ANOMALY DETECTION IN TIME SERIES DATA

Department of Computer Science

Pearl Miglani 011855836

Varsha Niharika Mallampati 011860768

Chinmay Chabbi 011858333

Abstract

Anomaly detection in time series data is a critical task with applications in various domains such as finance, healthcare, and industrial monitoring. In this project, we aim to implement anomaly detection on time series data using unsupervised deep learning techniques. We choose this problem because anomaly detection is crucial for identifying unusual behavior or events that deviate from normal patterns in time series data. Detecting anomalies early can help prevent potential issues, reduce downtime, and improve overall system reliability and performance. This project aims to experiment on unsupervised datasets and algorithms to conclude a relationship between types of datasets and best suited deep learning algorithms for most accuracy.

Introduction

Time series data is integral to various applications, from financial market analysis to patient health monitoring. Anomalies in this data often signal critical, actionable insights. However, accurately detecting these anomalies are challenging due to the complexity and dynamic nature of the data. Anomaly Detection has many real world use case like-

• Improving security by identifying suspicious activities.

• Enhancing fraud detection in financial services Monitoring health and well-being by detecting anomalies in patient data.

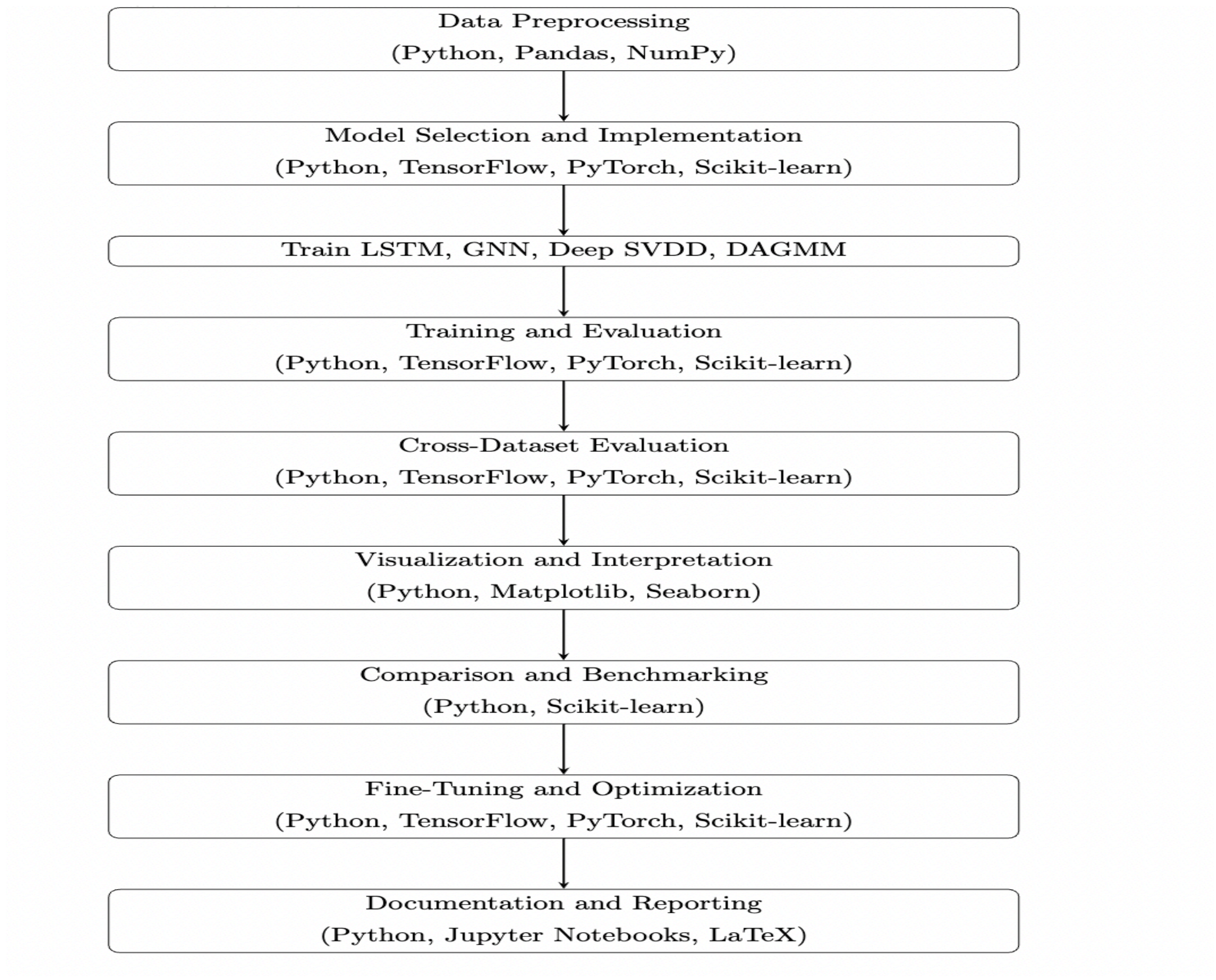
Our project aims to determine the ideal unsupervised deep learning algorithm for anomaly detection in time series data by experimenting and calculating the time and accuracy on four state-of-the-art datasets namely Water Distribution (WADI)[5] ,Server Machine Dataset(SMD)[6], Soil Moisture Active Passive (SMAP)[7] satellite and Mars Science Laboratory (MSL)[7] rover Datasets.

Literature Review

A brief review of various unsupervised deep learning methods include:

1. **Long Short-Term Memory (LSTM)** is a specialized type of recurrent neural network (RNN) designed to model sequential data by addressing the vanishing gradient problem. LSTMs utilize memory cells and gating mechanisms to selectively retain and update information over time, enabling them to capture long-range dependencies in sequential data more effectively than traditional RNNs. With components like input, forget, and output gates, LSTMs can regulate the flow of information through the network, making them well-suited for tasks such as time series prediction, natural language processing, and speech recognition.
2. **Graph Neural Networks (GNNs)** are specialized neural networks designed for graph- structured data. They can learn representations of nodes and edges by aggregating information from neighboring elements, enabling them to capture complex relationships within graphs. GNNs are used for tasks like node classification, link prediction, and graph classification, making them valuable in domains such as social network analysis, recommendation systems, and drug discovery.
3. **Deep Support Vector Data Description (Deep SVDD)** is an unsupervised anomaly detection method. It learns a low-dimensional representation of normal data instances, encapsulating them within a tight hypersphere in the learned feature space. Deep SVDD identifies anomalies as data points falling outside this hypersphere, making it suitable for unsupervised learning scenarios where labeled anomaly data is scarce.
4. The **Deep Autoencoding Gaussian Mixture Model (DAGMM)** is an unsupervised anomaly detection technique. It combines autoencoder neural networks and Gaussian mixture models (GMMs) to learn a low-dimensional representation of data and model its distribution simultaneously. DAGMM detects anomalies by identifying instances with low probability under the learned data distribution, making it effective for detecting anomalies in high-dimensional datasets without labeled anomaly data.

Technical Plan


Data Preprocessing
(Python, Pandas, NumPy)
Model Selection and Implementation
(Python, TensorFlow, PyTorch, Scikit-learn)
Train LSTM, GNN, Deep SVDD, DAGMM
Training and Evaluation
(Python, TensorFlow, PyTorch, Scikit-learn)
Cross-Dataset Evaluation
(Python, TensorFlow, PyTorch, Scikit-learn)
Visualization and Interpretation
(Python, Matplotlib, Seaborn)
Comparison and Benchmarking
(Python, Scikit-learn)
Fine-Tuning and Optimization
(Python, TensorFlow, PyTorch, Scikit-learn)
Documentation and Reporting
(Python, Jupyter Notebooks, LaTeX)

State of the art datasets used for training and testing are WADI(5.99% anomalies), SMD(4.16% anomalies), SMAP(13.13% anomalies) and MSL(10.72% anomalies). The deep learning methods used are LSTM and DAGMM.

Technical Implementation

Implementation for LSTM:

1. **Import Libraries**
    a. tensorflow or keras for building neural networks.
    b. numpy for numerical operations.
2. **Data Preparation**
    a. Prepare your sequential data (time series, text data, etc.).
    b. Split the data into training and testing sets.
3. **Build the LSTM Model**
    a. Initialize a sequential model.
    b. Add LSTM layers with desired parameters (number of units, activation function, etc.).
    c. Add any additional layers like dense layers for output.
4. **Compile the Model**
    a. Specify loss function (e.g., mean squared error for regression tasks, categorical cross-entropy for classification).
    b. Choose an optimizer (e.g., Adam, RMSprop).
    c. Optionally, specify metrics for evaluation.
5. **Train the Model**
    a. Fit the model to the training data.
    b. Specify batch size and number of epochs.
6. **Evaluate the Model**
    a. Evaluate the model performance on the test data.
    b. Calculate metrics like accuracy, loss, etc.

Implementation for Graph Neural Networks (GNNs):

1. **Import Libraries**
    a. tensorflow or pytorch for building neural networks.
    b. dgl, networkx, or igraph for working with graph structures.
2. **Data Preparation**
    a. Represent your data as a graph (nodes and edges).
    b. Split the data into training and testing sets.
3. **Build the GNN Model**
    a. Define node and edge features.
    b. Construct the GNN model architecture, including message passing layers and readout functions.

4. **Train the Model**
    a. Use graph-level or node-level tasks for training.
    b. Specify loss function and optimizer.
    c. Fit the model to the training data.
5. **Evaluate the Model**
    a. Evaluate the model performance on the test data.
    b. Calculate appropriate evaluation metrics for graph-related tasks.

Implementation for Deep Support Vector Data Description (Deep SVDD):

1. **Import Libraries**
    a. tensorflow or pytorch for building neural networks.
    b. numpy for numerical operations.
2. **Data Preparation**
    a. Prepare your data, ensuring it is suitable for unsupervised learning.
    b. Normalize the data if needed.
3. **Build the Deep SVDD Model**
    a. Define the neural network architecture, typically consisting of encoder layers.
    b. Implement a loss function that encourages the encoder to learn a compact representation of normal data.
4. **Train the Model**
    a. Train the model using only normal data.
    b. Optimize the encoder weights to minimize the defined loss.
5. **Detect Anomalies**
    a. After training, use the learned encoder to encode test data.
    b. Identify anomalies as data points lying outside a certain threshold from the learned normal data representation.

Implementation for Deep Autoencoding Gaussian Mixture Model (DAGMM):

1. **Import Libraries**
    a. tensorflow or pytorch for building neural networks.
    b. numpy for numerical operations.
    c. scikit-learn for GMM implementation.
2. **Data Preparation**
    a. Prepare your high-dimensional data.
    b. Normalize the data if necessary.
3. **Build the DAGMM Model**

    a. Implement an autoencoder neural network to learn a low-dimensional representation of the data.
    b. Combine the autoencoder with a Gaussian Mixture Model (GMM) to model the data distribution.

4. **Train the Model**
    a. Train the DAGMM model using unsupervised learning techniques.
    b. Optimize the parameters of both the autoencoder and the GMM simultaneously.
5. **Detect Anomalies**
    a. After training, use the learned model to compute the log-likelihood of each data point.
    b. Identify anomalies as data points with low likelihood under the learned data distribution.

Complete Results

Code Implementation

1. **LSTM** **:** https://colab.research.google.com/drive/11En6wMH8gHU5H83vV9WglYKZEpsnXQnA#scrollTo=BcF1H5 2ZjhOk
2. **GNN** **:** https://colab.research.google.com/drive/1WY37EA-LJ5mJKhABwRwPDaOld3hoA0ot#scrollTo=7oq7DF2Y
3. **SVDD** **:** https://colab.research.google.com/drive/1iZ0USWekewrfDxd0bnLVpLI2qJ3bNd-j#scrollTo=NvvHycwzts_s
4. **DAGMM** **:** https://colab.research.google.com/drive/1N7apLHAvZtKeR57jqwRUuyTKpgeev3uc#scrollTo=wGgSgdV1xB9m

Evaluation

Results for LSTM , DAGMM, GNN and Deep SVDD methods

Accuracy	WADI	SMD	SMAP	MSL
LSTM	0.46	0.86	0.71	0.85
DAGMM	0.22	0.67	0.63	0.75
GNN	0.68	0.74	0.59	0.72
Deep SVDD	0.55	0.68	0.61	0.69

Analysis

The chosen datasets exhibit different characteristics so that one can determine the best algorithm for each characteristic.

The algorithms are marked to choose based on average accuracy, train-test sample ratio, dimensionality and anomaly percentage.

Dataset	Train-Test Ratio	Dimensions	Anomaly Percentage
WADI	6.07	123	5.99
SMD	0.999978	1064	4.16
SMAP	0.316	1375	13.13
MSL	0.792	1485	10.72

Average Accuracy:

LSTM shows average accuracy of 0.72 which is the highest as compared to other methods.

Performance Across Train-Test Ratios:

1. LSTM: Consistently performs well across different ratios.
2. DAGMM: Shows a decrease in performance as the train-test ratio decreases.
3. GNN: Performs well across different ratios, with a slight decrease in lower ratios.
4. Deep SVDD: Maintains relatively stable performance across different ratios.

Performance on High vs. Low Dimensional Data:

1. LSTM: Suitable for both high and low-dimensional data.
2. DAGMM: Can handle both types of data but may struggle with high-dimensional data.
3. GNN: Generally performs well on both high and low-dimensional data.
4. Deep SVDD: Primarily designed for high-dimensional data and works well in that context.

Performance Across Anomaly Percentages:

1. LSTM: Performs relatively consistently across different anomaly percentages.
2. DAGMM: Shows some variability in performance, possibly struggling with higher anomaly percentages.
3. GNN: Maintains stable performance across different anomaly percentages.
4. Deep SVDD: Performs well across different anomaly percentages, particularly in detecting anomalies in high-dimensional data.

Conclusion Based on the analysis:

1. Overall Accuracy Average: LSTM achieves the highest average accuracy.
2. Train:Test Ratios: LSTM and GNN exhibit more consistent performance across different ratios.
3. Dimensionality: Deep SVDD stands out for high-dimensional data.
4. Anomaly Percentages: GNN and Deep SVDD demonstrate more stable performance across varying anomaly percentages.

We conclude that the choice of method should ultimately depend on the specific characteristics of the data and the requirements of the application.

Future Work

Further optimization and fine-tuning:

- Explore hyperparameter optimization techniques to improve the performance of the models.
- Experiment with different architectures, learning rates, and regularization techniques to enhance the generalization capabilities of the models.

References

[1] K. Choi, J. Yi, C. Park and S. Yoon, ”Deep Learning for Anomaly Detection in Time- Series Data: Review, Analysis, and Guidelines,” in IEEE Access, vol. 9, pp. 120043120065, 2021, doi: 10.1109/ACCESS.2021.3107975. Keywords: Anomaly detection; Time series analysis; Guidelines; Deep learning; Data models; Biological system modeling; Time factors; Fault diagnosis; Industry applications; Internet-of-Things (IoT); Time series analysis.

[2] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, ”Deep autoencoding gaussian mixture model for unsupervised anomaly detection,” in International Conference on Learning Representations, Feb. 2018.

[3] Z. Z. Darban, G. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, ”Deep Learning for Time Series Anomaly Detection: A Survey,” arXiv preprint arXiv:2211.05244, 2022.

[4] T. Wen and R. Keyes, ”Time series anomaly detection using convolutional neural networks and transfer learning,” arXiv preprint arXiv:1905.13628, 2019.iv CPT-S 434 Neural Networks Design and Application

[5] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. 2017. WADI: a water distribution testbed for research in the design of secure cyber physical systems.

[6] Zeyan Li, Wenxiao Chen, and Dan Pei. 2018. Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder.

[7] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding.