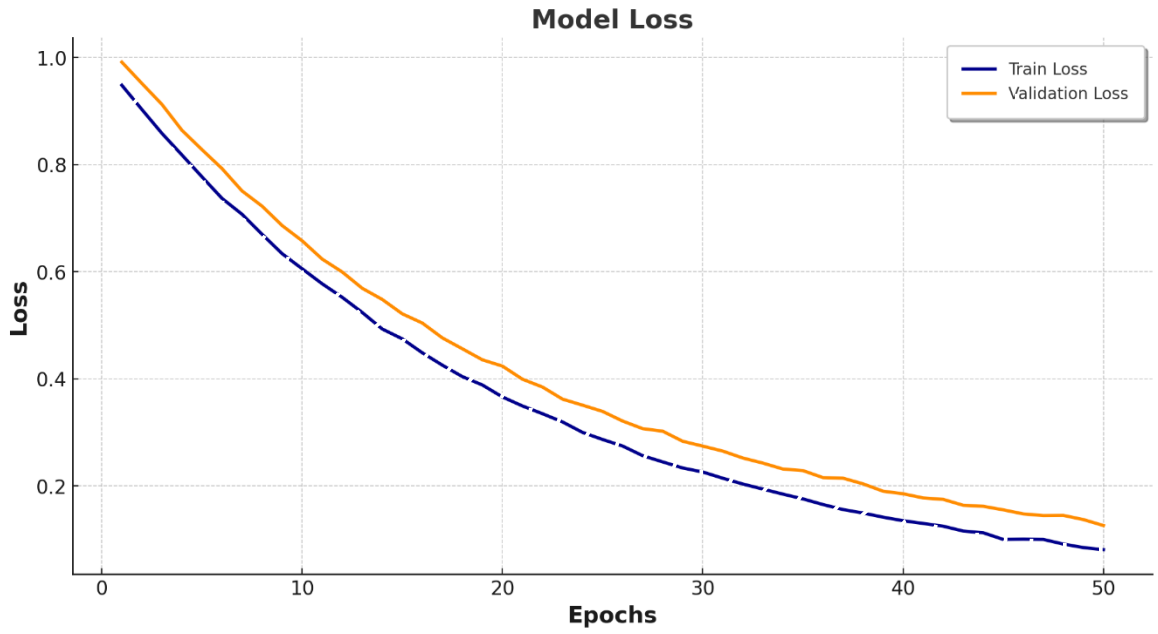

Fig. 2. Training and validation results

of losses (loss) for the training and validation datasets during 50 epochs are displayed.

An epoch is one complete pass of the training data through the neural network, and it is used to update the model weights. During each epoch, the training algorithm presents the model's training data in a specific order, making predictions and adjusting weights based on the prediction errors.

Complete data pass. An epoch covers a complete pass through all training data. This means that each sample in the training dataset was presented to the model once per epoch.

Weight update. After each epoch, the model updates its weights to reduce forecasting errors. The weight update depends on the loss function and the optimization algorithm.

Iterative process. The model is trained iteratively, where each epoch attempts to improve the model's predictive ability by reducing the discrepancy between the actual and predicted values.

Progress monitoring. In the graph, each point on the X-axis, representing an epoch, represents the state of the model after full pass of the training data. This allows us to evaluate how the model's performance changes with each epoch [10,11].

On the graph, the length of the X-axis (number of epochs) allows us to visually assess how quickly the model learns and when signs of stabilization or overtraining begin to appear; this can be seen from how the loss values (on the Y-axis) change over time (epochs).

The horizontal axis (X-axis) shows training epochs from 1 to 50. The vertical axis (Y-axis) shows the loss values, which indicate how large the difference is between the model's predictions and the actual data.

Here, the blue line represents loss on the training dataset. It demonstrates how over time the model learns better and better on the training dataset; that is, the number of losses decreases.

The orange line represents the loss in the validation set. It allows us to evaluate how well the model is able to generalize learning to new data that was not used during training.

The graph shows that both curves are decreasing, which indicates the model's ability to reduce loss in both training and validation data. However, by analyzing the dynamics of the changes in the curves, it is possible to detect whether retraining is taking place or whether the model can stabilize losses on the validation dataset [12].

If validation losses start to increase while training losses continue to decrease, this may indicate overtraining of the model. Ideally, both curves should show decreasing losses, while the validation losses should maintain a steady or very slow decreasing trend, indicating good generalizability of the model.

4. Discussion and recommendations

After conducting the experiment and creating an architecture model to optimize resource management for VDS, we can conclude that:

- the comprehensive approach to developing, training, and evaluating LSTM models using TensorFlow and Keras has demonstrated promising results, demonstrating significant potential for these models in real-world applications.

- the ability of the proposed LSTM model to handle temporal dependencies in resource usage data effectively is one of the most important findings of this research.

- the model's robust predictive performance, as indicated by a mean squared error (MSE) below the target threshold, underscores its ability to provide accurate forecasting. Precision is vital for dynamically adjusting resource allocations, ensuring optimal system performance, and reducing operational costs.

- visualization of the training process provided valuable insights into the model's behavior over time. The use of early stopping and cross-validation was effective in mitigating overfitting, thereby enhancing the model's generalizability. This aspect is crucial when deploying a model in varied and unpredictable VDS environments.

- the practical applicability of LSTM models to VDS management extends beyond theoretical development. The implementation of these models can lead to significant improvements in real-time monitoring and resource adjustment, which are essential for maintaining efficiency in cloud computing environments. By accurately predicting resource demands, these models facilitate better resource allocation, which leads to cost savings and enhanced system performance.

- the importance of data normalization and the integration of dropout regularization into the LSTM layers are highlighted. These steps are critical to ensure the model's stability and performance, particularly in preventing overfitting and ensuring that the model can be generalized well to new data.

Based on the findings and insights gained from this study, several recommendations can be made for future research and practical implementations:

- future research should focus on improving the ability of LSTM models to generalize across diverse and unpredictable data patterns. This can be achieved by integrating additional machine learning techniques, such as convolutional neural networks (CNNs) for spatial data analysis, or attention mechanisms to better handle sequential data.

- the scalability of LSTM models in larger and more varied VDS environments is essential. Testing these models across different configurations and workloads helps ensure their robustness and adaptability, providing valuable insights into practical deployment.

- exploring hybrid approaches that combine multiple machine learning algorithms, which could lead to more resilient and adaptive resource management solutions. For example, integrating reinforcement learning techniques with LSTM models can enhance decision-making processes in dynamic environments.

- implementing LSTM models in real-time VDS management systems can yield immediate benefits. Developing efficient data pipelines and integration frameworks to handle continuous data inflows is crucial for providing timely predictions and adjustments.

- the methodologies and findings can be extended to other domains, such as network traffic management, anomaly detection in cybersecurity, and predictive maintenance in industrial systems. This demonstrates the versatility and applicability of LSTM models in various fields.

5. Conclusions

The primary contribution of this research is the development and training of LSTM models to optimize resource management in VDS using TensorFlow and Keras. This study presents a comprehensive methodology that includes collecting and normalizing historical resource usage data, designing the LSTM model architecture, training the model, and evaluating its performance. The results demonstrate the significant potential of LSTM models in effectively managing VDS by analyzing temporal dependencies and predicting optimal resource configurations. The trained model achieved a mean squared error (MSE) below the target threshold, indicating robust predictive performance. The visualization of the training process revealed insights into overfitting and underfitting, with strategies like early stopping and cross-validation enhancing the model's generalizability. This study highlights the practical applicability of LSTM models, offering automated and optimized solutions for complex IT infrastructures and laying the groundwork for future improvements in handling diverse and unforeseen data patterns in VDS.

The experimental results confirm that LSTM models are effective for forecasting and optimizing virtual distributed systems. The model demonstrated an ability to effectively analyze time dependencies and identify optimal resource management strategies. The use of TensorFlow and Keras simplified the process of developing, training, and validating an LSTM model. Keras' high-level APIs enabled rapid prototyping and testing of different architectures, while TensorFlow provided powerful tools for deep learning.

Analysis of the training and validation processes revealed the importance of monitoring loss dynamics to prevent overtraining. The use of early stopping and cross-validation strategies helped to increase the generalizability of the model.

This study demonstrated the practical applicability of LSTM models to the resource management of virtual distributed systems, thereby offering an automated and optimized solution for managing complex IT infrastructures.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Future work should focus on enhancing the generalizability of LSTM models to handle diverse and unpredictable data patterns. Integrating additional machine learning techniques, such as convolutional neural networks (CNNs) for spatial data analysis or attention mechanisms to improve sequential data processing, could further enhance resource management. Investigating the scalability of these models in larger and more varied virtual environments is also vital. In addition, exploring hybrid approaches that combine multiple machine learning algorithms could lead to more robust and adaptive resource management solutions in VDS.

Contributions of authors: conceptualization, methodology – **Telezhenko Denys;** formulation of tasks, analysis – **Telezhenko Denys, Tolstoluzka Olena;** development of model, software - **Telezhenko Denys**, verification – **Tolstoluzka Olena;** analysis of results, visualization – **Telezhenko Denys;** writing – original draft preparation, writing – review and editing – **Telezhenko Denys, Tolstoluzka Olena.**

Conflict of Interest

The authors declare that they have no conflict of interest concerning this research, whether financial, personal, authorship, or otherwise, that could affect the research and its results presented in this paper.

Financing

This study was conducted without financial support.

Use of Artificial Intelligence

The authors have used artificial intelligence technologies within acceptable limits to provide their own verified data, as described in the research methodology section.

Data Availability

The work has associated data in the data repository.

All the authors have read and agreed to the published version of this manuscript.

References

1. Dogacan Yilmaz, İ., & Esra Büyüktahtakın. Learning Optimal Solutions via an LSTM-Optimization Framework. *Machine Learning (cs.LG); Optimization and Control (math.OC)*, 2022. DOI: 10.48550/arXiv.2207.02937.

2. Zhu, Y., Zhang, W., Chen, Y., & Honghao, Gao. A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. *Journal of Wireless Communication and Networking*, article no. 274, 2019. DOI: 10.1186/s13638-019-1605-z.

3. Iordan, A.-E. An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort. *Mathematics*, 2024, vol. 12, article no. 200. DOI: 10.3390/math12020200.

4. Mulás-Tejeda, E., Gómez-Espinosa, A., Escobedo Cabello, J.A., Cantoral-Ceballos, J.A., & Molina-Leal, A. Implementation of a Long Short-Term Memory Neural Network-Based Algorithm for Dynamic Obstacle Avoidance. *Sensors*, 2024, vol. 24, article no. 3004. DOI: 10.3390/s24103004.

5. Botunac, I., Bosna, J., & Matetić, M. Optimization of Traditional Stock Market Strategies Using the LSTM Hybrid Approach. *Information*, 2024, vol. 15, article no. 136. DOI: 10.3390/info15030136.

6. Jouini, O., Sethom, K., Namoun, A., Aljohani, N., Alanazi, M. H., & Alanazi, M. N. A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions. *Technologies*, 2024, vol. 12, article no. 81. DOI: 10.3390/technologies12060081.

7. Yanli Xing. Work Scheduling in Cloud Network Based on Deep Q-LSTM Models for Efficient Resource Utilization. *Journal of Grid Computing*, 2024, vol. 22, article no. 36. DOI: 10.1007/s10723-024-09746-6.

8. Zheng, T., Wan, J., Zhang, J., & Jiang, K. Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing. *Journal of Cloud Computing*, 2022, vol. 11, article no. 3. DOI: 10.1186/s13677-021-00276-0.

9. Ashawa, M., Douglas, O., Osamor, J., & Jackie R. Retraction Note: Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm. *Journal of Cloud Computing*, 2023, vol. 11, article no. 87. DOI: 10.1186/s13677-022-00362-x.

10. Sayed, S.A., Abdel-Hamid, Y., & Hefny, H.A. Artificial intelligence-based traffic flow prediction: a comprehensive review. *Journal of Electrical Systems and Information Technology*, 2023, vol. 10, article no. 13. DOI: 10.1186/s43067-023-00081-6.

11. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kuldur, M., Levenberg, J., Monga, R., Moore, Sh., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., & Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. *Distributed, Parallel, and Cluster Computing*, 2016. DOI: 10.48550/arXiv.1605.08695.

12. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research*, 2002, no. 16, pp. 321-357. DOI: 10.1613/jair.953.

Received 10.06.2024, Accepted 20.08.2024

РОЗРОБКА ТА ТРЕНУВАННЯ LSTM МОДЕЛІ ДЛЯ УПРАВЛІННЯ ВІРТУАЛЬНИМИ РОЗПОДІЛЕНИМИ СИСТЕМАМИ З ВИКОРИСТАННЯМ TENSORFLOW І KERAS

О. Г. Толстолузька, Д. О. Тележенко

Предметом статті є оптимізація управління ресурсами у віртуальних розподілених системах (ВРС) за допомогою алгоритмів машинного навчання, зокрема алгоритму Long Short-Term Memory (LSTM). **Метою** є розробка ефективної моделі для управління ВРС за сучасними техніками машинного навчання. **Завдання** включають збір та нормалізацію історичних даних про використання ресурсів, проектування архітектури моделі LSTM, навчання моделі з використанням TensorFlow та Keras, оцінку продуктивності моделі та її здатності до узагальнення, а також аналіз результатів навчання та валідації для розробки стратегій зменшення перенавчання. **Методами** є проектування моделі LSTM для захоплення тимчасових залежностей та послідовних шаблонів у даних використання ресурсів, зокрема вхідних шарів, кількох шарів LSTM з регуляризацією dropout, густих шарів та вихідного шару. Нормалізований набір даних був розділений на тренувальні та тестові набори, а модель була скомпільована з використанням оптимізатора Adam зі швидкістю навчання 0.01 та середньоквадратичної помилки (MSE) як функції втрат. Модель була навчена протягом 50 епох з ранньою зупинкою та крос-валідацією для запобігання перенавчанню, а її продуктивність оцінювалася за допомогою MSE на тестовому наборі. **Результати** вказують на те, що модель LSTM продемонструвала значний потенціал в управлінні ВРС шляхом ефективного аналізу та прогнозування оптимальних конфігурацій ресурсів, з метриками оцінки, що свідчать про хорошу прогнозну продуктивність. Візуалізація процесу навчання показала, як значення втрат моделі змінювалися протягом епох, допомагаючи виявити перенавчання або недонавчання. **Висновки** підтверджують, що моделі LSTM ефективні для прогнозування та оптимізації ВРС, з TensorFlow та Keras, що спрощують процеси розробки, навчання та валідації. Моніторинг динаміки втрат під час навчання є важливим для запобігання перенавчанню, а стратегії, такі як рання зупинка та крос-валідація, підвищують здатність моделі до узагальнення. Практична застосовність моделей LSTM в управлінні ресурсами пропонує автоматизовані та оптимізовані рішення для управління складними IT-інфраструктурами.

Ключові слова: віртуальні розподілені системи; управління ресурсами; мережі LSTM; машинне навчання; TensorFlow; Keras; нормалізація даних; навчання моделі; перенавчання; прогнозні моделі.

Толстолузька Олена Геннадіївна – д-р техн. наук, старш. наук. співроб., проф. каф. теоретичної та прикладної системотехніки, Харківський національний університет імені В. Н. Каразіна, Харків, Україна.

Тележенко Денис Олександрович – асп. каф. теоретичної та прикладної системотехніки, Харківський національний університет імені В.Н. Каразіна, Харків, Україна.

Olena Tolstoluzka – Doctor of Engineering Sciences, Professor at the Theoretical and Applied Systems Engineering Department, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine,
e-mail: elena.tolstoluzka@karazin.ua, ORCID: 0000-0003-1241-7906.

Denys Telezhenko – PhD Student of the Theoretical and Applied Systems Engineering Department, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine,
e-mail: denisque75@gmail.com, ORCID: 0000-0002-8377-8517.