LSTM Autoencoder Based Anomaly Detection Task for Temporal Climate Data

Fromsa Teshome Negasa[1], **Abdullah Mahmood**[2] **and Nadeer Hasan**[3]

[1, 2, 3] *Erasmus Mundus Joint Masters in Photonics for Security, Reliability, and Safety*

Delphine Maugars

Abstract—This report investigates the detection of temperature anomalies using machine learning techniques. An LSTM autoencoder model is employed to analyze historical temperature data from the Global Historical Climatology Network-Monthly (GHCN-M) dataset. The model is trained to identify anomalies in both global monthly mean temperatures and monthly temperatures for a specific location. The results demonstrate the effectiveness of the LSTM autoencoder in detecting anomalies, showing potential shifts in global and local climate patterns. The analysis provides valuable insights into the dynamics of climate change and can inform further research and decision-making related to climate change mitigation and adaptation strategies.

Keywords—*LSTM Autoencoder, temporal climate data,*

Contents

1. Introduction

Understanding the Earth's climate and its changes is crucial for addressing environmental challenges and ensuring a sustainable future. Analyzing these anomalies provides insight into long-term temperature trends and potential shifts in climate patterns.

In this report, we present an analysis of global temperature anomalies using machine learning. The analysis focuses on identifying anomalies in global monthly mean temperatures and explores temperature variations in specific locations. The goal is to provide a comprehensive overview of temperature anomaly detection using Long-Short Term Memory (LSTM) Auto-encoders.

2. Data Preprocessing

2.1. Overview of the Dataset

The analysis utilizes the Global Historical Climatology Network-Monthly (GHCN-M) dataset, a comprehensive collection of climate summaries from land surface stations around the world. This dataset provides monthly gridded temperature anomalies, representing deviations from a long-term average, from January 1880 to 2016 [1].

The data is organized by `year`, `month`, `latitude`, and `longitude`. Each gridded value represents the temperature anomaly in degrees Celsius, multiplied by 100 and stored as an integer. Missing values are indicated by `-9999`. The dataset covers the entire world with a 5 ° by 5 ° grid, resulting in 2,592 gridded data points per month.

The GHCN-M dataset is carefully curated to remove biases and ensure the representation of true climate variability. This makes it a reliable source for studying global temperature changes and identifying potential anomalies.

2.2. Handling Missing Values in the Dataset

The dataset contained a significant number of missing values, represented by `-9999`. In total, there were `1,400,604` missing values, accounting for 31.53% of the dataset. To address this, we experimented with several imputation methods to replace the missing values. However, the results consistently demonstrated that the removal of missing values led to better outcomes. Below are the methods we attempted for handling missing values:

1. **Monthly Mean Imputation:** Missing values for a given month were replaced with the mean of non missing values from the same month across all grid cells. This approach aims to preserve the seasonal patterns in the data, but can dilute anomalies and result in biased averages when missing values are clustered.
2. **Median Imputation:** The missing values for each month were replaced with the median of non-missing values in the same month.
3. **Mode (Most Frequent Value) Imputation:** The missing values were replaced with the most frequently occurring value in the corresponding month. This method resulted in significant distortion of the data distribution, especially for months with highly variable temperatures.
4. **K-Nearest Neighbors (KNN) Imputation:** The missing values were filled based on the weighted average of the nearest neighbors in the feature space. Although KNN imputation is often effective in dataset's with small gaps, its performance degraded with the high percentage of missing values in this dataset and increased computational complexity.

Ultimately, the decision to remove all missing values (`-9999`) proved to be more effective. This step reduced the dataset size but maintained its quality and reliability for our task.

2.3. Data Cleaning and Feature Engineering

Before applying machine learning models, the raw data underwent several pre-processing steps:

1. **Loading the data:** The dataset was loaded into a pandas DataFrame for efficient manipulation.
2. **Data Cleaning:** Invalid temperature values (`-9999`) were filtered out.
3. **Feature Engineering:** The latitude and longitude information was processed to create numerical representations.
   - Latitude and longitude signs were extracted to indicate north/south and east/west directions.
   - `lat` and `lon` values were converted to numerical formats by calculating the midpoint of grid cells.

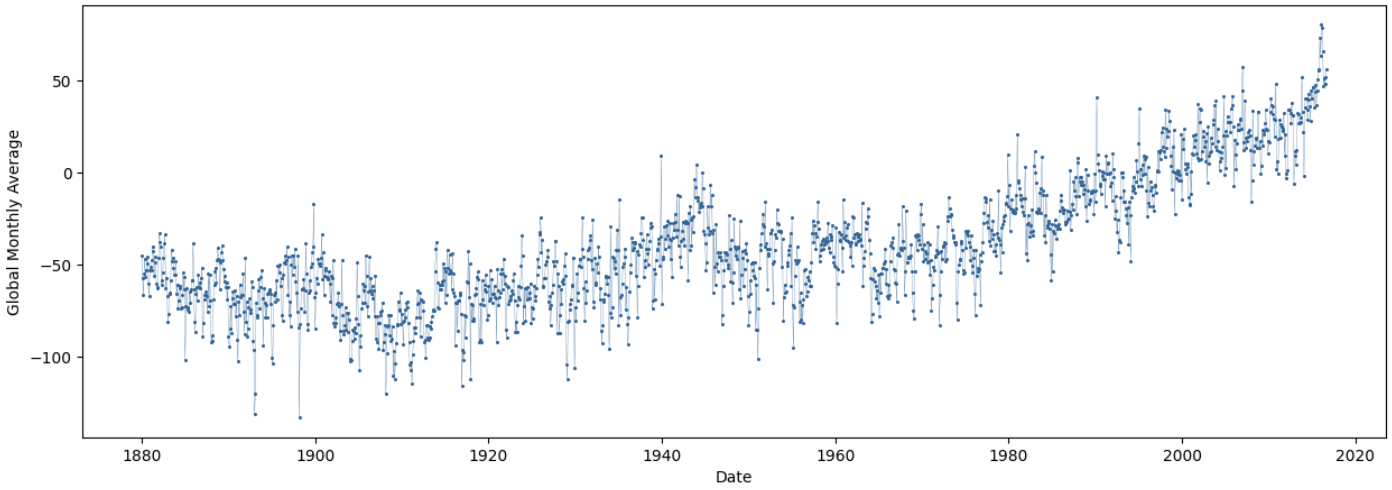


Figure 1. Global Monthly Average Temperature Over Time [1].

- A time column was added to represent the year and month.
- A `date` column was created by combining year and month into a datetime object.

These pre-processing steps transformed the raw data into a structured format, with 2863452 entries, 72 unique longitude values, and 38 unique latitude values. The head of the processed table is shown in Table 1.

Table 1. Sample data from the Global Historical Climatology Network

date	lat	lon	temp
1880-01-01	27.5	-177.5	-7
1880-01-01	22.5	-177.5	-44
1880-01-01	17.5	-177.5	-48
1880-01-01	-12.5	-177.5	-15
1880-01-01	-17.5	-177.5	44
⋮	⋮	⋮	⋮

3. Global Monthly Mean Temperature Anomalies

Here, we analyze global monthly mean temperature anomalies using a machine learning model to identify potential anomalies and trends.

3.1. Data Preparation

To prepare the data for this task, the following steps were performed:

Table 2. Sample data from the Global Historical Climatology Network

date	global_monthly_mean_temp
1880-01-01	-44.930750
1880-02-01	-56.910184
1880-03-01	-66.274477
1880-04-01	-54.772563
1880-05-01	-47.154472
⋮	⋮

1. **Calculating Global Monthly Means:** The average temperature for each month was calculated across all locations to obtain a `global monthly mean temperature` time series (Figure 1).

Table 3. Summary statistics of the global monthly mean temperature

stat	date	global_monthly_mean_temp
count	1641	1641.000000
mean	1948-05-01	-39.917914
min	1880-01-01	-132.803481
25%	1914-03-01	-65.026911
50%	1948-05-01	-45.682968
75%	1982-07-01	-20.202681
max	2016-09-01	80.182281
std	NaN	34.538044

2. **Data Splitting:** A train size of 80% was used to train and the remaining for testing.
3. **Data Scaling:** A `MinMaxScaler` was used to normalize the temperature values between 0 and 1. This step ensures that features with larger ranges do not dominate the model's learning process.
4. **Sequence Creation:** The data was transformed into sequences of a fixed length (12 months in this case) to capture temporal dependencies. Each sequence represents a 12-month window of temperature data.

3.2. Model Definition and Training

An *LSTM autoencoder* model was employed for anomaly detection. This type of model consists of an encoder that compresses the input sequence into a lower-dimensional representation and a decoder that reconstructs the original sequence from the compressed representation [6] [4]. The model is trained to minimize the difference between the input and reconstructed sequences. Anomalies are detected by identifying instances where the reconstruction error is significantly high, indicating that the model struggles to accurately reproduce the observed temperature patterns.

The LSTM autoencoder model used in this analysis comprises the following layers:

1. **Input Layer:** Accepts the sequence of temperature data.
2. **LSTM Layer (Encoder):** Encodes the input sequence into a hidden representation.
3. **Dropout Layer:** Prevents overfitting by randomly dropping out neurons during training.
4. **RepeatVector Layer:** Repeats the encoded representation to match the input sequence length.
5. **LSTM Layer (Decoder):** Decodes the repeated representation back into a sequence.
6. **Dropout Layer:** Prevents overfitting.

7. **TimeDistributed Dense Layer:** Outputs the reconstructed temperature sequence.

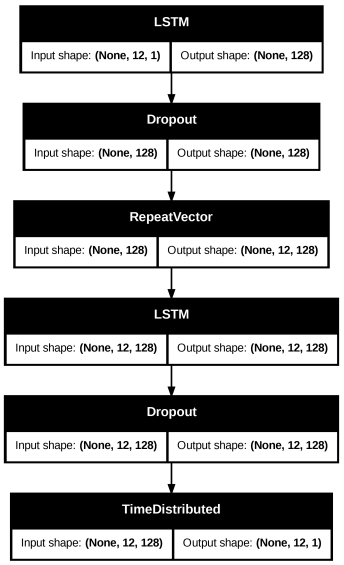


Figure 2. LSTM Autoencoder model.

The model was trained using the `Adam optimizer` and the mean absolute error (`MAE`) loss function. Early stopping was implemented to prevent over-fitting by monitoring the validation loss over 20 epochs and stopping training when it plateaued.

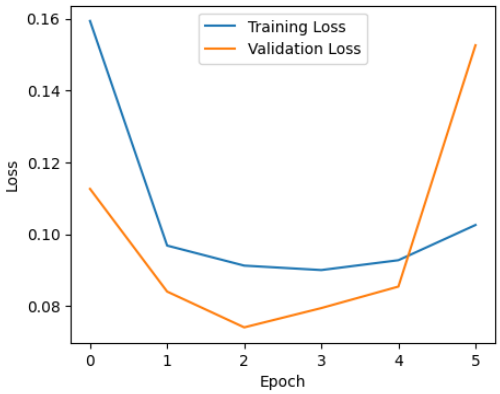


Figure 3. Training and Validation Loss for the LSTM Autoencoder Model

Figure 3 illustrates the training and validation loss curves during the model training process. The decreasing trend in both curves indicates that the model is learning to effectively reconstruct the temperature sequences. The early stopping mechanism ensures that the model avoids overfitting.

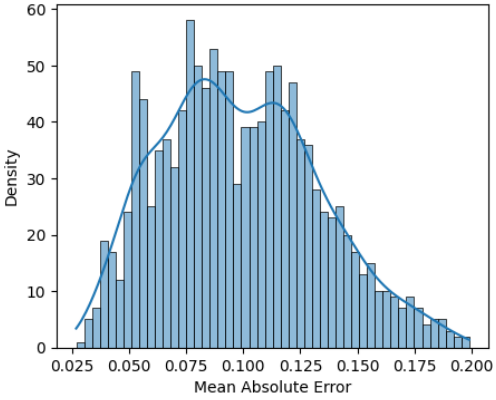
3.3. Anomaly Detection and Visualization

Anomalies were detected by comparing the reconstruction error of each data point to a predefined threshold. This threshold was determined by analyzing the distribution of reconstruction errors on the training data and setting it to a value that captures a desired level of sensitivity to anomalies.

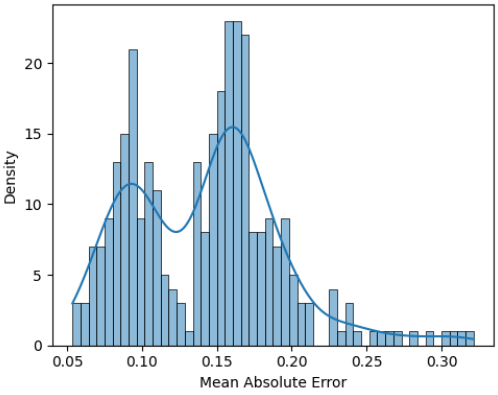
In this case, the threshold was set to 95% of the maximum reconstruction error (`0.1944`) observed on the training data to ensure that only the most significant deviations were flagged as anomalies.

Figure 4 shows the distribution of the Mean Absolute Error (MAE) for both the train (Figure 4a) and test (Figure 4b) sets.

Based on the train MAE distribution, the threshold was then set to be `0.1848`, as depicted in Figure 6. This figure illustrates the test MAE



(a) Distribution of Train MAE.



(b) Distribution of Test MAE

Figure 4. Distribution of Train and Test MAE

loss over time, alongside the established threshold. Points where the test MAE loss surpassed the threshold are identified as anomalies.

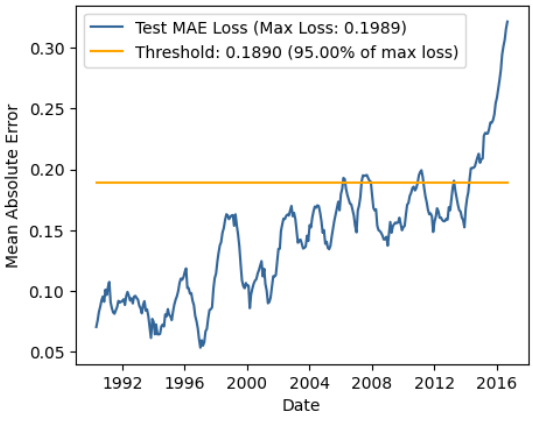


Figure 6. Test MAE over time for global mean temperature anomalies.

The identified anomalies are depicted in Figure 5 which is shown over the test set of global monthly mean temperature's.

4. Monthly Temperature Anomalies Based on Location

We extended the analysis to specific locations to investigate temperature anomalies at a finer geographical scale because the global mean temperature retrieval was not adequate to our analysis. So we took the same LSTM autoencoder model architecture and training procedure used in the previous section are applied here, with the key

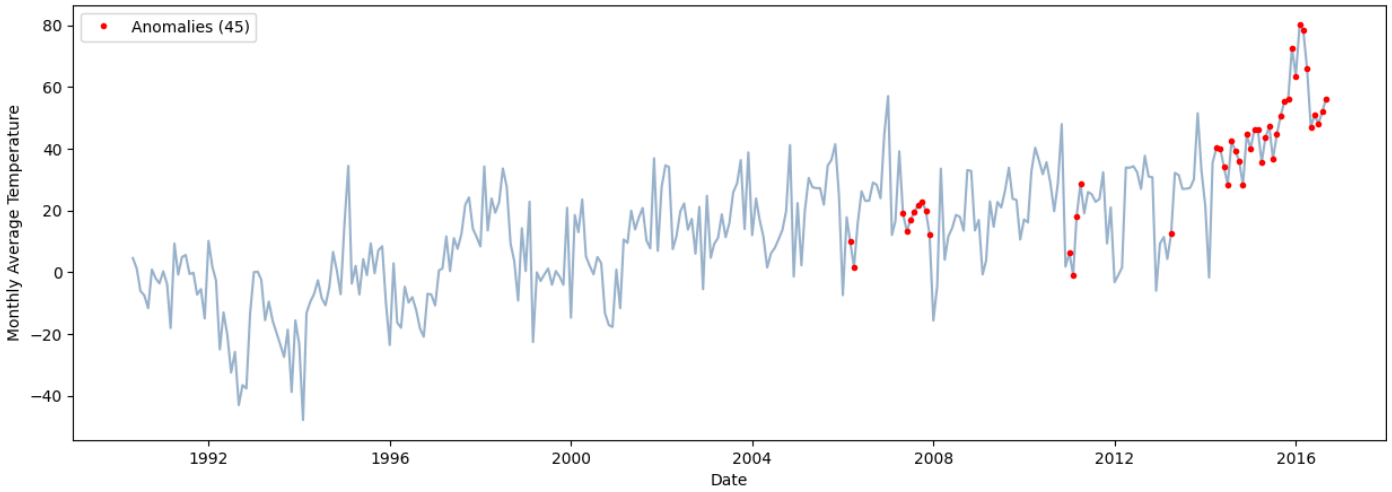


Figure 5. Detected Temperature Anomalies in Global Monthly Mean Temperatures.

difference being the input data. Instead of using global monthly mean temperatures, we focus on monthly temperature data for individual locations.

4.1. Getting Location Data

To analyze temperature anomalies for a specific location, we need to extract the relevant data from the original dataset. This is achieved by selecting data points that correspond to the desired latitude and longitude coordinates.

```
def get_loc_data(data, tgt_lat, tgt_lon):

    # Calculate distance to target coordinates
    data['dist'] = np.sqrt((data['lat'] -
    tgt_lat) ** 2 + (data['lon'] - tgt_lon) **
    2)

    # Find the closest match
    closest_idx = data['dist'].idxmin()
    closest_lat = data.loc[closest_idx, 'lat']
    closest_lon = data.loc[closest_idx, 'lon']

    # Filter data for the closest match
    loc_data = data[(data['lat'] == closest_lat)
     & (data['lon'] == closest_lon)]

    if loc_data.empty:
        print("No matching location found.")
        return None
    else:
        print(f"Closest coordinates: ({
    closest_lat}, {closest_lon})")
        return loc_data[['date', 'temp']], (
    closest_lat, closest_lon)
```

Code 1. Location data getter function

For example, the data for Paris is obtained by identifying the closest match to its latitude and longitude coordinates within the dataset. We specify the target coordinates of Paris as `(48.8566, 2.3522)` for latitude and longitude, respectively.

The function `get_location_data()` (Code 1) is used to find the closest match in the dataset. Calculate the distance between the target coordinates and all the coordinates in the data set and select the data point with the minimum distance. For Paris, the closest match found is at latitude 47.5 and longitude 2.5. This ensures that we are analyzing the temperature data for the location closest to Paris within the dataset's grid, even if there is not an exact match.

Table 4. Actual and Closest Match Coordinates for Paris

Location	Latitude	Longitude
Paris (Actual)	48.8566	2.3522
Paris (Closest Match)	47.5	2.5

4.2. Anomaly Detection and Visualization for Different Locations

Using the location-specific data, we follow the same anomaly detection steps as described in the previous section. The trained LSTM autoencoder model is applied to the location specific temperature data, and anomalies are identified by comparing the reconstruction error of each data point to the predefined relative threshold.

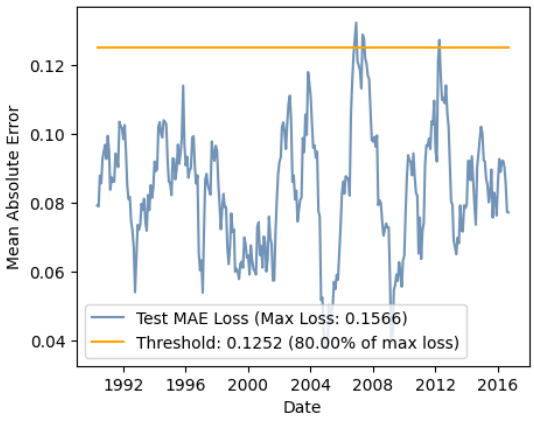
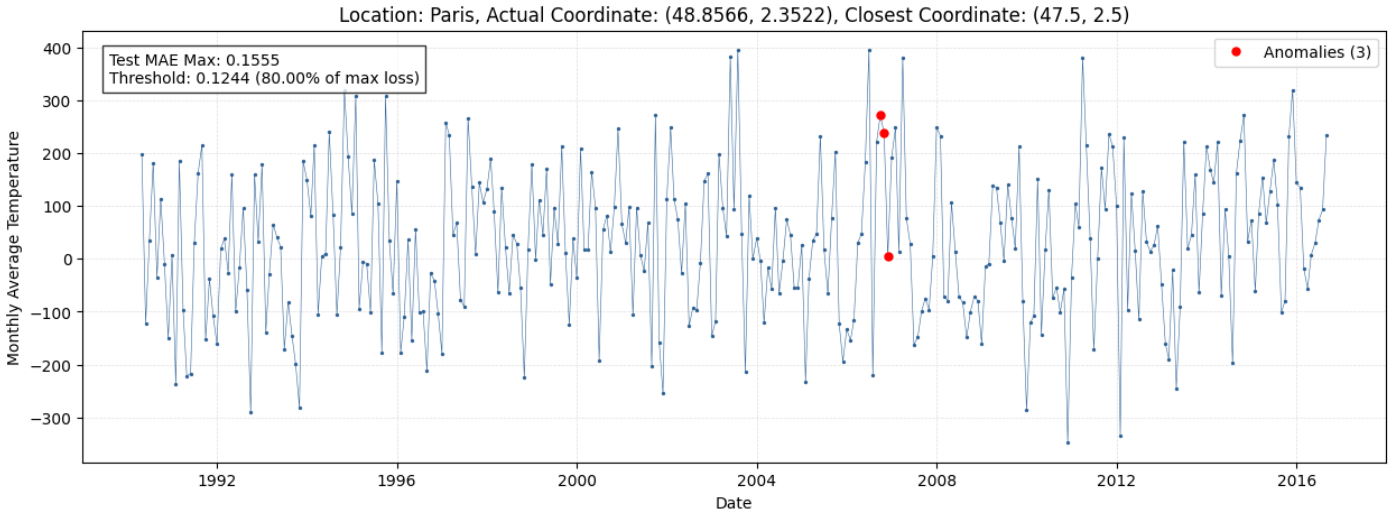


Figure 7. Test MAE over time for Paris monthly temperature anomalies.

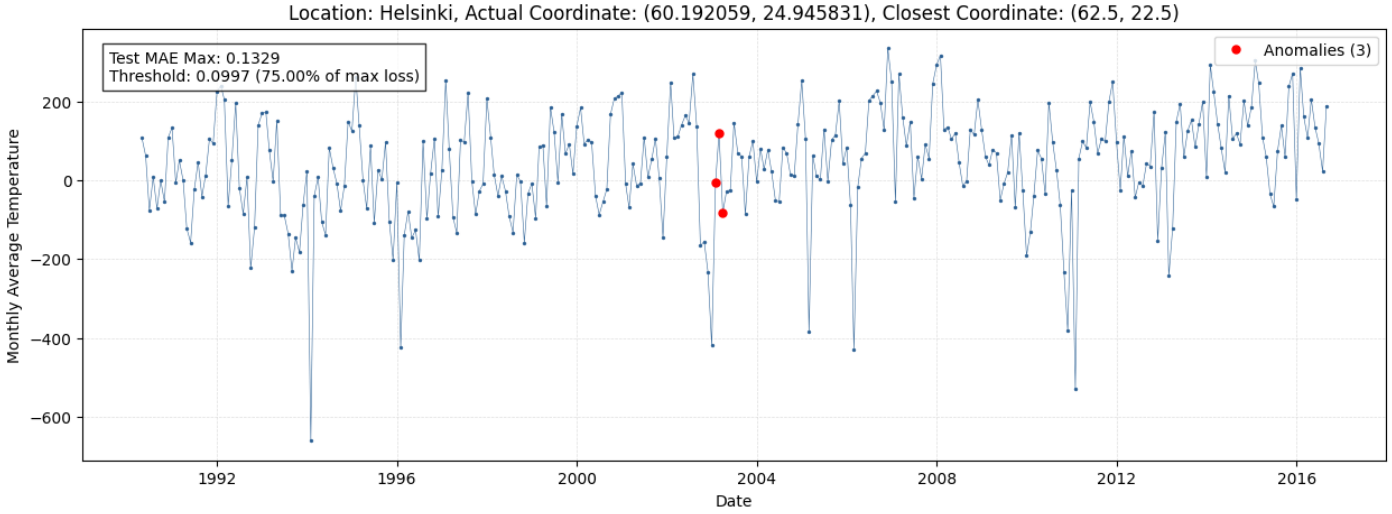
Similar to the global anomaly detection, the threshold is determined based on the maximum reconstruction error observed during the training phase. In the case of Paris, the maximum training MAE was found to be approximately `0.1566`. We found it appropriate to set it 80% of the maximum training error, resulting in a threshold of `0.1252`. Any data point with a reconstruction error exceeding this threshold is flagged as an anomaly, as shown in Figure 7.

Figure 8 displays the detected temperature anomalies in the monthly temperature time series for various locations. The red dots indicate the identified anomalies.

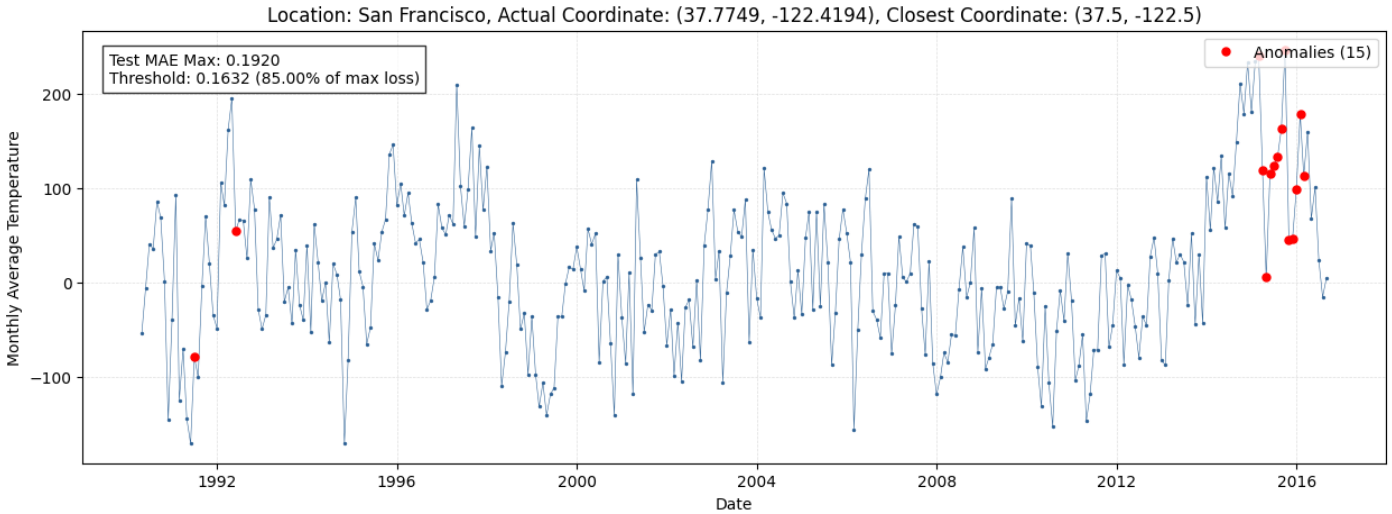
By applying the LSTM autoencoder model to location-specific data and carefully selecting the anomaly detection threshold, we can gain insights into temperature anomalies at a finer geographical scale.



(a) Detected Temperature Anomalies in Paris Monthly Temperatures.



(b) Detected Temperature Anomalies in Helsinki Monthly Temperatures.



(c) Detected Temperature Anomalies in San Francisco Monthly Temperatures.

Figure 8. Detected Temperature Anomalies in Monthly Temperatures for (a) Paris, (b) Helsinki, and (c) San Francisco.

This approach is better for a more comprehensive understanding of temperature variation on specific regions.

5. Other Models Used

In addition to the primary LSTM autoencoder model presented earlier, other network architectures were explored: an LSTM autoencoder with increased depth and a Bidirectional LSTM autoencoder.

5.1. LSTM-Autoencoder: 2-layer Depth

The LSTM autoencoder with 2-layer depth aims to enhance the model's capacity to learn complex patterns by increasing the number of LSTM layers in both the encoder and decoder. This architecture, illustrated in Figure 9, involves two LSTM layers in the encoder followed by two LSTM layers in the decoder, potentially allowing for a more nuanced representation of temporal dependencies [2].

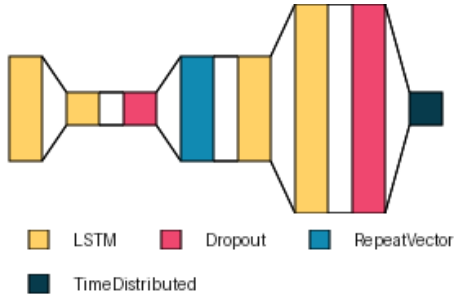


Figure 9. Architecture of the LSTM Autoencoder with 2-layer Depth

However, with the relatively limited size of the dataset used in this analysis, increasing the model's depth introduced the risk of overfitting. Overfitting occurs when a model learns the training data too well and fails to generalize to unseen data. This can lead to poor performance on the testing set and reduced accuracy in anomaly detection.

5.2. Bidirectional LSTM-Autoencoder

The Bidirectional LSTM autoencoder incorporates Bidirectional LSTM layers, which process the input sequence in both forward and backward directions [5]. This approach, depicted in Figure 10, aims to capture temporal dependencies more comprehensively by considering information from both past and future time steps.

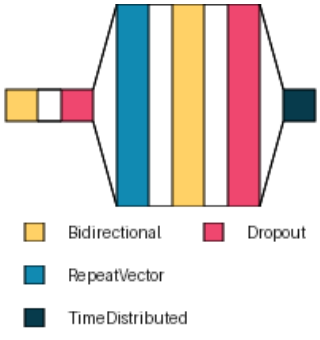


Figure 10. Architecture of the Bidirectional LSTM Autoencoder

While Bidirectional LSTMs can be powerful for sequence modeling, they also increase the model's complexity and computational requirements. In this particular case, the added complexity did not translate into significant improvements in anomaly detection performance.

Furthermore, both the deeper LSTM autoencoder and the Bidirectional LSTM autoencoder were found to be more prone to overfitting compared to the simpler LSTM autoencoder model due to their increased capacity to learn complex patterns, which can lead to memorization of the training data rather than generalization.

5.3. Hierarchical LSTM Autoencoder

The Hierarchical LSTM Model was designed to capture patterns in the data at both short-term (monthly) and long-term (yearly) levels. To achieve this, the model was constructed with two branches: one branch processes monthly data sequences using an LSTM autoencoder, while the other processes yearly data sequences using a similar LSTM structure [3].

These two branches were then merged, and additional dense layers were added to combine the information and generate the final predictions (Figure 11 shows the model architecture). This hierarchical structure allows the model to analyze both fine-grained and overarching trends in the dataset, offering a more comprehensive understanding of temporal dependencies.

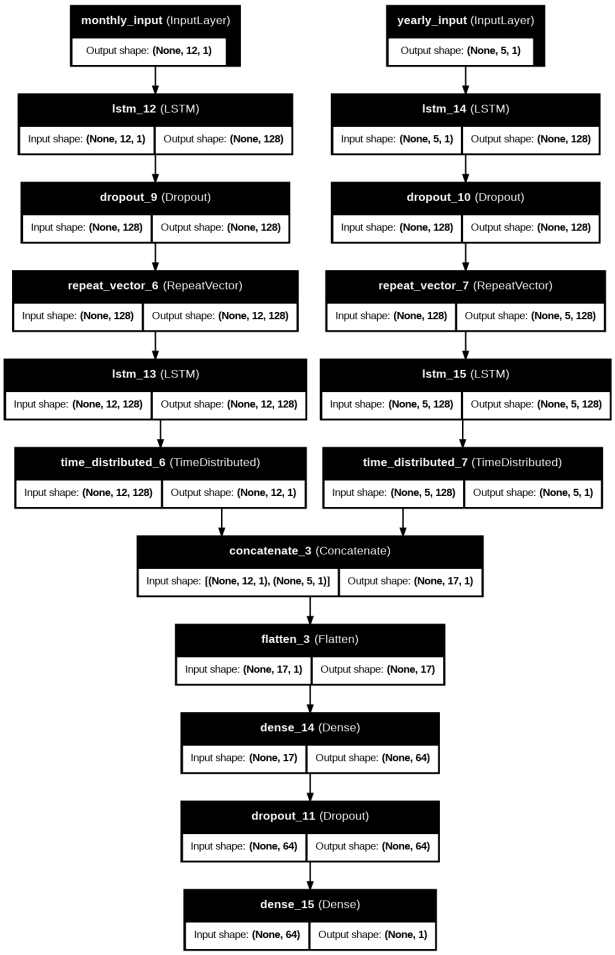


Figure 11. Architecture of the Hierarchical LSTM Autoencoder

The resulting loss is high and the Test MAE (Mean Absolute Error) was "11.11". Figure 12 shows that the gap between the training and validation loss curves is decreasing gradually and starting from the epoch 55 the gap is somehow fixed.

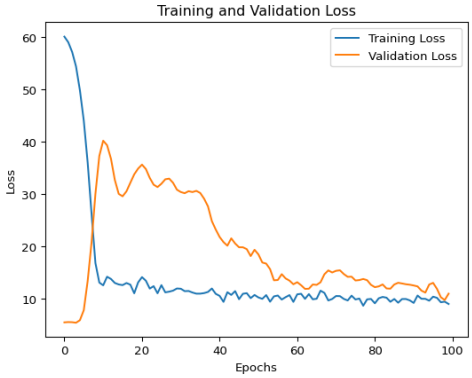


Figure 12. Training and Validation Loss oh hierarchical LSTM Autoencoder

Therefore, while exploring alternative architectures is valuable, the simpler LSTM autoencoder model was deemed more suitable for this specific temperature anomaly detection task due to its balance between performance and complexity, and its reduced susceptibility to overfitting on the given dataset.

6. Conclusion

In summary, we presented an analysis of global and location-specific temperature anomalies using an LSTM autoencoder model. The model was trained on historical temperature data from the Global Historical Climatology Network-Monthly (GHCN-M) dataset and applied to detect anomalies in both global monthly mean temperatures and monthly temperatures for Paris as a specific location.

The results demonstrated that LSTM autoencoder model are capable in identifying temperature anomalies. The analysis revealed a number of significant anomalies in both the global and location-specific temperature time series, suggesting potential shifts in climate patterns and local climate variations. By identifying temperature anomalies, we can gain a deeper understanding of the factors driving these fluctuations and their potential impacts on different regions.

Overall, this report highlights the potential of machine learning in analyzing climate data and identifying temperature anomalies, contributing to a better understanding of climate change and its implications.

7. Appendix

A. Code Repository

The full source code for this assignment is available on GitHub at https://github.com/PHYRA47/Computer-Vision/tree/main/project3-machine-learning-lstm-autoencoder-for-time-series-data.

References

[1] J. H. Lawrimore, M. J. Menne, B. E. Gleason, *et al.*, *Global historical climatology network - monthly (ghcn-m), version 3*, https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00839, [indicate subset used]. NOAA National Centers for Environmental Information. doi:10.7289/V5X34VDR [access date], 2011.

[2] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems", *Scientific Reports*, vol. 9, no. 1, 2019. DOI: 10.1038/s41598-019-55320-6. [Online]. Available: https://doi.org/10.1038/s41598-019-55320-6.

[3] P. Agarwal, J. Ivan, A. Elkamel, and H. Budman, "Hierarchical deep lstm for fault detection and diagnosis for a chemical process", *Processes*, vol. 10, no. 12, pp. 2557–2557, 2022. DOI: 10.3390/pr10122557. [Online]. Available: https://doi.org/10.3390/pr10122557.

[4] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, *Lstm-autoencoder based anomaly detection for indoor air quality time series data*, https://arxiv.org/abs/2204.06701, Accessed: 2024-12-11, 2022.

[5] K. Berahmand, F. Daneshfar, E. S. Salehi, Y. Li, and Y. Xu, "Autoencoders and their applications in machine learning: A survey", *Artificial Intelligence Review*, vol. 57, no. 2, 2024. DOI: 10.1007/s10462-023-10662-6. [Online]. Available: https://doi.org/10.1007/s10462-023-10662-6.

[6] Curiousily, *Deep-learning-for-hackers: Time series anomaly detection*, https://github.com/curiousily/Deep-Learning-For-Hackers/blob/master/14.time-series-anomaly-detection.ipynb, Accessed: 2024-12-11, 2024.