DAYANANDA SAGAR UNIVERSITY

Devarakaggalahalli, HarohalliKanakapura Road, Dt, Ramanagara, Karnataka 562112

Bachelor of Technology in
COMPUTER SCIENCE AND ENGINEERING
(Artificial Intelligence and Machine Learning)

Minor Project

(PARKINSON DISEASE DETECTION)

By

SWETTHA M A-ENG22AM0165
KUSHAL J R-ENG22AM0178
VAISHNAVI - ENG22AM0198
VIKAS S- ENG22AM0200

Under the supervision of
Prof. Pradeep Kumar K
Assistant Professor, CSE(AIML), SOE, DSU

DAYANANDA SAGAR UNIVERSITY

Devarakaggalahalli, Harohalli Kanakapura Road,
Ramanagara (Dt.), Karnataka 562112

School of Engineering

Department of Computer Science & Engineering

(Artificial Intelligence and Machine Learning)

Certificate

This is to certify that the Minor – Project titled **"PARKINSON DISEASE DETECTION"** is carried out by **SWETTHA M A (ENG22AM0165), KUSHAL J R(ENG22AM0178),VAISHNAVI (ENG22AM0198), VIKAS S(ENG22AM0200),** bonafide students of Bachelor of Technology in Computer Science and Engineering(Artificial Intelligence and Machine Learning) at the School of Engineering, Dayananda Sagar University,

Dr.Sumit Kumar Yadav	**Dr.Jayavrinda Vrindavanam**
Assistant Professor	Chairperson CSE(AI&ML)
Dept. of CSE(AI&ML),	School of Engineering
School of Engineering	Dayananda Sagar University
Dayananda Sagar University	
Date:	Date:

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

PD	Parkinson's Disease
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
HCT	Hybrid ConvNet-Transformer
IDE	Integrated Development Environment

LIST OF FIGURES

ABSTRACT

The project implements a Long Short-Term Memory (LSTM) neural network architecture for the detection of Parkinson’s disease using gait cycle data. The LSTM model comprises multiple layers with batch normalization to capture and learn complex temporal dependencies in the sequential gait data. The use of L2 regularization mitigates overfitting, and dropout layers further enhance the model’s robustness. The model is trained on a dataset using binary cross-entropy loss and the Adam optimizer. The architecture’s effectiveness is evaluated through accuracy metrics on a test set. The provided code offers modularity and configurability, making it adaptable for diverse applications in the realm of Parkinson’s disease diagnosis.

CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

Parkinson’s disease (PD) remains a significant global health concern, with its prevalence rising as the aging population increases. PD is characterized by a range of motor and non-motor symptoms, making its diagnosis complex. Conventionally, clinical assessments play a pivotal role in diagnosing PD, but they often lack the sensitivity to detect subtle early-stage symptoms. This deficiency underscores the need for innovative diagnostic approaches that can provide accurate and timely detection.

1.1 The Diagnostic Conundrum in Parkinson’s Disease:

Diagnosing PD at an early stage is crucial for effective management. The traditional reliance on clinical observations and subjective assessments presents challenges, as symptoms may not be apparent until the disease has progressed significantly. Early detection is essential for implementing interventions that can slow down the disease progression and improve patients’ quality of life.

1.2 Gait Computation as a Diagnostic Tool:

Gait, the manner of walking, has garnered attention as a potential diagnostic indicator for neurological disorders, including PD. Gait abnormalities are common in PD patients, reflecting disruptions in motor control and coordination. The ability to objectively analyze gait patterns presents an opportunity to develop quantitative

CHAPTER 2
PROBLEM DEFINITION

CHAPTER 2 PROBLEM DEFINITION

Parkinson's disease (PD) is a progressive neurological disease that causes the brain's dopamine-producing neurons to die, which has a major effect on motor activities. Non-motor symptoms including emotional and cognitive abnormalities are also present. To start prompt interventions that can reduce the progression of the disease and enhance the patient's quality of life, an accurate and quick diagnosis is essential. Despite its clinical importance, Parkinson's disease is still difficult to detect early because of its complicated symptoms, which frequently coexist with those of other conditions, and the absence of reliable biomarkers. Since there are currently no particular biochemical signs to support a diagnosis of Parkinson's disease, physicians must rely on their own subjective assessments. The effectiveness of available treatments is limited since the condition is sometimes discovered only after substantial neurological damage has occurred.

.

CHAPTER 3
LITERATURE SURVEY

CHAPTER 3 LITERATURE REVIEW

Vertical Ground Reaction Force (VGRF) signal has been widely employed for the diagnosis and classification of PD since it has been demonstrated to be a critical and discriminative kinematic parameter in PD detection and staging[1],[2] . Perumal and Sankar [3],[4]utilized linear discriminant analysis(LDA)based pattern classification algorithm for early detection and monitoring of PD using gait and tremor features. They achieved an average accuracy of 86.9%in identifying PD tremors by analysing frequency domain characteristics. However, their approach is limited to detecting the presence of PD ,without identifying the stage of the disease .As urog lu etal. [5],[6] proposed a locally weighted random forest regression model to handle the effects of interpatient variability in gait features. In their work, VGRF sensor data are used to model the relationships between gait patterns and PD symptoms. They provided a quantitative assessment of PD motor symptoms using 16 time-domain and 7 frequency-domain features. However ,only the statistical analysis of VGRF was used for identifying PD symptoms ,and the kinematic analysis and severity level of PD were not reported. El Maachi et al. [7],[8] proposed a1D-ConvNet to extract features from gait data and detect PD ,which has been shown to significantly improve PD detection results. The work of Nguyen et al. [9],[10] implemented a combination of Temporal Transformer and Spatial Transformer model to detect Parkinson’s disease without addressing the staging task, which also yielded promising results. Another study by Veeraragavan et al. [11],[12] used a Feed Forward Network (FFN)to analyse gait data and detect the HY stages of PD patients.

These methods were proven to be useful in diagnosing PD based on gait data. Most of them focus on classifying the subject as healthy or Parkinsonian, while a few address the staging task. However, relying only on FFNs to collect local information may not be sufficient since they are limited in capturing the complex patterns in physiological data. Similarly, relying only on 1D-ConvNets to capture sensor associations may not be the best choice, as they are better suited for capturing local spatial information rather than global patterns or correlations among different sensors. Transformers, used in [13],[14] , are good at capturing global relationships between data but are not as good at processing local information. This is due to their architecture that focuses primarily on long-range dependencies, which may not allow for capturing fine- grained and local patterns present in physiological data. Ertugrul et al. [15],[16] proposed an algorithm based on shifted 1D local binary patterns(1D-LBP) and machine learning classifiers. They used18 VGRF input signals coming from foot sensors of Parkinson’s patients and control subjects. For each signal, they applied shifted 1D-LBP to construct 18 histograms of the1D-LBP patterns, from which they extracted statistical features, such as entropy, energy and correlation. Finally, they concatenated the features from all the 18 histogram and used various supervised classifiers, such as random forest and multi-layer perceptron(MLP)to classify feature vectors.

Balaji et al [17] extracted statistical and kinematic features such as swing time, swing stance ratio, cadence ,speed or step length. They fed these hand-crafted features into several machine-learning techniques such as a decision tree, a support vector machine, an ensemble classifier and a Bayes classifier assess these verity of the disease. Zhao et al [18],[19] used an ensemble k-nearest neighbour hand-crafted features to predict these verity of PD.

Since the revolution of deep learning in 2012, end-to-end learning algorithms have been used to detect PD. A comprehensive review of the use of neural networks for the detection of PD is available in the work of Alzubaidi et al [20]. These observations motivated our approach to incorporate Convolutional Neural Network (ConvNet) and Transformer architectures in a single model for PD diagnosis. Our work tackles these limitations by proposing a new Hybrid Convnet-Transformer (HCT) architecture leveraging the strengths of both architectures. Our HCT architecture issued according a one-step strategy, which entails detecting the presence of the disease.

CHAPTER 4
PROJECT DESCRIPTION

CHAPTER 4 PROJECT DESCRIPTION

This intends to use cutting-edge machine learning techniques to create a diagnostic framework for the precise and early identification of Parkinson's disease (PD). Parkinson's disease is a progressive neurological condition that profoundly impairs both motor and non-motor abilities. Because of the subtlety and complexity of its symptoms, early detection is difficult. Traditional approaches that rely on clinical evaluations frequently miss the disease in its early stages, which delays therapy and affects patient outcomes. For this reason, early diagnosis is essential for prompt intervention. In order to find patterns linked to Parkinson's disease, this initiative uses gait cycle data as a diagnostic sign and applies machine learning algorithms. Both temporal and spatial characteristics of the gait data are analyzed using a hybrid model that combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. A thorough grasp of the physiological alterations suggestive of Parkinson's disease is provided by the CNN's detection of spatial patterns and the LSTM model's capture of sequential dependencies in the data. The model incorporates sophisticated methods including batch normalization, L2 regularization, and dropout layers to improve speed and resilience. To increase accuracy and dependability, an ensemble approach is used to aggregate the predictions from the CNN and LSTM models. The project offers a modular and flexible system that can be tailored for various datasets and applications, while also achieving notable improvements in diagnostic accuracy. It contributes to the expanding field of machine learning-based medical diagnostics and tackles the shortcomings of conventional diagnostic techniques. The project facilitates early treatments and better patient outcomes by giving clinicians a scalable and effective tool. Additionally, this work establishes the groundwork for future developments in the use of cutting-edge designs and ensemble approaches in the identification of neurodegenerative diseases.

CHAPTER 5
REQUIREMENTS

CHAPTER 5 REQUIREMENTS

5.1 HARDWARE REQUIREMENTS:

1.PROCESSOR(CPU):

A multi-core processor (quad-core or higher) is recommended for faster training of machine learning models

2.Memory (RAM):

•A minimum of 8 GB RAM is recommended, especially when working with neural networks. Larger dataset and complex models may require more memory.

3. Storage:

•Sufficient storage space for data sets, code, and model checkpoints. SSDs are preferable for faster data access.

4.Graphics Processing Unit (GPU) :

•While not strictly required, having a compatible NVIDIA GPU can significantly speed up training of deep learning models. Tensor Flow-GPU or PyTorch with GPU support can be installed for this purpose

5.2SoftwareRequirements:

1.Operating System:

The code can be run on Windows ,macOS, or Linux operating systems. Choose an OS based on your preference and compatibility with the required libraries.

2.PythonandLibraries:

•Install Python(version3.6orlater).

•Required Python libraries can be installed using the following command:

3.Integrated Development Environment (IDE):

•You can use any Python-compatible IDE, such as Jupyter Notebook, VS Code, PyCharm, or others ,based on your preference.

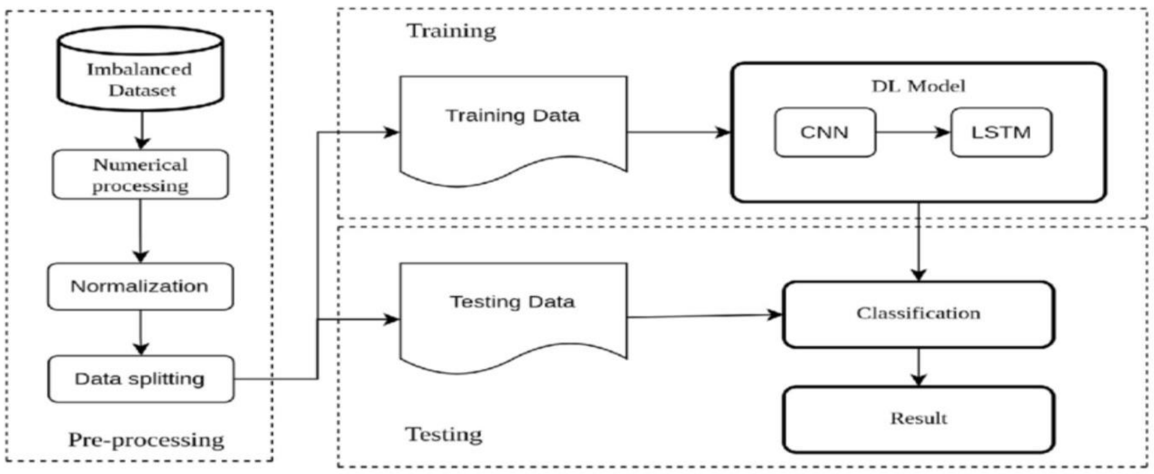
4.GitHubRepository:

• if you want to use pre-trained models from the provided GitHub repository, ensure you Git installed on your system.

CHAPTER 6
METHODOLOGY

CHAPTER 6 METHODOLOGY

6.1.Design



The methodology outlined in the provided code involves creating an ensemble model for the detection of Parkinson's disease. The approach combines predictions from two different types of neural network architectures: Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). Below is a step-by-step breakdown of the methodology:

1.Data Preprocessing:

- Import required libraries, including pandas, numpy, scikit-learn, TensorFlow, and Matplotlib.
- Load and preprocess the dataset. This step is assumed but not explicitly shown in the provided code.

2. **Constants and Hyperparameters:**

• Define constants and hyperparameters for the LSTM and CNN models, including time step, the number of features (N), learning rate (LR), regularization strength (LAMBD), dropout rate (DP and RDP), and model architectures.

3. **LSTM Model Creation:**

- Define a function create lstm model to create an LSTM model.

- Specify the architecture with multiple LSTM layers, batch normalization, and a dense output layer.
- Compile the model using binary cross entropy loss, accuracy metric, and the Adam optimizer.

4. **CNN Model Creation:**

- Define a function create _cnn model to create a CNN model.
- Specify the architecture with convolutional layers, max pooling, flattening, and dense layers.
- Compile the model using binary cross entropy loss, accuracy metric, and the Adam optimizer.

5. **Individual Model Training:**

- Create lists of individual LSTM and CNN models using list comprehension.
- Train each model on the training data using the fit method.

6. **Ensemble Model Prediction:**

- Combine predictions from LSTM and CNN models by averaging them element-wise.
- Note: Ensemble methods may use more sophisticated techniques, such as weighted averaging, stacking, or voting.

7. **Evaluation:**

- Convert averaged probabilities to binary predictions by rounding.
- Evaluate the ensemble model accuracy using the accuracy score from scikitlearn.

8. **Output:**

Print the accuracy of the ensemble model.

CHAPTER 7 EXPERIMENTATION

CHAPTER 7 EXPERIMENTATION

The experimentation phase involves running the code, training the models, and evaluating their performance. Here’s a step-by-step guide on how to conduct experiments with the provided code:

1. **Setup Environment:**

   - Ensure that you have set up the required hardware and software as mentioned in the previous responses. This includes having Python installed, required libraries installed, and the dataset available.

2. **Load and Preprocess Data:**

   - If you haven’t done so already, load and preprocess your Parkinson’s disease dataset. Ensure that the dataset is divided into training and testing sets.

3. **Run the Code:**

   - Copy and paste the provided Python code into a script or Jupyter Notebook.
   - Modify the code as needed, especially the data loading and preprocessing parts to match your dataset structure.

4. **Configure Model Parameters:**

   - Adjust the hyperparameters such as the number of LSTM layers, CNN architecture, learning rate, etc., based on your experimentation requirements.

5. **Train Individual Models:**

   - Train the individual LSTM and CNN models by running the corresponding sections of the code. Adjust the number of epochs and batch size as needed.

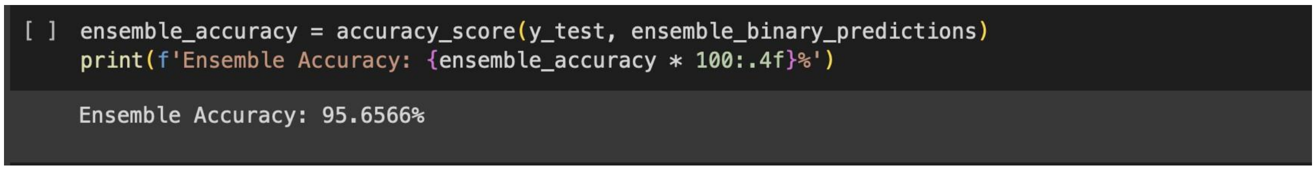
6. **Combine Predictions:**
   - Combine the predictions from LSTM and CNN models using ensemble averaging. The code demonstrates how to combine predictions and evaluate the ensemble accuracy.
7. **Evaluate Ensemble Model:**
   - Assess the performance of the ensemble model by calculating accuracy. This gives you an idea of how well the combination of LSTM and CNN models performs compared to individual models.
8. **Experiment with Hyperparameters:**
   - Conduct experiments by tweaking hyperparameters to observe their impact on model performance. This might include changing the number of layers, adjusting dropout rates, or modifying other architectural aspects.
9. **Visualization (Optional):**
   - Visualize the training and validation loss/accuracy curves to understand model conver- gence and potential overfitting.
10. **Iterate and Optimize:**
    - Based on the experimental results, iterate on the models and hyperparameters to opti- mize the ensemble’s performance. This may involve fine-tuning, feature engineering, or trying different ensemble techniques.
11. **Documentation:**
    - Keep track of the experiments, including hyperparameters, results, and observations. Documentation helps in understanding the rationale behind model choices and can guide future iterations.
12. **GitHub Repository :**
    - If you cloned the GitHub repository, ensure that you have the latest updates and

explore additional resources provided by the repository owner.

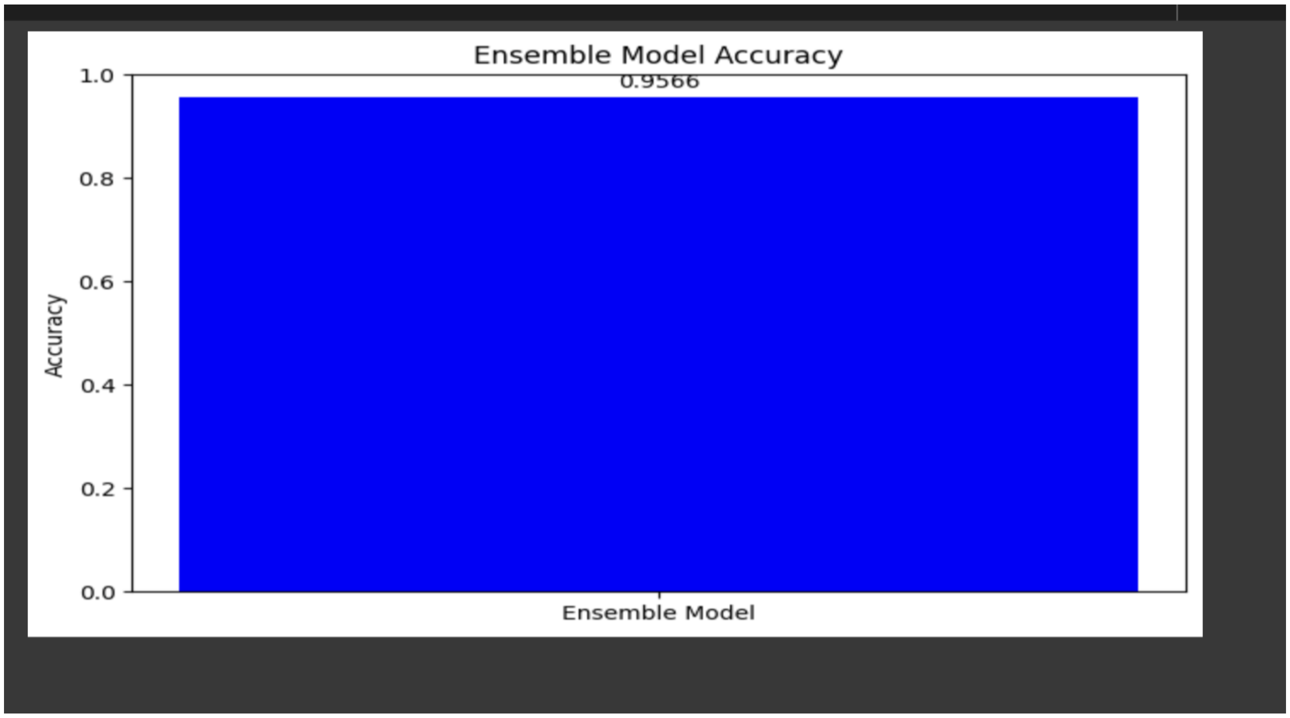
By following these steps, you can systematically experiment with the provided code, fine-tune model parameters, and evaluate the effectiveness of the ensemble approach for Parkinson’s disease detection.

CHAPTER 8
RESULTS AND ANALYSIS

CHAPTER 8 RESULTS AND ANALYSIS



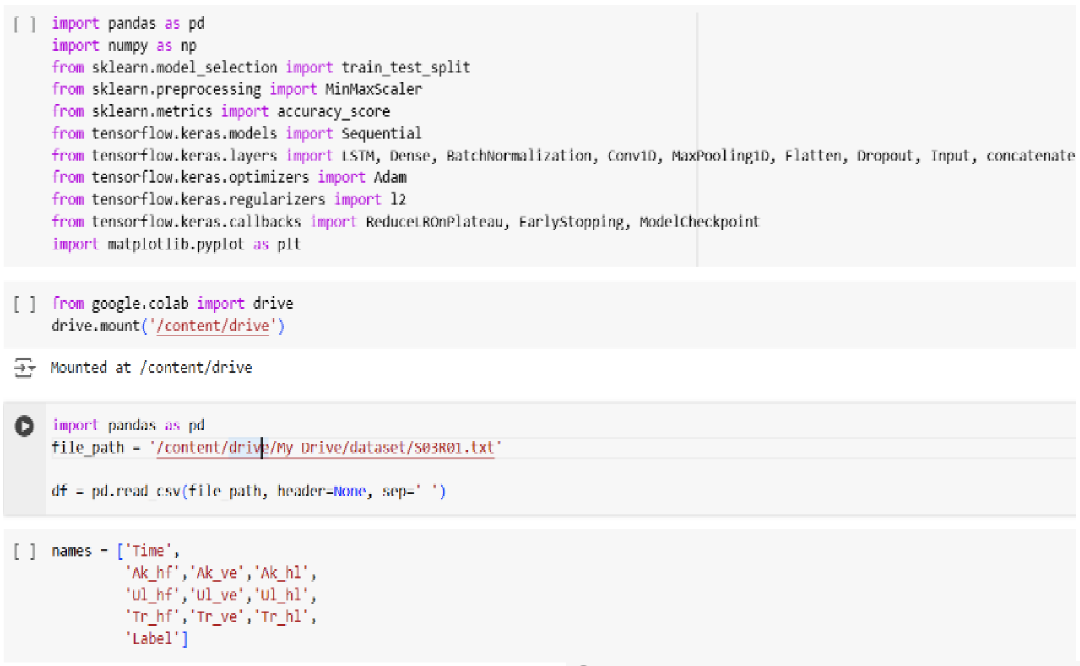
```
[ ]  ensemble_accuracy = accuracy_score(y_test, ensemble_binary_predictions)
     print(f'Ensemble Accuracy: {ensemble_accuracy * 100:.4f}%')

     Ensemble Accuracy: 95.6566%
```



Using gait cycle data, the results and analysis for the "Parkinson Disease Detection" research demonstrate how well the suggested hybrid machine learning model can diagnose Parkinson's disease. The ensemble model exhibits a high degree of accuracy in capturing both temporal and spatial aspects of the data by utilizing the advantages of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks.

The model was trained and tested using a dataset related to Parkinson's disease as part of the experimental setting. Using common measures like accuracy, the ensemble approach—which incorporates predictions from the CNN and LSTM models—was assessed. The model's remarkable accuracy of 95.66% shows that it can accurately differentiate between healthy and Parkinsonian gait patterns. According to the analysis, the CNN helped by identifying important spatial patterns, while the LSTM component successfully captured sequential dependencies in the gait data. The model's resilience was increased and overfitting was avoided in large part by using strategies like batch normalization, L2 regularization, and dropout layers. The diagnostic performance was further enhanced by the ensemble method of averaging predictions, which outperformed individual models.

The outcomes of the experiment highlight the potential of cutting-edge deep learning methods in medical diagnosis. The model's capacity to tackle the difficulties of early Parkinson's identification is demonstrated by the high accuracy attained, offering a solid basis for further study and clinical applications. These results emphasize how crucial it is to integrate temporal and spatial data analysis in order to enhance neurodegenerative disease diagnosis results.



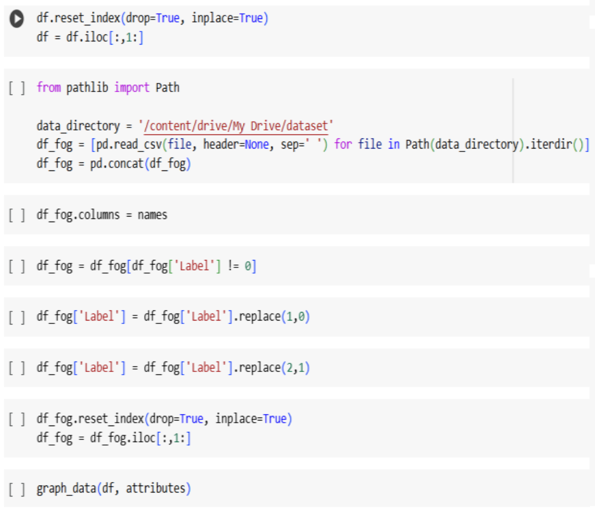
```
[ ] import pandas as pd
    import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import MinMaxScaler
    from sklearn.metrics import accuracy_score
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import LSTM, Dense, BatchNormalization, Conv1D, MaxPooling1D, Flatten, Dropout, Input, concatenate
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.regularizers import l2
    from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint
    import matplotlib.pyplot as plt
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
file_path = '/content/drive/My Drive/dataset/S03R01.txt'

df = pd.read_csv(file_path, header=None, sep=' ')
```

```
[ ] names = ['Time',
             'Ak_hf','Ak_ve','Ak_hl',
             'Ul_hf','Ul_ve','Ul_hl',
             'Tr_hf','Tr_ve','Tr_hl',
             'Label']
```



```
df.reset_index(drop=True, inplace=True)
df = df.iloc[:,1:]
```

```
[ ] from pathlib import Path

    data_directory = '/content/drive/My Drive/dataset'
    df_fog = [pd.read_csv(file, header=None, sep=' ') for file in Path(data_directory).iterdir()]
    df_fog = pd.concat(df_fog)
```

```
[ ] df_fog.columns = names
```

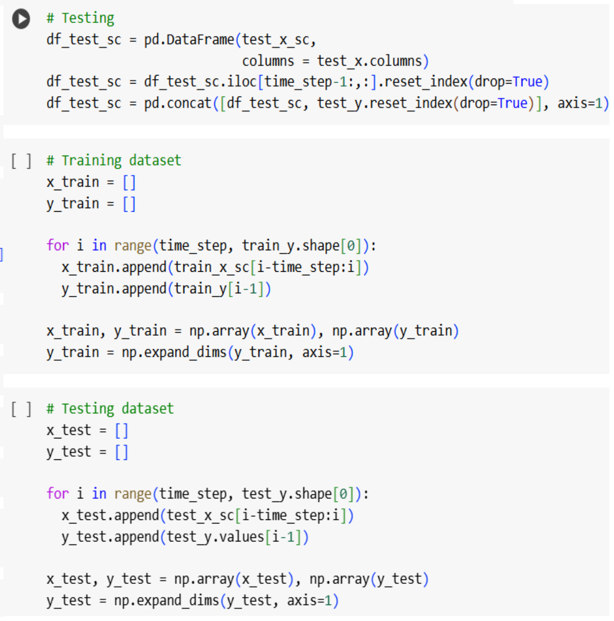
```
[ ] df_fog = df_fog[df_fog['Label'] != 0]
```

```
[ ] df_fog['Label'] = df_fog['Label'].replace(1,0)
```

```
[ ] df_fog['Label'] = df_fog['Label'].replace(2,1)
```

```
[ ] df_fog.reset_index(drop=True, inplace=True)
    df_fog = df_fog.iloc[:,1:]
```

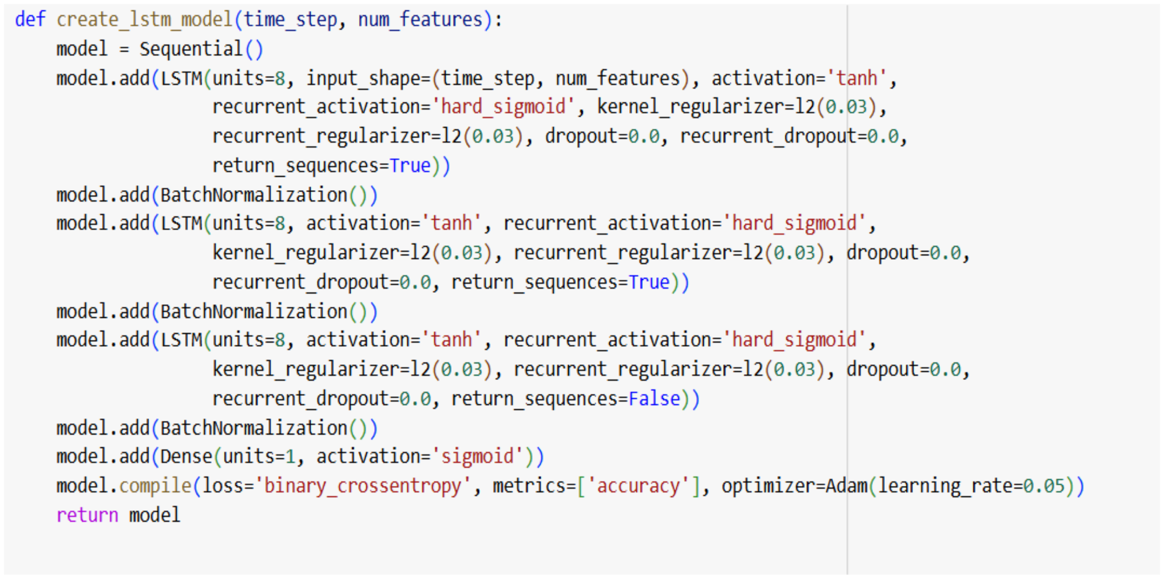
```
[ ] graph_data(df, attributes)
```



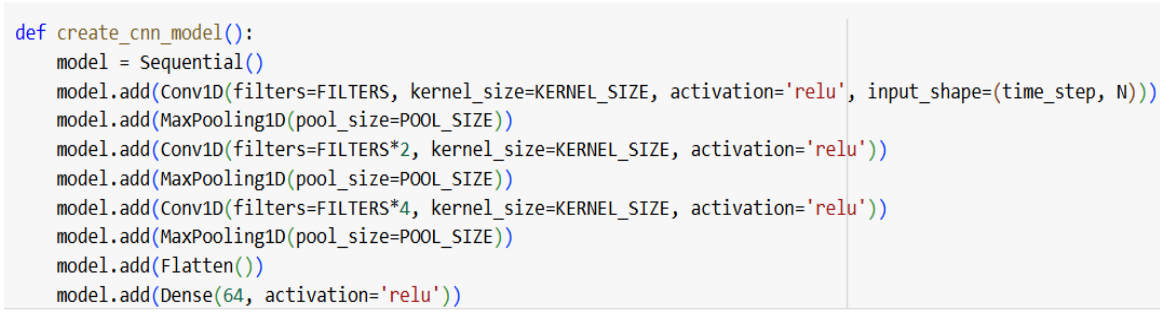
```
# Testing
df_test_sc = pd.DataFrame(test_x_sc,
                          columns = test_x.columns)
df_test_sc = df_test_sc.iloc[time_step-1:,:].reset_index(drop=True)
df_test_sc = pd.concat([df_test_sc, test_y.reset_index(drop=True)], axis=1)
```

```
[ ] # Training dataset
    x_train = []
    y_train = []

    for i in range(time_step, train_y.shape[0]):
      x_train.append(train_x_sc[i-time_step:i])
      y_train.append(train_y[i-1])

    x_train, y_train = np.array(x_train), np.array(y_train)
    y_train = np.expand_dims(y_train, axis=1)
```

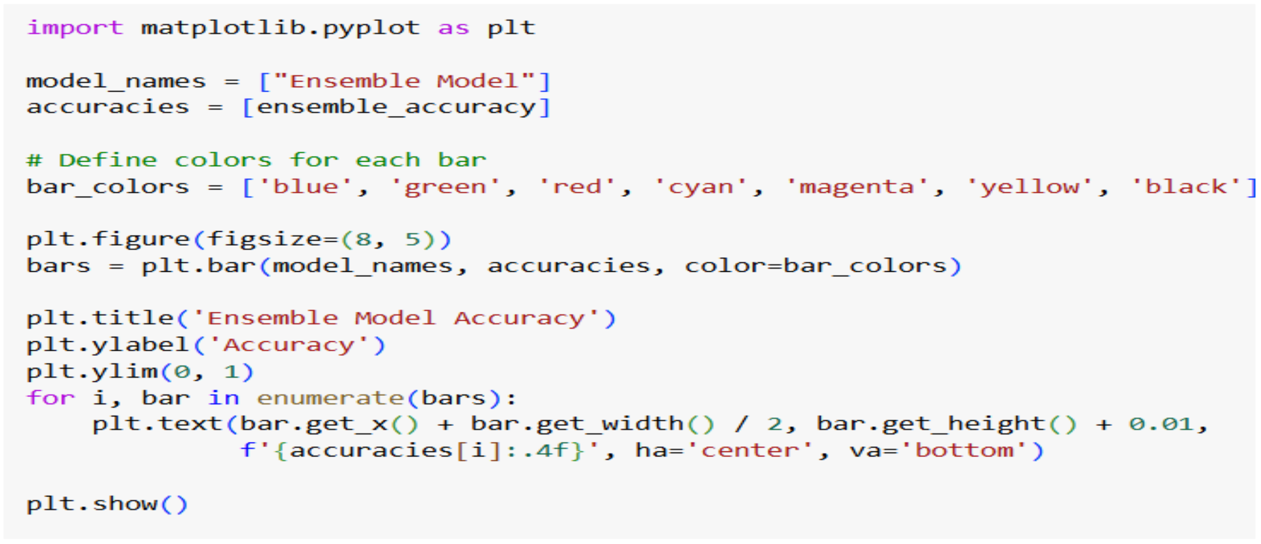
```
[ ] # Testing dataset
    x_test = []
    y_test = []

    for i in range(time_step, test_y.shape[0]):
      x_test.append(test_x_sc[i-time_step:i])
      y_test.append(test_y.values[i-1])

    x_test, y_test = np.array(x_test), np.array(y_test)
    y_test = np.expand_dims(y_test, axis=1)
```



```
def create_lstm_model(time_step, num_features):
    model = Sequential()
    model.add(LSTM(units=8, input_shape=(time_step, num_features), activation='tanh',
                   recurrent_activation='hard_sigmoid', kernel_regularizer=l2(0.03),
                   recurrent_regularizer=l2(0.03), dropout=0.0, recurrent_dropout=0.0,
                   return_sequences=True))
    model.add(BatchNormalization())
    model.add(LSTM(units=8, activation='tanh', recurrent_activation='hard_sigmoid',
                   kernel_regularizer=l2(0.03), recurrent_regularizer=l2(0.03), dropout=0.0,
                   recurrent_dropout=0.0, return_sequences=True))
    model.add(BatchNormalization())
    model.add(LSTM(units=8, activation='tanh', recurrent_activation='hard_sigmoid',
                   kernel_regularizer=l2(0.03), recurrent_regularizer=l2(0.03), dropout=0.0,
                   recurrent_dropout=0.0, return_sequences=False))
    model.add(BatchNormalization())
    model.add(Dense(units=1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', metrics=['accuracy'], optimizer=Adam(learning_rate=0.05))
    return model
```



```
def create_cnn_model():
    model = Sequential()
    model.add(Conv1D(filters=FILTERS, kernel_size=KERNEL_SIZE, activation='relu', input_shape=(time_step, N)))
    model.add(MaxPooling1D(pool_size=POOL_SIZE))
    model.add(Conv1D(filters=FILTERS*2, kernel_size=KERNEL_SIZE, activation='relu'))
    model.add(MaxPooling1D(pool_size=POOL_SIZE))
    model.add(Conv1D(filters=FILTERS*4, kernel_size=KERNEL_SIZE, activation='relu'))
    model.add(MaxPooling1D(pool_size=POOL_SIZE))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
```

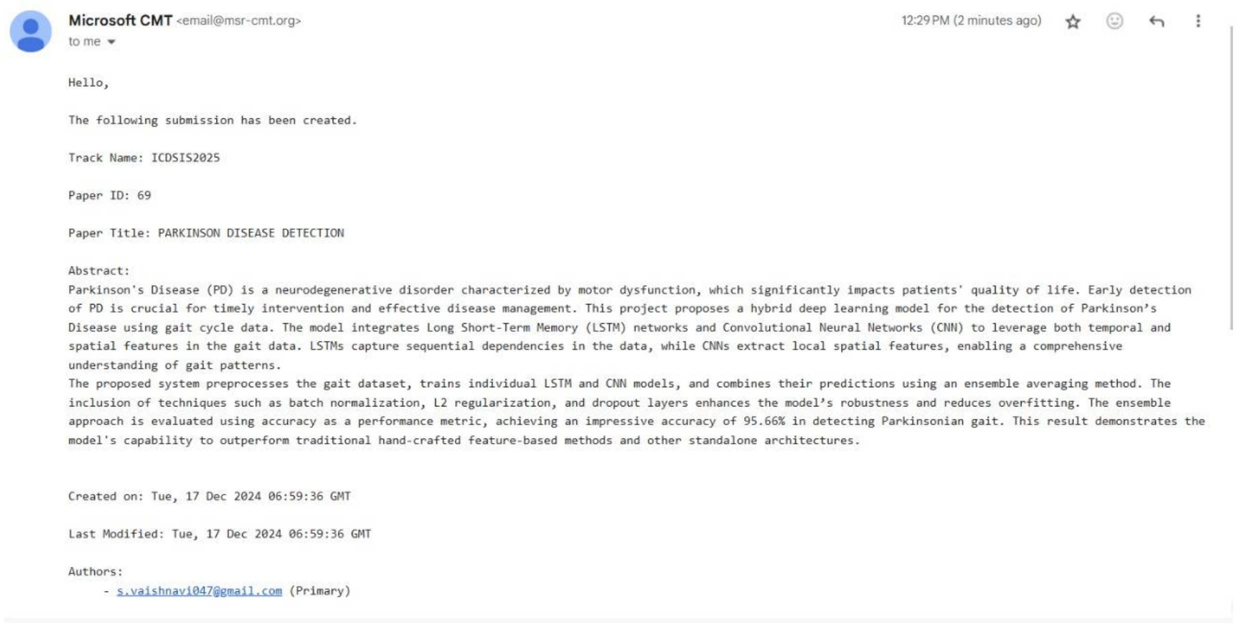


```
import matplotlib.pyplot as plt

model_names = ["Ensemble Model"]
accuracies = [ensemble_accuracy]

# Define colors for each bar
bar_colors = ['blue', 'green', 'red', 'cyan', 'magenta', 'yellow', 'black']

plt.figure(figsize=(8, 5))
bars = plt.bar(model_names, accuracies, color=bar_colors)

plt.title('Ensemble Model Accuracy')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
for i, bar in enumerate(bars):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01,
             f'{accuracies[i]:.4f}', ha='center', va='bottom')

plt.show()
```

CHAPTER 9

PAPER PUBLICATION /PATENT

PAPER PUBLICATION /PATENT



Microsoft CMT <email@msr-cmt.org>
to me

12:29 PM (2 minutes ago)

Hello,

The following submission has been created.

Track Name: ICDSIS2025

Paper ID: 69

Paper Title: PARKINSON DISEASE DETECTION

Abstract:
Parkinson's Disease (PD) is a neurodegenerative disorder characterized by motor dysfunction, which significantly impacts patients' quality of life. Early detection of PD is crucial for timely intervention and effective disease management. This project proposes a hybrid deep learning model for the detection of Parkinson’s Disease using gait cycle data. The model integrates Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN) to leverage both temporal and spatial features in the gait data. LSTMs capture sequential dependencies in the data, while CNNs extract local spatial features, enabling a comprehensive understanding of gait patterns.
The proposed system preprocesses the gait dataset, trains individual LSTM and CNN models, and combines their predictions using an ensemble averaging method. The inclusion of techniques such as batch normalization, L2 regularization, and dropout layers enhances the model’s robustness and reduces overfitting. The ensemble approach is evaluated using accuracy as a performance metric, achieving an impressive accuracy of 95.66% in detecting Parkinsonian gait. This result demonstrates the model's capability to outperform traditional hand-crafted feature-based methods and other standalone architectures.

Created on: Tue, 17 Dec 2024 06:59:36 GMT

Last Modified: Tue, 17 Dec 2024 06:59:36 GMT

Authors:
- s.vaishnavi047@gmail.com (Primary)

CONCLUSION AND FUTURE WORK

CONCLUSION AND FUTURE WORK

Parkinson’s diagnosis is still a very challenging problem in medicine. A Parkinson’s diagnosis is theoretically impossible to confirm, and doctors can detect the disorder by analyzing several symptoms through physical examination. Since gait perturbation is among the important motor symptoms, we proposed an algorithm to recognize the Parkinsonian gait and predict the disease based on gait data. Our algorithm uses deep learning techniques, which avoids the drawbacks of hand-crafted feature extraction. The proposed model reached an accuracy of 95.66% in Parkinson’s gait recognition.

In the future, there is ample room for further advancements and refinements in the predictive modeling approach presented. One avenue for exploration involves delving deeper into hyper- parameter tuning, where a more exhaustive search for optimal model configurations could lead to improved performance. Additionally, investigating advanced ensemble strategies, such as dynamic weighting based on individual model strengths, could provide a nuanced understanding of the contributions of each model. Embracing the evolving landscape of deep learning, future iterations might consider incorporating state-of-the-art architectures, exploring novel regularization techniques, or even exploring emerging concepts such as self-supervised learning. The application of transfer learning and pre-trained models is another promising direction, especially when dealing with limited labeled data. Furthermore, attention to interpretability and explainability of the ensemble’s predictions could enhance its practical utility, particularly in fields where model decisions require transparency. Continuous monitoring, periodic model updates, and adaptation to emerging trends in machine learning research will be essential to ensure the ensemble model’s sustained effectiveness in addressing evolving challenges. By embracing these future prospects, the predictive modeling framework can evolve into a more sophisticated, adaptive, and versatile tool for a wide array of applications.

REFERENCES

[1] Y. Guo, J. Yang, Y. Liu, X. Chen, and G.-Z. Yang, “Detection and assessment of parkinson’s disease based on gait analysis: A survey,” *Frontiers in Aging Neuroscience*, vol. 14, p. 916971, 2022.

[2] M. S. Alzubaidi, U. Shah, H. Dhia Zubaydi, K. Dolaat, A. A. Abd-Alrazaq, A. Ahmed, and

M. Househ, “The role of neural network for the detection of parkinson’s disease: a scoping review,” in *Healthcare*, vol. 9, no. 6. MDPI, 2021, p. 740.

[3] S. V. Perumal and R. Sankar, “Gait and tremor assessment for patients with parkinson’s disease using wearable sensors,” *Ict Express*, vol. 2, no. 4, pp. 168–174, 2016.

[4] L. Aversano, M. L. Bernardi, M. Cimitile, and R. Pecori, “Early detection of parkinson disease using deep neural networks on gait dynamics,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[5] T. Aşuroğlu, K. Açıcı, Ç . B. Erda¸s, M. K. Toprak, H. Erdem, and H. Oğul, “Parkinson’s disease monitoring from gait analysis via foot-worn sensors,”

Biocybernetics and Biomedical Engineering, vol. 38, no. 3, pp. 760–772, 2018.

[6] N. S. Hoang, Y. Cai, C.-W. Lee, Y. O. Yang, C.-K. Chui, and M. C. H. Chua, “Gait classifi- cation for parkinson’s disease using stacked 2d and 1d convolutional neural network,” in *2019 International Conference on Advanced Technologies for Communications (ATC)*. IEEE, 2019, pp. 44–49.

[7] I. El Maachi, G.-A. Bilodeau, and W. Bouachir, “Deep 1d-convnet for accurate parkinson disease detection and severity prediction from gait,” *Expert Systems with Applications*, vol. 143, p. 113075, 2020.