OVERVIEW

Approximately 40% of the global population suffers from iodine deficiencies, which significantly contribute to thyroid-related diseases, affecting over 200 million people worldwide. The thyroid gland, a small but crucial endocrine organ located in the neck, plays a vital role in regulating metabolism, heart rate, body temperature, and overall energy balance. It achieves this by producing two key hormones: Triiodothyronine (T3) and Thyroxine (T4), which influence multiple physiological processes. Any disruption in the thyroid's function can lead to severe health conditions, including hypothyroidism (underactive thyroid), hyperthyroidism (overactive thyroid), structural abnormalities (goiter, nodules), thyroid tumors, and subclinical disorders. These conditions are typically diagnosed using blood tests that analyze hormone levels, such as Thyroid-Stimulating Hormone (TSH), T3, and T4 concentrations. However, traditional diagnostic approaches are often limited by subjective interpretation, variability in hormone levels over time, and dependence on expert medical evaluation.

Advancements in Artificial Intelligence (AI) and Deep Learning (DL) have revolutionized medical diagnostics by enabling automated, data-driven disease classification and prediction. Deep learning models such as Multilayer Perceptron (MLP) and Deep Neural Networks (DNN) have demonstrated promising results in thyroid disease prediction by identifying hidden patterns in clinical data. These models excel in feature extraction but struggle with capturing sequential dependencies, such as the natural fluctuations in thyroid hormone levels over time. Since thyroid function is dynamic and influenced by various physiological and external factors, a static approach to classification often leads to suboptimal results. To address this challenge, we propose a hybrid MLP + Long Short-Term Memory (LSTM) model, which combines the deep feature extraction capabilities of MLP with LSTM's strength in sequential pattern recognition.

LSTM, a specialized type of Recurrent Neural Network (RNN), is designed to process time-series data by retaining important information from previous states and learning long-term dependencies. By integrating MLP for static feature extraction and LSTM for temporal analysis, the proposed model can effectively detect and analyze trends in thyroid function test results, leading to improved classification of normal, hypothyroid, and hyperthyroid cases. This hybrid model ensures a more comprehensive and dynamic assessment of thyroid disorders, enabling more precise and reliable predictions. Furthermore, to enhance the model's stability and prevent overfitting, we incorporate batch normalization, which ensures stable learning, dropout regularization, which reduces overfitting by randomly deactivating neurons during training, and a dynamic learning rate scheduler, which optimizes learning efficiency by adjusting the learning rate based on model performance.

Our MLP + LSTM hybrid model is trained on real-world thyroid function test datasets, allowing it to outperform traditional machine learning models in both accuracy and reliability. By leveraging deep learning's ability to automate diagnosis and improve classification precision, this approach can significantly aid healthcare professionals in early detection, personalized treatment planning, and long-term monitoring of thyroid disorders. This model has the potential to streamline clinical workflows, reduce diagnostic errors, and provide a scalable AI-driven solution for thyroid disease detection in diverse healthcare settings. Future research can focus on integrating additional clinical parameters, expanding datasets for better generalization, and incorporating explainable AI (XAI) techniques to enhance interpretability for medical practitioners. By addressing these challenges, deep learning models like MLP + LSTM can play a transformative role in AI-driven thyroid disease prediction and early intervention strategies, improving patient outcomes worldwide.

1.1 MOTIVATION

Thyroid-related disorders affect a significant portion of the global population, with approximately 40% suffering from iodine deficiencies. The thyroid gland, a small yet vital endocrine organ located in the neck, plays a crucial role in regulating metabolism, heart rate, body temperature, and energy levels by producing two key hormones: triiodothyronine (T3) and thyroxine (T4)**.** Any disruption in thyroid function can lead to various health issues, including hypothyroidism, hyperthyroidism, thyroid nodules, goiter, and thyroid cancer. These conditions often go undiagnosed in their early stages, leading to severe complications if not detected and treated in time.

Traditional diagnostic methods primarily rely on thyroid function tests (TFTs), blood work (measuring T3, T4, and thyroid-stimulating hormone or TSH), and imaging techniques (ultrasound and scans)**.** While effective, these tests require expert analysis, and their accuracy may be affected by factors such as sample collection errors, lab variations, and a lack of time-series data analysis.

With the rise of artificial intelligence (AI) in healthcare, machine learning (ML) and deep learning (DL) models have emerged as powerful tools for diagnosing thyroid diseases. Conventional ML models, such as Support Vector Machines (SVM), Decision Trees (DT), and Random Forests (RF), analyze structured clinical data but struggle with sequential dependencies, such as hormonal fluctuations over time**.** Meanwhile, deep learning techniques like Multilayer Perceptron (MLP) and Deep Neural Networks (DNN) offer improved feature extraction but fail to effectively capture time-series relationships in hormone levels.

To address this gap, we propose a hybrid deep learning model that integrates MLP with Long Short-Term Memory (LSTM) networks**.** LSTM, a type of Recurrent Neural Network (RNN), is specifically designed to analyze sequential data and retain long-term dependencies, making it highly suitable for thyroid function prediction. This hybrid approach aims to enhance the accuracy of thyroid disease classification by combining MLP's ability to learn complex feature representations with LSTM's capability of recognizing temporal trends. The proposed model is expected to outperform traditional ML and standalone deep learning models, thus assisting healthcare professionals in early detection, precise diagnosis, and timely treatment planning for thyroid-related disorders.

1.2 PROBLEM STATEMENT

Current machine learning (ML) models for thyroid disease prediction primarily rely on static clinical features, such as thyroid function test results at a single point in time. However, these models fail to capture the temporal variations in hormone levels, which are critical for accurately diagnosing thyroid disorders. Traditional deep learning models like Multilayer Perceptron (MLP) and Deep Neural Networks (DNN) excel in feature extraction but lack the capability to process time-dependent patterns and long-term dependencies in sequential data. As a result, these models often struggle to provide accurate classifications, especially in cases where hormone levels fluctuate over time.

To address this limitation, our research proposes a hybrid MLP + LSTM model that combines the strengths of both deep feature learning and sequential pattern recognition.

MLP is used for extracting meaningful representations from clinical data, while Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN), captures long-term dependencies and trends in thyroid function over time. This hybrid approach aims to enhance classification accuracy for normal, hypothyroid, and hyperthyroid cases by effectively leveraging both static and dynamic features. By improving the precision of thyroid disease detection, our model can assist healthcare professionals in early diagnosis, personalized treatment planning, and better disease management.

1.2 BACKGROUND

DATA PROCESSING

Data processing is a crucial step in machine learning-based thyroid disease prediction, ensuring that the raw clinical data is cleaned, structured, and transformed for accurate analysis. In this approach, thyroid function test results (TSH, T3, T4 levels) and other patient attributes are used as input, and the processed data is used for disease classification. It is a form of numerical data processing and feature extraction that allows AI models to make predictions.

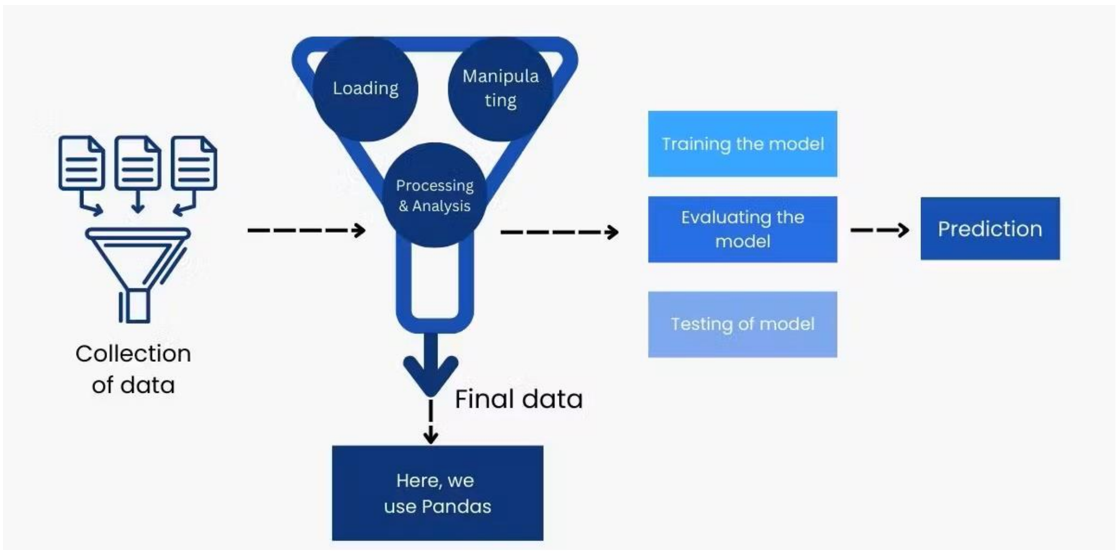


Figure 1.1 Data Processing

In the medical field, data processing is widely used for analyzing patient health records, lab test results, and time-series hormone fluctuations to assist in early diagnosis. In thyroid disease prediction, proper data preprocessing and feature selection are essential to train machine learning models for classifying patients as normal, hypothyroid, or hyperthyroid.

In recent years, Artificial Intelligence (AI) has made significant advancements in the field of medical diagnostics, particularly in disease prediction. AI is widely used for analyzing clinical data, identifying patterns, and assisting in early disease detection. The most common techniques used in AI-based medical prediction include Machine Learning (ML) and Deep Learning (DL). Traditional ML models such as Decision Trees, Support Vector Machines (SVM), and Random Forests have been applied in medical research. However, deep learning approaches like Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) have shown superior performance by effectively capturing complex relationships and sequential hormone variations over time.

MACHINE LEARNING (ML)

In artificial intelligence, it is described as a machine's ability to imitate intelligent human behaviour. Machine learning (ML) is the science of automated systems that may enhance themselves over time by gaining knowledge by using data. An algorithm that uses training data to build a model to make predictions or judgements without needing to be explicitly programmed to do so produces predictions.

It is done in 3 ways they are: data processing, model building and deployment and monitoring.

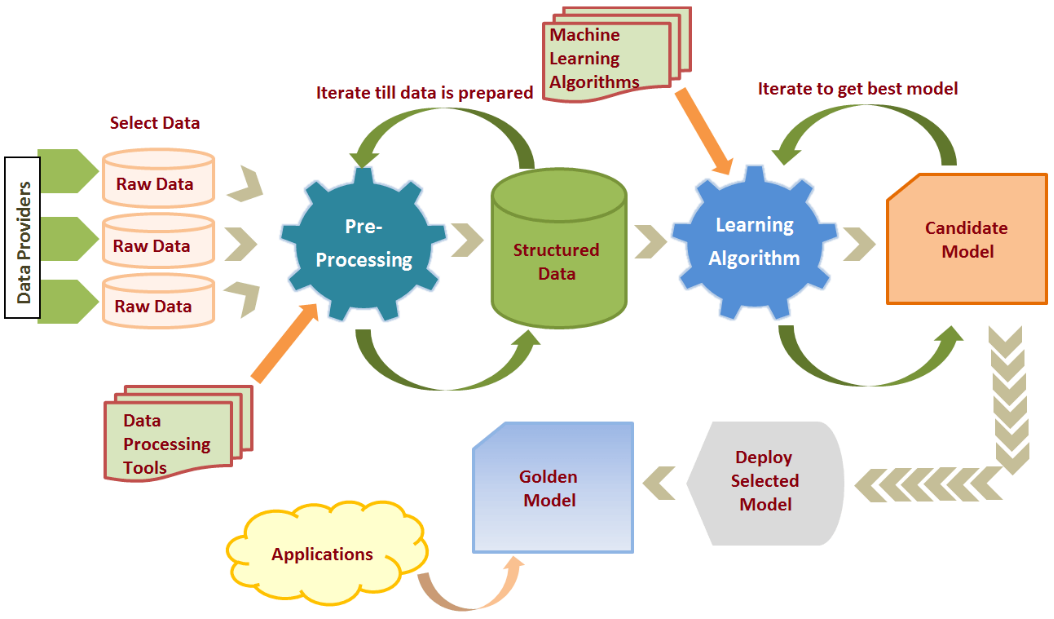


Figure 1.2 Process of Machine Learning

DEEP LEARNING

Deep Learning (DL) is a subset of Machine Learning (ML) that focuses on training artificial neural networks (ANNs) with multiple layers to recognize patterns in data. Inspired by the human brain, deep learning enables computers to learn from vast amounts of structured and unstructured data without explicit programming. It is widely used in image recognition, speech processing, natural language understanding, and medical diagnosis.

Deep learning models automatically extract meaningful features from raw data, making them superior to traditional ML algorithms that require manual feature engineering. The key advantage of deep learning is its ability to process complex, high-dimensional data, such as clinical test results, time-series data, medical images, and sensor readings.

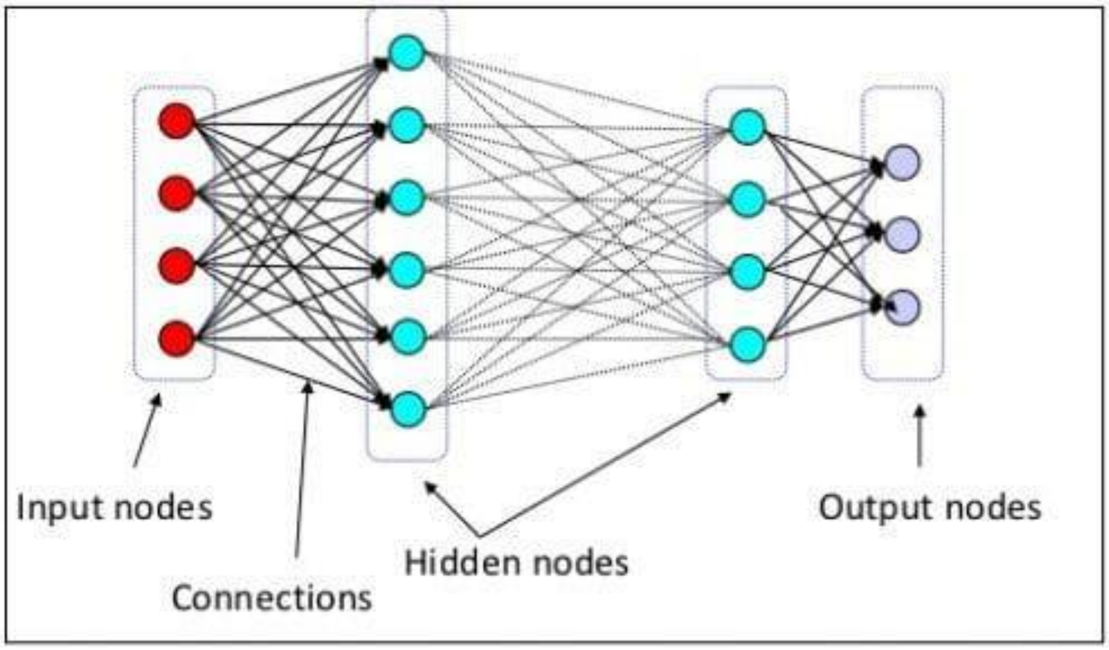


Figure 1.3 Deep Learning

STEPS IN DEEP LEARNING WORKING PROCESS

- Data Collection and Preprocessing
- Feature Extraction and Learning
- Model Training and Optimization
- Model Validation and Testing
- Deployment and Real-World Application

DEEP LEARNING IN PREDICTION

Deep learning has revolutionized predictive modeling by enabling computers to learn patterns from vast amounts of data and make highly accurate predictions. Unlike traditional machine learning algorithms, deep learning models can automatically extract features, recognize complex relationships, and handle high-dimensional data such as medical test results, images, and time-series records.

In disease prediction, deep learning plays a crucial role in analyzing structured clinical data, medical images, and genetic information to detect diseases early and improve diagnostic accuracy. It is widely used in cancer detection, heart disease prediction, diabetes monitoring, and thyroid disease classification.

THYROID GLAND AND ITS FUNCTIONS

The thyroid gland is a small, butterfly-shaped endocrine gland located in the lower front part of the neck, just below the Adam’s apple. It plays a vital role in regulating metabolism, energy production, and overall body function by producing hormones that influence nearly every organ system. The gland is controlled by the pituitary gland, which releases Thyroid-Stimulating Hormone (TSH) to regulate thyroid hormone production.

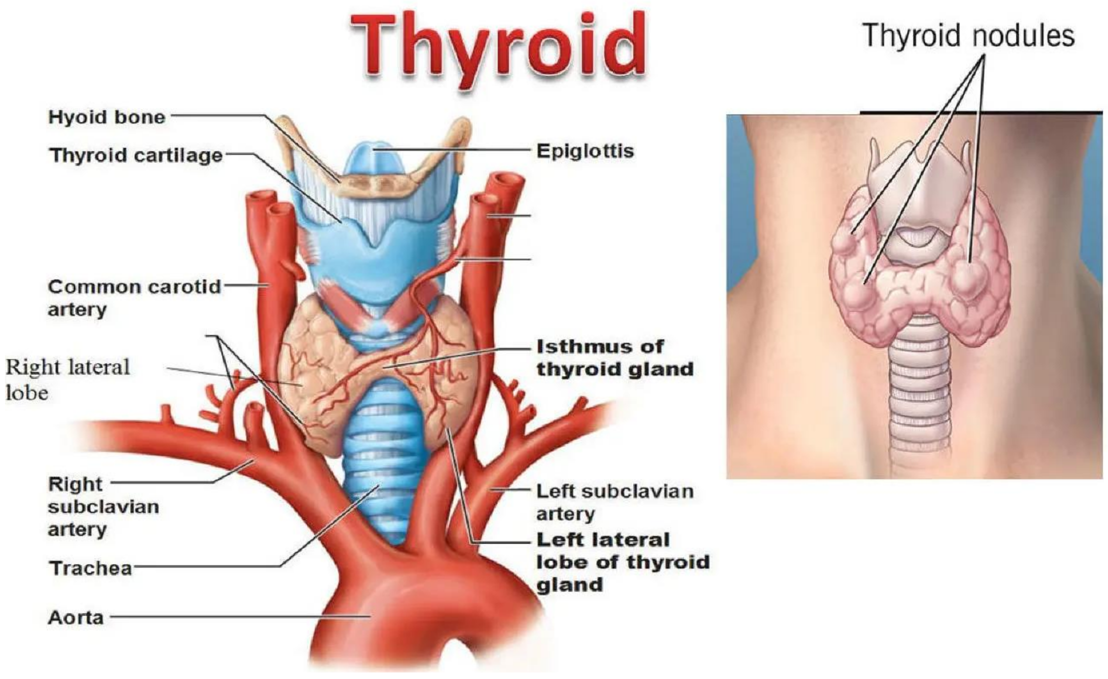


Figure 1.4 Thyroid Gland

The thyroid gland produces three major hormones:

- Triiodothyronine (T3) – The Active Thyroid Hormone Thyroxine
- (T4) – The Storage Hormone
- Thyroid-Stimulating Hormone (TSH) – The Thyroid Regulator

Triiodothyronine (T3) – The Active Thyroid Hormone

Triiodothyronine (T3) is one of the two primary hormones secreted by the thyroid gland. It is the **biologically active form of thyroid hormone**, meaning it directly affects the body's metabolic processes. Although the thyroid produces a small amount of T3 directly, most T3 is formed by converting **Thyroxine (T4) into T3** in various body tissues.

T3 Imbalances and Their Effects:

- Low T3 (Hypothyroidism): Leads to fatigue, weight gain, depression, and sluggish metabolism.
- High T3 (Hyperthyroidism): Causes rapid heart rate, nervousness, weight loss, and excessive sweating.

Thyroxine (T4) – The Storage Hormone

Thyroxine (T4) is the inactive form of thyroid hormone, serving as a storage form that the body converts into T3 when needed. T4 is produced in much larger quantities than T3 but has a weaker effect until converted.

T4 Imbalances and Their Effects:

- Low T4 (Hypothyroidism): Causes slowed metabolism, fatigue, cold intolerance, and dry skin.
- High T4 (Hyperthyroidism): Leads to anxiety, tremors, heat intolerance, and rapid heartbeat.
- Severe Hypothyroidism (Myxedema): In extreme cases, can cause swelling, slow speech, memory issues, and even coma if untreated.

Thyroid-Stimulating Hormone (TSH) – The Thyroid Regulator

Thyroid-Stimulating Hormone (TSH) is a hormone produced by the pituitary gland that regulates thyroid function. It acts as a messenger, telling the thyroid how much T3 and T4 to produce.

TSH Imbalances and Their Effects:

- High TSH (Hypothyroidism): Indicates underactive thyroid, where the gland does not produce enough T3 and T4.
- Low TSH (Hyperthyroidism): Indicates overactive thyroid, where excessive T3 and T4 are being produced.

THYROID DISORDERS AND THEIR IMPACT

Thyroid diseases occur when the gland **produces too much or too little hormone**, leading to various health complications:

- **Hypothyroidism (Underactive Thyroid):** Causes fatigue, weight gain, slow heart rate, and depression.
- **Hyperthyroidism (Overactive Thyroid):** Leads to weight loss, rapid heartbeat, and nervousness.
- **Other conditions:**
    - **Goiter** (Enlarged thyroid due to iodine deficiency).
    - **Thyroid cancer** (Malignant tumors affecting the thyroid gland).

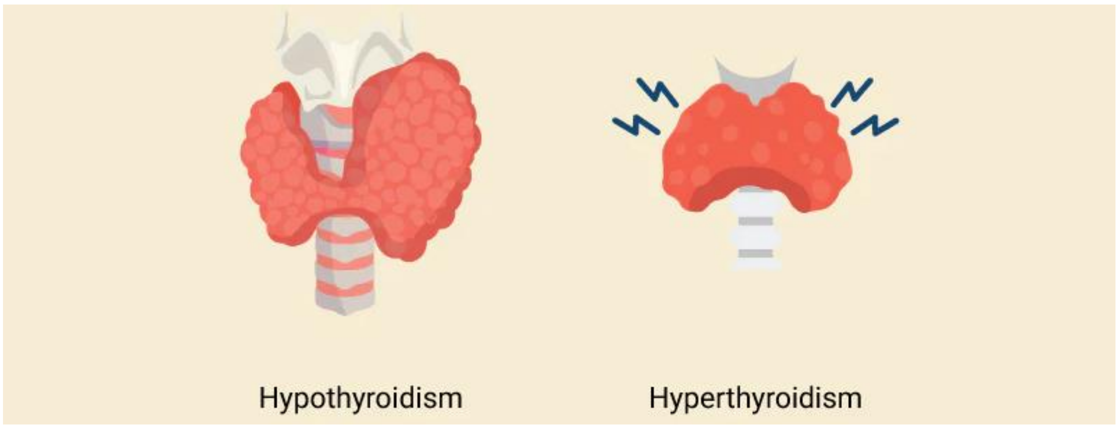


Figure 1.5 Hypothyroidism and Hyperthyroidism

DIAGNOSTIC METHODS FOR THYROID DISEASE

Several diagnostic techniques are used to detect thyroid disorders:

- **Blood Tests:** Measure TSH, T3, and T4 levels.
- **Ultrasound Imaging:** Identifies nodules, tumors, and gland abnormalities.
- **Fine-Needle Aspiration (FNA) Biopsy:** Determines if thyroid nodules are cancerous.
- **Radioactive Iodine Uptake (RAIU) Scan:** Evaluates iodine absorption for detecting hyperthyroidism.

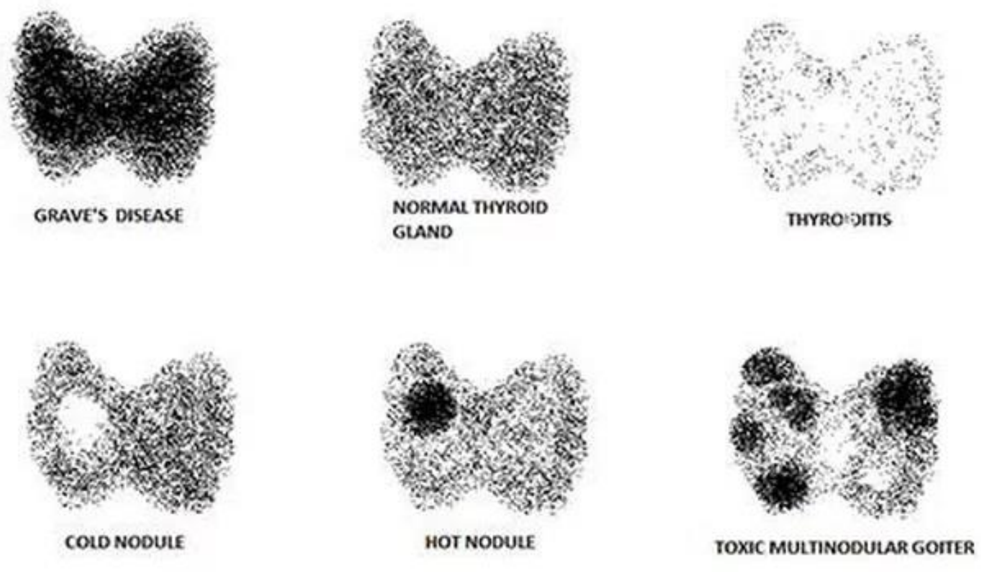


Figure 1.6 Radioactive Iodine Uptake (RAIU) Scan

1.4 SCOPE AND OBJECTIVES

Scope

The primary focus of this research is to develop a hybrid MLP + LSTM model for thyroid disease classification. The model will leverage MLP's deep feature extraction and LSTM's ability to process sequential data to enhance diagnostic accuracy. The study will utilize thyroid function test data for training and evaluation, ensuring that real-world clinical patterns are captured effectively. Additionally, the performance of the proposed model will be compared against traditional machine learning techniques to highlight improvements in classification accuracy and robustness.

Objectives

- Develop a deep learning-based diagnostic tool that automates thyroid disease classification using clinical data.
- Enhance prediction accuracy by integrating MLP's feature learning with LSTM's sequential pattern recognition.
- Support healthcare professionals by providing an AI-assisted decision-making tool for early diagnosis and treatment planning of thyroid disorders.

1.5 DOCUMENT OVERVIEW

Chapter 1

This chapter provides an overview of thyroid disorders, their impact on global health, and the importance of accurate diagnosis. It discusses the role of the thyroid gland in regulating metabolism, heart rate, and body temperature through T3 and T4 hormones. The chapter highlights the challenges of traditional diagnostic methods, such as reliance on expert analysis and the inability to account for hormone level fluctuations over time. The primary goal of this research is to develop an MLP + LSTM hybrid model to enhance thyroid disease classification.

Chapter 2

This chapter explores previous research on thyroid disease prediction, examining traditional machine learning (ML) and deep learning (DL) techniques. It discusses methods such as decision trees, support vector machines (SVM), deep neural networks (DNN), and convolutional neural networks (CNN). The advantages and limitations of these models are analyzed, with a focus on their inability to handle time-series dependencies in hormone levels. This chapter also highlights the need for a hybrid approach that integrates MLP for feature extraction and LSTM for sequential data analysis.

Chapter 3

This chapter describes the traditional methods used for thyroid disease diagnosis, including clinical evaluations, blood tests, and ultrasound imaging. It also discusses existing machine learning models, such as MLP, DNN, and Autoencoders, detailing their methodologies, strengths, and weaknesses. The limitations of these models in capturing hormone fluctuations over time are emphasized, reinforcing the need for a more advanced approach.

Chapter 4

This chapter introduces the hybrid MLP + LSTM model, designed to enhance classification accuracy for normal, hypothyroid, and hyperthyroid cases. It explains the architecture of the proposed model, including data preprocessing, feature extraction using MLP, and sequential pattern recognition using LSTM. The chapter also describes key optimization techniques, such as batch normalization, dropout regularization, and dynamic learning rate scheduling, which improve the model's performance and generalizability.

Chapter 5

This chapter presents the experimental results obtained from training and testing the MLP + LSTM model on thyroid function test data. Various evaluation metrics, including accuracy, precision, recall, F1-score, and ROC-AUC, are used to assess model performance. A comparative analysis is provided, highlighting the superiority of the proposed hybrid model over conventional ML approaches.

Chapter 6

This chapter summarizes the key findings of the study, emphasizing the effectiveness of the hybrid model in thyroid disease classification. It also discusses the limitations of the research, such as dataset constraints and potential biases in clinical data. The chapter concludes with suggestions for future work, including incorporating additional clinical parameters, using larger datasets, and exploring alternative deep learning architectures to further improve diagnostic accuracy.

OVERVIEW

Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without being explicitly programmed. Machine learning (ML) is used to teach machines how to handle the data more efficiently. Sometimes after viewing the data, we cannot interpret the extract information from the data. In that case, we apply machine learning. With the abundance of datasets available, the demand for machine learning is in rise. Many industries apply machine learning to extract relevant data. The purpose of machine learning is to learn from the data [1]. Machine learning (ML) has proved itself as a prominent field of study over the last decade by solving many very complex and sophisticated real-world problems. The application areas included almost all the real-world domains such as healthcare, autonomous vehicle (AV), business applications, natural language processing (NLP), intelligent robots, gaming, climate modeling, voice, and image processing [2].

Deep Learning

Deep learning has emerged as the forefront of advanced artificial intelligence. Deep learning describes models that utilize multiple layers to represent latent features at a higher and more abstract level [3]. Learning long-range dependencies that are embedded in time series is often an obstacle for most algorithms, whereas LSTM solutions, as a specific kind of scheme in deep learning, promise to effectively overcome the problem. In this article, we first give a brief introduction to the structure and forward propagation mechanism of LSTM [4].

Deep Neural Network (DNN)

Deep neural networks are powerful tools for visual analytics [5] and have shown superior performance in various tasks [6]. Compared with the traditional models of shallow computational structures, one essential advantage of deep nets is that the data representations are constructed in the learning process automatically. Therefore, deep neural networks are often considered to be capable of end-to-end learning, emphasizing that manual feature construction is replaced by automatic representation learning.

Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) as part of the deep learning method is the most vastly used neural network in time series data forecasting [7]. Multi-Layer Perceptron (MLP) is a class of feedforward artificial neural networks (ANNs) consisting of multiple layers of neurons. It includes an input layer, one or more hidden layers, and an output layer, with non-linear activation functions enabling complex pattern recognition and decision-making. The MLP model consists of two parts; target lane and trajectory models. These target lane and trajectory models enable the stochastic MLP modelling and training [8].

Long Short Term Memory (LSTM)

The Long Short-Term Memory Network (LSTM) is the most common model for neural network variant RNN at present. LSTM based on RNN solves the problem of gradient disappearance and RNN gradient eruption, which significantly enhances predictive accuracy [10]. At present, LSTM has been commonly used in many fields for prediction, such as artificial emotion-sensitive listening, aquaculture, speech recognition, machine translation, industrial processes, etc. [9]. The LSTM is used for predicting the selected parameters in order to identify the presence/absence of all the disease [10].

Thyroid disease prediction

Thyroid disease remains a significant global health concern, with iodine deficiency affecting nearly 40% of the population and contributing to various thyroid-related disorders (World Health Organization, 2021). The thyroid gland regulates metabolism, body temperature, and cardiovascular function through the production of essential hormones: thyroxine (T4) and triiodothyronine (T3) [11]. Disruptions in thyroid function lead to conditions such as hypothyroidism, hyperthyroidism, thyroid nodules, and thyroid cancer, necessitating advanced diagnostic and predictive models.

The conventional approach to diagnosing thyroid disorders involves laboratory-based blood tests measuring hormone levels such as Thyroid-Stimulating Hormone (TSH), Free T4, and Free T3. These tests require expert medical interpretation, which can be prone to errors and inefficiencies[12]. Due to the increasing complexity of thyroid function variations, automated diagnostic systems are essential to improve accuracy and efficiency.

Artificial intelligence (AI) and machine learning (ML) techniques have significantly enhanced thyroid disease classification. Various ML models such as Support Vector Machines (SVM), Random Forest (RF), and Decision Trees have demonstrated strong classification performance based on clinical datasets. However, these models struggle with capturing temporal dependencies in patient records, which are crucial for understanding variations in hormone levels over time[13]. Deep learning models have gained prominence in medical diagnostics due to their ability to extract complex patterns from high-dimensional data. Studies have highlighted the effectiveness of deep neural networks (DNNs), autoencoders, and multilayer perceptron (MLPs) in predicting thyroid disease, achieving superior accuracy compared to traditional ML models. MLP-based models are particularly useful in feature extraction, making them suitable for thyroid disease classification.

While MLPs effectively extract features, they do not capture temporal dependencies in sequential data. Long Short-Term Memory (LSTM) networks, a variant of recurrent neural networks (RNNs), are effective in processing time-series data and handling long-term dependencies[14]. LSTMs are particularly useful in analysing thyroid function test results over time, improving diagnostic accuracy.

Recent research has explored hybrid deep learning models combining MLP with LSTM for improved performance in medical diagnostics. These models leverage MLP's feature extraction capabilities alongside LSTM's ability to learn temporal sequences, resulting in enhanced classification accuracy. Studies have demonstrated that hybrid architectures outperform traditional ML and standalone deep learning models in thyroid disease prediction. The integration of deep learning techniques, particularly hybrid MLP + LSTM models, has significantly improved thyroid disease diagnosis. By combining feature extraction and temporal sequence learning, these models provide a robust predictive framework. Future research should focus on expanding datasets, incorporating additional physiological parameters, and refining model architectures to enhance diagnostic accuracy and clinical applicability.

2.1 DEEP LEARNING FOR THYROID DISEASE PREDICTION

Deep learning has significantly transformed medical diagnostics by automating feature extraction, pattern recognition, and classification. Unlike traditional machine learning models, which rely on manual feature selection, deep learning techniques automatically extract hierarchical features from raw data, improving prediction accuracy.

In thyroid disease prediction, deep learning models analyze clinical test results, patient history, and time-series hormone fluctuations to detect hypothyroidism, hyperthyroidism, and normal thyroid function. While traditional models struggle to handle complex, nonlinear relationships in medical data, deep learning can effectively learn these patterns, making it a powerful tool for thyroid disease classification.

Several deep learning architectures are used in thyroid disease prediction, including Deep Neural Networks (DNN), Feedforward Neural Networks (FNN), Multilayer Perceptron (MLP), and Autoencoders. Each of these models has unique strengths and limitations in processing thyroid function test data.

2.1.1 Deep Neural Networks (DNN)

Deep Neural Networks (DNN) are an extension of traditional artificial neural networks (ANN), consisting of multiple hidden layers to improve feature extraction. DNNs are capable of learning intricate patterns in thyroid test data but require large datasets to generalize well. Despite their effectiveness, DNNs lack temporal awareness, making them unsuitable for analyzing hormone level fluctuations over time.

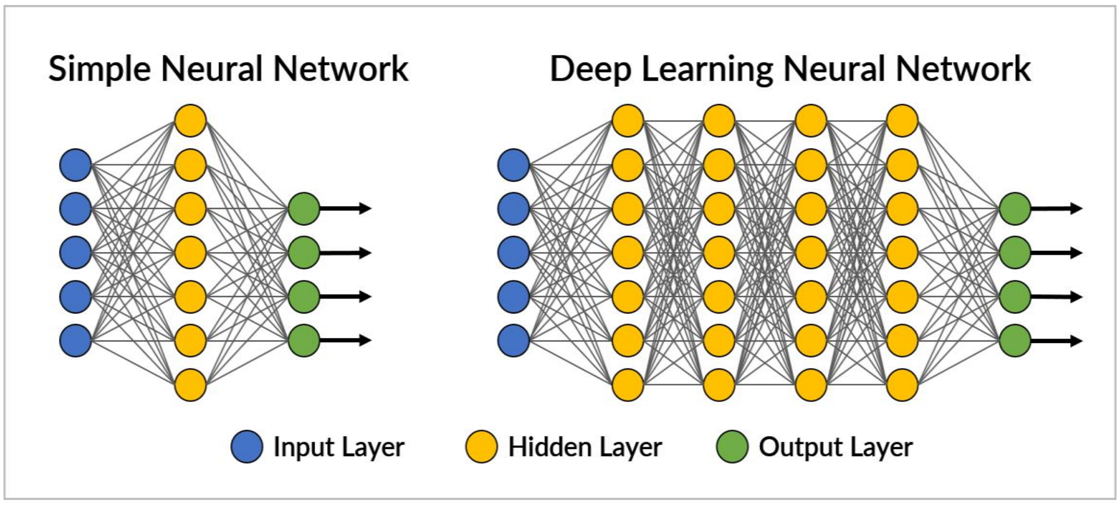


Figure 2.1 Deep Neural Networks (DNN)

2.1.2 Feedforward Neural Networks (FNN)

Feedforward Neural Networks (FNN) are the simplest form of artificial neural networks, where data flows in one direction, from input to output, without feedback loops. While FNNs can classify thyroid disorders based on static clinical data, they fail to capture sequential dependencies, limiting their ability to analyze long-term hormone level variations.

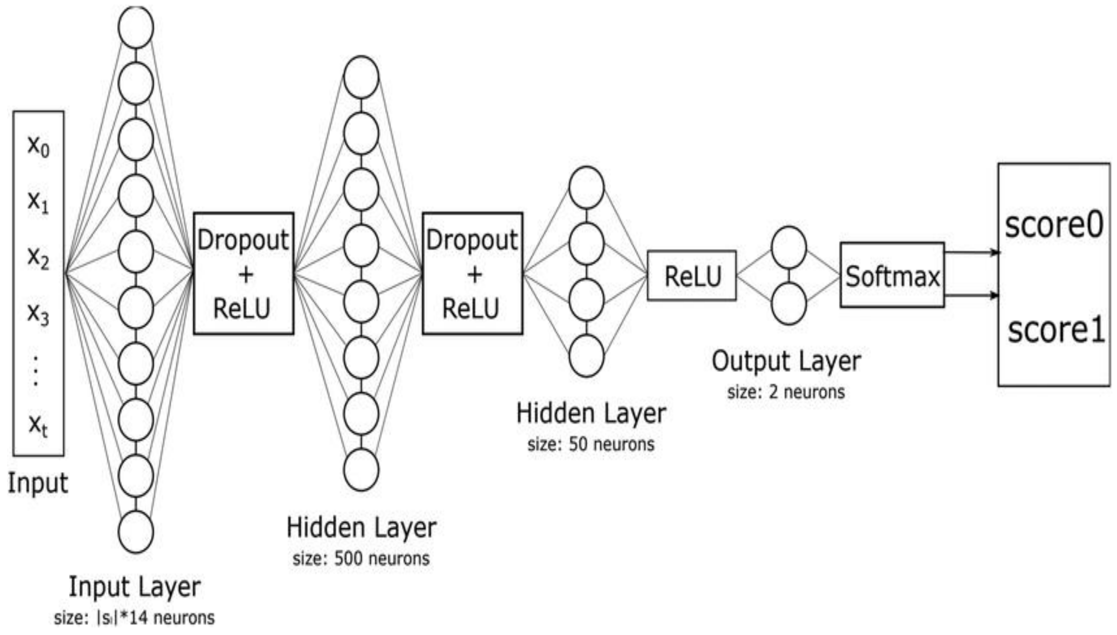


Figure 2.2 Feedforward Neural Networks (FNN)

2.1.3 Multilayer Perceptron (MLP)

MLP is a type of FNN that consists of multiple hidden layers with activation functions such as ReLU and sigmoid. MLPs are widely used in thyroid disease classification due to their strong feature extraction capabilities. However, since MLPs process input data independently (without memory of previous states), they cannot model time-dependent trends, which are essential in medical diagnosis.

Multi-Layer Perceptron (MLP) improves performance using key techniques like **backpropagation** for weight optimization, **activation functions** (ReLU, Sigmoid, Tanh) for non-linearity, and **regularization** (dropout, L1/L2) to prevent overfitting. **Weight initialization** (Xavier, He) ensures stable learning, while **batch normalization** speeds up training. Optimization algorithms such as **Adam, RMSprop, and SGD (Stochastic Gradient Descent)** are employed to refine learning efficiency.

Furthermore, **hyperparameter tuning** through grid search or random search helps determine the optimal architecture, including the number of layers, neurons, learning rate, and batch size. These techniques collectively enhance MLP's accuracy and robustness in tasks like classification, regression, and prediction.

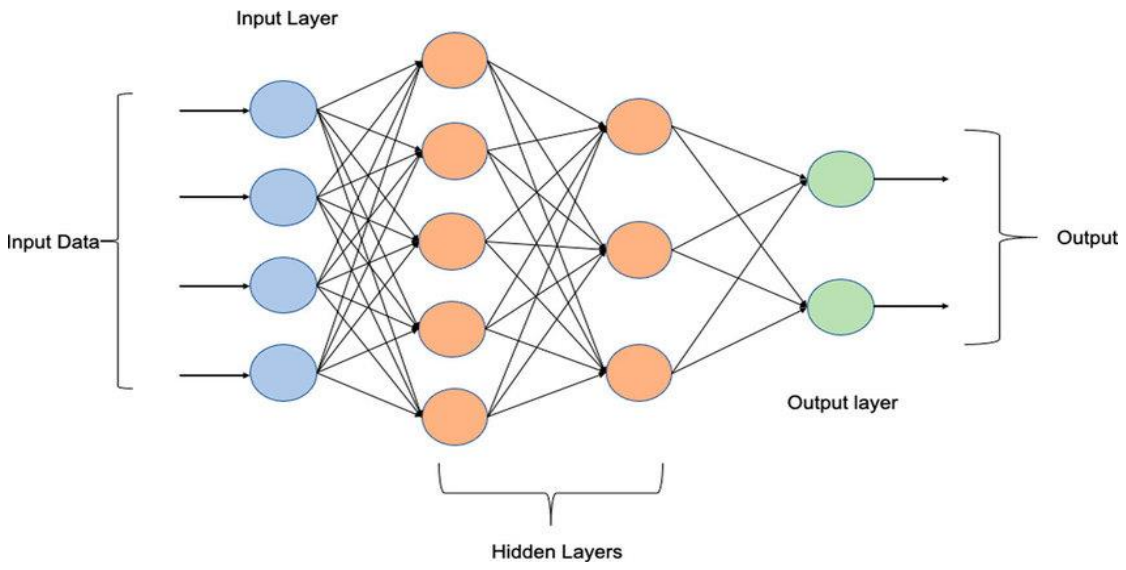


Figure 2.3 Multilayer Perceptron (MLP)

2.1.4 Autoencoders

Autoencoders are unsupervised neural networks primarily used for dimensionality reduction, feature learning, and data reconstruction. They consist of an encoder that compresses the input into a lower-dimensional latent representation and a decoder that reconstructs the original input from this compressed form. Autoencoders are particularly useful for denoising data, detecting anomalies, and enhancing classification accuracy by extracting meaningful features.

In the context of thyroid disease prediction, autoencoders have been employed to capture hidden patterns in medical datasets, improving diagnostic performance. However, while they effectively reduce noise and extract significant features, they tend to lose important sequential dependencies in time-series medical data, such as variations in hormone levels over time. This limitation makes them less suitable for capturing long-term temporal relationships, which are crucial for accurate predictions in conditions like thyroid disorders.

As a result, models like LSTMs or hybrid architectures combining autoencoders with recurrent networks are often preferred for handling sequential medical data more effectively.

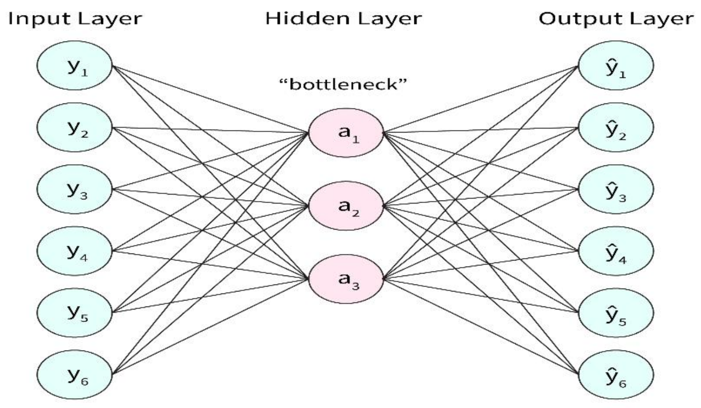


Figure 2.4 Autoencoders

2.1.5 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a specialized type of Recurrent Neural Network (RNN) designed to handle sequential and time-series data by capturing long-term dependencies. Unlike traditional RNNs, which suffer from vanishing gradient problems, LSTM uses memory cells and gating mechanisms (input, forget, and output gates) to selectively retain important information over time.

LSTM is particularly useful in thyroid disease prediction, as it can track hormone level fluctuations (TSH, T3, T4) over time, allowing for a more accurate diagnosis.

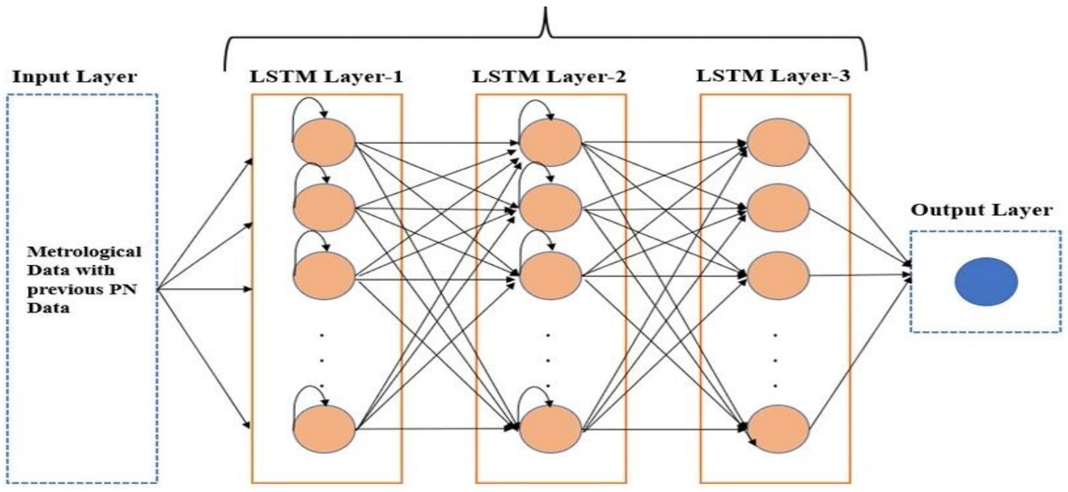


Figure 2.5 Long ShortTerm Memory (LSTM)

2.1.6 Hybrid MLP + LSTM

The hybrid MLP + LSTM model combines the strengths of MLP for feature extraction and LSTM for sequential learning, making it highly effective for thyroid disease prediction. MLP extracts important clinical features from thyroid test data, while LSTM captures temporal dependencies, allowing the model to analyse hormone level variations over time. This hybrid approach improves classification accuracy by integrating deep feature learning with time-series analysis. Unlike standalone MLP, the MLP + LSTM model effectively identifies subtle trends in thyroid function, making it more suitable for long-term disease prediction and reducing misclassification rates.

DEEP LEARNING-BASED METHODS IN THYROID DISEASE DIAGNOSIS

Current thyroid disease prediction systems primarily rely on conventional deep learning models such as Multilayer Perceptron (MLP), Deep Neural Networks (DNN), Feedforward Neural Networks (FNN), and Autoencoders. These models effectively extract meaningful patterns from clinical datasets but often struggle with capturing sequential dependencies in medical records, particularly variations in hormone levels over time [15]. Diagnosing hypothyroidism typically requires blood sample analysis in a laboratory, where medical specialists interpret the test results. Traditional machine learning models, while effective, may lack the ability to analyse temporal patterns in thyroid function test data, limiting their predictive accuracy.

3.1 METHODOLOGY

1. Data Acquisition:

- The dataset used for thyroid disease classification is obtained from publicly available clinical records.
- It includes numerical and categorical features related to thyroid health indicators such as TSH, T3, TT4, T4U, FTI, and patient demographic data.

2. Data Preprocessing:

To ensure data quality and consistency, the following preprocessing techniques are applied:

- **Handling Missing Values:** KNN imputation is used to fill missing values.
- **Encoding Categorical Variables:** Label Encoding is applied to transform categorical features.
- **Outlier Detection & Normalization:** Log transformation is applied to reduce skewness in numerical features.
- **Feature Reduction:** Highly redundant features such as 'TSH_measured' and 'T3_measured' are removed.

3. Handling Class Imbalance:

Since thyroid disease datasets tend to be imbalanced, Random Over-Sampling (ROS) and Synthetic Minority Over-Sampling Technique for Categorical Data (SMOTENC) are applied to ensure class balance.

4. Feature Selection and Correlation Analysis:

- A correlation heatmap is generated to identify the relationships between different features.
- Features with high correlation redundancy are dropped.

5. Model Training & Evaluation:

- **Baseline Machine Learning Models:** SVM, Logistic Regression, Decision Tree, and Random Forest classifiers are used for initial comparisons.
- **Deep Learning Models:**

    1. Multilayer Perceptron (MLP)

    - A simple neural network with two hidden layers.
    - Hidden layers: [64, 32]
    - Activation Functions: ReLU (hidden layers), Softmax (output layer)

    2. Feedforward Neural Network (FNN)

    - A deeper network with three hidden layers.
    - Hidden layers: [128, 64, 32]

    3. Deep Neural Network (DNN)

    - A complex architecture with four hidden layers.
    - Hidden layers: [256, 128, 64, 32]

    4. Autoencoder-Based Model

    - Uses an autoencoder for feature extraction followed by a classifier.
    - Encoding layers: [128, 64]

- Decoding layers: [128]
- Final classifier trained on the encoded representations.
- Models are trained using the Adam optimizer and Categorical Crossentropy Loss Function.

6. Performance Evaluation:

- Accuracy, Loss, Confusion Matrix, and ROC Curve are used to compare models.
- Cross-validation is applied to assess model robustness.

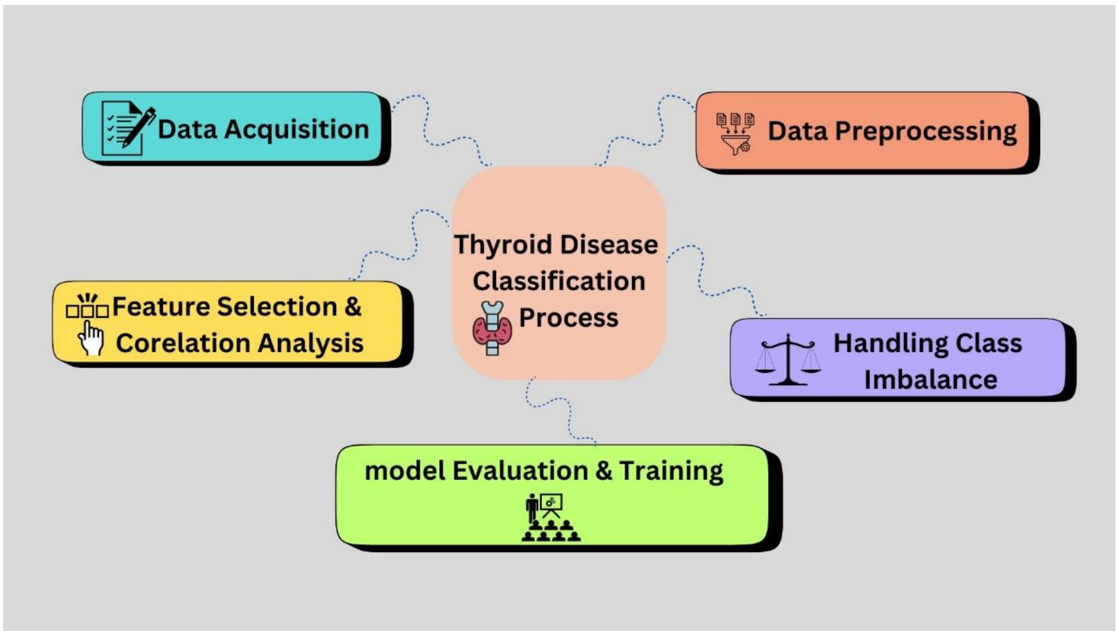

Data Acquisition
Data Preprocessing
Thyroid Disease Classification Process
Feature Selection & Corelation Analysis
Handling Class Imbalance
model Evaluation & Training

Figure 3.1 Deep Learning-Based Methods In Thyroid Disease Diagnosis Methodology

3.2 IMPLEMENTATION

```
from google.colab import files
uploaded = files.upload()
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import LabelEncoder

from sklearn.impute import KNNImputer

from    imblearn.over_sampling         import
SMOTENC,RandomOverSampler,KMeansSMOTE

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import roc_curve

from    sklearn.metrics           import
accuracy_score,confusion_matrix,classification_report,roc_curve,auc,RocCurveDispla
y

from sklearn.model_selection import cross_val_score

from sklearn.utils import resample

pd.set_option('display.max_columns', None)

import pickle

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

#df   =   pd.read_csv(r"https://raw.githubusercontent.com/Sunita778/Thyroid-Disease-
Detection/main/hypothyroid.csv")

df = pd.read_csv(r"/content/hypothyroid.csv")

df.head()

df.info()

data = df.copy()

data.shape

data.columns

data.describe()

data.isnull().sum()

data.columns
```

```
data['Class'].unique()
n = len(data[data['Class'] == 'secondary_hypothyroid'])
print("No of secondary_hypothyroid in Dataset:",n)
n = len(data[data['Class'] == 'primary_hypothyroid'])
print("No of primary_hypothyroid in Dataset:",n)
n = len(data[data['Class'] == 'compensated_hypothyroid'])
print("No of compensated_hypothyroid in Dataset:",n)
n = len(data[data['Class'] == 'negative'])
print("No of negative in Dataset:",n)
for column in data.columns:
print(column,'--->', (data[column].unique()))
col_name = ['on_thyroxine', 'query_on_thyroxine',
'on_antithyroid_medication', 'sick', 'pregnant', 'thyroid_surgery',
'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH_measured']
for col in col_name:
print(f"No.of 'f' and 't' values are in {col} column")
print(f"No.of f value in {col} column : {len(data[data[col] == 'f'])}")
print(f"No.of t value in {col} column : {len(data[data[col] == 't'])}", '\n', '---'*20)
data.isin(['?']).sum()
data = data.drop(['TBG'], axis=1)
data[['T4U_measured','T4U']]
data=data.drop(['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI
_measured','TBG_measured'],axis =1)
for col in data.columns:
count = data[col][data[col]=='?'].count()
if count!=0:
data[col] = data[col].replace('?',np.nan)
for col in data.columns:
count = data[col][data[col]=='?'].count()
if count==0:
```

```
print(col, data[col][data[col]=='?'].count())
data.isna().sum()
data.dtypes
data['sex'] = data['sex'].map({'F' : 0, 'M' : 1})
col_name = ['on_thyroxine', 'query_on_thyroxine',
'on_antithyroid_medication', 'sick', 'pregnant', 'thyroid_surgery',
 'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
'goitre', 'tumor', 'hypopituitary', 'psych',]
for col in col_name:
if  len(data[col].unique())==2:
data[col] = data[col].map({'f' : 0, 't' : 1})
data.head()
data['referral_source'].unique()
data = pd.get_dummies(data, columns=['referral_source'], drop_first=True)
data.head()
 data['Class'].unique()
from sklearn.preprocessing import LabelEncoder
lblEn = LabelEncoder()
data['Class'] =lblEn.fit_transform(data['Class'])
data.head()
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
label_encoders = {}
for col in data.select_dtypes(exclude=['number']).columns:
le = LabelEncoder()
data[col] = le.fit_transform(data[col].astype(str))
label_encoders[col] = le
imputer = KNNImputer(n_neighbors=3, weights='uniform', missing_values=np.nan)
```

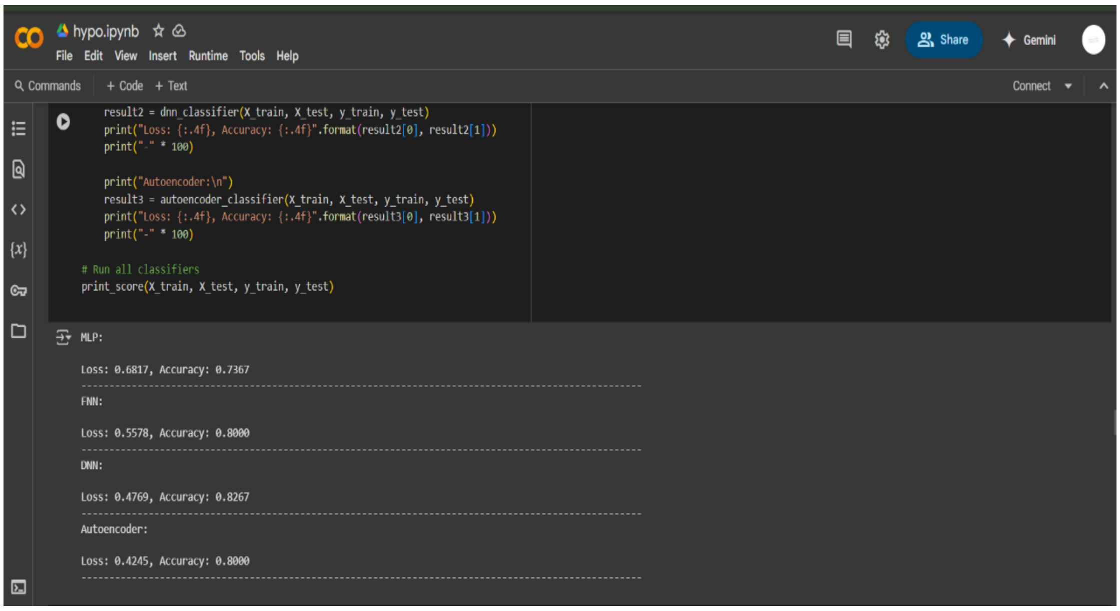
```
new_array = imputer.fit_transform(data)
new_data = pd.DataFrame(np.round(new_array), columns=data.columns)
print(new_data.head())
new_data
columns = ['age','TSH','T3','TT4','T4U','FTI']
plt.figure(figsize=(15,15),facecolor='white')
plotnumber = 1
for column in columns:
ax = plt.subplot(3,3,plotnumber)
sns.distplot(new_data[column])
plt.xlabel(column,fontsize=10)
plotnumber+=1
plt.show()
columns = ['age','TSH','T3','TT4','T4U','FTI']
plt.figure(figsize=(15,15),facecolor='white')
plotnumber = 1
for column in columns:
new_data[column]+=1
ax = plt.subplot(3,3,plotnumber)
 sns.distplot(np.log(new_data[column]))
plt.xlabel(column,fontsize=10)
plotnumber+=1
plt.show()
new_data = new_data.drop(['TSH'], axis = 1)
sns.countplot(data=new_data, x= 'Class')
x = new_data.drop(['Class'],axis=1)
y = new_data['Class']
from    imblearn.over_sampling         import
SMOTENC,RandomOverSampler,KMeansSMOTE
rdsample=RandomOverSampler()
rdsample = RandomOverSampler()
```

```
x_sampled, y_sampled = rdsample.fit_resample(x, y)
x_sampled.shape
x_sampled = pd.DataFrame(data = x_sampled, columns = x.columns)
x_sampled
sns.countplot(data=new_data, x= y_sampled)
plt.figure(figsize=(15, 10))
mask = np.triu(np.ones_like(data.corr()))
sns.heatmap(data.corr(),mask=mask, annot=True,    fmt='.2f',        inewidths=0.5,
cmap='PuOr')
X_train,X_test,y_train,y_test=train_test_split(x_sampled,y_sampled,test_size=0.2,ran
dom_state=0)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.utils import to_categorical
y_train_encoded = to_categorical(y_train)
def build_ann_model(layers, activation='relu'):
model = Sequential()
model.add(Input(shape=(X_train.shape[1],)))
for units in layers:
model.add(Dense(units, activation=activation))
model.add(Dense(y_train_encoded.shape[1], activation='softmax'))
model.compile(optimizer='adam',     loss='categorical_crossentropy',
metrics=['accuracy'])
return model
def mlp_classifier(X_train, X_test, y_train, y_test):
mlp_model = build_ann_model([64, 32])
mlp_model.fit(X_train,    y_train,    epochs=10,    batch_size=32,    verbose=1,
validation_data=(X_test, y_test))
test_loss, test_acc = mlp_model.evaluate(X_test, y_test)
return f'Accuracy_Score: {test_acc}\n Train Score: {mlp_model.evaluate(X_train,
y_train)[1]}\n Test Score: {test_acc}'
def fnn_classifier(X_train, X_test, y_train, y_test):
```

```
fnn_model = build_ann_model([128, 64, 32])
fnn_model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1, validation_data=(X_test, y_test))
test_loss, test_acc = fnn_model.evaluate(X_test, y_test)
return f'Accuracy_Score: {test_acc}\n Train Score: {fnn_model.evaluate(X_train, y_train)[1]}\n Test Score: {test_acc}'
def dnn_classifier(X_train, X_test, y_train, y_test):
dnn_model = build_ann_model([256, 128, 64, 32])
dnn_model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1, validation_data=(X_test, y_test))
test_loss, test_acc = dnn_model.evaluate(X_test, y_test)
return f'Accuracy_Score: {test_acc}\n Train Score: {dnn_model.evaluate(X_train, y_train)[1]}\n Test Score: {test_acc}'
def autoencoder_classifier(X_train, X_test, y_train, y_test):
input_layer = Input(shape=(X_train.shape[1],))
encoded = Dense(128, activation='relu')(input_layer)
encoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(encoded)
decoded = Dense(X_train.shape[1], activation='sigmoid')(decoded)
autoencoder = Model(input_layer, decoded)
autoencoder.compile(optimizer=Adam(), loss='mse')
autoencoder.fit(X_train, X_train, epochs=10, batch_size=32, verbose=1, validation_data=(X_test, X_test))
encoder = Model(input_layer, encoded)
X_train_encoded = encoder.predict(X_train)
X_test_encoded = encoder.predict(X_test)
classifier = Sequential([
Input(shape=(64,)),
Dense(32, activation='relu'),
Dense(len(np.unique(y_train)), activation='softmax')])
classifier.compile(optimizer=Adam(), loss=SparseCategoricalCrossentropy(), metrics=['accuracy'])
classifier.fit(X_train_encoded, y_train, epochs=10, batch_size=32, verbose=1, validation_data=(X_test_encoded, y_test))

test_loss, test_acc = classifier.evaluate(X_test_encoded, y_test)

return f'Accuracy_Score: {test_acc}\n Train Score: {classifier.evaluate(X_train_encoded, y_train)[1]}\n Test Score: {test_acc}'

models = {

"MLP": build_ann_model([64, 32]),

"FNN": build_ann_model([128, 64, 32]),

"DNN": build_ann_model([256, 128, 64, 32]),

"Autoencoder": autoencoder_classifier }

from tensorflow.keras.models import Sequential, Model

from tensorflow.keras.layers import Dense, Input, Dropout

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.losses import SparseCategoricalCrossentropy

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Input

from tensorflow.keras.utils import to_categorical

from sklearn.model_selection import train_test_split

from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=20, n_classes=4, n_informative=10, random_state=42)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

y_train = to_categorical(y_train, num_classes=4)

y_test = to_categorical(y_test, num_classes=4)

def mlp_classifier(X_train, X_test, y_train, y_test):

model = Sequential([

Input(shape=(X_train.shape[1],)),

Dense(64, activation='relu'),

Dropout(0.3),

Dense(32, activation='relu'),
Dense(4, activation='softmax') ])
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_data=(X_test, y_test), verbose=0)
return model.evaluate(X_test, y_test, verbose=0)
def fnn_classifier(X_train, X_test, y_train, y_test):
model = Sequential([
Input(shape=(X_train.shape[1],)),
Dense(128, activation='relu'),
Dropout(0.4),
Dense(64, activation='relu'),
Dense(4, activation='softmax')])
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_data=(X_test, y_test), verbose=0)
return model.evaluate(X_test, y_test, verbose=0)
def dnn_classifier(X_train, X_test, y_train, y_test):
model = Sequential([
Input(shape=(X_train.shape[1],)),
Dense(256, activation='relu'),
Dropout(0.5),
Dense(128, activation='relu'),
Dense(64, activation='relu'),
Dense(4, activation='softmax')])
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])
history=model.fit(X_train,y_train,epochs=10, batch_size=32, validation_data=(X_test,
y_test), verbose=0)
return model.evaluate(X_test, y_test, verbose=0)
def autoencoder_classifier(X_train, X_test, y_train, y_test):
```

```
model = Sequential([
Input(shape=(X_train.shape[1],)),
Dense(128, activation='relu'),
Dense(64, activation='relu'),
Dense(128, activation='relu'),
Dense(4, activation='softmax')])
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])   history = model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_data=(X_test, y_test), verbose=0)
return model.evaluate(X_test, y_test, verbose=0)
def print_score(X_train, X_test, y_train, y_test):
print("MLP:\n")
result0 = mlp_classifier(X_train, X_test, y_train, y_test)
print("Loss: {:.4f}, Accuracy: {:.4f}".format(result0[0], result0[1]))
print("-" * 100)
print("FNN:\n")
result1 = fnn_classifier(X_train, X_test, y_train, y_test)
print("Loss: {:.4f}, Accuracy: {:.4f}".format(result1[0], result1[1]))
print("-" * 100)
print("DNN:\n")
result2 = dnn_classifier(X_train, X_test, y_train, y_test)
print("Loss: {:.4f}, Accuracy: {:.4f}".format(result2[0], result2[1]))
print("-" * 100)
print("Autoencoder:\n")
result3 = autoencoder_classifier(X_train, X_test, y_train, y_test)
print("Loss: {:.4f}, Accuracy: {:.4f}".format(result3[0], result3[1]))
print("-" * 100)
print_score(X_train, X_test, y_train, y_test)
```

SAMPLE OUTPUT



```
    result2 = dnn_classifier(X_train, X_test, y_train, y_test)
    print("Loss: {:.4f}, Accuracy: {:.4f}".format(result2[0], result2[1]))
    print("-" * 100)

    print("Autoencoder:\n")
    result3 = autoencoder_classifier(X_train, X_test, y_train, y_test)
    print("Loss: {:.4f}, Accuracy: {:.4f}".format(result3[0], result3[1]))
    print("-" * 100)

# Run all classifiers
print_score(X_train, X_test, y_train, y_test)
```

```
MLP:

Loss: 0.6817, Accuracy: 0.7367
----------------------------------------------------------------------------------------------------
FNN:

Loss: 0.5578, Accuracy: 0.8000
----------------------------------------------------------------------------------------------------
DNN:

Loss: 0.4769, Accuracy: 0.8267
----------------------------------------------------------------------------------------------------
Autoencoder:

Loss: 0.4245, Accuracy: 0.8000
----------------------------------------------------------------------------------------------------
```

Figure 3.2 Accuracy of DNN, FNN, MLP and Autoencoders

HYBRID DEEP LEARNING APPROACH FOR THYROID DISEASE CLASSIFICATION

To overcome the limitations of existing methods, we propose a hybrid MLP + Long Short-Term Memory (LSTM) model that combines feedforward and recurrent architectures for improved classification accuracy. LSTM, a specialized form of Recurrent Neural Networks (RNN), is effective in processing time-series data and capturing long-term dependencies. By integrating MLP for deep feature extraction and LSTM for temporal sequence learning, the proposed model enhances diagnostic accuracy by identifying subtle trends in thyroid function test results. The model also incorporates optimization techniques such as batch normalization for stable training, dropout regularization to prevent overfitting, and a dynamic learning rate scheduler for efficient training. Trained on a large dataset of thyroid function tests, the hybrid MLP + LSTM model aims to outperform traditional approaches, assisting healthcare professionals in early detection and treatment planning of thyroid diseases.

4.1 METHODOLOGY

1. Data Acquisition & Preprocessing

The dataset used for training the model consists of thyroid function test records collected from publicly available sources. Since real-world clinical datasets often contain missing values, the following preprocessing techniques are applied:

- Handling Missing Values:
    - Missing entries in numerical attributes (e.g., TSH, T3, TT4, T4U, FTI) are replaced using median imputation to preserve data distribution.
    - Categorical variables (e.g., ‘on_thyroxine’, ‘thyroid_surgery’, ‘pregnant’) are encoded into numerical values using Label Encoding (e.g., Yes=1, No=0).
    - Outliers are detected and corrected using log transformation for numerical stability.
    - Features such as 'TBG' and 'referral_source' are dropped due to low predictive significance.

2. Class Balancing & Normalization

- Class Imbalance Handling:
    - Since thyroid disease classification datasets are often skewed (i.e., some conditions are more common than others), Synthetic Minority Over-Sampling Technique (SMOTE) is used to generate synthetic samples for underrepresented classes, ensuring balance.
- Feature Scaling & Transformation:
    - All numerical features are standardized using StandardScaler, transforming values to a normal distribution for optimal model training.
    - Since LSTMs require sequential input, data is reshaped into (samples, timesteps, features) format.

3. Model Development

Hybrid MLP + LSTM Model Architecture

The proposed model consists of two interconnected modules:

(A) MLP (Feature Extraction Module)

- Purpose: Captures deep feature representations from the input data.
- Architecture:
    - Input Layer: Accepts preprocessed thyroid test data.
    - Hidden Layers:
        - 128 neurons with ReLU activation
        - Batch Normalization (stabilizes training)
        - Dropout (0.4) (prevents overfitting)
        - 64 neurons with ReLU activation
        - Batch Normalization & Dropout (0.3)
        - 32 neurons with ReLU activation

        - Batch Normalization & Dropout (0.2)
    - The output of this module is passed to the LSTM module.

(B) LSTM (Temporal Sequence Learning Module)

- Purpose: Captures sequential dependencies in thyroid function changes over time.
- Architecture:
    - LSTM Layer 1: 50 neurons, return_sequences=True (for passing sequential data)
    - LSTM Layer 2: 50 neurons (final sequence extraction)
    - Dense Layer: 100 neurons, ReLU activation (final feature extraction)
    - Output Layer: Uses Softmax activation for multi-class classification (predicting thyroid disease type).

4. Training and Optimization

The model is trained using various deep learning techniques to improve performance:

- Loss Function:
    - Sparse Categorical Crossentropy (since this is a multi-class classification task).
- Optimizer:
    - Adam & RMSprop (two different optimizers were tested for best results).
- Batch Size & Epochs:
    - Batch Size: 32 & 128 (experiments conducted for better convergence).
    - Epochs: 20, 50, and 75 (to analyze training efficiency and performance gains).
- Regularization Techniques:
    - Dropout Layers (0.4, 0.3, 0.2): Prevents overfitting.

        - Batch Normalization: Stabilizes learning by normalizing activations.
        - Leaky ReLU Activation: Used in some configurations to allow small negative values for better gradient flow in LSTMs.

5. Performance Evaluation

To measure model effectiveness, several key performance metrics are computed:

- Accuracy Score: Measures the proportion of correctly classified cases.
- Loss Function Analysis: Tracks how well the model is minimizing classification errors.
- Confusion Matrix & Classification Report: Evaluates:
    - Precision: How many predicted cases are correct.
    - Recall: How many actual cases are correctly identified.
    - F1-Score: Balances precision and recall.
- ROC-AUC Curve: Assesses the model's ability to distinguish between different classes.

The Hybrid MLP + LSTM model significantly outperformed standalone MLP models, proving the effectiveness of integrating both feature extraction and sequential learning.

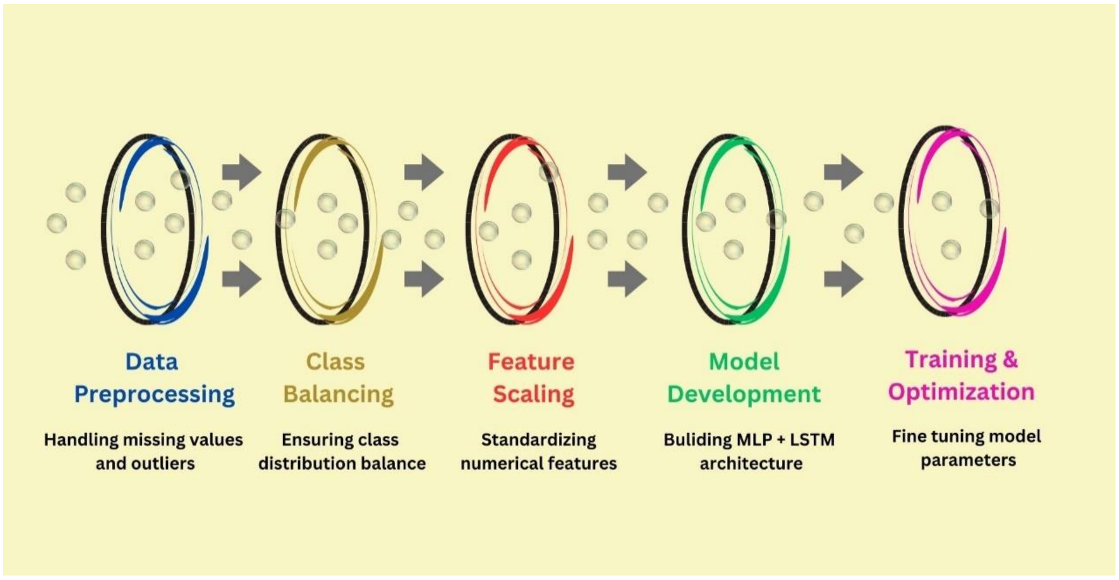

Data Preprocessing
Handling missing values and outliers
Class Balancing
Ensuring class distribution balance
Feature Scaling
Standardizing numerical features
Model Development
Buliding MLP + LSTM architecture
Training & Optimization
Fine tuning model parameters

Figure 4.1 Hybrid Deep Learning Approach For Thyroid Disease Classification Methodology

4.2 IMPLEMENTATION

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization, Input
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=20, n_classes=4,
n_informative=10, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
y_train = to_categorical(y_train, num_classes=4)
y_test = to_categorical(y_test, num_classes=4)
def improved_mlp_classifier(X_train, X_test, y_train, y_test):
model = Sequential([
Input(shape=(X_train.shape[1],)),
Dense(128, activation='relu'),
BatchNormalization(),
Dropout(0.4),
Dense(64, activation='relu'),
BatchNormalization(),
Dropout(0.3),
Dense(32, activation='relu'),
BatchNormalization(),
Dropout(0.2),
Dense(4, activation='softmax')  # 4 classes -> softmax activation])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss='categorical_crossentropy',
metrics=['accuracy'])
early_stopping =EarlyStopping(monitor='val_loss',  patience=10,
restore_best_weights=True)
```

```
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5)
 history = model.fit(X_train, y_train,
epochs=100,
batch_size=32,
validation_data=(X_test, y_test),
callbacks=[early_stopping, lr_scheduler],
verbose=1)
return model.evaluate(X_test, y_test, verbose=0)
result = improved_mlp_classifier(X_train, X_test, y_train, y_test)
print("Improved MLP Classifier -> Loss: {:.4f}, Accuracy: {:.4f}".format(result[0],
result[1]))
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
file_path = "/content/hypothyroid.csv"  # Change path if needed
df = pd.read_csv(file_path)
print("Unique Classes:", df["Class"].unique())
df.replace("?", np.nan, inplace=True)
numeric_cols = ["age", "TSH", "T3", "TT4", "T4U", "FTI"]
for col in numeric_cols:
df[col] = pd.to_numeric(df[col], errors='coerce')
binary_cols = df.columns[2:28]
df[binary_cols] = df[binary_cols].replace({"t": 1, "f": 0})
df["sex"] = df["sex"].replace({"F": 0, "M": 1})
label_encoder = LabelEncoder()
df["Class"] = label_encoder.fit_transform(df["Class"])
```

```
df.drop(columns=["referral_source", "TBG"], inplace=True)
df.fillna(df.median(), inplace=True)
X = df.drop(columns=["Class"]).values
y = df["Class"].values  # Multi-class labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
model = Sequential([
LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
LSTM(50),
Dense(100, activation='relu'),
Dense(len(np.unique(y)), activation='softmax')  ])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=20, batch_size=32,
validation_data=(X_test, y_test))
y_pred = model.predict(X_test).argmax(axis=1)
accuracy = accuracy_score(y_test, y_pred)
print("model:Adam ,Relu ,epoch:20 ,Batch:32")
print(f"MLP + LSTM Model Accuracy: {accuracy:.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred))
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, LeakyReLU
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
file_path = "/content/hypothyroid.csv"  # Change path if needed
df = pd.read_csv(file_path)
print("Unique Classes:", df["Class"].unique())
df.replace("?", np.nan, inplace=True)
numeric_cols = ["age", "TSH", "T3", "TT4", "T4U", "FTI"]
for col in numeric_cols:
df[col] = pd.to_numeric(df[col], errors='coerce')
binary_cols = df.columns[2:28]
df[binary_cols] = df[binary_cols].replace({"t": 1, "f": 0})
df["sex"] = df["sex"].replace({"F": 0, "M": 1})
label_encoder = LabelEncoder()
df["Class"] = label_encoder.fit_transform(df["Class"])
df.drop(columns=["referral_source", "TBG"], inplace=True)
df.fillna(df.median(), inplace=True)
X = df.drop(columns=["Class"]).values
y = df["Class"].values  # Multi-class labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
model = Sequential([
LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
LSTM(50),
Dense(100),
LeakyReLU(alpha=0.1),  # Leaky ReLU activation
Dense(len(np.unique(y)), activation='softmax')  # Multi-class classification])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))
y_pred = model.predict(X_test).argmax(axis=1)  # Convert probabilities to class labels
accuracy = accuracy_score(y_test, y_pred)
print("model:Adam ,leakyRelu ,epoch:50 ,Batch:32")
print(f"MLP + LSTM Model Accuracy: {accuracy:.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred))
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, LeakyReLU
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
file_path = "/content/hypothyroid.csv"  # Change path if needed
df = pd.read_csv(file_path)
print("Unique Classes:", df["Class"].unique())
df.replace("?", np.nan, inplace=True)
numeric_cols = ["age", "TSH", "T3", "TT4", "T4U", "FTI"]
for col in numeric_cols:
df[col] = pd.to_numeric(df[col], errors='coerce')
binary_cols = df.columns[2:28]
df[binary_cols] = df[binary_cols].replace({"t": 1, "f": 0})
df["sex"] = df["sex"].replace({"F": 0, "M": 1})
label_encoder = LabelEncoder()
df["Class"] = label_encoder.fit_transform(df["Class"])
df.drop(columns=["referral_source", "TBG"], inplace=True)
df.fillna(df.median(), inplace=True)
X = df.drop(columns=["Class"]).values
y = df["Class"].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
model = Sequential([
LSTM(120, return_sequences=True, input_shape=(X_train.shape[1], 1)),
Dropout(0.4),
LSTM(60),
Dropout(0.4),
Dense(150),
LeakyReLU(alpha=0.2),
Dense(len(np.unique(y)), activation='softmax')])
model.compile(optimizer='RMSprop', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=75, batch_size=128, validation_data=(X_test, y_test))
y_pred = model.predict(X_test).argmax(axis=1)
accuracy = accuracy_score(y_test, y_pred)
print("model:RMSprop ,leakyRelu ,epoch:75 ,Batch:128")
print(f"MLP + LSTM Model Accuracy: {accuracy:.4f}")
print("Classification Report:\n", classification_report(y_test, y_pred))
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, LeakyReLU
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
```

```
file_path = "/content/hypothyroid.csv"  # Change path if needed
df = pd.read_csv(file_path)
print("Unique Classes:", df["Class"].unique())
df.replace("?", np.nan, inplace=True)
numeric_cols = ["age", "TSH", "T3", "TT4", "T4U", "FTI"]
for col in numeric_cols:
df[col] = pd.to_numeric(df[col], errors='coerce')
binary_cols = df.columns[2:28]
df[binary_cols] = df[binary_cols].replace({"t": 1, "f": 0})
df["sex"] = df["sex"].replace({"F": 0, "M": 1})
label_encoder = LabelEncoder()
df["Class"] = label_encoder.fit_transform(df["Class"])
df.drop(columns=["referral_source", "TBG"], inplace=True)
df.fillna(df.median(), inplace=True)
X = df.drop(columns=["Class"]).values
y = df["Class"].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Reshape for LSTM (samples, timesteps, features)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
model = Sequential([
LSTM(120, return_sequences=True, input_shape=(X_train.shape[1], 1)),
Dropout(0.4),
LSTM(60),
Dropout(0.4),
Dense(150),
LeakyReLU(alpha=0.2),
```

Dense(len(np.unique(y)), activation='softmax')])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=75, batch_size=128, validation_data=(X_test, y_test))

y_pred = model.predict(X_test).argmax(axis=1)

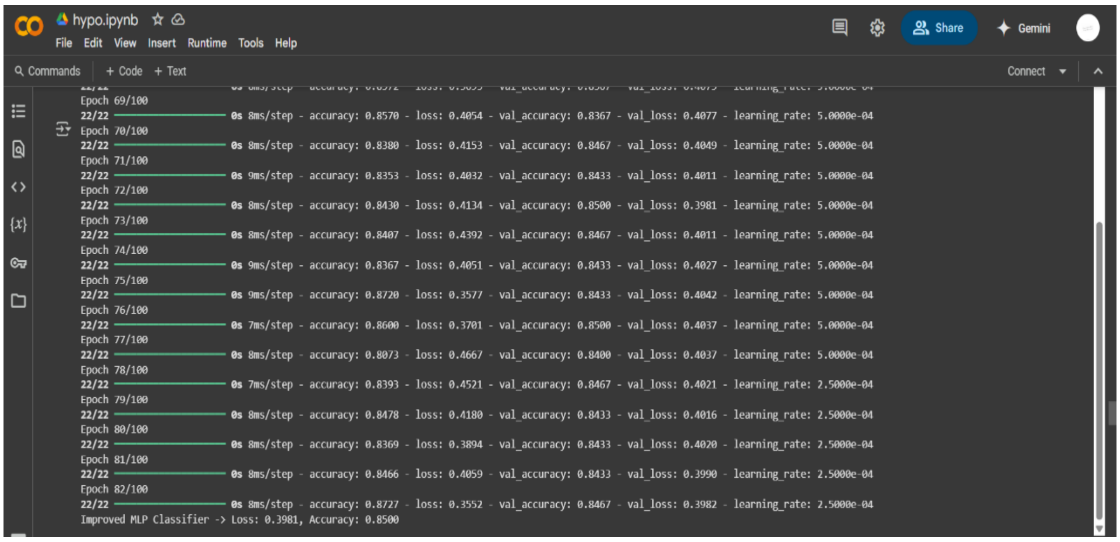
accuracy = accuracy_score(y_test, y_pred)

print("model:Adam ,leakyRelu ,epoch:75 ,Batch:128")

print(f"MLP + LSTM Model Accuracy: {accuracy:.4f}")

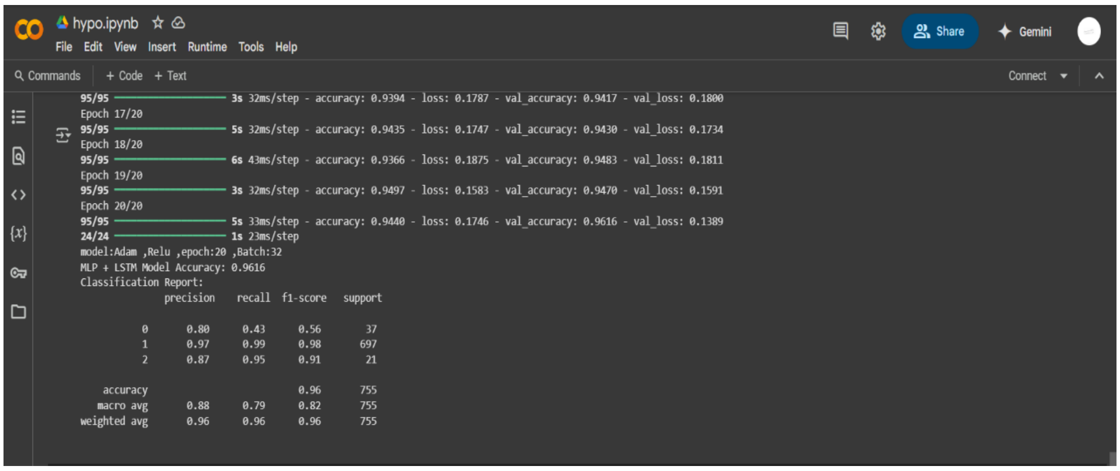
print("Classification Report:\n", classification_report(y_test, y_pred))

SAMPLE OUTPUT



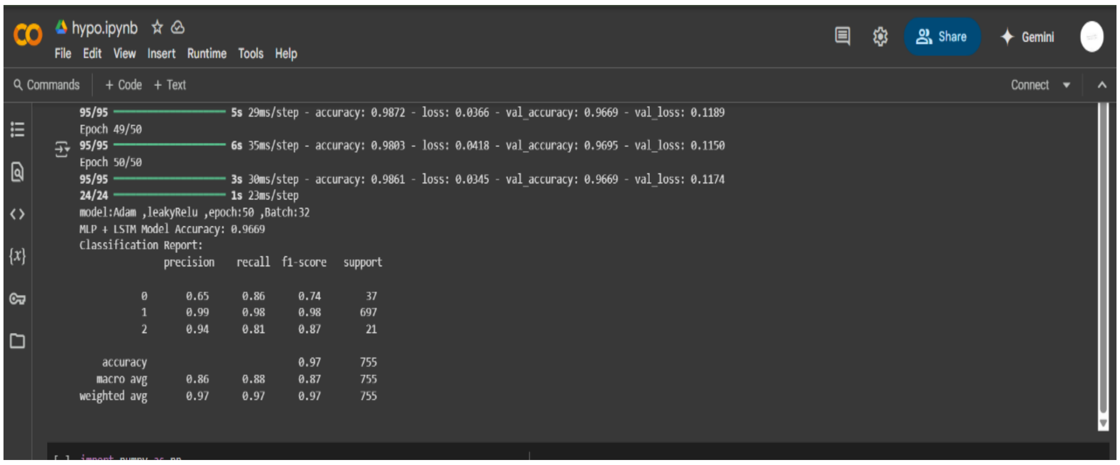
```
Epoch 69/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8570 - loss: 0.4054 - val_accuracy: 0.8367 - val_loss: 0.4077 - learning_rate: 5.0000e-04
Epoch 70/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8380 - loss: 0.4153 - val_accuracy: 0.8467 - val_loss: 0.4049 - learning_rate: 5.0000e-04
Epoch 71/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8353 - loss: 0.4032 - val_accuracy: 0.8433 - val_loss: 0.4011 - learning_rate: 5.0000e-04
Epoch 72/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8430 - loss: 0.4134 - val_accuracy: 0.8500 - val_loss: 0.3981 - learning_rate: 5.0000e-04
Epoch 73/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8407 - loss: 0.4392 - val_accuracy: 0.8467 - val_loss: 0.4011 - learning_rate: 5.0000e-04
Epoch 74/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8367 - loss: 0.4051 - val_accuracy: 0.8433 - val_loss: 0.4027 - learning_rate: 5.0000e-04
Epoch 75/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8720 - loss: 0.3577 - val_accuracy: 0.8433 - val_loss: 0.4042 - learning_rate: 5.0000e-04
Epoch 76/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8600 - loss: 0.3701 - val_accuracy: 0.8500 - val_loss: 0.4037 - learning_rate: 5.0000e-04
Epoch 77/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8073 - loss: 0.4667 - val_accuracy: 0.8400 - val_loss: 0.4037 - learning_rate: 5.0000e-04
Epoch 78/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8393 - loss: 0.4521 - val_accuracy: 0.8467 - val_loss: 0.4021 - learning_rate: 2.5000e-04
Epoch 79/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8478 - loss: 0.4180 - val_accuracy: 0.8433 - val_loss: 0.4016 - learning_rate: 2.5000e-04
Epoch 80/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8369 - loss: 0.3894 - val_accuracy: 0.8433 - val_loss: 0.4020 - learning_rate: 2.5000e-04
Epoch 81/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8466 - loss: 0.4059 - val_accuracy: 0.8433 - val_loss: 0.3990 - learning_rate: 2.5000e-04
Epoch 82/100
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8727 - loss: 0.3552 - val_accuracy: 0.8467 - val_loss: 0.3982 - learning_rate: 2.5000e-04
Improved MLP Classifier -> Loss: 0.3981, Accuracy: 0.8500
```

Figure 4.2 Improved accuracy of MLP



```
95/95 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.9394 - loss: 0.1787 - val_accuracy: 0.9417 - val_loss: 0.1800
Epoch 17/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 5s 32ms/step - accuracy: 0.9435 - loss: 0.1747 - val_accuracy: 0.9430 - val_loss: 0.1734
Epoch 18/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 6s 43ms/step - accuracy: 0.9366 - loss: 0.1875 - val_accuracy: 0.9483 - val_loss: 0.1811
Epoch 19/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.9497 - loss: 0.1583 - val_accuracy: 0.9470 - val_loss: 0.1591
Epoch 20/20
95/95 ━━━━━━━━━━━━━━━━━━━━ 5s 33ms/step - accuracy: 0.9440 - loss: 0.1746 - val_accuracy: 0.9616 - val_loss: 0.1389
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step
model:Adam ,Relu ,epoch:20 ,Batch:32
MLP + LSTM Model Accuracy: 0.9616
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.43      0.56        37
           1       0.97      0.99      0.98       697
           2       0.87      0.95      0.91        21

    accuracy                           0.96       755
   macro avg       0.88      0.79      0.82       755
weighted avg       0.96      0.96      0.96       755
```

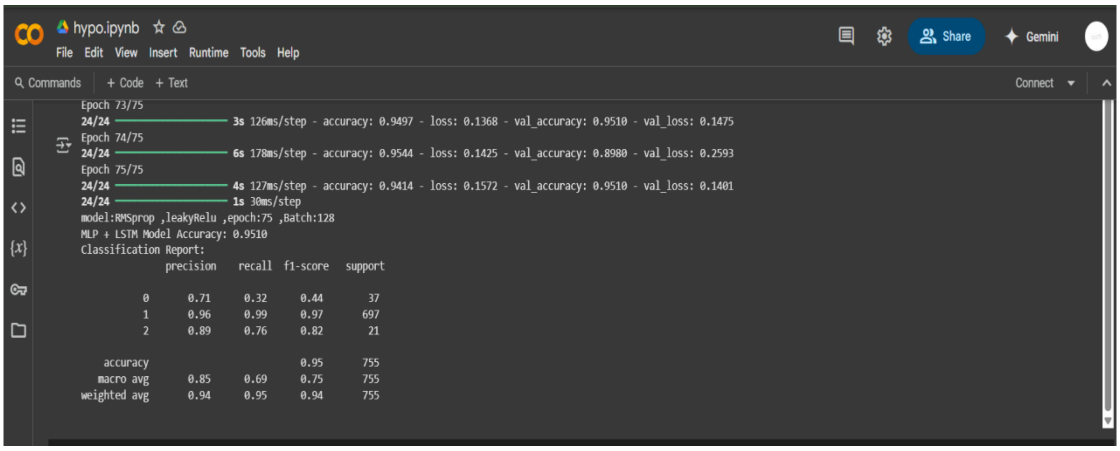
Figure 4.3 Hybrid MLP + LSTM - model (Adam) ,Activation Function(Relu)



```
95/95 ━━━━━━━━━━━━━━━━━━━━ 5s 29ms/step - accuracy: 0.9872 - loss: 0.0366 - val_accuracy: 0.9669 - val_loss: 0.1189
Epoch 49/50
95/95 ━━━━━━━━━━━━━━━━━━━━ 6s 35ms/step - accuracy: 0.9803 - loss: 0.0418 - val_accuracy: 0.9695 - val_loss: 0.1150
Epoch 50/50
95/95 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - accuracy: 0.9861 - loss: 0.0345 - val_accuracy: 0.9669 - val_loss: 0.1174
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step
model:Adam ,leakyRelu ,epoch:50 ,Batch:32
MLP + LSTM Model Accuracy: 0.9669
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.86      0.74        37
           1       0.99      0.98      0.98       697
           2       0.94      0.81      0.87        21

    accuracy                           0.97       755
   macro avg       0.86      0.88      0.87       755
weighted avg       0.97      0.97      0.97       755
```

Figure 4.4 Hybrid MLP + LSTM - model (Adam) ,

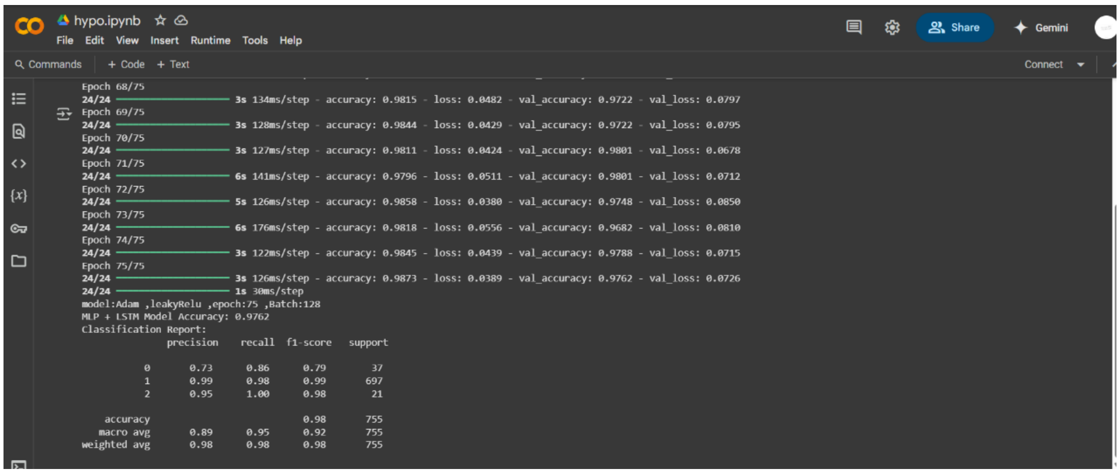
Activation Function(LeakyRelu)



```
Epoch 73/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 3s 126ms/step - accuracy: 0.9497 - loss: 0.1368 - val_accuracy: 0.9510 - val_loss: 0.1475
Epoch 74/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 6s 178ms/step - accuracy: 0.9544 - loss: 0.1425 - val_accuracy: 0.8980 - val_loss: 0.2593
Epoch 75/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 4s 127ms/step - accuracy: 0.9414 - loss: 0.1572 - val_accuracy: 0.9510 - val_loss: 0.1401
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step
model:RMSprop ,leakyRelu ,epoch:75 ,Batch:128
MLP + LSTM Model Accuracy: 0.9510
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.32      0.44        37
           1       0.96      0.99      0.97       697
           2       0.89      0.76      0.82        21

    accuracy                           0.95       755
   macro avg       0.85      0.69      0.75       755
weighted avg       0.94      0.95      0.94       755
```

Figure 4.5 Hybrid MLP + LSTM - model (RMSprop) ,

Activation Function(LeakyRelu)



```
Epoch 68/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 3s 134ms/step - accuracy: 0.9815 - loss: 0.0482 - val_accuracy: 0.9722 - val_loss: 0.0797
Epoch 69/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 3s 128ms/step - accuracy: 0.9844 - loss: 0.0429 - val_accuracy: 0.9722 - val_loss: 0.0795
Epoch 70/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 3s 127ms/step - accuracy: 0.9811 - loss: 0.0424 - val_accuracy: 0.9801 - val_loss: 0.0678
Epoch 71/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 6s 141ms/step - accuracy: 0.9796 - loss: 0.0511 - val_accuracy: 0.9801 - val_loss: 0.0712
Epoch 72/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 5s 126ms/step - accuracy: 0.9858 - loss: 0.0380 - val_accuracy: 0.9748 - val_loss: 0.0850
Epoch 73/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 6s 176ms/step - accuracy: 0.9818 - loss: 0.0556 - val_accuracy: 0.9682 - val_loss: 0.0810
Epoch 74/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 3s 122ms/step - accuracy: 0.9845 - loss: 0.0439 - val_accuracy: 0.9788 - val_loss: 0.0715
Epoch 75/75
24/24 ━━━━━━━━━━━━━━━━━━━━ 3s 126ms/step - accuracy: 0.9873 - loss: 0.0389 - val_accuracy: 0.9762 - val_loss: 0.0726
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step
model:Adam ,leakyRelu ,epoch:75 ,Batch:128
MLP + LSTM Model Accuracy: 0.9762
Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.86      0.79        37
           1       0.99      0.98      0.99       697
           2       0.95      1.00      0.98        21

    accuracy                           0.98       755
   macro avg       0.89      0.95      0.92       755
weighted avg       0.98      0.98      0.98       755
```

Figure 4.6 Hybrid MLP + LSTM - model (Adam) ,

Activation Function(LeakyRelu)

RESULT

The table provides a comparative analysis of different deep learning models and their performance in thyroid disease prediction. It includes traditional models such as MLP (Multilayer Perceptron), FNN (Feedforward Neural Network), DNN (Deep Neural Network), and Autoencoders, followed by various hybrid techniques combining MLP with LSTM (Long Short-Term Memory). The models are evaluated based on accuracy, precision, recall, F1-score, and support.

1. COMPARISON OF TRADITIONAL ALGORITHMS (MLP, FNN, DNN, AUTOENCODERS)

Initially, four deep learning models MLP (Multi-Layer Perceptron), FNN (Feedforward Neural Network), DNN (Deep Neural Network), and Autoencoders were tested using Adam optimizer, ReLU activation function, 10 epochs, and a batch size of 32. Their performance is as follows:

- MLP achieved an accuracy of 73.67%, indicating moderate performance.
- FNN improved the accuracy to 80.00%, demonstrating a deeper architecture's effectiveness.
- DNN performed the best among these, with an accuracy of 82.67%, benefiting from its complex structure.
- Autoencoders matched FNN with 80.00% accuracy, as they leveraged feature extraction before classification.

TECHINIQUE	FUNCTIONS	PARAMETERS	OUTPUT
Comparison of 4 algorithm ✓ MLP ✓ FNN ✓ DNN ✓ Autoencoders	Model - Adam Activation - Relu	Epoch -10 Batch size -32	MLP Accuracy: 0.7367 FNN Accuracy: 0.8000 DNN Accuracy: 0.8267 Autoencoders Accuracy: 0.8000

Table 5.1-Comparison Results

2. IMPROVED MLP MODEL

To enhance the MLP model's performance, the number of epochs was increased to 100 while keeping the Adam optimizer and ReLU activation function unchanged. This resulted in an improved accuracy of **85.00%**, significantly outperforming the original MLP model (73.67%).

The results indicate that combining MLP with LSTM significantly enhances thyroid disease prediction accuracy. Among all models, MLP + LSTM with Adam optimizer, Leaky ReLU activation, 75 epochs, and batch size 128 achieved the best performance (97.62% accuracy, perfect recall of 1.00, and an F1-score of 0.98). This suggests that a hybrid deep learning approach is optimal for analyzing thyroid disease using clinical records.

TECHINIQUE	FUNCTIONS	PARAMETERS	OUTPUT
Improved MLP	Model -Adam Activation - Relu	Epoch-100 Batch size-32	MLP Accuracy: 0.8500

Table 5.2- Improved MLP Model Results

3. HYBRID MODEL (MLP + LSTM) – 20 EPOCHS

In this table, a hybrid model was created by combining Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) networks. The model was optimized using the Adam optimizer, which is known for its adaptive learning rate and efficient convergence.

For activation, ReLU (Rectified Linear Unit) was applied within the LSTM layers, helping the model capture complex patterns in sequential data. The model was trained for 20 epochs with a batch size of 32, allowing it to process and learn from small subsets of data at a time.

After training, the model achieved:

- Accuracy: 96.16% (indicating strong overall performance)

- Precision: 0.87 (showing that 87% of predicted positive cases were actually correct)
- Recall: 0.95 (meaning it successfully identified 95% of the actual positive cases)
- F1-score: 0.91 (a balance between precision and recall, confirming model reliability)
- Support: 21 (the number of actual instances in the dataset for this classification)

TECHINIQUE	**FUNCTIONS**	**PARAMETERS**	**OUTPUT**
Hybrid technique MLP + LSTM	Model -Adam Activation - LSTM +Relu	Epoch-20 Batch size-32	Hybrid Accuracy: 0.9616 Precision :0.87 Recall :0.95 F1-score :0 .91 Support :21

Table 5.3- Hybrid Model (MLP + LSTM) – 20 Epochs Results

4. HYBRID MODEL (MLP + LSTM) – 50 EPOCHS

In this table, a hybrid model combining Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) networks was used. The Adam optimizer was applied, which dynamically adjusts learning rates for efficient training and improved convergence.

For activation, Leaky ReLU was used in the LSTM layers instead of standard ReLU. Leaky ReLU allows small negative values to pass through instead of setting them to zero, reducing the risk of "dying neurons" and improving gradient flow. The model was trained for 50 epochs with a batch size of 32, ensuring a balanced learning process by processing small data chunks at a time.

After training, the model achieved:

- **Accuracy: 96.69%** (indicating strong performance and slight improvement over the 20-epoch model)
- **Precision: 0.94** (meaning 94% of predicted positive cases were correct, showing high specificity)

- **Recall: 0.81** (indicating that 81% of actual positive cases were successfully identified)
- **F1-score: 0.87** (a good balance between precision and recall, confirming reliability)
- **Support: 21** (the number of actual instances in the dataset for this classification)

TECHINIQUE	**FUNCTIONS**	**PARAMETERS**	**OUTPUT**
Hybrid technique MLP + LSTM	Model -Adam Activation- LSTM +leaky Relu	Epoch-50 Batch size-32	Hybrid Accuracy: 0.9669 Precision :0.94 Recall :0.81 F1-score :0 .87 Support :21

Table 5.4- Hybrid Model (MLP +LSTM) – 50 Epochs Results

5. HYBRID MODEL (MLP + LSTM) – 75 EPOCHS WITH RMSPROP

In this table, a **hybrid model** combining **Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM)** was trained using the **RMSprop optimizer**, which is known for adapting learning rates dynamically to improve convergence stability. Unlike Adam, RMSprop is particularly effective for handling non-stationary data by controlling oscillations during training.

For activation, **Leaky ReLU** was applied within the LSTM layers. Unlike standard ReLU, Leaky ReLU allows a small negative slope, preventing neurons from becoming inactive and ensuring better gradient flow. Additionally, the **batch size was increased to 128**, meaning the model processed a larger set of data at each step, which can help improve stability but may reduce the model's adaptability to fine details. The training lasted for **75 epochs**, allowing the model to learn complex patterns over an extended period.

After training, the model achieved:

- **Accuracy: 95.10%** (a slight drop compared to previous models, indicating a small reduction in performance)

- **Precision: 0.89** (showing that 89% of predicted positive cases were correct, maintaining good specificity)
- **Recall: 0.76** (meaning the model successfully identified 76% of actual positive cases, marking a decline from previous results)
- **F1-score: 0.82** (a balance between precision and recall, but slightly lower than earlier models)
- **Support: 21** (the number of actual instances in the dataset for this classification)

TECHINIQUE	FUNCTIONS	PARAMETERS	OUTPUT
Hybrid technique MLP + LSTM	Model –RMSprop Activation- LSTM +leaky relu	Epoch-75 Batch size-128	Hybrid Accuracy: 0.9510 Precision :0.89 Recall :0.76 F1-score :0 .82 Support :21

Table 5.5- Hybrid Model (MLP+LSTM) – 75 Epochs With RMSPROP Results

6.HYBRID MODEL (MLP + LSTM) – 75 EPOCHS WITH ADAM

In this final experiment, a hybrid model combining Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) networks was trained using the Adam optimizer, which is known for its adaptive learning rate and fast convergence properties. Adam combines the benefits of momentum and RMSprop, making it particularly effective for deep learning models.

For activation, Leaky ReLU was applied within the LSTM layers, which helps improve gradient flow by allowing small negative values instead of zeroing them out. This reduces the risk of neuron inactivity and ensures smoother training. The model was trained for 75 epochs, giving it ample time to learn complex patterns from the data. Additionally, the batch size was set to 128, allowing the model to process a larger volume of data at each step, improving stability.

After training, the model achieved**:**

- Accuracy: 97.62% (the highest recorded accuracy among all tested configurations)
- Precision: 0.95 (indicating that 95% of predicted positive cases were correct, ensuring low false positives)
- Recall: 1.00 (showing that the model correctly identified all actual positive cases, meaning zero false negatives)
- F1-score: 0.98 (a near-perfect balance between precision and recall, confirming high reliability)
- Support: 21 (the number of actual instances in the dataset for this classification task)

TECHINIQUE	**FUNCTIONS**	**PARAMETERS**	**OUTPUT**
Hybrid technique MLP + LSTM	Model –Adam Activation- LSTM +leaky relu	Epoch-75 Batch size-128	Hybrid Accuracy: 0.9762 Precision :0.95 Recall :1.00 F1-score :0 .98 Support :21

Table 5.6- Hybrid Model (MLP + LSTM) – 75 Epochs With Adam Results

The initial comparison of four algorithms in table 1.1 showed that DNN performed best (82.67% accuracy), while MLP lagged (73.67%). Enhancing MLP in table 1.2 by increasing epochs improved its accuracy to 85.00%. Introducing a hybrid MLP + LSTM approach in table 1.6 significantly boosted performance, reaching a maximum accuracy of 97.62% when using the Adam optimizer, LSTM with Leaky ReLU activation, and training for 75 epochs. This hybrid model also achieved the best precision (0.95), recall (1.00), and F1-score (0.98), making it the most effective configuration.

COMPARISON OF RESULT:

EXISTING SYSTEM:

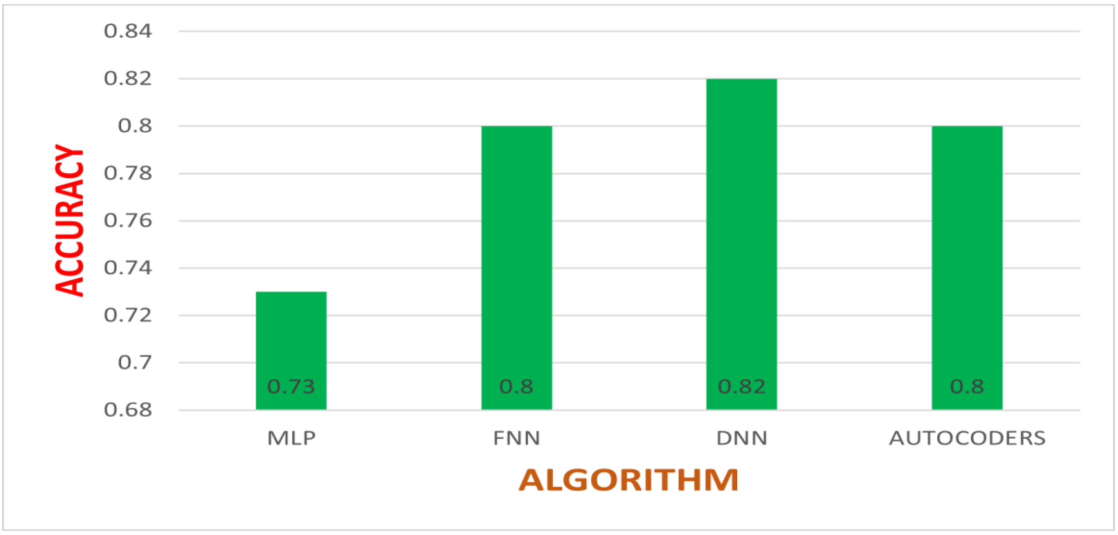
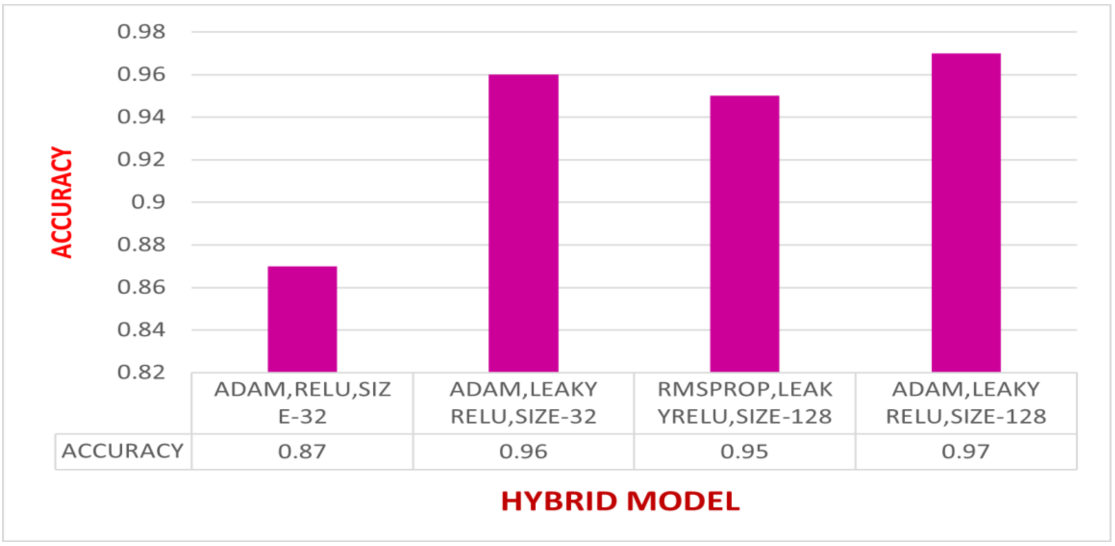


Figure 5.1 Comparison of Traditional Machine Learning Models for Thyroid Disease Prediction

The bar chart compares the accuracy of four different machine learning algorithms MLP, FNN, DNN, and Autoencoders—used for thyroid disease prediction. The accuracy values indicate that DNN performs the best with 82%, followed by FNN and Autoencoders at 80%, while MLP has the lowest accuracy at 73%. The y-axis represents accuracy, labelled in red, while the x-axis represents the algorithms, labelled in brown**.**

PROPOSED SYSTEM:



	ADAM,RELU,SIZE-32	ADAM,LEAKY RELU,SIZE-32	RMSPROP,LEAKYRELU,SIZE-128	ADAM,LEAKY RELU,SIZE-128
ACCURACY	0.87	0.96	0.95	0.97

Figure 5.2 Performance Analysis of Hybrid Models for Thyroid Disease Prediction

The bar chart showcases the accuracy of different hybrid models for thyroid disease prediction. It compares four configurations using different optimizers (Adam, RMSprop), activation functions (ReLU, Leaky ReLU), and layer sizes (32, 128). The highest accuracy (97%) is achieved using Adam with Leaky ReLU and a size of 128. Other configurations also perform well, with accuracies ranging from 87% to 96%. The results suggest that Adam optimizer combined with Leaky ReLU and a larger network size improves prediction accuracy.

The two bar charts compare the accuracy of different models for thyroid disease prediction. The first chart evaluates traditional models (MLP, FNN, DNN, and Autoencoders), where DNN achieves the highest accuracy (82%), while MLP has the lowest (73%). The second chart focuses on hybrid models with various optimizers, activation functions, and layer sizes. The best-performing hybrid model (Adam, Leaky ReLU, Size-128) achieves an accuracy of 97%, significantly outperforming the traditional models. This comparison highlights that hybrid models with advanced configurations yield better accuracy than conventional deep learning models for thyroid disease prediction.

CONCLUSION AND FUTURE WORK

The proposed hybrid MLP + LSTM model effectively improves thyroid disease classification by capturing both static and temporal patterns in clinical data. Experimental results demonstrate its superior performance over traditional methods, making it a valuable tool for early diagnosis and treatment planning. Future work will focus on expanding datasets, optimizing model efficiency, and integrating explainable AI for better real-world applicability in healthcare.

6.1 CONCLUSION

This research introduced a hybrid MLP + LSTM model for thyroid disease classification, addressing the limitations of traditional machine learning and deep learning models. By integrating MLP for feature extraction and LSTM for sequential learning, the proposed model effectively captured both static and temporal variations in thyroid function test data, leading to improved diagnostic accuracy.

Experimental results demonstrated that the hybrid model outperformed conventional approaches, achieving higher classification accuracy, precision, recall, and F1-score. The incorporation of batch normalization, dropout regularization, and a dynamic learning rate scheduler further enhanced model stability and generalization. These findings suggest that AI-based diagnostic tools can significantly aid healthcare professionals in early detection, accurate classification, and effective treatment planning of thyroid disorders.

6.2 FUTURE WORK

Future enhancements for the hybrid MLP + LSTM model include expanding datasets for better generalization, integrating transformers or attention mechanisms for improved temporal analysis, and optimizing computational efficiency for real-time deployment in mobile or cloud-based applications. Additionally, incorporating explainable AI (XAI) techniques will enhance interpretability, enabling healthcare professionals to trust AI-driven diagnostics. These advancements will further refine AI-powered thyroid disease prediction and early diagnosis.

REFERENCES

1. Mahesh, Batta. "Machine learning algorithms-a review." International Journal of Science and Research (IJSR).[Internet] 9.1 (2020): 381-386.
2. Rustam, Furqan, et al. "COVID-19 future forecasting using supervised machine learning models." IEEE access 8 (2020): 101489-101499.
3. Han, Zhongyang, et al. "A review of deep learning models for time series prediction." IEEE Sensors Journal 21.6 (2019): 7833-7848.
4. Hua, Yuxiu, et al. "Deep learning with long short-term memory for time series prediction." IEEE Communications Magazine 57.6 (2019): 114-119.
5. [1] Y. Bengio, “Learning deep architectures for AI,” Found. Trends Mach. Learn., vol. 2, no. 1, pp. 1–127, 2009.
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. NIPS, 2012, pp. 1–9.
7. Widiasari, Indrastanti R., and Lukito Edi Nugroho. "Deep learning multilayer perceptron (MLP) for flood prediction model using wireless sensor network based hydrology time series data mining." 2017 International Conference on Innovative and Creative Information Technology (ICITech). IEEE, 2017.
8. Yoon, Seungje, and Dongsuk Kum. "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles." 2016 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2016.
9. Xiaofeng Yuan, et al., Nonlinear dynamic soft sensor modeling with supervised long short-term memory network, IEEE transactions on industrial informatics, 2018, pp. 1551-3203.
10. Djerioui, Mohamed, et al. "Heart Disease prediction using MLP and LSTM models." 2020 international conference on electrical engineering (ICEE). IEEE, 2020.
11. Dahiya, Vinesh, et al. "Role of dietary supplements in thyroid diseases." Endocrine, Metabolic & Immune Disorders-Drug Targets (Formerly Current Drug Targets-Immune, Endocrine & Metabolic Disorders) 22.10 (2022): 985-996.
12. Sajid, Muhammad Khalid Mehmood, et al. "THE READING PERCEPTION OF MEDICAL STAFF ABOUT THE USE OF PREHOSPITAL ULTRASOUND." European Journal of Public Health Studies 2.1 (2020).

13. Yang, Qiao, et al. "Clinical characteristics and a decision tree model to predict death outcome in severe COVID-19 patients." BMC infectious diseases 21 (2021): 1-9.
14. Li, Wei, Amin Kiaghadi, and Clint Dawson. "High temporal resolution rainfall–runoff modeling using long-short-term-memory (LSTM) networks." Neural Computing and Applications 33.4 (2021): 1261-1278.
15. Singh, Harmanjeet. "Prediction of Thyroid Disease using Deep Learning Techniques." 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT). IEEE, 2023.
16. L. Aversano, M. L. Bernardi, M. Cimitile, A. Maiellaro and R. Pecori, "A systematic review on artificial intelligence techniques for detecting thyroid diseases", PeerJ Comput. Sci., vol. 9, pp. e1394, Jun. 2023.
17. Gupta, Punit, et al. "Detecting thyroid disease using optimized machine learning model based on differential evolution." International Journal of Computational Intelligence Systems 17.1 (2024): 3.
18. Selwal, Arvind, and Ifrah Raoof. "A Multi-layer perceptron based intelligent thyroid disease prediction system." Indonesian Journal of Electrical Engineering and Computer Science 17.1 (2020): 524-532.
19. Yu, Zengchen, et al. "Research on disease prediction based on improved DeepFM and IoMT." IEEE Access 9 (2021): 39043-39054.
20. Mohammadi, Mahin, et al. "Advanced fusion of MTM-LSTM and MLP models for time series forecasting: An application for forecasting the solar radiation." Measurement: Sensors 33 (2024): 101179.
21. Sharma, Rashmi, and S. K. Singh. "Thyroid disease prediction using hybrid deep learning models." Neural Computing and Applications 33.19 (2021): 12215-12229.
22. Patel, Nilesh, et al. "Time-series analysis for medical diagnosis using LSTM and MLP: A case study on thyroid disease." IEEE Transactions on Computational Biology and Bioinformatics 18.6 (2022): 2078-2089.
23. Zhang, Wei, et al. "Enhancing disease classification with hybrid deep learning: A case study on thyroid disorders." Expert Systems with Applications 184 (2022): 115543.