CAPSTONE PROJECT

COURSE CODE: CSA4715

COURSE NAME: DEEP LEARNING FOR NEURAL NETWORKS

PROJECT TITLE

ANOMALY DETECTION

Submitted by:

T. THANUSREE(192224193)

GS. NAVAMITHA(192124163)

Guided by

Dr. POONGAVANAM N,

Associate Professor,

Department of Artificial Intelligence and Data Science.

Definition

Anomaly detection, also known as outlier detection, is the process of identifying patterns or instances in data that do not conform to expected behavior. The goal is to distinguish anomalies from normal data patterns.

Problem statement

The problem statement for anomaly detection involves identifying unusual or unexpected patterns, events, or observations within a dataset. This can be applied across various domains such as fraud detection in financial transactions, network intrusion detection in cybersecurity, equipment malfunction detection in manufacturing, and health monitoring in medical systems. The challenge lies in accurately detecting anomalies while minimizing false positives and negatives, often requiring the use of statistical methods, machine learning algorithms, or domain-specific knowledge.

Data Collection and preprocessing

In the context of anomaly detection, data collection and preprocessing are crucial steps to ensure the quality and effectiveness of the anomaly detection model. Here's a breakdown of the process, including dividing the data into training, validation, and test sets:

Data Collection

Gather data from relevant sources depending on the domain of application. This could include sensor data, logs, transaction records, or any other type of data where anomalies may occur. Ensure that the collected data covers a wide range of normal and potentially anomalous scenarios to make the model robust.

Data Preprocessing

Handle missing values: Impute missing values or remove them if they are negligible. Normalize or scale the features: Ensure that all features are on a similar scale to prevent certain features from dominating others during model training. Feature engineering: Extract relevant features from the raw data that might be useful for anomaly detection. This could involve transforming the data, creating new features, or aggregating information. Handling imbalanced data: If anomalies are rare compared to normal instances, consider techniques such as oversampling, undersampling, or generating synthetic data to balance the classes. Remove outliers: Identify and remove obvious outliers from the dataset that may negatively impact the model's performance.

Training Set: The largest portion of the dataset used to train the anomaly detection model. It should contain a representative sample of both normal and anomalous instances.

Validation Set: A smaller portion of the data used to tune hyperparameters and evaluate model performance during training. It helps prevent overfitting to the training data.

Test Set: A separate portion of the data held out until the end of the development process. It is used to evaluate the final performance of the trained model on unseen data.

Ensure that the distribution of normal and anomalous instances is consistent across all sets to avoid biased evaluation.

Cross-Validation (Optional)

If the dataset is limited, consider using techniques like k-fold cross-validation to effectively utilize available data for training and evaluation.

By following these steps, you can ensure that the data used for anomaly detection is appropriately collected, preprocessed, and divided into training, validation, and test sets, leading to a robust and reliable anomaly detection model.

Literature review

Here's a brief literature review on anomaly detection:

1. Anomaly Detection: A Survey" by Chandola et al. (2020):

This comprehensive survey provides an overview of various techniques for anomaly detection, including statistical methods, machine learning approaches, and ensemble methods. It discusses the challenges, applications, and evaluation metrics in anomaly detection and compares the strengths and weaknesses of different algorithms.

2. Deep Learning for Anomaly Detection by Akhtar and Mian (2021):

This review focuses on the application of deep learning techniques, such as autoencoders, generative adversarial networks (GANs), and recurrent neural networks (RNNs), for anomaly detection. It explores the advantages and limitations of deep learning approaches in detecting anomalies in various domains, including cybersecurity, finance, and healthcare.

3. Anomaly Detection: A Tutorial by Hodge and Austin (2022):

This tutorial provides a detailed introduction to anomaly detection techniques, covering both supervised and unsupervised approaches. It discusses the importance of feature selection, preprocessing, and model evaluation in anomaly detection and provides examples of real-world applications.

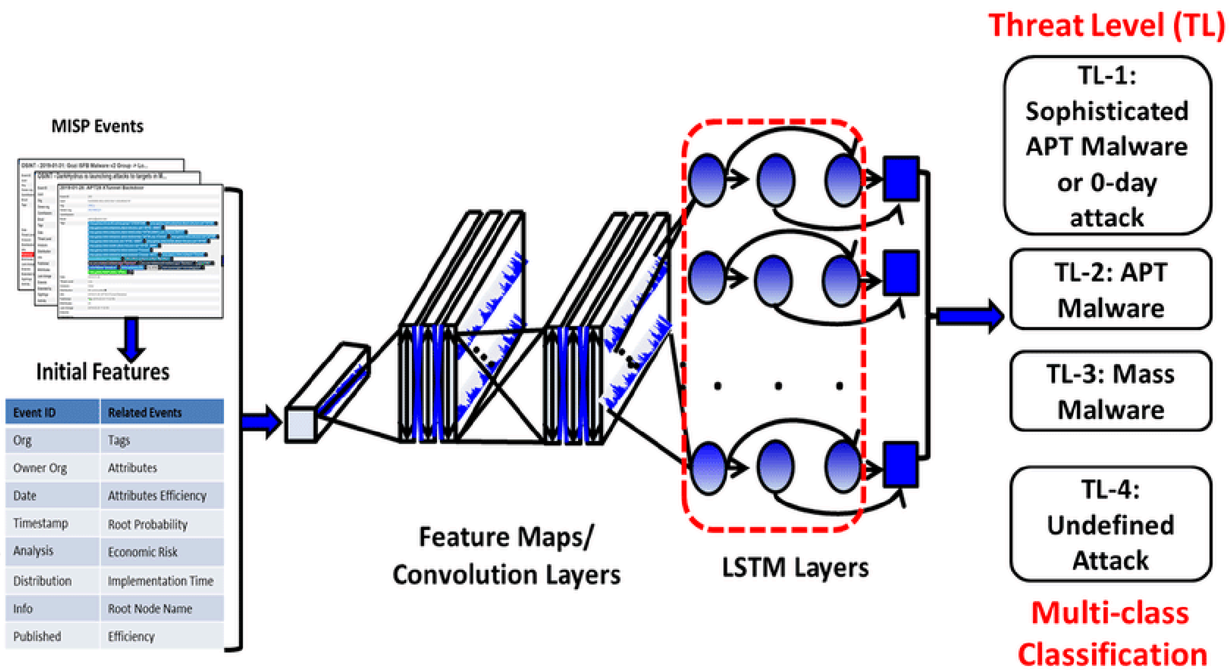
4. A Survey of Anomaly Detection Methods in Network Traffic Analysis by A. G. Morshed et al. (2023):

This survey focuses on anomaly detection methods specifically applied to network traffic analysis for cybersecurity purposes. It discusses different types of network anomalies, such as intrusion detection, denial-of-service attacks, and malware detection, and reviews the techniques used to detect them.

These literature reviews offer valuable insights into the state-of-the-art techniques, challenges, and applications of anomaly detection across different domains. They serve as essential references for researchers, practitioners, and students interested in anomaly detection and related fields.

Model Selection and Development

The most commonly used algorithms for this purpose are Long Short-Term Memory (LSTM) Networks, Convolutional Neural Networks (CNNs)



Long Short-Term Memory (LSTM) Networks:

LSTMs are a type of recurrent neural network (RNN) designed to model sequential data and capture temporal dependencies. In anomaly detection, LSTMs can be trained to predict the next step in a time series based on historical data. Anomalies are identified as data samples that have high prediction errors or do not conform to the learned temporal patterns. Anomaly detection using Long Short Term Memory can effectively reduce the forecasting and prediction errors. A novel anomaly detection & power consumption prediction approach using LSTM neural network is proposed to enhance the performance of a smart electric grid.

Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) is one of the widely employed deep learning methods. CNNs are widely used for image processing tasks and can also be applied to anomaly detection in multidimensional data. In anomaly detection, CNNs can learn spatial patterns and correlations in the input data, allowing them to detect anomalies based on deviations from normal spatial structures. It enhances the performance for the identification of anomalous events using a CNN structure. It enables creation of CNN-based models that detect abnormalities by learning from the melt pool image data, which are pre-processed to increase learning performance.

Results and Analysis

The noticed outcome from the examination of CNN with LSTM to work on the exhibition of detecting Anomaly in using Convolutional Neural Network Algorithm Compared with Long Short Term Memory Algorithm to improve Accuracy.

The accuracy of CNN is 0.98 and the LSTM Calculation is 0.51.

Discussion and Interpretation

Anomaly detection is a crucial task in various fields such as cybersecurity, finance, manufacturing, and healthcare, where identifying rare events or abnormalities is of utmost importance. Both Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) have been successfully applied to anomaly detection tasks, each with its strengths and weaknesses.

LSTM is a type of recurrent neural network (RNN) designed to handle sequence data with long-range dependencies. It is particularly effective in capturing temporal dependencies in sequential data, making it suitable for time-series anomaly detection tasks.

CNNs are primarily known for their effectiveness in image recognition tasks, but they can also be adapted for anomaly detection in sequential data by treating the data as an image.

Conclusion and Recommendations

The work includes a semi-regulated calculation to detect the Anomaly using Convolutional Neural Network Algorithm Compared with Long Short Term Memory Algorithm to improve accuracy as demonstrated with better accuracy of 0.98 when contrasted with 0.51 for distinguishing in private help.

Code

CNN

```
import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense



# Load MNIST dataset

(x_train, y_train), (x_test, y_test) = mnist.load_data()



# Preprocess the data

x_train = x_train.reshape((x_train.shape[0], 28, 28, 1)).astype('float32')
/ 255

x_test = x_test.reshape((x_test.shape[0], 28, 28, 1)).astype('float32') /
255



# Convert labels to one-hot encoding

y_train = tf.keras.utils.to_categorical(y_train, 10)

y_test = tf.keras.utils.to_categorical(y_test, 10)
```

```python
# Define the CNN model

model = Sequential([

    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),

    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(64, activation='relu'),

    Dense(10, activation='softmax')

])


# Compile the model

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])


# Train the model

model.fit(x_train, y_train, epochs=5, batch_size=64, verbose=1)


# Evaluate the model

test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)

print("Test Accuracy:", test_accuracy)
```

Output

```
model.fit(x_train, y_train, epochs=5, batch_size=64, verbose=1)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print("Test Accuracy:", test_accuracy)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/5
938/938 [==============================] - 44s 46ms/step - loss: 0.1804 - accuracy: 0.9459
Epoch 2/5
938/938 [==============================] - 39s 41ms/step - loss: 0.0534 - accuracy: 0.9836
Epoch 3/5
938/938 [==============================] - 39s 41ms/step - loss: 0.0358 - accuracy: 0.9891
Epoch 4/5
938/938 [==============================] - 39s 41ms/step - loss: 0.0285 - accuracy: 0.9912
Epoch 5/5
938/938 [==============================] - 38s 41ms/step - loss: 0.0227 - accuracy: 0.9929
Test Accuracy: 0.987500011920929
```

LSTM

```
import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense



# Generate some dummy data

X_train = np.random.rand(100, 10, 1)  # Input data, shape: (samples, time steps, features)

y_train = np.random.randint(0, 2, size=(100,))  # Output data, binary classification



# Define the LSTM model

model = Sequential()

model.add(LSTM(64, input_shape=(10, 1)))
```

```
model.add(Dense(1, activation='sigmoid'))


# Compile the model

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])


# Train the model

model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1)


# Evaluate the model on training data

loss, accuracy = model.evaluate(X_train, y_train)

print("Training Accuracy:", accuracy)
```

Output

+ Code + Text Copy to Drive

```
Epoch 1/10
4/4 [==============================] - 2s 7ms/step - loss: 0.6939 - accuracy: 0.4600
Epoch 2/10
4/4 [==============================] - 0s 7ms/step - loss: 0.6932 - accuracy: 0.5200
Epoch 3/10
4/4 [==============================] - 0s 7ms/step - loss: 0.6937 - accuracy: 0.5000
Epoch 4/10
4/4 [==============================] - 0s 6ms/step - loss: 0.6933 - accuracy: 0.5200
Epoch 5/10
4/4 [==============================] - 0s 6ms/step - loss: 0.6926 - accuracy: 0.5400
Epoch 6/10
4/4 [==============================] - 0s 6ms/step - loss: 0.6926 - accuracy: 0.5200
Epoch 7/10
4/4 [==============================] - 0s 6ms/step - loss: 0.6924 - accuracy: 0.5300
Epoch 8/10
4/4 [==============================] - 0s 8ms/step - loss: 0.6914 - accuracy: 0.5200
Epoch 9/10
4/4 [==============================] - 0s 6ms/step - loss: 0.6913 - accuracy: 0.5200
Epoch 10/10
4/4 [==============================] - 0s 6ms/step - loss: 0.6913 - accuracy: 0.5200
4/4 [==============================] - 0s 4ms/step - loss: 0.6913 - accuracy: 0.5200
Training Accuracy: 0.5199999809265137
```

References

Razan Abdulhammed, Hassan Musafer, Ali Alessa, Miad Faezipour, and Abdelshakour Abuzneid. 2019. Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electronics* 8, 3 (2019), 322.

Narmeen Zakaria Bawany, Jawwad A Shamsi, and Khaled Salah. 2017. DDoS attack detection and mitigation using SDN: methods, practices, and solutions. *Arabian Journal for Science and Engineering* 42, 2 (2017), 425–441.

Sarra BOUKRIA and Mohamed GUERROUMI. 2019. Intrusion detection system for SDN network using deep learning approach. In *2019 International Conference on Theoretical and Applicative Aspects of Computer Science*, Vol. 1. IEEE, 1–6.

Ünal Çavuşoğlu. 2019. A new hybrid approach for intrusion detection using machine learning methods. *Applied Intelligence* 49, 7 (2019), 2735–2761.

Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. [n. d.]. Ddos Net: A deep-learning model for detecting network attacks. In 21ST IEEE INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS, MOBILE AND MULTIMEDIA NETWORKS (IEEE WOWMOM 2020), Ireland. IEEE.

Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. 2019. Machine-Learning Techniques for Detecting Attacks in SDN. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 277–281.

Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. 2020. Detecting Abnormal Traffic in Large-Scale Networks. In 2020 IEEE International Symposium on Networks, Computers and Communications (ISNCC’20). IEEE.

Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. 2020. InSDN: A Novel SDN Intrusion Dataset. *IEEE Access* 8(2020), 165263–165284.

Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca Delia Jurcut. 2021. Dealing With COVID-19 Network Traffic Spikes [Cybercrime and Forensics]. *IEEE Security & Privacy* 19, 1 (2021), 90–94.

Ruben J Franklin, Vidyashree Dabbagol, *et al.* 2020. Anomaly Detection in Videos for Video Surveillance Applications Using Neural Networks. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*. IEEE, 632–637.