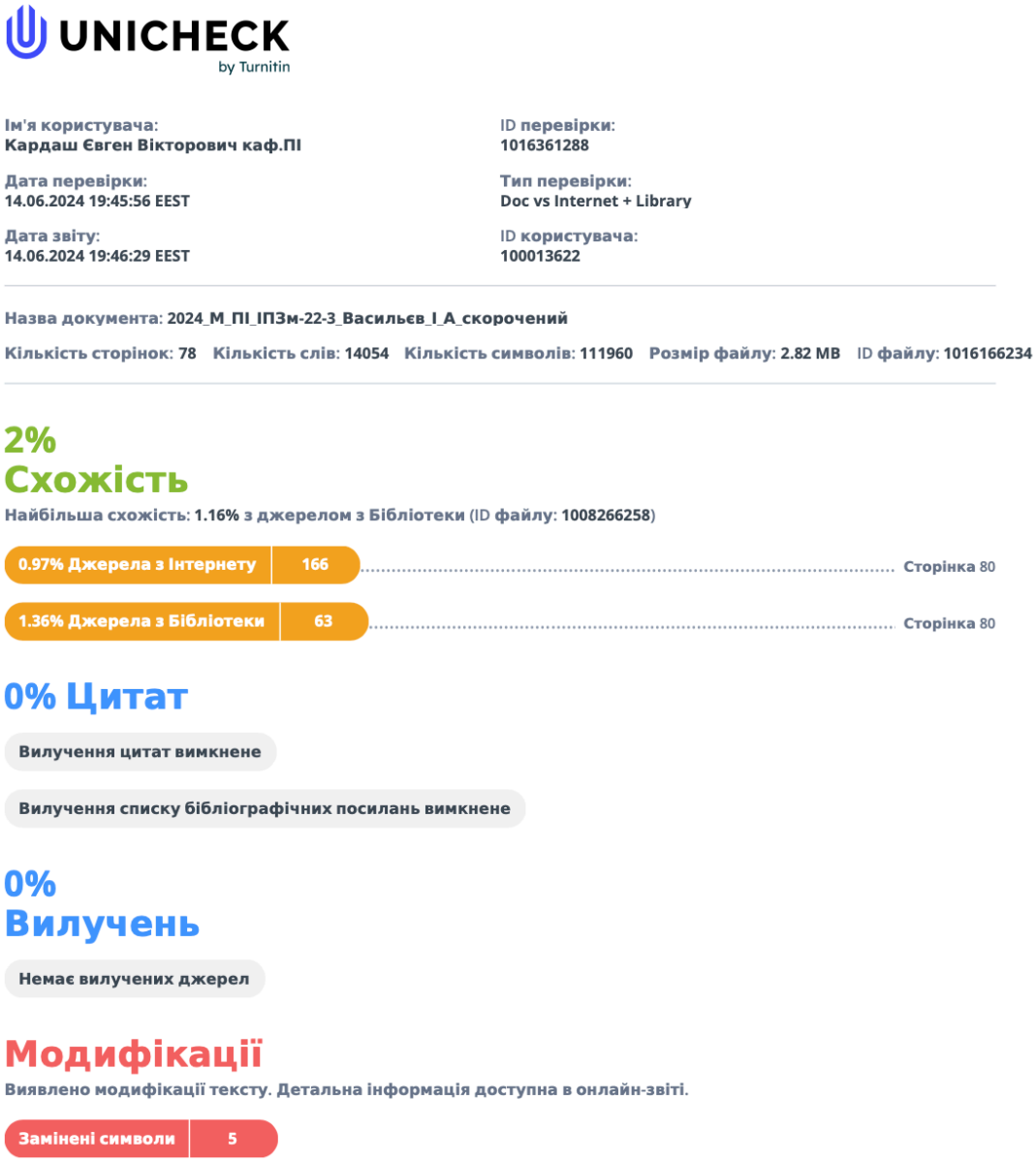
ДОДАТОК А

Результат перевірки на плагіат



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016361288

Дата перевірки:
14.06.2024 19:45:56 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
14.06.2024 19:46:29 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗм-22-3_Васильєв_І_А_скорочений

Кількість сторінок: 78 **Кількість слів:** 14054 **Кількість символів:** 111960 **Розмір файлу:** 2.82 MB **ID файлу:** 1016166234

2%
Схожість

Найбільша схожість: 1.16% з джерелом з Бібліотеки (ID файлу: 1008266258)

0.97% Джерела з Інтернету 166 ..... Сторінка 80

1.36% Джерела з Бібліотеки 63 ..... Сторінка 80

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 5

ДОДАТОК Б

Таблиця порівняння конкурентів

Характеристика	Siemens MindSphere	Splunk	ThingWorx	McAfee IoT Security	AWS IoT Analytics	IBM Watson IoT	Losant	Ubidots	Bosch IoT Suite
Масштабованість	Високий	Високий	Високий	Помірний	Високий	Помірний	Високий	Помірний	Високий
Ефективність	Надійна	Високий	Ефективний	Повищена безпека	Високий	Ефективний	Ефективний	Добре	Ефективний
Максимальне навантаження	Витримує значні навантаження	Високі навантаження	Витримує значні навантаження	Обробляє навантаження для безпеки	Витримує значні навантаження	Витримує різні навантаження	Витримує навантаження	Витримує навантаження	Може витримувати різноманітні навантаження
Функції безпеки	Адаптовано для промислового IoT	Повищена безпека	Підходить для промислового використання	Комплексна безпека	Надійні функції безпеки	Функції безпеки	Функції безпеки	Функції безпеки	Функції безпеки
Налаштування та інтеграція	Підтримує спеціальні програми	Широкі можливості налаштування	Спеціальні програми	Інтегруєтьс я з Безпекою	Безпроблемн а інтеграція з AWS	Інтегрується з IBM Cloud	Спеціальні програми	Програми спеціального керування	Параметри налаштування
Простота використання	Спеціально для промислових користувачів	Зручний інтерфейс	Зручний інтерфейс	Зосереджен о на безпеці	Зручний для користувача в AWS	Зручний інтерфейс	Розроблено з простотою використання	Зручна платформа	Зручний інтерфейс
Ефективність витрат	Індивідуальне ціноутворення	Вартість змінюється	Рентабельність для промисловості	Вартість змінюється	Економічно в AWS	Вартість змінюється	Економічні рішення	Економічні варіанти	Вартість змінюється
Гнучкість у виявленні аномалій	Спеціально для промислового	Дуже гнучкий	Гнучкість в інтеграції	Гнучкість у виявленні загроз	Гнучкість у налаштуванн ях	Гнучкість у реалізації	Гнучкість створення програм	Гнучка платформа	Гнучкість у реалізації

ДОДАТОК В

Програмний код

Програмний код моделі виявлення аномалій:

```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Input, Dropout, LayerNormalization, LSTM, Add, MultiHeadAttention, TimeDistributed, Conv1D, Bidirectional, Flatten
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import pandas as pd
import numpy as np

class ResidualBiLSTM(tf.keras.layers.Layer):
    def __init__(self, units, **kwargs):
        super(ResidualBiLSTM, self).__init__(**kwargs)
        self.units = units
        self.bilstm = Bidirectional(LSTM(units, return_sequences=True))
        self.layernorm = LayerNormalization()
        self.dense = Dense(units * 2)
        self.add = Add()

    def build(self, input_shape):
        super(ResidualBiLSTM, self).build(input_shape)

    def call(self, inputs):
        x = self.bilstm(inputs)
        x = self.layernorm(x)
        inputs_transformed = self.dense(inputs)
        return self.add([inputs_transformed, x])

def create_advanced_model(input_shape, num_heads=4, key_dim=64, ff_dim=128):
    inputs = Input(shape=input_shape)

    x = Conv1D(filters=64, kernel_size=3, padding='same', activation='relu')(inputs)

    x = ResidualBiLSTM(units=128)(x)
    x = ResidualBiLSTM(units=64)(x)

    attn_output = MultiHeadAttention(num_heads=num_heads, key_dim=key_dim)(x, x)
    attn_output = Dropout(0.3)(attn_output)
    out1 = LayerNormalization(epsilon=1e-6)(attn_output + x)

    ffn_output = TimeDistributed(Dense(units=1))(out1)
```

```
        outputs = Flatten()(ffn_output)

        outputs = Dense(1, activation='sigmoid')(outputs)

        model = Model(inputs=inputs, outputs=outputs)
        model.compile(optimizer='adam', loss='binary_crossentropy')
        return model

train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')

X_train = train_data.iloc[:, :-1].values
y_train = train_data.iloc[:, -1].values
X_test = test_data.iloc[:, :-1].values
y_test = test_data.iloc[:, -1].values

timesteps = 2  # Since we have only 2 features (temperature and humidity)
features = X_train.shape[1] // timesteps
assert X_train.shape[1] % timesteps == 0, "Features are not divisible by timesteps!"

X_train = X_train.reshape(-1, timesteps, features)
X_test = X_test.reshape(-1, timesteps, features)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
random_state=42)

with tf.keras.utils.custom_object_scope({'ResidualBiLSTM': ResidualBiLSTM}):
    input_shape = (timesteps, features)
    model = create_advanced_model(input_shape)

    early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

    history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_data=(X_val, y_val), callbacks=[early_stopping])

    model.save('bilstm_anomaly_detection_model.keras', save_format='keras')

    loaded_model = tf.keras.models.load_model('bilstm_anomaly_detection_model.keras',
custom_objects={'ResidualBiLSTM': ResidualBiLSTM})

    test_loss = loaded_model.evaluate(X_test, y_test)
    print("Test Loss:", test_loss)

    y_pred_probs = loaded_model.predict(X_test)
    y_pred_classes = (y_pred_probs > 0.5).astype(int)
```

```
        print("Classification Report:")
        print(classification_report(y_test, y_pred_classes))
        print("Confusion Matrix:")
        print(confusion_matrix(y_test, y_pred_classes))
        print("AUC:", roc_auc_score(y_test, y_pred_probs))
```

Програмний код серверу для обслуговування веб застосунку і застосування моделі:

```
from flask import Flask, jsonify
from flask_cors import CORS
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import serial
import csv
import time
from threading import Thread
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score


app = Flask(__name__)
CORS(app, resources={r"/*": {"origins": "*"}})

serial_port = '/dev/cu.usbmodem1101'
baud_rate = 9600
data_path = 'Model/test.csv'

class ResidualBiLSTM(tf.keras.layers.Layer):
    def __init__(self, units, **kwargs):
        super(ResidualBiLSTM, self).__init__(**kwargs)
        self.units = units
        self.bilstm    =    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units,
return_sequences=True))
        self.layernorm = tf.keras.layers.LayerNormalization()
        self.dense = tf.keras.layers.Dense(units * 2)
        self.add = tf.keras.layers.Add()

    def call(self, inputs):
        x = self.bilstm(inputs)
        x = self.layernorm(x)
        inputs_transformed = self.dense(inputs)
        return self.add([inputs_transformed, x])

def load_and_evaluate_model():
    test_data = pd.read_csv(data_path)
    X_test = test_data.iloc[:, :-1].values
```

```
            y_test = test_data.iloc[:, -1].values

            timesteps = 2
            features = X_test.shape[1] // timesteps
            assert X_test.shape[1] % timesteps == 0, "Features are not divisible by
timesteps!"

            X_test = X_test.reshape(-1, timesteps, features)

            loaded_model                                                    =
tf.keras.models.load_model('Model/bilstm_anomaly_detection_model.keras',
custom_objects={'ResidualBiLSTM': ResidualBiLSTM})

            y_pred_probs = loaded_model.predict(X_test)
            y_pred_classes = (y_pred_probs > 0.5).astype(int)

            report = classification_report(y_test, y_pred_classes, output_dict=True)
            matrix = confusion_matrix(y_test, y_pred_classes).tolist()
            auc = roc_auc_score(y_test, y_pred_probs)

            return {
                "report": report,
                "matrix": matrix,
                "auc": auc
            }

        def read_from_serial(port, baud):
            try:
                ser = serial.Serial(port, baud)
                with open(data_path, 'a', newline='') as csvfile:
                    csv_writer = csv.writer(csvfile)
                    if csvfile.tell() == 0:
                        csv_writer.writerow(['temperature', 'humidity', 'label'])
                    try:
                        while True:
                            line = ser.readline().decode('utf-8').strip()
                            print(line)
                            data = line.split(', ')
                            if len(data) == 3:
                                csv_writer.writerow(data)
                                csvfile.flush()
                    except KeyboardInterrupt:
                        print("Stopped by User")
                    finally:
                        ser.close()
                        csvfile.close()
            except serial.SerialException as e:
```

```
        print(f"Error: {e}")
        print("Retrying in 5 seconds...")
        time.sleep(5)
        read_from_serial(port, baud)

def read_serial_loop(port, baud):
    while True:
        read_from_serial(port, baud)
        time.sleep(5)

@app.route('/data', methods=['GET'])
def get_data():
    try:
        if os.path.exists(data_path):
            data = pd.read_csv(data_path)
            last_four_rows = data.tail(4)
            column_names = list(data.columns)
            response_data = {
                "columns": column_names,
                "data": last_four_rows.to_dict(orient='records')
            }
            response = jsonify(response_data)
            response.headers.add('Access-Control-Allow-Origin', '*')
            return response
        else:
            return jsonify({"error": "Data file not found"}), 404
    except Exception as e:
        print(f"Error: {e}")
        return jsonify({"error": str(e)}), 500

@app.route('/predict', methods=['GET'])
def predict():
    try:
        analysis = load_and_evaluate_model()
        response = jsonify(analysis)
        response.headers.add('Access-Control-Allow-Origin', '*')
        return response
    except Exception as e:
        print(f"Error: {e}")
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    serial_thread = Thread(target=read_serial_loop, args=(serial_port, baud_rate))
    serial_thread.start()
    app.run(port=5000, debug=False)
```

ДОДАТОК Г

Слайди презентації

«Дослідження методів виявлення аномалій даних у вимірюваннях пристроїв IoT»

Васильєв Ігор Антонович, ІПЗм-22-3
доцент кафедри іст, Хацько Н. Є.

25 червня 2024

Дослідження

Актуальність та стан галузі

- Швидкий розвиток Інтернету речей (IoT) викликає потребу в ефективних системах виявлення відхилень.

Чітке визначення напряму дослідження

- Вдосконалення алгоритмів машинного навчання для точного виявлення аномалій в потоках даних IoT.
- Розробка гібридних моделей, що комбінують різні підходи для підвищення ефективності.

Об’єкт дослідження

- Потокові дані від IoT-пристроїв, що містять інформацію про стан об’єктів та середовища їхнього функціонування.

Огляд літератури (аналогів)

Перелік основних джерел та теорій у галузі

Застосування алгоритму K-Means в аспекті кластеризації Industrial IoT:
Дослідження концентрується на вирішенні проблем швидкості зв'язку, використовуючи 4-рівневу мережеву топологію для цифрового виробництва та розвитку уніфікованої системи збору даних.

Архітектура туманно-хмарних обчислень для вирішення викликів IoT:
Поділ шару туману на три рівні дозволяє успішно кластеризувати різноманітні дані IoT за допомогою алгоритму DBSCAN.

Виявлення аномалій за допомогою ізольованого лісу та автокодування:
Методологія об'єднує алгоритм ізольованого лісу з автокодуванням для ефективного виявлення аномалій у даних про енергію.

Використання One-Class Support Vector Machine (OCSVM) для виявлення аномалій у зв'язку:
Новий алгоритм виявлення вторгнень, що базується на OCSVM, демонструє потенціал для ефективного визначення ненормальної поведінки в промислових системах керування.

Застосування LSTM для динамічного розподілу ресурсів у програмах:
Використання LSTM в поєднанні з динамічною маршрутизацією хмарного центру обробки даних показує потенціал у підвищенні ефективності розподілу ресурсів програм.

Зазначення прогалин у наявних дослідженнях

Обмеження моделей:
Більшість досліджень обмежуються деякими моделями чи алгоритмами, що може обмежувати їхню універсальність та застосовність у різних сценаріях.

Необхідність додаткового валідації:
Багато робіт вимагають додаткової валідації на реальних даних для підтвердження ефективності запропонованих методів.

Обмеженість досліджень:
Деякі дослідження можуть бути обмежені в обсязі або в рамках застосування, що може впливати на загальну відповідність їхніх висновків.

Недостатність порівняльного аналізу:
У деяких дослідженнях може бути недостатньо порівняльного аналізу з іншими сучасними методами.

Відсутність стандартизації:
Відсутність стандартизації у методиках оцінки ефективності алгоритмів може ускладнювати порівняння результатів різних досліджень.

Постановка задачі

Чітке формулювання проблеми

Проблема. Традиційні системи виявлення аномалій стикаються з такими проблемами, як затримка, обмеження масштабованості та вразливість до окремих точок збою.

Пропоноване рішення: інтеграція передових механізмів виявлення аномалій, таких як LSTM, із механізмами уваги.

Переваги: підвищена точність і ефективність у виявленні аномалій, визначення пріоритетів критичних даних і покращене прийняття рішень у складних середовищах даних.

Опис очікуваних результатів

Цілі: розробити прототип для виявлення аномалій за допомогою LSTM з механізмами уваги в середовищах IoT.

Очікувані результати: підвищення цілісності та безпеки даних, вирішення унікальних проблем зі складними даними в системах IoT.

Програми: інтерфейси моніторингу та керування в режимі реального часу для пристроїв IoT, що сприяють захисту даних та інноваціям в IoT.

Методологія

Опис використаних методів дослідження

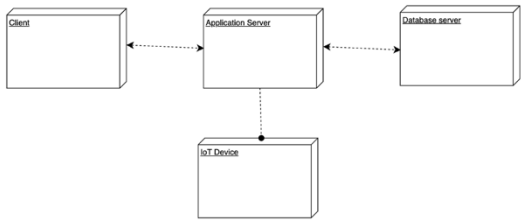
Визначення конкретних цілей та масштабів
Огляд літератури
Формулювання методології
Збір і попередня обробка даних
Вибір алгоритму та структура інтеграції
Реалізація та оптимізація моделі
Тестування та оцінка
Уточнення методології дослідження
Виклики та можливості інтеграції
Аналіз та інтерпретація результатів
Документація та звітність

Інструментарій та технології, використані в роботі

Frontend: React, JavaScript
Контроль версій: GitHub
Інтернет речей (IoT): Arduino IDE, Python
Алгоритми: Python, Visual Studio Code, WebStorm
Зберігання даних: Docker, Firebase, CSV
Машинне навчання: Jupyter Notebook, TensorFlow, PyTorch
Візуалізація даних: Tableau, Power BI, Matplotlib, Seaborn

Архітектура система для проведення експериментального дослідження

1) Клієнт (інтерфейс користувача):
   a) Представляє інтерфейс користувача, за допомогою якого користувачі взаємодіють із системою.
   b) Зв'язується з сервером додатків для надсилання запитів і отримання відповідей.
2) Сервер додатків:
   a) Керує бізнес-логікою, включаючи обробку даних, виявлення аномалій та оптимізацію.
   b) Зв'язується з сервером бази даних для зберігання та отримання даних.
   c) Зв'язується з пристроями IoT для передачі даних і оптимізації.
3) Сервер бази даних:
   a) Зберігає дані користувача, параметри, графіки та інформацію про пристрій IoT.
   b) За потреби надає дані на сервер додатків.
4) Пристрої IoT:
   a) Представляє фізичні пристрої в розумному середовищі.
   b) Зв'язується з сервером додатків для передачі даних і оптимізації.



Опис програмного забезпечення, що було використано у дослідженні

Опис процесу розробки

1. Збереження натренованої моделі до окремого файлу
2. Flask API для обслуговування моделей
3. Розробка серверної частини
4. Реалізація веб сервісу для керування системою
5. Реалізація IOT пристрою

Вибрані мови програмування та фреймворки

- React
- Node.js
- Python
- Javascript
- Flask
- Matplotlib

Зміст проведеного експерименту

Методи:
Використання LSTM з механізмами уваги для виявлення аномалій в IoT.

Вхідні дані:
Набори даних реального світу, включаючи дані NASA про турбовентилятори та інші.

Критерії:
Точність виявлення аномалій, затримка, масштабованість та відмовостійкість.

Вимірювання:
Вимірювання продуктивності за ключовими параметрами, включаючи точність, AUC, ROC показники.

Послідовність:

1. Обробка та попередня підготовка наборів даних.
2. Вибір та інтеграція алгоритмів машинного навчання.
3. Оптимізація моделі LSTM з увагою на архітектуру IoT.
4. Тестування та оцінка ефективності прототипу.

Результати експерименту

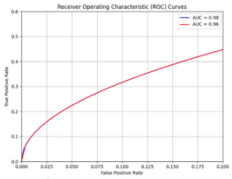
Таблиця 8.1 - ключові показники ефективності з кожного набору даних

Dataset	Accuracy	Precision	Recall	F1-score	AUC
NASA	0.95	0.94	0.96	0.95	0.98
IoT	0.92	0.91	0.93	0.92	0.96

Таблиця 8.2 – Порівняння результатів вимірювання серед конкурентів моделі

Algorithm	Accuracy	AUC
Isolation Forest	0.85	0.90
One-Class SVM	0.88	0.92
k-means clustering	0.82	0.88
LSTM-based model	0.92	0.96
Our Model	0.95	0.98

ROC Крива моделі



Графіки точності в застосунку на даних IoT

Аналіз отриманих результатів

Аналіз отриманих результатів: Результати дослідження були успішно порівняні з поставленими цілями, засвідчуючи ефективність моделі LSTM з механізмами уваги виявлення аномалій у розумних середовищах IoT. Отримані дані дозволили зробити висновки щодо точності виявлення аномалій, швидкості реакції та ефективності розробленого прототипу. Інтерпретація результатів показала, що застосування LSTM з механізмами уваги може суттєво поліпшити системи виявлення аномалій в IoT середовищах.

Висновки: Досягнення цілей дослідження підтверджують ефективність обраного підходу до виявлення аномалій у складних даних IoT. Рекомендації включають розвиток інтеграції LSTM з механізмами уваги у виробничі середовища та подальшу оптимізацію алгоритмів для підвищення продуктивності.

Наступні Кроки: Плани включають подальшу розробку та вдосконалення моделі, враховуючи отримані результати, а також розширення області застосування у різних секторах промисловості та бізнесу. Перспективи включають подальше тестування в реальних умовах та співпрацю з партнерами для впровадження рішень у практичних сценаріях.

Публікація результатів

Exploring LSTM with Attention for Anomaly Detection

Natalia Khatsko[a,1], Ihor Vasyliev[a], Dmytro Orlovskyi[b] and Kyrylo Khatsko[b]

[a] National University of Radio Electronics, Kharkiv, 61166, Ukraine
[b] National Technical University «Kharkiv Polytechnic Institute», Kharkiv, 61002, Ukraine

Abstract

This paper presents an innovative anomaly detection system exploring the integration of Long Short-Term Memory (LSTM) networks empowered with attention mechanisms. The study delves into the dynamic landscape of anomaly detection methodologies, emphasizing the potential opportunities afforded by LSTM with attention, particularly within complex industrial datasets. Leveraging the fusion of LSTM and attention mechanisms, this approach aims to discern subtle anomalies and intricate patterns within industrial data streams. By selectively focusing on critical elements within the data, the system enhances anomaly detection accuracy and interpretability. Detailed exploration of the system's design, algorithms, and implementation highlights the promising capabilities of LSTM with attention in uncovering nuanced anomalies within diverse industrial datasets. This study underscores the paradigm-shifting potential of LSTM with attention mechanisms in reshaping anomaly detection practices, offering prospects for enhanced anomaly discernment, precise pattern recognition, and improved understanding of intricate industrial data.

Keywords [1]

Edge-cloud computing architecture, LSTM, anomaly detection, data mining techniques, attention mechanism

1. Introduction and problem statement

In the vast realm of industrial operations, the ability to detect anomalies stands as a linchpin for ensuring optimal performance, reducing downtime, and preempting potential catastrophic breakdowns. However, conventional centralised anomaly detection systems grapple with persistent challenges such as latency, scalability limitations, and susceptibility to single points of failure. To transcend these hurdles and meet the burgeoning demands of modern industries, there's a burgeoning need for innovative solutions.

Enter the vision of a decentralized anomaly detection prototype integrating an edge cloud computing architecture. This groundbreaking approach aims to revolutionize anomaly detection practices by redistributing computational tasks across edge devices and cloud resources. The envisioned system seeks to herald a new era of real-time analysis, scalability enhancements, and fortified fault tolerance, enabling industries to swiftly identify anomalies and swiftly implement responsive maintenance strategies.

The application spectrum for this paradigm-shifting system spans across diverse industrial domains such as manufacturing, energy, transportation, and logistics, where proactive anomaly detection plays a pivotal role in ensuring smooth operations. The market's dire need for agile and robust anomaly detection methodologies [1] propels the urgency for sophisticated systems capable of real-time monitoring and adaptable scalability. The amalgamation of decentralized anomaly detection with edge cloud computing technology not only addresses these pressing market needs but also unveils

Proceedings of the Information Systems and Technology (IST-2023), November 28-December 01, 2023, Kharkiv, Ukraine
EMAIL: n.khatsko@gmail.com (N. Khatsko); ihor.vasyliev@nure.ua (I. Vasyliev); orlovskyi.dm@gmail.com (D. Orlovskyi); kyrylo.khatsko@khpi.edu.ua (K. Khatsko)
ORCID: 0000-0002-2143-8240 (N. Khatsko); 0003-0307-2526-5406 (I. Vasyliev); 0000-0002-8241-2988 (D. Orlovskyi);0000-0003-3315-1311 (K. Khatsko)

IST-2023
28 November - 01 December
Kharkiv 2023 Ukraine

Certificate

Ihor Vasyliev

participated in the
12th International Scientific and Technical Conference
«Information Systems and Technologies»
(IST-2023)

15 hours / 0.5 ECTS credits

Co-chairman of the organizing committee *Yuri Romanenkov*

Kharkiv 2023

Підсумки

Отримані результати є реалістичними і мають значну корисність для промислових та бізнесових секторів. Дослідження підтверджує можливість застосування розробленої моделі у реальних умовах, забезпечуючи ефективне виявлення аномалій у системах IoT.
Наступні кроки включають розширення обсягу досліджень на інші сектори та аспекти застосування, такі як медицина, фінанси та транспорт. Можливості для подальшого вдосконалення включають оптимізацію алгоритмів, розробку нових функціональних можливостей та розширення спектру використовуваних даних. Ці результати відкривають шлях до покращення ефективності систем управління, зменшення ризиків та підвищення якості обслуговування в різних галузях, спираючись на аналіз даних IoT та застосування навчальних моделей.

ДОДАТОК Д

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

1

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-3 (група)

Васильєв Ігор Антонович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

17.06.2024