ACKNOWLEDGEMENT

I am deeply grateful to Prof. Prakash Babu G, Assistant Professor at Acharya Institute of Technology, for granting me the incredible opportunity to explore pattern recognition in time series data as part of my internship. My heartfelt thanks also go to, my mentor, and Mr. MVD Raghava, Deputy Manager & T.R.Ganesh , Chief General Manager , System Logistics for their unwavering guidance, encouragement, and invaluable insights.

Their expertise has been instrumental in shaping my understanding and skills, while Mr. MVD Raghava, seamless coordination of project work ensured a smooth and enriching experience. This internship has been a transformative phase in my academic journey, significantly enhancing my practical knowledge and professional growth. I am sincerely thankful to Mr. MVD Raghava for their mentorship and support, which have left an indelible mark on my career aspirations.

Name - Aditya Ranjan

ABSTRACT

This project explores the development and implementation of an LSTM-based deep learning model for feature extraction and fault classification in transmission lines. The Long Short-Term Memory (LSTM) network, a type of recurrent neural network (RNN), is employed to capture temporal dependencies and extract meaningful features from time-series data. Additionally, Fourier Transform is applied to complement the feature extraction process by analyzing the frequency domain characteristics of the data. The extracted features are combined and subjected to clustering using the K-Means algorithm to identify fault signatures.

To evaluate the performance of the clustering process, metrics such as Silhouette Score and Davies-Bouldin Score are utilized, providing insights into cluster quality and separability. The results are further visualized through graphs generated from clustered data stored in organized directories. This approach demonstrates the effectiveness of LSTM models in time-series feature extraction and their integration with unsupervised learning for anomaly detection and classification tasks in power systems.

CONTENTS

LIST OF FIGURES

CHAPTER 1

About the company – Grid India

1. About Grid India

Grid Controller of India Limited, commonly known as Grid-India, is a state-owned enterprise responsible for overseeing the integrated operation of India's national electricity grid. Formerly known as Power System Operation Corporation Limited (POSOCO), the company was rebranded to Grid-India to reflect its pivotal role in grid management.

Figure 1.1 - Company Logo

2. Key Areas of Operation

Key Operations of Grid-India:

1. **Integrated Grid Management:** Grid-India ensures the seamless operation of the National Load Despatch Centre (NLDC) and five Regional Load Despatch Centres (RLDCs). These centers are crucial for the real-time monitoring and control of electricity flow across the country, ensuring stability and reliability in power supply.

2. **Facilitation of Electricity Markets:** The organization plays a significant role in promoting competitive and efficient wholesale electricity markets. It administers settlement systems and facilitates non-discriminatory access to the transmission network, thereby supporting market operations and transactions.

3. **Implementation of Power Sector Reforms:** Grid-India is designated as the nodal agency for major reforms in the power sector. This includes the implementation and operation of the Green Energy Open Access Portal, Renewable Energy Certificate (REC) Mechanism, transmission pricing, short-term open access in transmission, and the Deviation Settlement Mechanism.

4. **Promotion of Technological Innovation:** The company is committed to adopting the latest technologies and practices, with a strong emphasis on cyber security. This approach ensures the grid's resilience and adaptability to emerging challenges and technological advancements.

5. **Human Resource Development:** Grid-India places importance on nurturing human and intellectual capital, recognizing that skilled personnel are essential for efficient grid management and operation.

Through these operations, Grid-India plays a vital role in maintaining the reliability, security, and efficiency of India's power system, facilitating the seamless transfer of electricity across regions and supporting the nation's energy infrastructure.

CHAPTER 2

Introduction to domain of internship

2.1 Deep Learning

Deep learning, a subset of artificial intelligence (AI) and machine learning (ML), focuses on simulating the functionality of the human brain to process data and create patterns for decision-making. It revolves around the concept of artificial neural networks, which consist of layers of interconnected nodes designed to mimic biological neural systems. Over the past decade, deep learning has emerged as a revolutionary technology, driving advancements across multiple domains such as image processing, natural language understanding, speech recognition, and autonomous systems.

2.2 Core Principles

Deep learning models learn from vast amounts of data using hierarchical feature extraction. Unlike traditional machine learning models, which require manual feature engineering, deep learning autonomously learns high-level features from raw data. For instance, in image processing, lower network layers identify edges and shapes, while deeper layers capture complex objects or patterns.

The backbone of deep learning is its network architecture, which includes types such as convolutional neural networks (CNNs) for image-related tasks, recurrent neural networks (RNNs) for sequential data, and transformers for natural language processing. These architectures are powered by modern hardware accelerators like GPUs and TPUs, enabling the training of models on massive datasets with billions of parameters.

2.3 Applications

Deep learning has penetrated various fields with transformative impact:

- **Computer Vision**: Tasks like image classification, object detection, and facial recognition use CNNs extensively.
- **Natural Language Processing (NLP)**: Models like GPT and BERT perform tasks like translation, sentiment analysis, and text generation.
- **Healthcare**: Deep learning aids in medical imaging, disease diagnosis, and drug discovery.
- **Autonomous Systems**: Vehicles and drones leverage deep learning for navigation and decision-making.
- **Energy and Fault Detection**: In domains like power transmission, deep learning is used for fault prediction and anomaly detection in time-series data.

Why Deep Learning Matters

The success of deep learning lies in its ability to scale with data and compute power. As datasets grow larger and computation becomes more affordable, the performance of deep learning systems continues to improve. Its adaptability allows it to be applied to structured and unstructured data alike, making it a critical tool for solving complex, real-world problems.

Deep learning represents a significant step toward achieving AI's potential to emulate human cognition, and its ongoing advancements promise to redefine the technological landscape in years to come.

CHAPTER 3

Technology and Tools

The implementation of time-series clustering in the provided code leverages a combination of modern technologies and tools to efficiently preprocess data, extract meaningful features, and perform clustering. Below is a detailed description of these technologies and tools:

3.1. Python Programming Language

Python serves as the primary language for implementing the solution. Its simplicity, versatility, and extensive ecosystem make it ideal for machine learning and data processing tasks. Python's rich libraries provide the foundation for data handling, feature extraction, and model building.

Figure 3.1 – Python

3.2. Jupyter Notebook

Jupyter Notebook is used as the development environment. It enables:

- **Interactive Coding**: Combining code, outputs, and markdown in a single document for easy visualization.
- **Data Visualization**: Facilitating exploratory data analysis and graph plotting directly within the notebook.
- **Reproducibility**: Sharing and understanding code seamlessly.

Figure 3.2 – Jupyter Notebook

3.3. Libraries for Data Manipulation

- **Pandas**:
  - Used for reading and processing CSV files.
  - Handles data cleaning, transformation, and missing value imputation with its flexible data structures (e.g., DataFrames).
  - Powers efficient manipulation of large time-series datasets.
- **NumPy**:
  - Provides efficient handling of numerical data.
  - Facilitates Fourier Transform calculations and matrix operations essential for deep learning preprocessing.

3.4. Libraries for Machine Learning and Deep Learning

- **Scikit-learn**:
  - Used for clustering (K-Means) and clustering evaluation metrics.
  - Key Functions:
    - KMeans: Groups time-series data based on extracted features.
    - metrics.silhouette_score and metrics.davies_bouldin_score: Evaluate clustering performance.
- **TensorFlow**:
  - Builds the LSTM-based deep learning model for feature extraction.
  - Provides tools to:
    - Define Sequential neural networks.
    - Train models on time-series data.
    - Perform backpropagation and optimization.

3.5. Libraries for Feature Extraction

- **SciPy**:
  - Used for computing Fast Fourier Transform (FFT), converting time-series data from the time domain to the frequency domain.
  - Enables extraction of frequency components as features, capturing periodic patterns in the data.
- **Sklearn's Preprocessing**:
  - MinMaxScaler: Scales data values to a fixed range [0, 1], ensuring uniform feature representation across datasets.

3.6. Libraries for Data Visualization

- **Matplotlib**:
  - Visualizes time-series data (VRM values) as line plots.
  - Produces graphs for clustered data, aiding in understanding and interpreting patterns.
  - Saves visualizations for later use in presentations or reports.

3.7. File and Folder Management

- **OS Library**:
  - Manages files and directories.
  - Automatically organizes data into cluster-specific folders (e.g., VRM Cluster {i}).
  - Ensures systematic storage of CSV files and visualizations based on cluster assignments.

3.8. Techniques and Models

- **LSTM (Long Short-Term Memory) Networks**:
  - A type of recurrent neural network (RNN) designed for sequential data.
  - Captures temporal dependencies in time-series data using sliding windows.
  - Extracts meaningful, high-dimensional features for clustering.
- **K-Means Clustering**:
  - Groups data into 10 clusters based on combined LSTM and FFT features.
  - An unsupervised learning method, ideal for discovering patterns in unlabeled time-series data.
- **Feature Engineering**:
  - Combines LSTM and FFT-derived features for a robust representation of time-series characteristics.

3.9. Evaluation Metrics

- **Silhouette Score**:
  - Evaluates the cohesion and separation of clusters.
  - Provides insights into the quality of clustering.
- **Davies-Bouldin Index (DBI)**:
  - Measures intra-cluster compactness and inter-cluster separation.

CHAPTER 4

LSTM Model Creation

The Long Short-Term Memory (LSTM) network is designed to process sequential data, extracting temporal patterns from the time-series voltage measurements.

Steps Involved:

1. **Sliding Window Technique**:
   a. The time-series data (VRM) is divided into smaller sequences using a fixed window size (sequence_length = 30).
   b. Each sequence serves as an input sample for the LSTM model, ensuring the model captures local temporal dependencies.
2. **Model Architecture**:
   a. **Input Layer**: Accepts time-series data sequences as input.
   b. **Hidden Layers**:
      i. Three LSTM layers with 100, 50, and 25 units, respectively.
      ii. Each LSTM layer captures temporal dependencies at varying levels of abstraction.
      iii. Dropout layers (20%) are added to prevent overfitting.
   c. **Output Layer**:
      i. A dense layer reduces the LSTM output to a fixed size (30 features).
      ii. These 30 features are the learned temporal representations of the time-series data.
3. **Training the Model**:
   a. The LSTM network is trained in an **autoencoder setup**:
      i. **Input and Output**: The input sequence is used as the target, training the model to reconstruct the sequence.
      ii. **Loss Function**: Measures the reconstruction error, guiding the model to learn meaningful patterns.
   b. After training, the learned feature representations from the dense layer are extracted for clustering.

```
# Create a more complex LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(sequence_length, 1)))  # First LSTM layer
model.add(Dropout(0.2))  # Dropout layer to prevent overfitting
model.add(LSTM(50, return_sequences=True))  # Second LSTM layer
model.add(Dropout(0.2))  # Another Dropout layer
model.add(LSTM(25))  # Third LSTM layer
model.add(Dropout(0.2))  # Dropout layer
model.add(Dense(30))  # Output layer
model.compile(optimizer='adam', loss='mse')

# Fit the model
model.fit(X, X, epochs=30, batch_size=32)  # Increased epochs for better training
```

```
D:\College\7th Semester\Internship\Application\venv\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_d
im` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Epoch 1/30
11789/11789 ━━━━━━━━━━━━━━━━━━━━ 239s 20ms/step - loss: 0.0237
Epoch 2/30
11789/11789 ━━━━━━━━━━━━━━━━━━━━ 242s 21ms/step - loss: 0.0036
Epoch 3/30
11789/11789 ━━━━━━━━━━━━━━━━━━━━ 248s 21ms/step - loss: 0.0033
Epoch 4/30
11789/11789 ━━━━━━━━━━━━━━━━━━━━ 249s 21ms/step - loss: 0.0032
```

Figure 4.1 –LSTM Model Creation

CHAPTER 5

LSTM & Fourier Transform Feature Extraction

The Fourier Transform captures frequency-domain characteristics of the time-series data, providing complementary information to the LSTM-derived temporal features.

Steps Involved:

1. **Fast Fourier Transform (FFT)**:
    a. Converts each sequence from the time domain to the frequency domain.
    b. Outputs a spectrum of frequencies and their amplitudes.
2. **Feature Selection**:
    a. The first 10 Fourier coefficients (magnitude) are selected as features.
    b. These coefficients represent dominant frequency components, capturing periodic patterns or trends in the data.

The final feature vector for each time-series sample is created by combining the features derived from the LSTM model and Fourier Transform.

Steps Involved*:*

1. **Concatenation**:
    a. The 30 LSTM features and 10 Fourier features are concatenated to form a 40-dimensional feature vector.
    b. This combined feature set provides a comprehensive representation of the time-series data:
        i. LSTM Features: Capture sequential dependencies and temporal patterns.
        ii. Fourier Features: Capture frequency-domain characteristics and periodicity.
2. **Normalization**:
    a. Ensures all features are scaled consistently, making them suitable for clustering algorithms.

Benefits of Combining LSTM and Fourier Features

1. **Complementary Insights**:
    a. LSTM focuses on time-domain dependencies, while Fourier Transform emphasizes frequency-domain characteristics.
    b. Combining both ensures that the model captures diverse patterns in the data.
2. **Improved Clustering**:
    a. The combined feature set enhances the ability of the K-Means algorithm to group similar samples, as it leverages both temporal and frequency information.

Output of the Feature Extraction Process

- For each time-series sequence, a 40-dimensional feature vector is created.
- These feature vectors are input to the K-Means algorithm for clustering.

This approach demonstrates the power of hybrid feature engineering, combining advanced deep learning techniques with traditional signal processing methods to achieve accurate and meaningful clustering of time-series data.

Figure 5.1 –Features Extraction

CHAPTER 6

K-Means Clustering: Overview and Implementation

K-Means is one of the most widely used clustering algorithms in unsupervised machine learning. It partitions a dataset into *kk*k distinct groups (clusters) based on feature similarity. Below is a detailed explanation of K-Means clustering, with a focus on its methodology, advantages, limitations, and application in the given code.

1. Definition

K-Means clustering aims to minimize the intra-cluster variance (how close data points are to the cluster center) while maximizing the inter-cluster separation (how distinct clusters are from each other).

Each cluster is represented by its centroid, which is the average of all data points within the cluster.

2. Working Mechanism

The K-Means algorithm follows these steps:

Step 1: Initialization

- Randomly select *kk*k points from the dataset as initial centroids. These centroids serve as the starting point for cluster assignment.

Step 2: Assignment

- Assign each data point to the nearest centroid based on a distance metric (usually Euclidean distance). This step forms *kk*k clusters.

Step 3: Update

- Recalculate the centroids as the mean of all data points assigned to each cluster.

Step 4: Repeat

- Repeat the assignment and update steps until the centroids stabilize (i.e., they no longer move) or a maximum number of iterations is reached.

Step 5: Output

- Final cluster assignments for each data point.
- Centroids of the clusters.

3. Distance Metric

- **Euclidean Distance**: Commonly used to measure the similarity between points.
- *d(x,c)=∑i=1n(xi−ci)2d(x, c) = \sqrt{\sum_{i=1}^{n} (x_i - c_i)^2}*d(x,c)=∑i=1n (xi −ci)2 , where *xx*x is a data point, and *cc*c is a centroid.

4. Strengths

- **Scalability**: Efficient for large datasets with a low computational cost for each iteration.
- **Simplicity**: Easy to understand and implement.
- **Versatility**: Works well for various types of numerical data.

5. Limitations

- **Choice of *kk*k**:
  - The number of clusters *kk*k must be defined beforehand.
  - Determining the optimal *kk*k is often non-trivial and requires techniques like the Elbow Method or Silhouette Analysis.
- **Sensitivity to Initialization**:
  - Poor initialization may lead to suboptimal clusters.
  - Methods like K-Means++ are used to improve initialization.
- **Assumes Spherical Clusters**:
  - Works best when clusters are convex and evenly sized.
- **Outlier Sensitivity**:
  - Outliers can disproportionately affect centroids and cluster assignments.

6. Application in the Uploaded Code

In the provided code, K-Means clustering is applied to the feature vectors generated by combining LSTM and Fourier Transform outputs.

Steps in the Code:

1. **Feature Space**:
   a. Uses a 40-dimensional feature vector (30 LSTM features + 10 Fourier features) as input to the K-Means algorithm.

2. **Cluster Count (*kk*k)**:
   a. *k=10k = 10*k=10, indicating the dataset is divided into 10 clusters based on feature similarity.

3. **Clustering Output**:
   a. Each data sample is assigned a cluster label (0 to 9).
   b. The clusters represent groups of time-series data with similar temporal and frequency characteristics.

4. **Data Organization**:
   a. Files corresponding to each cluster are moved into separate folders, allowing easy interpretation and visualization of clustered data.
5. **Evaluation**:
   a. Metrics like Silhouette Score and Davies-Bouldin Index assess the quality of clustering, indicating the cohesion within clusters and separation between clusters.

7. Advantages of K-Means in This Project

- **Unsupervised Nature**:
  - No labeled data is required, making it ideal for the given time-series dataset.
- **Simplicity**:
  - Straightforward implementation ensures efficient clustering of high-dimensional data.
- **Scalability**:
  - Handles large datasets and multidimensional features effectively.

8. Use Case for Time-Series Data

In the context of time-series clustering:

- K-Means identifies distinct patterns or anomalies (e.g., fault signatures in voltage measurements).
- Grouping similar data helps in fault detection, anomaly analysis, and predictive maintenance.

K Means Clustering

10 Clusters

[3]:
```
# KMeans clustering
kmeans = KMeans(n_clusters=10, random_state=0)
kmeans.fit(combined_features)
labels = kmeans.labels_

# Create directories for clusters
os.makedirs('Signature Fault Clusters Version 16 Final/VBM', exist_ok=True)
for i in range(10):
    os.makedirs(os.path.join('Signature Fault Clusters Version 16 Final/VBM', f'VBM Cluster {i}'), exist_ok=True)

# Move files to their respective clusters
for i, filename in enumerate(file_names):
    cluster_label = labels[i]
    source_path = os.path.join(folder_path, filename)
    destination_path = os.path.join('Signature Fault Clusters Version 16 Final/VBM', f'VBM Cluster {cluster_label}', filename)
    os.rename(source_path, destination_path)
print(f"Files successfully clustered.")
```

```
Files successfully clustered.
```

Figure 6.1 – K Means Clustering

CHAPTER 7

Summary

This project focuses on clustering time-series voltage measurements from transmission lines to detect patterns, anomalies, or potential faults. Using advanced machine learning techniques, the project aims to group similar time-series sequences into clusters, providing insights into system behavior and aiding in fault detection or predictive maintenance. The methodology involves leveraging deep learning for temporal feature extraction and traditional signal processing techniques for frequency-domain analysis, creating a robust hybrid feature set for clustering.

Steps Involved

1. Data Preprocessing

The raw voltage data is processed to ensure it is suitable for analysis:
Missing values are handled to maintain dataset integrity.
Normalization is applied using Min-Max scaling to standardize the feature range.
Time-series data is segmented into fixed-length sequences using a sliding window approach to prepare for further processing.

2. Feature Extraction

Feature extraction is a crucial step in this project, combining time-domain and frequency-domain techniques:
LSTM Autoencoder:
A Long Short-Term Memory (LSTM) network is used to process time-series data.
The autoencoder setup trains the model to reconstruct input sequences, allowing it to learn meaningful temporal patterns.
After training, 30 high-level temporal features are extracted from the dense layer.
Fourier Transform:
The Fast Fourier Transform (FFT) converts time-series data into the frequency domain, capturing periodic patterns.
The first 10 Fourier coefficients are selected, representing dominant frequency components.
Feature Combination:
The 30 LSTM features and 10 Fourier features are concatenated, creating a 40-dimensional feature vector for each time-series sequence.

3. Clustering with K-Means

The combined feature vectors are clustered using the K-Means algorithm:
The number of clusters (*kk*) is set to 10, dividing the dataset into groups with similar temporal and frequency characteristics.
Each sequence is assigned a cluster label (0 to 9), representing its membership.

4. Evaluation Metrics

The performance of clustering is evaluated using two unsupervised metrics:

Silhouette Score:

Measures how well-separated clusters are.

A score of **0.3253** indicates moderate clustering quality, suggesting some overlap between clusters but reasonable separation overall.

Davies-Bouldin Index (DBI):

Measures cluster compactness and separation.

A DBI value of **1.0299** indicates relatively compact and well-separated clusters, representing good clustering performance.

These metrics provide critical feedback on clustering quality, ensuring the process effectively captures the underlying patterns in the data.

5. Visualization and Organization

The clustered data is visualized by plotting time-series sequences within each cluster. This helps in understanding the characteristics of different clusters. Additionally, data and plots are organized into separate directories based on cluster labels, enabling easier interpretation and analysis.

Key Insights

The project demonstrates the effectiveness of a hybrid feature engineering approach, combining LSTM-based temporal features with Fourier-based frequency features. The clustering results, supported by evaluation metrics, reveal meaningful groupings in the voltage data. This provides actionable insights for fault detection, anomaly analysis, and system optimization. The use of advanced techniques ensures scalability and adaptability, making the approach suitable for other time-series datasets and clustering tasks.

Conclusion

This project highlights a robust methodology for analyzing time-series data through unsupervised learning. By combining temporal and frequency-domain insights, it achieves meaningful clustering results that can be used for predictive maintenance and anomaly detection in transmission line systems. The evaluation metrics further validate the quality of the clustering, showcasing its utility in addressing real-world challenges in the energy domain.

References

1. **Deep Learning and LSTM Networks**:
   a. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.
   b. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
2. **Clustering Techniques**:
   a. MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1: Statistics.
   b. Lloyd, S. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129–137.
3. **Feature Extraction**:
   a. Brigham, E. O. (1988). *The Fast Fourier Transform and Its Applications*. Prentice-Hall.
   b. Abdi, H., & Williams, L. J. (2010). Principal Component Analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4), 433–459.
4. **Evaluation Metrics**:
   a. Rousseeuw, P. J. (1987). Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.
   b. Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2), 224–227.
5. **Tools and Libraries**:
   a. Chollet, F. (2015). *Keras: Deep Learning Framework*.
   b. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
   c. NumPy, Pandas, Matplotlib, and Seaborn for data processing and visualization.

Appendix

Skills and Knowledge Gained During the Project

1. **Understanding of LSTM Networks**:
   a. Hands-on experience in designing, training, and extracting features from LSTM autoencoders to capture temporal patterns in time-series data.
   b. Insights into how LSTMs handle sequential dependencies and long-term memory retention.
2. **Signal Processing Techniques**:
   a. Application of Fourier Transform for extracting frequency-domain features, learning how periodic patterns in time-series data translate into frequency components.

3. **Unsupervised Learning**:
   a. Comprehensive understanding of K-Means clustering and its application in high-dimensional feature spaces.
   b. Exploration of cluster evaluation metrics like Silhouette Score and Davies-Bouldin Index to validate clustering performance.
4. **Hybrid Feature Engineering**:
   a. Experience in combining deep learning (LSTM features) and traditional methods (FFT features) to build a robust feature space for clustering.
5. **Data Preprocessing and Visualization**:
   a. Proficiency in handling large datasets, normalization, and sequence segmentation.
   b. Visualization of clustered data to interpret and present results effectively.
6. **Python Tools and Libraries**:
   a. Mastery of Python libraries such as TensorFlow/Keras for deep learning, Scikit-learn for clustering, and Matplotlib/Seaborn for data visualization.

Personal and Professional Growth

- Enhanced problem-solving skills by applying theoretical concepts to a real-world problem in the energy domain.
- Improved technical writing and reporting skills through documentation of findings.
- Learned project management by breaking the problem into modular tasks and systematically solving them.

This project provided a solid foundation in advanced machine learning and signal processing, opening up opportunities to work on similar challenges in other domains.