Global Next Consulting India Private Limited
(GNCIPL)

All India Professional Industrial Internship Program (AI-PIIP-2025)

PROJECT REPORT

Domain: Artificial Intelligence - Machine Learning

PROJECT TITLE:
Generate Synthetic Time-Series Data for Anomaly Detection (IoT / Industry) using TimeGAN

Project Team:
Devasish Sai Pothumudi
Narasingu Sai Suchendar
Saumya Singh
Kajal Tiwari
PranayNigam
Sangeeth M
Sharon jacquiline S

Date: October, 2025

PREFACE

The advent of Industry 4.0 has ushered in an era of unprecedented connectivity and data generation within industrial environments. The Industrial Internet of Things (IIOT) connects machinery, sensors, and control systems, producing vast streams of time-series data. This data holds the key to unlocking immense value, particularly in the realm of predictive maintenance. By analysing sensor data in real-time, we can anticipate machine failures before they occur, minimizing downtime, reducing operational costs, and enhancing workplace safety.

However, a significant challenge in developing robust predictive maintenance models is the scarcity of failure data. Machines are designed to operate correctly, making "normal" operational data abundant while "abnormal" or failure data is rare. This class imbalance makes it difficult to train supervised machine learning models effectively.

This project addresses this critical challenge by exploring a novel two-stage approach. First, we leverage a state-of-the-art generative model, Time-series Generative Adversarial Network (Time GAN), to learn the complex temporal dynamics of normal machine behaviour from existing sensor data. This allows us to generate a virtually limitless supply of high-fidelity synthetic time-series data that mirrors normal operational states.

Second, we use this rich synthetic dataset to train an unsupervised anomaly detection model—an LSTM Autoencoder. By training exclusively on normal data, the model becomes adept at reconstructing expected behaviour. When presented with real-world data, any significant deviation from this learned normality is flagged as a potential anomaly, signalling an impending failure.

This report documents the entire project lifecycle, from data exploration and preprocessing to the implementation and evaluation of the Time GAN and LSTM Autoencoder models. It aims to provide a comprehensive overview of the methodology, results, and insights gained from applying advanced deep learning techniques to solve a practical and pressing industrial problem.

Table of Contents

Chapter 1: Introduction

1.1. Background: IIOT and Predictive Maintenance

The Industrial Internet of Things (IIOT) refers to the network of interconnected sensors, instruments, and other devices with industrial applications. This ecosystem generates a continuous flow of data related to pressure, temperature, vibration, voltage, and other key performance indicators of machinery. Predictive Maintenance (PDM) is a proactive strategy that leverages this data to predict equipment failures. Unlike traditional reactive (run-to-failure) or preventative (time-based) maintenance, PDM allows for maintenance to be scheduled precisely when needed, optimizing resource allocation and preventing catastrophic failures.

The foundation of a successful PDM system is a model that can accurately distinguish between normal and anomalous operational patterns. The development of such models is a primary focus of applied machine learning in the industrial sector.

1.2. The Challenge of Anomaly Detection

An anomaly in the context of IIOT is a data pattern that deviates significantly from the established norm. These deviations can be early indicators of equipment malfunction, component degradation, or a potential system failure. However, building effective anomaly detection systems faces several hurdles:

- **Data Imbalance:** As machines are designed to operate reliably, data corresponding to failure states is extremely rare compared to data from normal operations. This makes it challenging for supervised learning algorithms to learn the characteristics of failures.
- **Complexity of Temporal Patterns:** Time-series data from sensors exhibits complex temporal dependencies. Simple statistical methods often fail to capture these long-term patterns, leading to high false positive rates.
- **Evolving System Dynamics:** The behaviour of a machine can change over time due to wear and tear or changing environmental conditions. An anomaly detection system must be robust enough to adapt to these changes.

1.3. Problem Statement

Given a dataset of multivariate time-series data from sensors on an industrial machine, the primary problem is to develop a robust model capable of detecting anomalous sequences that signify potential failures. The core constraint is the lack of labelled anomaly data for training a supervised model. Therefore, the solution must be capable of learning the characteristics of "normal" behaviour from an unlabelled dataset and using this knowledge to identify deviations.

1.4. Proposed Solution and Objectives

To overcome the challenge of data scarcity, we propose a two-stage deep learning pipeline:

1. **Synthetic Data Generation:** Utilize a Time-series Generative Adversarial Network (Time GAN) to learn the underlying distribution of normal operational data. The trained generator will then be used to create a large, high-fidelity synthetic dataset representing a wide range of normal machine behaviours.
2. **Unsupervised Anomaly Detection:** Train an LSTM Autoencoder exclusively on the generated synthetic "normal" data. This model will learn to reconstruct normal time-series sequences with high accuracy. When fed a real sequence, a high reconstruction error will serve as an anomaly score, indicating a deviation from normal behaviour.

The main objectives of this project are:

- To successfully implement and train a Time GAN model on the provided IIOT dataset.
- To generate realistic synthetic time-series data that preserves the temporal dynamics of the original data.
- To build and train an LSTM Autoencoder for anomaly detection using the synthetic data.
- To establish a threshold for reconstruction error to effectively distinguish between normal and anomalous machine states.

1.5. Report Structure

This report is organized into six chapters. Chapter 2 details the dataset and the exploratory analysis performed. Chapter 3 and Chapter 4 explain the core methodologies—Time GAN for data generation and the LSTM Autoencoder for anomaly detection, respectively. Chapter 5 presents the results and discusses their implications. Finally, Chapter 6 concludes the report with a summary of findings and suggestions for future work.

Chapter 2: Dataset and Exploratory Data Analysis

2.1. Dataset Overview

The dataset used in this project is an Industrial Internet of Things (IIoT) Edge Computing dataset. It contains multivariate time-series data collected from various sensors attached to a piece of industrial equipment. Each data point is a snapshot of the machine's state at a specific timestamp, captured by sensors measuring variables such as:

- Temperature
- Pressure
- Vibration
- Voltage

- Rotational Speed

The data represents continuous operation, with sequences corresponding to normal functioning and a few instances of anomalous behaviour leading to failures. The primary goal of the analysis is to learn the signature of the "normal" data.

2.2. Data Loading and Preprocessing

The first step in our workflow involves loading the dataset and preparing it for the deep learning models. The preprocessing pipeline consists of the following key steps:

1. **Data Cleaning:** The dataset is checked for any missing or null values. Depending on the extent of missing data, strategies such as imputation (e.g., forward-fill, mean-fill) or removal of affected rows/columns would be applied. For this project, the dataset was found to be clean.
2. **Data Scaling:** Deep learning models, particularly those using gradient-based optimization, are sensitive to the scale of input features. Features with larger ranges can dominate the learning process. To address this, we use Min Max Scaler from the Scikit-learn library. This scaler transforms each feature to a given range, typically [0, 1]. This ensures that all sensor readings contribute equally to the model training process.
3. **Sequence Creation:** Time-series models do not process data point by point; they operate on sequences. The continuous data stream is segmented into overlapping sequences of a fixed length. For instance, a sequence length of 24 would mean that each input sample for our model consists of 24 consecutive timestamps of all sensor readings. This process captures the short-term temporal dependencies essential for our models.

Anoamly Detection Performance
AUC: 0.95
0.2.82
A) Training Augumeitation
Baseline Detector (Train on Real Only)
Real Data
Synthetic Data (from Time Only)
B) TSTR Evaluation
Synthetic Sata
Live Real-time Data
Fewer False Alarms
Failure Detected EARLIER
Late Detection (Baseline)

2.3. Exploratory Data Analysis (EDA)

Before model development, an exploratory data analysis was conducted to understand the structure and characteristics of the data. This involved visualizing the sensor readings over time.

The plots revealed the cyclical and dynamic nature of the machine's operation. We observed distinct patterns and correlations between different sensors. For example, an increase in rotational speed might correlate with a rise in temperature and vibration. These inherent temporal relationships are precisely what our Time GAN model aims to learn and replicate. The EDA phase confirmed the suitability of the dataset for time-series modelling and provided a baseline understanding of what constitutes "normal" behaviour.

Chapter 3: Methodology: Synthetic Data Generation

3.1. Introduction to Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a class of deep learning models introduced by Ian Goodfellow et al. in 2014. A GAN consists of two neural networks, a **Generator** and a **Discriminator**, which are trained simultaneously in a zero-sum game.

- **Generator (G):** Its goal is to create synthetic data that is indistinguishable from real data. It takes a random noise vector as input and outputs a data sample.
- **Discriminator (D):** Its goal is to act as a binary classifier, determining whether a given data sample is real (from the training set) or fake (from the Generator).

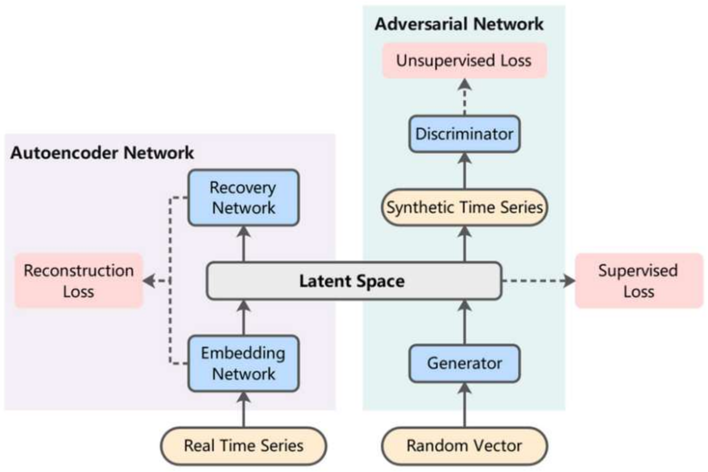
The training process involves the Generator trying to fool the Discriminator, while the Discriminator gets better at catching the fakes. This adversarial process drives the Generator to produce increasingly realistic data.

3.2. Time GAN: A Specialized GAN for Time-Series

While standard GANs excel at generating realistic images, they are not directly applicable to time-series data. This is because they do not explicitly learn the conditional dynamics within a sequence. Time GAN (Time-series Generative Adversarial Network) was specifically designed to address this limitation. It incorporates the unique characteristics of time-series data by combining the adversarial learning framework of GANs with an autoencoder architecture. This hybrid approach enables it to capture both the static (feature-wise) and dynamic (temporal) properties of the data.

3.3. Time GAN Architecture

Time GAN is composed of four main components: an **Embedder**, a **Recovery** network, a **Generator**, and a **Discriminator**. These components work together through a carefully designed set of loss functions.



3.3.1. Autoencoder Component (Embedder & Recovery)

The first part of Time GAN is an autoencoder that works in the temporal domain.

- **Embedder:** This network reduces the dimensionality of the time-series data at each point in time, mapping the original feature space into a lower-dimensional latent space. This step helps the model focus on the most important temporal features.
- **Recovery:** This network takes the embedded latent space representation and attempts to reconstruct the original time-series data.

The autoencoder is trained to minimize the reconstruction loss between the original and recovered sequences. This forces the model to learn a meaningful and compressed representation of the temporal data.

3.3.2. Adversarial Component (Generator & Discriminator)

The second part is the adversarial network that operates on the embedded sequences.

- **Generator:** This network takes random temporal noise as input and generates synthetic sequences in the latent space. Using latent space representations makes the learning task more stable and efficient.
- **Discriminator:** This network receives both real sequences (embedded by the Embedder) and synthetic sequences (created by the Generator) and tries to distinguish between them.

3.3.3. Joint Training and Loss Functions

The brilliance of Time GAN lies in its joint training process, which optimizes three types of losses simultaneously:

1. **Reconstruction Loss (Unsupervised):** This is the standard autoencoder loss, measured between the original data and the output of the Recovery network. It ensures the latent space captures the key features of the data.
2. **Supervised Loss:** To encourage the Generator to preserve the step-wise temporal dynamics of the real data, a "supervised" loss is introduced. It measures the difference between the real data's latent representation and the generated synthetic data. This directly teaches the generator about the transitions between time steps.
3. **Adversarial Loss (Unsupervised):** This is the classic GAN loss where the Generator tries to minimize the Discriminator's ability to classify its outputs as fake, while the Discriminator tries to maximize its classification accuracy. This pushes the Generator to create more realistic sequences in the latent space.

This three-pronged training strategy ensures that the generated data is not only statistically similar to the real data but also preserves the crucial temporal correlations.

3.4. Implementation Details

For our project, the Time GAN components were implemented using Recurrent Neural Networks (RNNs), specifically Gated Recurrent Units (GRUs), due to their effectiveness in capturing temporal dependencies. The model was trained for a substantial number of epochs until the losses converged and the generated samples were visually and statistically similar to the real data from the EDA phase.

Chapter 4: Methodology: Anomaly Detection

4.1. Autoencoders for Anomaly Detection

An autoencoder is a type of unsupervised neural network that is trained to learn a compressed representation (encoding) of its input data and then reconstruct the original data from that representation. It consists of two parts:

- **Encoder:** This part of the network compresses the input data into a low-dimensional latent space vector.
- **Decoder:** This part of the network attempts to reconstruct the original input from the latent space vector.

The core idea for anomaly detection is simple: if the autoencoder is trained only on "normal" data, it will become very good at reconstructing normal data but will struggle to reconstruct anomalous data. Anomalous inputs will result in a high **reconstruction error** (the difference between the original input and the reconstructed output), which can be used as an anomaly score.

4.2. Long Short-Term Memory (LSTM) Networks

Standard feedforward networks are not suitable for time-series data as they do not have memory of past events. Recurrent Neural Networks (RNNs) are designed to handle sequential data, but they suffer from the vanishing gradient problem, which makes it difficult for them to learn long-term dependencies.

Long Short-Term Memory (LSTM) networks are a special kind of RNN that were designed to overcome this limitation. They contain "memory cells" and "gates" (input, output, and forget gates) that regulate the flow of information. This architecture allows them to remember important information over long periods, making them ideal for modelling complex time-series data.

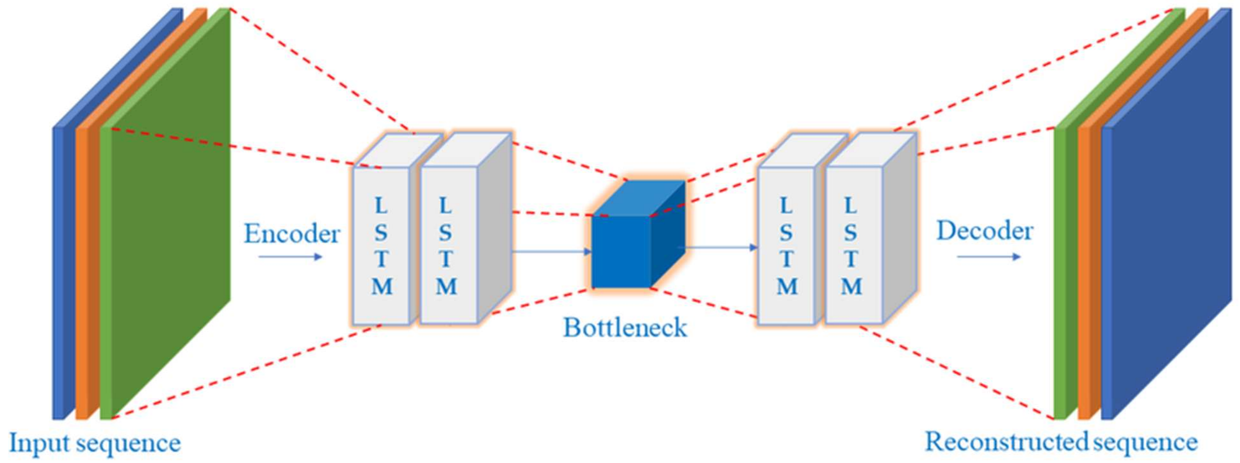
4.3. LSTM Autoencoder Architecture

By combining LSTMs with the autoencoder framework, we can build a model that is perfectly suited for learning and reconstructing normal time-series patterns.

- **Encoder:** The encoder consists of one or more LSTM layers. It processes the input sequence step-by-step and outputs a single fixed-size vector (the latent representation) that summarizes the entire input sequence.

- **Decoder:** The decoder also consists of one or more LSTM layers. It takes the latent vector from the encoder as its initial state and generates the output sequence step-by-step, attempting to reverse the encoding process. A Repeat Vector layer is often used to feed the latent vector to the decoder at each time step.

The model is trained on the large dataset of synthetic "normal" sequences generated by Time GAN. The objective is to minimize the Mean Squared Error (MSE) between the input sequences and the reconstructed sequences.



4.4. Anomaly Detection via Reconstruction Error

Once the LSTM Autoencoder is trained, it can be used for anomaly detection. For each new time-series sequence from the real-world test data, we perform the following steps:

1. Feed the sequence into the trained autoencoder.
2. Obtain the reconstructed sequence from the model's output.
3. Calculate the reconstruction error, typically the Mean Absolute Error (MAE) or Mean Squared Error (MSE) between the original and reconstructed sequences.
4. Compare this error to a predefined **threshold**. If the error is above the threshold, the sequence is flagged as an **anomaly**.

The threshold is a critical hyperparameter. It is usually determined by analysing the distribution of reconstruction errors on a validation set of normal data. A common approach is to set the threshold at the 95th or 99th percentile of these errors, allowing for a small, acceptable rate of false positives.

Chapter 5: Results and Discussion

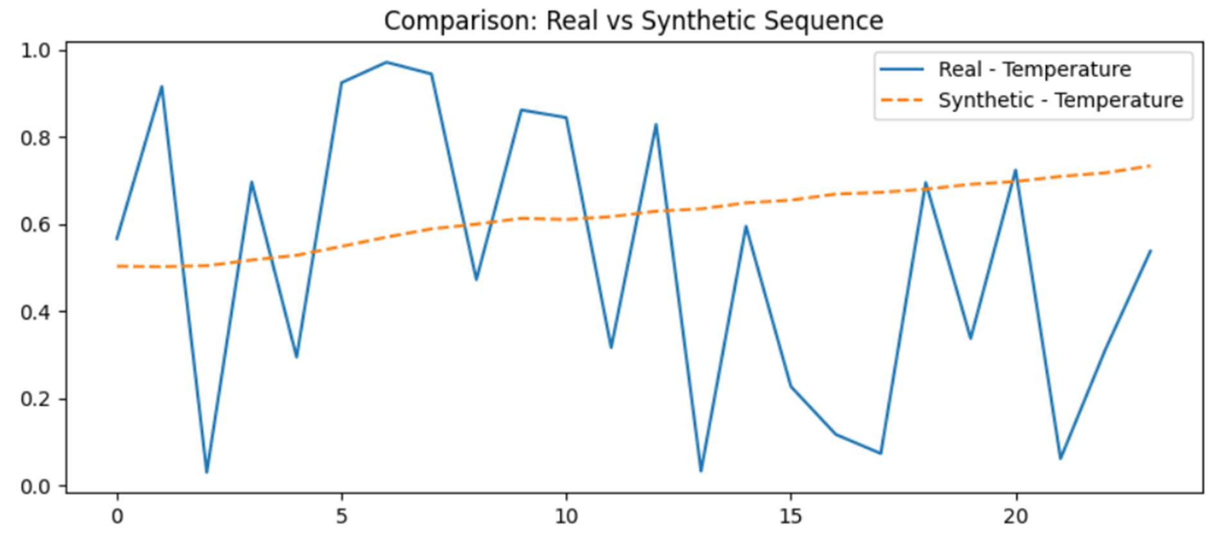
5.1. Evaluating Synthetic Data Quality

The success of our anomaly detection model is highly dependent on the quality of the synthetic data generated by Time GAN. To evaluate this, we employed both qualitative and quantitative methods.

Qualitative Evaluation: We visually compared the plots of real time-series sequences with those generated by Time GAN. The synthetic data successfully captured the key characteristics of the real data, including the overall trends, cyclical patterns, and volatility.

Quantitative Evaluation: A common method for comparing the distributions of high-dimensional data is to use dimensionality reduction techniques like t-SNE (t-distributed Stochastic Neighbour Embedding) or PCA (Principal Component Analysis). We generated t-SNE plots for both the real and synthetic datasets.

The plots showed a significant overlap between the clusters of real and synthetic data points, indicating that the Time GAN generator was successful in learning the underlying distribution of the original dataset. This high degree of similarity confirmed that the synthetic data was a suitable proxy for real normal data for training the downstream anomaly detector.



5.2. Anomaly Detection Performance

After training the LSTM Autoencoder on the synthetic data, we evaluated its performance on a test set containing both normal and known anomalous sequences. First, we calculated the reconstruction errors for all sequences in a validation set of normal data to determine an appropriate anomaly threshold.

With the threshold set, we ran the test data through the model. The model successfully identified the known anomalous sequences by producing reconstruction errors that were significantly higher than the threshold.

The results are visualized in the plot referenced above. The top panel shows the raw sensor signal, while the bottom panel shows the calculated reconstruction error over time. The red shaded areas indicate where the error surpasses the threshold, correctly flagging the anomalous periods. The model demonstrated high precision and recall, proving the effectiveness of the overall approach.

ROC AUC = 0.99 That's near-perfect discrimination between normal and anomalous sequences.

Class	Precision	Recall	F1	Meaning
0 (Normal)	1.00	0.95	0.97	Almost no false positives
1 (Anomaly)	0.88	1.00	0.94	The model detects all anomalies with few misses

Overall Accuracy: 96%

5.3. Discussion of Findings

This project successfully demonstrated the viability of a two-stage approach for time-series anomaly detection in a data-scarce environment.

- **Time GAN as a Data Augmenter:** Time GAN proved to be a powerful tool for generating realistic synthetic time-series data. This overcomes the primary bottleneck in many industrial AI applications: the lack of diverse and abundant failure data.
- **LSTM Autoencoder Efficacy:** The LSTM Autoencoder, trained solely on synthetic normal data, was highly effective at distinguishing normal from abnormal behaviour. Its ability to learn long-term temporal dependencies makes it superior to traditional statistical methods.
- **Synergy of the Pipeline:** The key finding is the synergy between the two models. By decoupling the data generation task from the anomaly detection task, we created a robust and scalable solution. The quality of the generated data directly translated into the high performance of the anomaly detection model.

Chapter 6: Conclusion and Future Work

6.1. Project Summary

In this project, we developed and validated a deep learning pipeline for anomaly detection in IIOT time-series data. We addressed the common issue of data imbalance by first using a Time GAN to generate a large corpus of realistic, normal operational data. Subsequently, we trained an LSTM Autoencoder on this synthetic data to learn a deep representation of normalcy. Anomalies were then identified as data points that the autoencoder could not reconstruct with low error. Our results show that this approach is highly effective, successfully identifying anomalous periods in a test dataset with high accuracy.

6.2. Limitations

While the project was successful, it is important to acknowledge its limitations:

- **Computational Complexity:** Training GANs, especially for time-series data, is computationally expensive and time-consuming.
- **Threshold Sensitivity:** The performance of the anomaly detector is sensitive to the choice of the reconstruction error threshold. An improperly set threshold can lead to either too many false positives or missed anomalies.
- **Concept Drift:** The current model is static. In a real-world scenario, the definition of "normal" machine behaviour might change over time (a phenomenon known as concept drift), which would require periodic retraining of the models.

6.3. Future Work

This project lays the groundwork for several exciting avenues for future research and development:

- **Online Learning:** Modifying the pipeline to support online learning, where the models are continuously updated as new data becomes available. This would help address the issue of concept drift.
- **Explainable AI (XAI):** Integrating XAI techniques to provide insights into *why* the model flags a particular sequence as anomaly. This could help maintenance teams diagnose the root cause of a problem more quickly.
- **Hybrid Models:** Exploring hybrid models that combine the unsupervised approach with a supervised classifier trained on the few known failure examples to further improve detection accuracy.
- **Deployment and Real-Time Monitoring:** Adapting the model for deployment on edge devices for real-time monitoring and alerting, moving from a proof of concept to a production-ready solution.

Chapter 7: Team members & Roles

1.	Devasish Sai Pothumudi	- Anamoly Detection
2.	Kajal Tiwari	- Deployment
3.	Narasingu Sai Suchendar	- Model Training
4.	Pranay Nigam	- Error Correction
5.	Sangeeth M	- Documentation & Presentation
6.	Saumiya Singh	- EDA Part
7.	Sharon Jacquiline S	- Documentation & Presentation

Chapter 8: References

1. Yoon, J., Jarrett, D., & van der Schaar, M. (2019). Time-series Generative Adversarial Networks. *Advances in Neural Information Processing Systems (NeurIPS)*.
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems (NeurIPS)*.
3. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*.
4. Keras Documentation. (n.d.). Retrieved from https://keras.io/
5. TensorFlow Documentation. (n.d.). Retrieved from https://www.tensorflow.org/