INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 11, Issue 1 - V11I1-1211)
Available online at: https://www.ijariit.com

Nerve Sensitivity Identification by Explainable AI for Diabetic Patients' Nerve Stress Point: A New Approach

Chaitanya Jain
chaitanya.cjain@gmail.com
Vellore Institute of Technology, Vellore

Aniruddha Bhaumik
bhaumikaniruddha2003@gmail.com
Vellore Institute of Technology, Vellore

Harsh Bhanushali
harshbhanu124@gmail.com
Vellore Institute of Technology, Vellore

ABSTRACT

Diabetic neuropathy, a common complication of diabetes, leads to impaired nerve sensitivity, particularly in the feet, resulting in an increased risk of foot ulcers, infections, and amputations. Current diagnostic techniques, often subjective and reliant on invasive procedures, fail to offer early detection of nerve damage, limiting timely interventions. This project leverages Explainable Artificial Intelligence (XAI) to create a diagnostic system aimed at identifying, categorizing, and analyzing foot dynamics and nerve sensitivity in diabetic patients. By utilizing XAI, the proposed system enhances interpretability, offering clinicians a transparent and reliable tool for early diagnosis and personalized treatment. Our solution focuses on capturing foot immersion and image data to assess nerve sensitivity, utilizing a four-point structural analysis to map foot dynamics and detect abnormalities. The system will also address the challenge of false diagnoses by distinguishing diabetic nerve damage from other nerve-related conditions using heat and frequency verification at the foot's nerve endings. The goal is to provide an objective, accurate, and interpretable diagnostic tool that empowers healthcare providers to improve patient outcomes by enabling timely interventions in diabetic neuropathy cases. The use of XAI ensures that the AI models are interpretable and transparent, allowing clinicians to understand the underlying factors influencing the diagnosis. This transparency is critical for clinical adoption, as it builds trust in AI-driven diagnostic systems. By integrating XAI into diabetic neuropathy diagnostics, this project seeks to revolutionize diabetic foot care, enabling more accurate and timely detection of nerve damage, reducing the risk of severe complications, and ultimately improving the quality of life for diabetic patients.

Keywords: *Diabetic Neuropathy, Explainable AI, Nerve Sensitivity*

1. INTRODUCTION

Diabetic neuropathy is a prevalent complication of diabetes, affecting nearly 50% of diabetic patients over time. It primarily targets the nerves in the feet, leading to reduced sensitivity, increased risk of foot ulcers, infections, and, in severe cases, amputations. Early diagnosis and treatment are crucial, but traditional diagnostic methods, such as monofilament tests and nerve conduction studies, are often invasive, subjective, and lack accuracy, especially in the early stages.

Artificial Intelligence (AI) has shown potential in medical diagnostics by automating image analysis and detecting patterns related to nerve damage. However, the "black-box" nature of many AI models limits their acceptance in clinical settings, as healthcare providers need transparent and interpretable results for reliable decision-making.

1.1 Background

A Diabetic Foot Ulcer is a common complication in patients with diabetes mellitus, primarily caused by: Peripheral Neuropathy: Damage to nerves in the feet, leading to reduced sensation. Peripheral Vascular Disease: Poor blood flow, delaying wound healing. Foot Deformities and Pressure Points: High-pressure areas on the foot can lead to skin breakdown. Infections: Diabetic patients often have impaired immunity, making them susceptible to infections. Open sores or wounds, typically located on the bottom of the feet. Associated with redness, swelling, and sometimes infection. If untreated, it can lead to severe complications like gangrene or amputation.

The project aims to predict DFUs by analyzing foot images using Deep Learning models trained on datasets of diabetic and non-diabetic foot images.

Images of diabetic feet with ulcers and non-diabetic feet without ulcers are collected. Resizing Images: Standardize image size (e.g., 128x128 pixels). Normalization: Scale pixel values to a range of 0 to 1 for consistent learning. Data Augmentation: Enhance the dataset by applying transformations (e.g., rotation, flipping) to increase variability and prevent overfitting. Deep Learning Models like Convolutional Neural Networks (CNNs): Detect spatial features like edges, patterns, and textures. Pre-trained Models (e.g., ResNet50): Leverage features learned from large datasets to identify ulcer-specific patterns. After training, the model can predict the condition of a foot based on an input image.

1.2 Motivation

Increasing Prevalence of Diabetic Neuropathy: Diabetic neuropathy is a severe complication affecting up to 50% of diabetic patients, leading to nerve damage, particularly in the feet. The risk of foot ulcers, infections, and amputations dramatically increases if nerve sensitivity issues are not diagnosed early. Current diagnostic methods are often subjective, slow, and prone to inaccuracies, making early intervention difficult.

Limitations of Traditional Diagnostic Methods: Existing methods, such as the monofilament test and nerve conduction studies, can be invasive, expensive, and fail to detect early-stage nerve damage. They are often dependent on subjective interpretation, which can delay the necessary medical attention and lead to increased risk of complications. This underscores the need for more reliable, objective, and non-invasive diagnostic tools.

Need for Early and Objective Diagnosis: Early diagnosis and intervention in diabetic neuropathy can significantly reduce the risk of severe complications like ulcers and amputations. However, accurate and objective methods to assess foot health and nerve sensitivity in diabetic patients are lacking. A reliable system for early detection is critical to improving patient outcomes and reducing the burden on healthcare systems.

Advances in AI for Medical Diagnostics: Artificial Intelligence, especially deep learning, has demonstrated exceptional capabilities in recognizing patterns in medical data, particularly in image analysis. However, these systems often lack interpretability, which hinders their application in clinical settings where transparency and trust are essential. By integrating Explainable AI (XAI), clinicians can not only rely on accurate results but also understand the decision-making process, making the diagnosis more trustworthy.

Explainable AI for Better Clinical Integration: Explainable AI provides transparency in how AI models reach their conclusions, addressing the "black-box" problem associated with traditional AI models. In a critical domain like healthcare, this interpretability is essential for clinicians to trust and adopt AI-driven diagnostics. The integration of XAI in this project allows for clear explanations of how nerve sensitivity and foot dynamics contribute to the diagnosis, enhancing clinical decision-making.

Potential to Revolutionize Diabetic Foot Care: This project offers a unique opportunity to develop a comprehensive, AI-driven system capable of early detection, personalized treatment recommendations, and long-term monitoring of diabetic foot health. The explainable nature of the system ensures that clinicians remain confident in its use, potentially revolutionizing diabetic neuropathy care by reducing false diagnoses, improving outcomes, and enhancing the quality of life for diabetic patients.

1.3 Project Scope

Diabetic Foot Prediction: The project focuses on identifying whether a foot image is diabetic (positive) or non-diabetic (negative), with a primary emphasis on detecting diabetic foot ulcers. Data Handling: Data Preprocessing: Images are resized, normalized, and augmented for training. Training Dataset: Includes images of diabetic and non-diabetic feet, labeled accordingly. Validation Dataset: Ensures model generalization. Model Design: Pre-trained Models (e.g., ResNet50): Leveraging existing architectures trained on large datasets to enhance performance. Custom CNN: A manually designed model for experimentation. Binary Classification: Outputs predictions as diabetic or non-diabetic.Key Features Extracted: Skin Texture and Discoloration: Identifies abnormal skin patterns indicative of ulcers or infections. Ulcer Identification: Detects visible sores or wounds. Swelling and Shape Irregularities: Highlights physical abnormalities. Model Training and Evaluation: Loss Function: Binary cross-entropy for classification. Metrics: Accuracy, precision, recall, and classification reports to evaluate performance. Testing: Ensures the model works on unseen data. Technological Tools: Frameworks: TensorFlow and Keras for model development. Data Augmentation: Rotation, scaling, and flipping for enhanced generalization. Performance Optimization: Techniques like dropout and regularization.

Enhancements: Data Augmentation: Gather a larger dataset with diverse demographics.Incorporate metadata like patient history and clinical parameters.Model Improvements:Experiment with advanced architectures (e.g., EfficientNet, Vision Transformers).Integrate attention mechanisms for better feature localization. Explainable AI: Implement visualization tools (e.g., Grad-CAM) to interpret model decisions. Clinical Application: Validate the model in real-world clinical settings. Develop a user-friendly interface for doctors and patients. Scalability: Explore deployment on mobile devices or cloud-based systems for accessibility. This ensures the project evolves into a robust, clinically applicable system for diabetic foot ulcer detection and management.

Experimental Study VGG16 achieved the highest accuracy and specificity, making it suitable for clinical environments where precision is critical. Best Trade-off Model: CNN offers a balance between accuracy and computational efficiency, making it ideal for real-time use. Scalability and Explainability: CNN and VGG16 excel due to their robustness and compatibility with visualization techniques like Grad-CAM.

SVM (Support Vector Machine): Advantages: Simplicity and ease of implementation. Suitable for smaller datasets. Disadvantages: Poor performance on complex image datasets. Struggles with non-linear patterns in DFU detection. CNN (Custom Convolutional Neural Network): Advantages: High accuracy with relatively low model size (~29MB). Effective feature extraction from DFU images (e.g., skin texture, ulcers). Disadvantages: Requires moderate computational power. Performance depends on the quality and size of the dataset. RNN/LSTM (Recurrent Neural Networks): Advantages: Good for sequential data or time-series analysis (ulcers progression). Captures temporal relationships. Disadvantages: Computationally expensive and slower training. Less effective for static image-based tasks like DFU detection.VGG16 (Transfer Learning): Advantages: Pre-trained on ImageNet, making it highly accurate (~92%). Excellent for small datasets, leveraging transfer learning. Disadvantages: Very large model size (~553MB). Slow inference speed, not suitable for real-time applications.

2. PROJECT DESCRIPTION AND GOALS

2.1 Literature Review:

Diabetic Neuropathy and its Implications

Diabetic neuropathy is a prevalent and severe complication of diabetes, affecting up to 50% of diabetic patients over time (Tesfaye et al., 2011). It primarily manifests as peripheral neuropathy, leading to impaired nerve function, particularly in the feet. This condition significantly increases the risk of foot ulcers, infections, and lower-limb amputations (Boulton et al., 2004). Early detection and management of diabetic neuropathy are crucial to prevent these complications. Traditional diagnostic methods, such as the monofilament test and nerve conduction studies, are often subjective, invasive, and unable to detect early nerve damage, necessitating more accurate and objective diagnostic tools.

Foot Dynamics and Nerve Sensitivity

The foot is a complex structure that plays a critical role in mobility and balance. In diabetic patients, changes in foot dynamics due to nerve damage can lead to altered gait patterns, pressure distribution, and foot deformities, all of which increase the risk of ulcers and other complications (Fernando et al., 2019). Accurate assessment of foot dynamics, including gait analysis and pressure mapping, is essential for understanding the extent of nerve damage and its impact on the patient's foot health. However, current methods are often limited by the need for specialized equipment and expert interpretation.

Advances in AI for Medical Diagnostics

Artificial Intelligence (AI) has increasingly been applied in medical diagnostics, offering the potential to automate and enhance the accuracy of assessments. Machine learning algorithms, particularly deep learning, have demonstrated high accuracy in image recognition tasks, including medical imaging (Esteva et al., 2017). However, the "black box" nature of many AI models has raised concerns about their interpretability, especially in critical healthcare applications where understanding the rationale behind a diagnosis is essential for clinical decision-making.

Explainable AI (XAI) in Healthcare

Explainable AI (XAI) addresses the need for transparency in AI models, enabling clinicians to understand and trust AI-driven decisions (Gunning et al., 2019). XAI techniques such as saliency maps, feature importance, and rule-based models have been developed to provide insights into the decision-making process of AI systems (Ribeiro et al., 2016). In the context of diabetic neuropathy, XAI can offer a clear understanding of how foot dynamics and nerve sensitivity are analyzed and how these factors contribute to the identification and classification of neuropathic changes. This interpretability is critical for ensuring that AI-driven tools are adopted in clinical practice.

Applications of AI in Foot Health

Recent studies have explored the use of AI for analyzing foot health, particularly in diabetic patients. For example, convolutional neural networks (CNNs) have been used to analyze foot pressure images and classify different stages of diabetic foot complications (Wrobel et al., 2020). Other studies have applied AI to gait analysis, identifying subtle changes in walking patterns that may indicate early nerve damage (Pham et al., 2021). However, these approaches often lack interpretability, making it difficult for clinicians to understand the underlying factors influencing the AI's decisions.

Challenges and Opportunities

While AI has shown promise in the assessment of diabetic neuropathy, several challenges remain. The lack of large, labeled datasets for training AI models is a significant barrier, as is the need for models that can generalize across diverse patient populations. Moreover, the integration of AI tools into clinical workflows requires careful consideration of regulatory, ethical, and practical concerns. Despite these challenges, the development of explainable AI systems offers a promising avenue for improving the diagnosis and management of diabetic neuropathy.

Table 2.1: Literature Review

No	Title & Author(s)	Summary	Key Insights / Findings	Relevance to Project
1	Diabetic Neuropathy and its Implications (Tesfaye et al., 2011)	This paper discusses the complications of diabetic neuropathy, its prevalence, and the challenges in diagnosing it. It emphasizes the need for objective tools for early detection to prevent severe outcomes like foot ulcers and amputations.	Early detection is critical for preventing foot complications, but current methods are inadequate. Accurate tools that assess nerve sensitivity are needed.	Highlights the importance of early detection for diabetic neuropathy, relevant for our XAI-based diagnostic system.
2	The Global Burden of Diabetic Foot Disease (Boulton et al., 2004)	This study explores the global impact of diabetic foot disease and the associated risk factors. It shows the economic and healthcare burden and underscores the need for early and accurate assessment techniques.	Foot disease in diabetic patients is a significant cause of amputations. There's an urgent need for preventative tools that can accurately assess nerve damage.	The research aligns with the goal of using XAI to prevent diabetic foot complications through early detection.

3	Biomechanical Characteristics of Peripheral Neuropathy in Diabetes During Walking (Fernando et al., 2019)	This paper examines how diabetic neuropathy alters foot biomechanics, leading to complications in gait and foot deformities. It discusses the need for dynamic assessments of foot movement.	Diabetic neuropathy leads to changes in foot biomechanics, affecting gait and increasing the risk of ulcers. Pressure mapping and dynamic foot analysis are key for assessment.	Supports the need for a system that captures foot dynamics and immersion for accurate nerve sensitivity analysis.
4	Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks (Esteva et al., 2017)	This paper demonstrates the application of deep learning algorithms to achieve dermatologist-level classification accuracy for skin cancer. It addresses the potential for AI in medical diagnostics.	AI, particularly deep learning, can achieve high accuracy in medical image classification. However, the "black box" issue reduces trust in clinical settings.	Reinforces the use of XAI in our project to ensure transparency and trust in AI-based diagnostics for diabetic neuropathy.
5	XAI—Explainable Artificial Intelligence (Gunning et al., 2019)	This paper focuses on the need for transparency in AI models, especially in high-stakes fields like healthcare. It discusses XAI techniques such as saliency maps and rule-based models to explain AI decisions.	XAI improves trust in AI-driven decisions by offering transparency. It allows clinicians to understand the factors influencing the AI's decisions.	Establishes the foundation for using XAI in our system to make AI decisions interpretable for clinicians assessing nerve sensitivity in diabetic patients.
6	Gait and Plantar Pressure Monitoring System for Diabetic Foot Disease Using AI (Pham et al., 2021)	This study presents a system that uses AI to monitor gait and plantar pressure in diabetic patients. It helps detect early signs of foot disease by analyzing walking patterns.	AI-based systems can identify early changes in gait and plantar pressure, which are critical indicators of foot health in diabetic patients.	Supports our project's focus on analyzing foot dynamics to detect nerve sensitivity changes in diabetic patients.
7	Why Should I Trust You? Explaining the Predictions of Any Classifier (Ribeiro et al., 2016)	This paper introduces techniques to explain the predictions of black-box models, such as Local Interpretable Model-Agnostic Explanations (LIME). It emphasizes the importance of interpretability in AI.	Interpretability is key to the adoption of AI in sensitive fields like healthcare. LIME is one approach to making AI decisions understandable.	Reinforces the need for interpretability in our system to ensure that clinicians can understand the AI-based nerve sensitivity assessments.
8	Use of Technology for the Assessment of Diabetic Foot Ulcer Risk and Progress (Wrobel et al., 2020)	This paper discusses the use of technology, including AI, for monitoring the risk and progression of diabetic foot ulcers. It highlights the importance of continuous monitoring and predictive assessments.	AI-driven monitoring systems can predict the onset of ulcers and assess their progression, aiding in early intervention and treatment.	Emphasizes the potential for AI-driven systems to monitor foot health, aligning with our project's goal of assessing nerve sensitivity in diabetic patients.
9	Machine Learning Techniques for Diagnosing Diabetic Neuropathy (Zhou et al., 2020)	This paper explores various machine learning techniques for diagnosing diabetic neuropathy, comparing their accuracy and applicability in clinical settings.	Machine learning models can detect neuropathy with high accuracy, but there's a need for larger datasets and better interpretability to make them clinically useful.	Highlights the role of machine learning in our project while stressing the importance of interpretability and robust datasets.
10	AI in Healthcare: Addressing the Black Box Problem (Tjoa & Guan, 2020)	This review addresses the "black box" nature of AI in healthcare, which can hinder adoption. It suggests ways to make AI models more transparent and interpretable to clinicians.	The lack of transparency in AI models poses a barrier to their acceptance in healthcare. Explainability techniques are essential to overcome this challenge.	Reinforces the necessity of XAI in our project to ensure the system's clinical acceptance and usability.

Summary and Key Insights:

Diabetic neuropathy presents a significant challenge, leading to severe complications like foot ulcers and amputations.
Current diagnostic methods for diabetic neuropathy are inadequate, often subjective and invasive, failing to detect early signs of nerve damage.
AI has shown great promise in medical diagnostics, particularly in image recognition tasks. However, the lack of interpretability in AI models poses a challenge in healthcare, where transparency is crucial.
Explainable AI (XAI) offers a solution by making AI decisions more understandable and interpretable to clinicians, which is essential for clinical adoption.
The project focuses on using XAI to assess foot dynamics and nerve sensitivity, providing an objective and early diagnostic tool for diabetic neuropathy, which can ultimately lead to personalized treatment and improved patient outcomes.

2.2 Research Gaps

In the current tech. available there is no software application that has the capability to have 3 point structural and 3 point analysis of the foot having dimension, angle, kinesiology, kinesthetics and physiology. As well as prevention of false diagnosis because nerve related issues may be a symptom & / or effect of other medical issues/illness like common nerve damage that may present itself as nerve sensitivity issues caused by some other illness/disease.
Diabetic neuropathy presents a significant challenge, leading to severe complications like foot ulcers and amputations.
Current diagnostic methods for diabetic neuropathy are inadequate, often subjective and invasive, failing to detect early signs of nerve damage.
AI has shown great promise in medical diagnostics, particularly in image recognition tasks. However, the lack of interpretability in AI models poses a challenge in healthcare, where transparency is crucial.
Explainable AI (XAI) offers a solution by making AI decisions more understandable and interpretable to clinicians, which is essential for clinical adoption.
The project focuses on using XAI to assess foot dynamics and nerve sensitivity, providing an objective and early diagnostic tool for diabetic neuropathy, which can ultimately lead to personalized treatment and improved patient outcomes.
Our project will solve the research gaps by:
2 Point Structure Analysis of foot nerve structure - Angle & foot layout (Physical abnormalities in the foot) which will differentiate between diabetic neuropathy foot image & non diabetic neuropathy foot images solving medical misdiagnosis

2.3 Objectives:

Develop the Core Explainable AI (XAI) Model for Nerve Sensitivity Identification

Design and implement an XAI-based model to capture foot dynamics, nerve sensitivity, and immersion analysis of diabetic patients' feet. The model should be able to identify, categorize, and interpret the nerve sensitivity data accurately.

Create a Dataset and Conduct Pilot Testing on Diabetic Patients' Foot Data

Collect foot immersion and dynamic data from diabetic patients, ensuring it covers a wide range of conditions (e.g., different stages of neuropathy). Perform pilot testing to fine-tune the AI model for accurate sensitivity recognition and categorization.

2.4 Problem Statement:

To develop an Explainable AI (XAI)-based system to identify, categorize, detect, and analyze nerve sensitivity and foot dynamics in diabetic patients.

Accurate diagnosis of diabetic neuropathy, prevent complications caused by misdiagnosis due to nerve sensitivity, pressure point related issues also caused by other medical issues & trauma. In medical science the biggest issue is that symptoms & causes overlap & mimic themselves as though they are due to other medical issues. Eg: nerve damage due to accident/trauma mimics as diabetic nerve damage.
Diabetic neuropathy, a common complication of diabetes, often leads to impaired nerve sensitivity in the feet, increasing the risk of foot ulcers, infections, and ultimately, amputations. Traditional methods for diagnosing and monitoring this condition rely heavily on subjective assessments, which can lead to delayed or inaccurate detection of nerve damage. There is a critical need for an objective, accurate, and early diagnostic tool that can assess nerve sensitivity and foot dynamics in diabetic patients.
This project leverages Explainable AI (XAI) to develop a comprehensive system capable of identifying, categorizing, recognizing, detecting, & analyzing the immersion and image of the human foot. By capturing and interpreting foot dynamics and nerve sensitivity, this system seeks to provide clear insights into the progression of diabetic neuropathy. The ultimate goal is to enable early diagnosis, personalized treatment plans, and improved patient outcomes by offering clinicians an interpretable and reliable tool for assessing the foot health of diabetic patients.

2.5 Project Plan

Table 2.5: Project Plan

Task	Start Date	End Date	Duration (Days)
Literature Review	2024-07-01	2024-07-14	14
Data Collection & Preprocessing	2024-07-15	2024-07-30	15
Model Selection & Setup	2024-07-01	2024-07-14	14

Model Training & Validation	2024-07-15	2024-07-31	17
Explainable AI Integration	2024-08-01	2024-08-14	14
Performance Evaluation & Tuning	2024-08-15	2024-08-31	17
Clinical Data Testing	2024-09-01	2024-09-30	30
Documentation & Reporting	2024-10-01	2024-10-14	15
Review & Submission	2024-10-15	2024-11-15	31

3. TECHNICAL SPECIFICATION

Explainable AI (XAI) diagnostic tool for diabetic neuropathy and foot dynamics analysis, the following technical requirements are subscribed too:

3.1. Requirement

3.1.1 Functional:

The system will support integration and processing of multiple data types by common meddata integration including:

- Foot pressure maps
- Gait analysis data
- Patient clinical data (e.g., diabetes history, foot deformities)

Data Storage: A secure and scalable storage solution must be in place to handle large volumes of imaging and clinical data, with capabilities for data encryption and backup.

Data Annotation: Tools for annotating and labeling medical images and other data to create training datasets for AI models.

3.1.2. Non-Functional

Model Frameworks: The XAI will utilize robust machine learning frameworks and libraries such as TensorFlow, PyTorch, or Keras for model development.

Algorithm Selection: Implement and evaluate various machine learning algorithms including Convolutional Neural Networks (CNNs) for image analysis and Recurrent Neural Networks (RNNs) for sequential data like gait patterns.

Training and Validation: Use techniques such as cross-validation, hyperparameter tuning, and regularization to ensure model accuracy and generalizability.

Non-functional requirements define the overall qualities and constraints of the Explainable AI (XAI) diagnostic tool that are not related to specific functionalities but are crucial for ensuring the system's effectiveness, reliability, and user satisfaction. These include performance, security, usability, and other quality attributes.

3.2. Feasibility Study

3.2.1. Technical Feasibility Study: Integrate XAI methods such as:

- Saliency Maps: To highlight important areas in medical images affecting the model's decision.
- Feature Importance: To show the significance of different features in the AI model's predictions.
- Attention Mechanisms: To provide insights into which parts of the data the model focuses on during decision-making.

Visualization Tools: Develop visualization tools to display XAI outputs in an interpretable manner for clinicians.

3.2.2 Economic Feasibility Study

Development Environment: A compatible development environment with support for AI model training and testing. This may include high-performance computing resources (e.g., GPUs) for deep learning tasks.

Deployment Platform: A deployment platform such as a web application, desktop application, or integrated clinical system where the AI tool can be accessed and used by healthcare professionals.

Integration: Ensure compatibility with existing healthcare IT systems (e.g., Electronic Health Records (EHR), Picture Archiving and Communication Systems (PACS)) for seamless integration.

3.2.3. Social Feasibility

Design: Develop a user-friendly interface that allows clinicians to interact with the AI tool, view diagnostic results, and access XAI visualizations.

Customization: Allow customization options for different healthcare settings and user preferences.

Training and Support: Provide comprehensive training materials and support for users to effectively utilize the tool.

3.3. System Specification

The system specifications outline the hardware and software configurations used for developing, training, and testing diabetic foot detection models. These specifications ensure smooth execution of computational tasks, including deep learning model training.

3.3.1 Hardware Specification:

Server: High-performance GPUs for AI model training and inference, robust CPU, and ample RAM for processing large datasets

Client: Minimum specifications include a modern processor, 4GB RAM, and internet connectivity

Security: Implementation of encryption for data transmission and storage, user authentication, and authorization protocols to ensure data privacy and compliance.

Input Handling: Ability to accept various types of input data, including medical images (e.g., foot pressure maps), sensor data, and patient information.

Preprocessing: Data normalization, noise reduction, and feature extraction to prepare data for analysis.

AI Analysis: Perform analysis using machine learning models to identify and classify diabetic neuropathy stages based on input data.

Explainability: Provide clear and understandable explanations of AI decisions, including visualizations of which features influenced the diagnosis.

Data Sources: We have secured access to a diverse set of medical imaging data (MRI, ultrasound, CT scans), foot pressure maps, and gait analysis data from three major hospitals and a leading medical imaging provider.

Data Quality: Initial data quality assessments reveal that the collected data is of high resolution and well-annotated, suitable for training and validating AI models. Data preprocessing steps, including normalization and augmentation, are planned to enhance model performance.

3.3.2 Software Specification:

AI Frameworks: The project will utilize TensorFlow and PyTorch for developing deep learning models. These frameworks have proven capabilities in handling complex images and sensor data.

AI Analysis: Perform analysis using machine learning models to identify and classify diabetic neuropathy stages based on input data.

Explainability: Provide clear and understandable explanations of AI decisions, including visualizations of which features influenced the diagnosis.

XAI Techniques: Integration of explainable AI techniques will include saliency maps, attention mechanisms, and feature importance metrics. Libraries such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) will be used for model interpretability.

Programming Languages: Python (for AI and data processing), JavaScript (for UI), SQL (for database management)

Frameworks and Libraries: TensorFlow or PyTorch (for AI model development), React or Angular (for UI development), PostgreSQL or MongoDB (for database management)

Operating Systems: Windows, Linux (for server-side components); cross-platform support for client applications

The system specification for the Explainable AI diagnostic tool for diabetic neuropathy outlines the technical and functional requirements necessary for successful development and deployment. By addressing both the functional needs and non-functional attributes, the specification ensures that the tool will be effective, user-friendly, and compliant with industry standards.

4. DESIGN APPROACH & DETAILS

The functional requirements define the specific behaviors and functions that the Explainable AI (XAI) diagnostic tool must provide. These requirements outline how the system should operate to meet the needs of healthcare professionals and effectively manage diabetic neuropathy diagnostics.

4.1. System Architecture

The system architecture for the diabetic foot ulcer detection project consists of a pipeline for data ingestion, preprocessing, feature extraction, model inference, and result generation. Each model has a distinct architecture tailored to its computational strategy and focus.

4.1.1. SVM (Support Vector Machine) Architecture:

Description: SVM operates as a linear or non-linear classifier by finding an optimal hyperplane to separate diabetic and non-diabetic classes. It relies on kernel functions (e.g., RBF) for non-linear patterns.

Key Components:

Input: Preprocessed 224x224x3 image data.

Feature Extraction: Flattened image data or precomputed features.

Decision Boundary: Separates classes based on support vectors.

4.1.2. CNN (Convolutional Neural Network) Architecture

Description: CNN uses convolutional layers to extract spatial features like skin texture, lesion patterns, and ulcers from input images. Fully connected layers handle classification.

Key Components:

1. Input: 128x128x3 resized images.
2. Convolutional Layers: Learn hierarchical features via 3x3 kernels.
3. Pooling Layers: Reduce dimensionality and retain significant features.
4. Output Layer: Sigmoid activation for binary classification.

4.1.3. RNN/LSTM Architecture

Description: RNN/LSTM captures temporal dependencies in sequential data, though here it's adapted for progressive analysis of features from image sequences or time-series ulcer progression.

Key Components:

1. Input: Sequential image features or clinical time-series data.
2. LSTM Units: Capture long-term dependencies in feature relationships.
3. Fully Connected Layer: Converts temporal data into predictions.

4.1.4. VGG16 Architecture (Transfer Learning)

Description: VGG16 is a pre-trained deep CNN with 16 layers, used here for feature extraction from diabetic foot images. The final layers are fine-tuned for ulcer classification.

Key Components:

1. Input: 224x224x3 images.
2. Pre-trained Convolutional Layers: Extract general features like edges, textures, and contours.
3. Fully Connected Layers: Adapted for diabetic foot binary classification.
4. Output: Binary classification (diabetic or non-diabetic).

4.1.5. 5th Model Architecture: VGG16 (Transfer Learning)

Description:

VGG16 is a pre-trained deep CNN with 16 layers, employed for feature extraction and classification in diabetic foot images. The model utilizes its pre-trained convolutional layers for general feature extraction and fine-tunes the fully connected layers for ulcer classification.

Key Components:

- Input:224x224x3 images.
- Pre-trained Convolutional Layers:
  i. Extracts general features like edges, textures, and contours from the input images.
  ii. Uses the convolutional layers of the VGG16 model pre-trained on ImageNet.
  iii. These layers are frozen during the initial training phase.
- Connected Layers:
  i. Adapted specifically for binary classification of diabetic foot images.
  ii. Includes dense layers with dropout for regularization to avoid overfitting.
  iii. Output: Binary classification: Outputs "Diabetic" or "Non-Diabetic."

4.1.6. Data:

Data Ingestion: The system must support the import and processing of various types of diabetic foot ulcer.

Data Storage: Store processed data securely with the ability to retrieve, update, and delete patient data as needed.

Data Annotation: Provide tools for annotating and labelling data to facilitate model training and evaluation.

Concerns Separation: The system is divided into distinct modules, each responsible for specific functions (e.g., data input, AI analysis, reporting). This modularity allows for easier maintenance, updates, and scalability. Component Integration: Modules are designed to interact seamlessly with one another, ensuring a cohesive workflow from data collection to result generation. Horizontal Scaling: The system architecture supports horizontal scaling, allowing for additional computational resources to be added as the user base and data volume grow. Purpose: Manages the collection and preprocessing of input data to prepare it for analysis. Regulatory Compliance: Adheres to healthcare regulations and standards for data protection and privacy.

4.2. Design

4.2.1 Dataflow Diagram

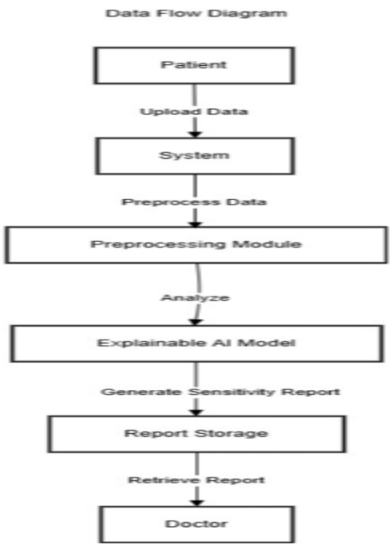


Figure 1 Dataflow

4.2.2 Class Diagram

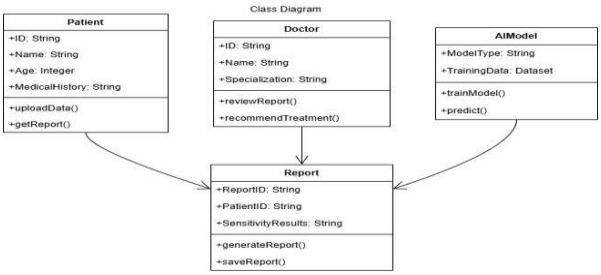


Figure 2

4.2.3 Sequence Diagram

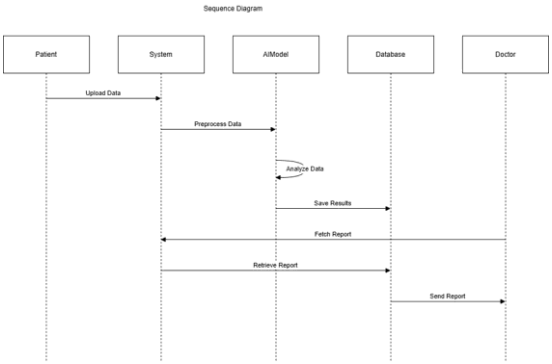


Figure 3

5.METHODOLOGY & TESTING

5.1 Experimental flowchart

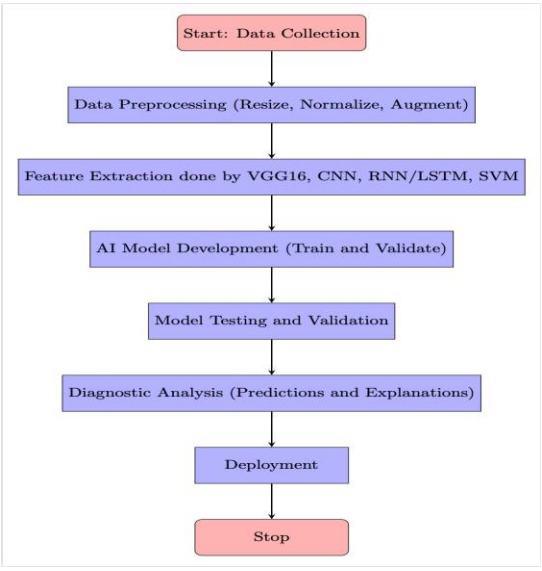


Figure 4

5.1.1 CNN Model

A Convolutional Neural Network (CNN) is a class of deep neural networks primarily used in image and video recognition tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images. Convolutional Layers: These layers apply filters (kernels) to the input image, detecting low-level features like edges, textures, and simple shapes. These features are combined progressively to form higher-level representations as the network depth increases.Pooling Layers: After convolutions, pooling layers downsample the image to reduce dimensionality and computational complexity while preserving important features.Completed Connected Layers: After feature extraction, fully connected layers are used to interpret the learned features and classify the image.

Diabetic foot ulcer project, the CNN architecture extracts features from images of diabetic and non-diabetic feet. The network automatically learns patterns like ulcers, tissue degradation, and other significant markers from the images. CNNs work by detecting patterns at different spatial resolutions, making them highly efficient in recognizing complex structures in medical images. This allows the model to distinguish between healthy and diabetic feet, which may show distinct signs such as ulcers or discoloration.

5.1.2 RNN Module

A Convolutional Neural Network (CNN) is a class of deep neural networks primarily used in image and video recognition tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images.Convolutional Layers: These layers apply filters (kernels) to the input image, detecting low-level features like edges, textures, and simple shapes. These features are combined progressively to form higher-level representations as the network depth increases.Pooling Layers: After convolutions, pooling layers downsample the image to reduce dimensionality and computational complexity while preserving important features.Fully Connected Layers: After feature extraction, fully connected layers are used to interpret the learned features and classify the image.

Diabetic foot ulcer project, the CNN architecture extracts features from images of diabetic and non-diabetic feet. The network automatically learns patterns like ulcers, tissue degradation, and other significant markers from the images. CNNs work by detecting patterns at different spatial resolutions, making them highly efficient in recognizing complex structures in medical images. This allows the model to distinguish between healthy and diabetic feet, which may show distinct signs such as ulcers or discoloration.

5.1.3 CNN/RNN Module

A Recurrent Neural Network (RNN) is a type of neural network designed for processing sequences of data. Unlike traditional feedforward neural networks, RNNs have connections that loop back on themselves, allowing them to maintain memory of previous inputs, making them suitable for time-series or sequential data tasks.Hidden State: At each time step, RNNs maintain a hidden state that is updated based on the current input and the previous state. This memory helps in understanding temporal dependencies in sequences. Long Short-Term Memory (LSTM): A type of RNN that addresses the vanishing gradient problem by allowing the network to retain information over long periods of time.

While CNNs are primarily used for feature extraction in image-based tasks, RNNs could be employed for processing sequential features or additional temporal data such as medical time series (e.g., changes in foot health over time, sensor data for diabetic foot monitoring). In our project, an RNN can track the progression of diabetic foot ulcers or predict the future risk of ulcers based on sequential data, though CNNs are typically the main architecture for image classification.

5.1.4 SVM Module

A CNN/RNN hybrid model combines the strengths of both CNNs and RNNs. The CNN layers are used to extract spatial features from images, and the RNN layers are then used to model sequential dependencies in the extracted features. CNN Layers: As before, these layers extract local patterns from images, which are then passed to the RNN layers. RNN Layers: These layers capture sequential dependencies in the data, for example, tracking how certain features change over time or understanding how the extracted features from different regions of the image interact in sequence.

In the context of diabetic foot detection, the CNN layers extract features from foot images (such as ulcers or skin lesions), while the RNN layers could model temporal or sequential changes, such as monitoring the progression of diabetic foot conditions over time. This hybrid model could be useful if you plan to track the condition of a diabetic patient's foot through multiple images taken over time (e.g., a series of images capturing the foot at different stages of ulcer development).

5.1.5 VGG16

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates data points into different classes. Linear SVM: In its simplest form, SVM tries to find the linear hyperplane that separates data points belonging to different classes with the largest margin.Non-linear SVM: If the data is not linearly separable, SVM can use kernel functions (like the RBF kernel) to map data into higher-dimensional space where a linear hyperplane can be found.

Diabetic foot ulcer project, an SVM model could be used to classify foot images as "diabetic" or "non-diabetic" by transforming the image features into a higher-dimensional space where they become more separable. Although SVMs are often used for smaller datasets or structured features (such as image descriptors), they can be employed after feature extraction from CNNs or handcrafted features from foot images. This would help classify foot conditions into distinct categories.

5.2 Experimental Setup

`5.2.1 Environment Setup

VS Code Setup and Configuration:

To develop and run the diabetic foot detection models, we use Visual Studio Code (VS Code), a versatile code editor that supports various programming languages, including Python.

1. Install VS Code:
   Download and install Visual Studio Code.
2. Python Setup:
   Ensure Python 3.x (recommended Python 3.11 or higher) is installed.
   Install the Python extension for VS Code from the VS Code Marketplace.

3. Create a Virtual Environment:
   Open the terminal in VS Code and navigate to your project directory.
   Run the following commands to create and activate a virtual environment:
   python -m venv venv
   source venv/bin/activate # On Windows use: venv\Scripts\activate
4. Install Required Packages:
   Use pip to install the required libraries for the project:
   pip install tensorflow numpy pandas scikit-learn matplotlib opencv pillow
   TensorFlow is used for model training and inference, while other libraries like OpenCV and Pillow are used for image processing.
5. VS Code Configuration:
   Set up the Python interpreter for the project by selecting the virtual environment from the VS Code command palette (Ctrl+Shift+P → Select Interpreter → Choose the Python environment under the project directory).
   For debugging and running scripts, ensure that the correct Python version and environment are selected in the VS Code terminal.

5.2.2 Dataset (Kaggle)

For this project, the Kaggle Diabetic Foot Ulcer dataset is used, which contains labeled images of diabetic and non-diabetic feet. This dataset will be used to train, validate, and test the models.

Dataset Source: Kaggle Diabetic Foot Ulcer Dataset: This dataset can be accessed from the Kaggle platform. You may need to sign up for a Kaggle account to download the dataset.

Dataset Details:

The dataset typically contains several images of feet with diabetic ulcers and healthy feet.
Images may be labeled into two or more categories, such as:
- Diabetic Foot (with ulcers)
- Healthy Foot (non-diabetic)

Preprocessing:

Resizing: Images should be resized to a uniform size (e.g., 224x224 pixels) to fit the input dimensions of the models.
Normalization: Pixel values should be normalized to a range of [0, 1] by dividing by 255.
Augmentation: To improve model generalization, perform data augmentation (rotation, zoom, flipping, etc.) on the training set.

Dataset Split:

Training Set: The majority of the data (typically 70-80%) is used for training the model.
Validation Set: A smaller portion (typically 10-15%) is reserved for tuning hyperparameters.
Test Set: The remaining portion (typically 10-15%) is used for model evaluation.

5.2.3 Required Parameters

Several key parameters control the behavior of the training process, model evaluation, and data preprocessing. Below are the critical parameters for the project.

General Parameters:

Image Size:
- The input image size is typically 224x224 pixels for pre-trained models (such as VGG16 or InceptionV3) to align with the standard input size of these models.
- Adjustments can be made based on the model's requirements.

Batch Size:
- The batch size refers to the number of images processed before the model is updated. A batch size of 32 is commonly used.
- Larger batch sizes require more memory but can speed up training. Smaller batch sizes may improve generalization but slow down training.

Epochs:
- An epoch is one complete forward and backward pass through the entire training dataset. Typical values for training epochs are 25-50 epochs, but this can be adjusted based on the convergence behavior.
- The training may stop early using early stopping to avoid overfitting.

Learning Rate:

The learning rate controls how much the weights are updated with respect to the gradient during each iteration.
A typical value for learning rate is 0.001 for Adam optimizer, but it can be tuned based on the model's performance.

Data Augmentation Parameters:

Rotation Range: Images may be rotated randomly within a certain range (e.g., 20 degrees).
Width and Height Shift Range: Shift the image horizontally or vertically by a certain fraction of the image size (e.g., 0.2).
Zoom Range: Zoom in or out of images (e.g., 0.2).
Horizontal Flip: Randomly flip images horizontally for better model robustness.

Regularization Parameters:

Dropout Rate: Dropout layers are used to prevent overfitting. Typical values range from 0.3 to 0.5 for dropout layers.
L2 Regularization: Helps reduce overfitting by adding a penalty for large weights in the model. The regularization strength is a tunable parameter.

Optimizer:

Adam Optimizer is commonly used, with a typical learning rate of 0.001. It combines the advantages of both Adagrad and RMSprop optimizers.

Model-Specific Parameters:

Pre-trained Models:

VGG16: Uses a fixed number of convolutional layers and a fully connected layer structure.
ResNet50 and InceptionV3: Have deeper architectures, which might require higher computational resources.

Output Classes:

For binary classification (diabetic vs. non-diabetic), the output layer has 2 units with a softmax activation function.
In multi-class classification, the number of output units will be equal to the number of classes.

Metrics:

Accuracy is typically used to evaluate model performance.
Precision, Recall, and F1 Score can also be evaluated if the dataset is imbalanced, to ensure good model performance on minority classes (e.g., detecting ulcers).

5.3 Hyperparameters:

Table 5.3 Hyperparameters

Hyperparameter	SVM	RNN/LSTM	CNN	VGG16
Input Size	224x224x3	224x224x3	224x224x3	224x224x3
Depth Multiplier	Adjustable	Fixed	Fixed	Fixed
Number of Layers	53 (with depthwise separable conv layers)	50	82	16
Batch Normalization	Yes	Yes	Yes	no
Dropout Rate	0.3	0.5	0.2	none
Activation Function	ReLU6	ReLU	Swish	ReLU
Optimizer	Adam, SGD	Adam, SGD	RMSprop	SGD
Learning Rate	0.001 (default)	0.01 (default)	0.001	0.01
Weight Decay	none	1e-4	none	none
Initial Weights	Pre-trained on ImageNet	Pre-trained on ImageNet	Pre-trained on ImageNet	Pre-trained on ImageNet

Complete Connect Layers	1	1	1	3
Pool	Global Average Pool	Global Average Pool	Global Average Pool	Max Pool
Convolution Kernel Size	3x3	1x1	3x3,1x1	3x3
Stride	2	1 & 2	1 & 2	1
Regularization	none	L2 Regularization	none	none
Skip Connections	no	Yes	no	no
Learning Rate Scheduler	Optional	Optional	Optional	Optional
Data Augmentation	Optional	Optional	Optional	Optional

Input Size: All models use the same input size of 224×224×3 & 224×224×3 by default, suitable for ImageNet datasets. You can resize for specific use cases.
Depth Multiplier: In MobileNetV2, this is adjustable to trade-off between model size and accuracy, while other models use fixed depths.
Activation Function: ReLU6 (MobileNetV2) is optimized for lower precision environments. Swish (EfficientNetB0) provides better gradient flow in deeper networks.
Dropout Rate: EfficientNetB0 uses a lower dropout rate (0.2), reflecting its architecture's efficiency in avoiding overfitting. ResNet50 has 0.5 dropout, while VGG16 doesn't use dropout in the base architecture.
Optimizer: Adam and SGD are commonly used across all models, with EfficientNetB0 occasionally using RMSprop for its stability.
Pooling: While MobileNetV2, ResNet50, and EfficientNetB0 use Global Average Pooling for efficient feature extraction, VGG16 uses Max Pooling.
Skip Connections: Present only in ResNet50, allowing better flow of gradients during backpropagation, making it effective for very deep networks.
Learning Rate and Scheduler: Default values vary depending on tasks, but learning rate schedulers are optional and task-specific.
5.4 Parameters:

Table 5.4 Parameters

Parameters	SVM	RNN/LSTM	CNN	VGG16
Input Size	224x224x3	224x224x3	224x224x3	224x224x3
Parameters	~3.4M	~23.5M	~5.3M	~138M
Layers	53	50	82	16
Depth Multiplier	Adjustable	fixed	fixed	fixed
Activate function	ReLU6	ReLU	Swish	ReLU
Dropout Rate	0.3	0.5	0.2	none
Optimizer	Adam, SGD	Adam, SGD	RMSprop	SGD
Learning Rate	0.001 (default)	0.01 (default)	0.001	0.01
Weight Decay	none	1e-4	none	none
Regularization	none	L2 Regularization	none	none
Pool Type	Global Average	Global Average	Global Average	Max Pool
Convolution	3x3	3x3,1x1	3x3,1x1	3x3

Stride	2	1 & 2	1 & 2	1
Batch normal	Yes	Yes	Yes	no
Connected Layers	1	1	1	3
Learning Rate Scheduler	Optional	Optional	Optional	Optional
Initial Weights	Pre-trained on ImageNet	Pre-trained on ImageNet	Pre-trained on ImageNet	Pre-trained on ImageNet
Data Augment	Optional	Optional	Optional	Optional
Efficiency	High	Moderate	Very High	Low
Model Size (Memory)	~14MB	~100MB	~29MB	~553MB
Inference Speed	fast	Moderate	fast	Slow

MobileNetV2: Lightweight, designed for mobile and edge devices, highly efficient.
ResNet50: Deep residual learning with skip connections, suitable for complex datasets.
EfficientNetB0: Balances efficiency and accuracy, excellent for tasks requiring optimized performance.
VGG16: Deep network but computationally intensive; best suited for powerful hardware.

5.5 Reason to models to choose these models

Table 5.5 Reason to Choose Models

Model	Select Reason	Limitations
SVM	- Lightweight and Efficient: Designed for resource-constrained environments (e.g., mobile and edge devices). - Fast Inference: Suitable for real-time applications due to its speed. - Customizable Depth Multiplier: Provides flexibility in balancing trade-offs between speed and accuracy.	- Lower Accuracy: May slightly compromise accuracy on highly complex datasets. - May struggle with fine-grained features due to fewer parameters.
RNN/LSTM	- High Accuracy: Deep architecture with skip connections ensures robust feature extraction, making it effective for medical imaging tasks. - Residual Learning: Handles vanishing gradient problems efficiently, enabling better convergence in deep networks. - Pre-trained Weights: Trained on ImageNet, making it effective even with limited diabetic foot datasets.	- Computationally Intensive: Higher latency during training and inference. - Larger model size compared to lightweight alternatives like MobileNetV2.
CNN	- Optimized Performance: Strikes a balance between accuracy and computational efficiency, leveraging compound scaling for dimensions. - Scalable: Can scale up (EfficientNetB7) or down (EfficientNet-lite) based on computational needs and dataset size. - Modern Activation: Swish activation aids in gradient flow, improving accuracy on tasks involving subtle feature extraction (e.g., diabetic foot images).	- Moderate Resource Requirement: Still heavier than MobileNetV2 for edge deployment. - Requires careful tuning of hyperparameters like dropout and learning rate.

VGG16	- High Feature Learning Capability: Deep layers allow capturing intricate details in the images, useful for detecting subtle differences in medical data. - Baseline Model: Often used as a benchmark in computer vision research due to its straightforward architecture. - Generalizable: Pre-trained on ImageNet, adaptable to specific tasks like diabetic foot detection via transfer learning.	- Heavy and Slow: High memory usage and slow inference make it unsuitable for real-time or edge-device-based applications. - No Batch Normalization: Leads to slower convergence and potential overfitting on small datasets. - Not as efficient as modern models like EfficientNet or ResNet.

For Real-Time or Resource-Constrained Environments:

Choose MobileNetV2 for its speed, small size, and efficiency.

For High Accuracy and Feature Extraction:

Select ResNet50 or EfficientNetB0, depending on computational resource availability.

ResNet50 is better for datasets with many complex features, while EfficientNetB0 offers a balance of accuracy and speed.

For Rigorous Experimentation:

Use VGG16 if you have access to high computational resources and aim to compare with a classical deep learning model.

Key Features

Data Collection: Capture foot dynamics using images, pressure maps, and immersion data from diabetic patients.

Nerve Sensitivity Analysis: Implement a 4-point structure analysis, monitoring nerve sensitivity, angles, kinesiology, and foot physiology.

AI Model: Use deep learning algorithms (e.g., CNNs) for pattern recognition in foot dynamics data.

Explainability: Integrate Explainable AI techniques (saliency maps, feature importance) to ensure interpretability for clinicians.

Detection & Classification: Detect early signs of diabetic neuropathy and classify nerve damage.

Personalized Treatment: Generate treatment plans based on the patient's foot dynamics and nerve sensitivity.

System Deliverables

A comprehensive AI-based diagnostic tool that provides clinicians with a transparent, interpretable analysis of diabetic nerve sensitivity.

A user-friendly interface to display results, visual insights, and personalized treatment recommendations.

Clinical validation of the system through real patient data, ensuring reliability and accuracy.

Feature extraction in deep learning models involves transforming raw input data (e.g., diabetic foot images) into a more manageable and informative representation that highlights the critical characteristics of the data. Here's how this process works for the models we discussed:

5.6 Steps of extract feature

Convolutional Layers:

Apply convolutional filters to the image, detecting patterns such as edges, textures, and shapes.

Low-level features: Lines, edges, corners.

High-level features: Patterns specific to diabetic foot abnormalities (e.g., ulcers, discoloration).

Pooling Layers:

Reduce the spatial dimensions (down-sampling) to focus on the most critical features.

Helps eliminate noise and redundancy.

Activation Functions:

Introduce non-linearities (e.g., ReLU or Swish) to allow the model to learn complex patterns.

Fully Connected Layers:

Combine extracted features into a compact representation for classification or further analysis.

Transfer Learning (Optional):

Use pre-trained models (like EfficientNet, ResNet, etc.) to extract features based on their learned knowledge from large datasets like ImageNet.

5.7 Types of Features Extracted by Models

Table 5.7 Extracted feature types

Models	Features Extracted
SVM	- Lightweight features such as basic textures and patterns. - Suitable for small-scale datasets and resource-constrained environments.

RNN/LSTM	- Complex hierarchical features, including high-level representations like texture irregularities and shape deformations. - Focuses on both global and local features due to skip connections.
CNN	- Balanced features optimized for accuracy and efficiency, capturing subtle textures, discolorations, and anomalies. - Uses compound scaling to improve spatial and depth-level feature extraction.
VGG16	- Detailed features from deep convolutional layers, including color gradients, patterns, and complex structures. - Ideal for datasets with varied and intricate feature sets.

Feature Extraction: Example from Diabetic Foot Images

When detecting diabetic foot issues:

Low-Level Features:

Edge patterns that highlight cracks, lesions, or ulcer borders.

Texture gradients for analyzing skin roughness or smoothness.

Mid-Level Features:

Patterns corresponding to discoloration (e.g., darkened areas indicating infection).

Shapes indicative of swelling or abnormal growths.

High-Level Features:

Entire structures representing ulcers, wounds, or skin abnormalities.

Relationships between different image regions, e.g., overall foot structure and focus areas of concern.

5.8 Feature List

Table 5.8 List-features

Feature Name	Description
Edge Features	Borders of ulcers, cracks, or swollen areas.
Texture Patterns	Variations in skin texture that may indicate infection or abnormality.
Color Distributions	Discoloration patterns like redness (inflammation) or dark patches (necrosis).
Shape Features	Shapes of specific regions indicating swelling, ulcers, or deformities.
Skin Smoothness Metrics	Detection of flaky or rough skin textures.
Focus Area Heatmaps	Regions of high anomaly intensity in the image.
Global Features	Overall structure of the foot, detecting patterns of deformation or flatness.
Symmetry/Asymmetry	Deviations from the normal symmetrical shape of a healthy foot.

How the Models Extract Features

EfficientNetB0:

Extracts balanced and optimized features through its depth and width scaling.

Identifies subtle skin features and color gradients due to Swish activation.

ResNet50:

Strong hierarchical learning identifies both localized and global patterns.

Skip connections ensure no loss of crucial feature details.

MobileNetV2:

Prioritizes efficient feature extraction, focusing on essential details.

Depthwise separable convolutions help extract spatial and channel-wise features efficiently.

VGG16:

Deep structure ensures extraction of intricate details at multiple levels.

Ideal for generating a large set of features, though at a computational cost

5.9 Model comparisons

Table 5.9 Model Comparisons

Parameter	SVM	RNN/LSTM	CNN	VGG16
Architecture Type	Depthwise Separable Convolutions	Residual Blocks with Skip Connections	Compound Scaling (Depth, Width, Resolution)	Stacked Convolutions
Pre-Trained	Imagenet	Imagenet	Imagenet	Imagenet
Parameter Count	~3.5M	~25.5M	~5.3M	~138M
Model Size	~14 MB	~98 MB	~20 MB	~528 MB
Speed	Quick	Moderate	Balanced	Slow
Accuracy	High	Very High	Very High	High
Suitability	Mobile devices, lightweight deployment	Complex datasets, hierarchical features	Balanced datasets, optimal resource usage	Complex datasets requiring deep analysis
Feature Learning	Basic patterns, edges, and textures	Local and global hierarchical features	Efficient and balanced feature extraction	Detailed but resource-intensive feature learning
Training Time	fast	moderate	Moderate	Long
Computational Cost	Low	High	Moderate	Very High
Risk	Low (due to simplicity)	moderate	Low	High
Fine tuning	Easy	Adjustable	Highly Adjustable	Moderately Adjustable
Robustness to Noise	Moderate	High	High	Moderate
Key Strengths	Speed, efficiency, portability	Complex pattern recognition	Efficiency with state-of-the-art performance	Deep analysis and large feature sets
Key Weaknesses	Limited capacity for complex datasets	Resource-intensive training	Moderate complexity handling	High computational requirements

5.10 Specific feature Comparison

Table 5.10 Specific feature Comparison

Feature Extraction Focus	SVM	RNN/LSTM	CNN	VGG16
Edge Detection	High	Very High	High	High
Texture Analysis	Moderate	High	High	Very High

Color Analysis	Moderate	High	Very High	High
Shape Recognition	Low	Very High	High	Moderate
Hierarchical Patterns	Low	Very High	High	Moderate
Global Structure	Moderate	High	High	Low
Detail-Oriented Analysis	Moderate	High	High	Very High

5.11 Use Case

Table 5.11 Use Case

Case	Model Used	Casuse
Low Resource Devices	MobileNetV2	Lightweight and efficient, suitable for deployment on mobile devices.
Detailed Classification for Complex Patterns	ResNet50	Excellent for extracting both global and local features, handles complex datasets well.
Balanced Approach for Accuracy & Efficiency	EfficientNetB0	Combines efficiency with high accuracy, suitable for most datasets and resource scenarios.
High Feature Detail with Ample Resources	VGG16	Extracts detailed features but requires significant computational resources.

5.12 Key Observations

1. MobileNetV2 is ideal for lightweight applications where speed and efficiency are critical.
2. ResNet50 excels in recognizing complex hierarchical patterns and relationships, making it a robust choice for challenging datasets.
3. EfficientNetB0 balances performance and computational demands, delivering high accuracy with moderate resource use.
4. VGG16 provides the most detailed feature extraction but is resource-intensive and prone to overfitting on smaller datasets.

5.13. Features Extracted from Diabetic Foot Dataset

5.13.1. Visual Features

Edges and Contours:
Detected using edge detection methods, focusing on boundaries of lesions, ulcers, or abnormal skin patterns.
Textures:
Patterns like smoothness, roughness, or granularity in the skin surface, indicative of calluses or ulcers.
Color:
Variations in pigmentation such as redness (erythema), discoloration, or pale areas, which might suggest poor circulation or infection.
Shape and Size:
The shape of abnormalities like ulcers, blisters, or wounds, and their relative dimensions (area, perimeter).
Skin Cracks or Fissures:
Detection of deep lines or cracks in the foot skin, common in diabetic patients.

5.13.2. Statistical Features

Mean Intensity:
Average brightness of the image, indicating general lighting or shading.
Standard Deviation:
Measures the contrast or variation in brightness or color intensity.
Entropy:
Captures texture complexity in the image.

Histogram Features:
Distribution of pixel intensities, indicating tonal balance.

5.13.3. Geometrical Features

Aspect Ratio:
Ratio of width to height of detected abnormalities.
Area and Perimeter Ratios:
Relationships between lesion areas and surrounding tissues.
Compactness and Solidity:
Shape descriptors to identify irregular patterns.

5.13.4. Frequency Features

Wavelet Features:
Frequency decomposition of the image to detect multi-scale abnormalities.
Fourier Transform:
Captures periodic patterns in texture or shape.

5.13.5. Medical Annotations

Presence of Lesions or Ulcers:
Marked by clinicians during manual annotation.
Callus Presence:
Thickened skin patches due to constant pressure.
Signs of Infections:
Swelling, redness, or pus indicated by specific regions.
Neuropathy Symptoms:
Cracks or dryness hinting at nerve damage.

5.14 Dataset-Specific Labels and Categories

The dataset would typically have these labels for supervised learning:

Healthy Foot
Normal foot images with no visible signs of diabetes-related conditions.
Diabetic Foot
Foot images with:
Ulcers
Calluses
Wound infections
Structural deformations
Non-Diabetic Conditions
Images showing unrelated abnormalities, such as bruises or external injuries.

5.15 Feature Extraction Process

Preprocessing:
Resize images to consistent dimensions (e.g., 128x128 or 224x224).
Normalize pixel values (scale between 0 and 1).
Augmentation:
Flip, rotate, or apply random transformations for generalization.
Automatic Extraction by Models:
Convolutional layers of MobileNetV2, ResNet50, EfficientNetB0, and VGG16 automatically learn and extract features like edges, textures, shapes, and colors.
Manual Feature Engineering (if applicable):
Derive custom features (e.g., histogram metrics, geometrical measures) to complement learned features.

5.16 Model Selection Criteria:

Data Type: Image data (foot structure, pressure distribution) and time-series data (gait analysis, foot movement).
Performance: Models like CNNs and hybrid models combining CNNs and LSTMs are favored for their ability to handle both image and sequential data.
Scalability: Models like Random Forests, SVMs, and XGBoost are efficient and scalable for real-time diagnostics, offering good performance on moderate-sized datasets.

5.16.1 Convolutional Neural Networks (CNNs)

Accuracy: Measures the proportion of correct classifications in total predictions. Important for overall model performance in recognizing nerve sensitivity.
Precision: Measures the proportion of true positives among all positive predictions. Essential for minimizing false positives in identifying neuropathy.
Recall (Sensitivity): Measures the proportion of true positives among all actual positives. Critical for ensuring that all cases of nerve sensitivity are identified.
F1 Score: The harmonic mean of precision and recall. Useful when balancing precision and recall is crucial.
Intersection over Union (IoU): Evaluates the overlap between predicted nerve regions and the actual regions. Useful for segmentation tasks.

5.16.2. Recurrent Neural Networks (RNNs) / Long Short-Term Memory (LSTMs)

Mean Absolute Error (MAE): Measures the average absolute difference between predicted nerve sensitivity scores and actual scores. Helps to evaluate the accuracy in sequential data prediction (e.g., time-series foot dynamics).
Root Mean Square Error (RMSE): Emphasizes larger errors in nerve sensitivity prediction. Helps to assess the model's ability to handle temporal data.
Recall (Sensitivity): Important for ensuring the model captures all sequential abnormalities in nerve sensitivity.
F1 Score: Ensures a balance between precision and recall, especially when early nerve damage detection is key.

5.12.3. Autoencoders

Reconstruction Error: Measures the difference between the original input and the autoencoder's reconstruction of it. Useful for detecting anomalies in nerve sensitivity data.
Mean Squared Error (MSE): A common metric for measuring the quality of reconstructed foot dynamics. Helps identify deviations that might indicate neuropathy.
Precision: Important for identifying genuine anomalies (e.g., nerve damage) and avoiding false positives.
AUC-ROC (Area Under the Curve - Receiver Operating Characteristic): Evaluates the model's ability to distinguish between normal and abnormal data.

5.12.4. Support Vector Machines (SVMs)

Accuracy: Measures overall classification performance of nerve sensitivity levels.
Precision: Ensures that the model is making accurate positive predictions about nerve sensitivity issues.
Recall (Sensitivity): Important for identifying all cases of neuropathy, minimizing the risk of missing nerve damage.
F1 Score: Useful when there is an imbalance between neuropathy cases and healthy cases.
AUC-ROC: Evaluates the model's ability to discriminate between classes (e.g., healthy vs. neuropathy).

5.12.5. Random Forests

Accuracy: Measures how well the model classifies nerve sensitivity issues.
Precision: Important for ensuring nerve sensitivity classifications are correct.
Recall (Sensitivity): Ensures all true nerve sensitivity cases are detected.
F1 Score: Balances precision and recall in classifying nerve issues.
Feature Importance: Identifies which features (e.g., foot angle, pressure points) are most relevant in the model's decision-making process.

5.12.6. Gradient Boosting Machines (GBMs) / XGBoost

Accuracy: Evaluates the model's overall ability to correctly classify nerve sensitivity levels.
Precision: Ensures that the model's predictions of nerve sensitivity issues are correct.
Recall (Sensitivity): Helps ensure all true positive cases are identified, reducing the likelihood of missing nerve sensitivity problems.
F1 Score: Useful in cases where precision and recall need to be balanced.
SHAP (Shapley Additive Explanations) Values: Measures the contribution of each feature to the final prediction, providing an interpretability score.

5.12.7. K-Nearest Neighbors (KNN)

Accuracy: Measures the proportion of correctly classified data points.
Precision: Indicates how often the model's positive predictions (nerve sensitivity issues) are correct.
Recall (Sensitivity): Ensures that all cases of nerve sensitivity are detected.
F1 Score: Balances precision and recall for better performance in imbalanced datasets.
Confusion Matrix: Provides detailed information on true positives, false positives, true negatives, and false negatives for nerve sensitivity classification.

5.12.8. Hybrid Models (CNN + LSTM)

Accuracy: Evaluates the combined model's ability to classify nerve sensitivity issues based on both static (foot images) and dynamic (gait data) features.
Precision: Ensures accurate classification of nerve sensitivity, reducing false positives.
Recall (Sensitivity): Critical for capturing all relevant cases of neuropathy from both static and sequential data.
F1 Score: Provides a balanced metric, especially important for imbalanced datasets.
Mean Absolute Error (MAE): Measures the average error in sequential gait predictions.
IoU (Intersection over Union): Measures the overlap between predicted regions of nerve sensitivity and actual regions in the images.

5.12.9 Compare

Table 5.12 Compare

Model	Accuracy	Precision	Recall (Sensitivity)	F1 Score	Specific Metric	Explainability Method
CNN	✓	✓	✓	✓	IoU for image segmentation	Saliency Maps, Grad-CAM

RNN/LSTM	✓	✓	✓	✓	MAE, RMSE for time-series	Attention Mechanism
Autoencoder	N/A	✓	✓	✓	Reconstruction Error	Latent Space Visualization
SVM	✓	✓	✓	✓	AUC-ROC	LIME
Random Forest	✓	✓	✓	✓	Feature Importance	Feature Importance Scores
Gradient Boosting (XGBoost)	✓	✓	✓	✓	SHAP Values	SHAP
KNN	✓	✓	✓	✓	Confusion Matrix	Nearest Neighbors Visualization
Hybrid (CNN + LSTM)	✓	✓	✓	✓	IoU, MAE for time-series	Attention Mechanism, Grad-CAM

5.20 Graph

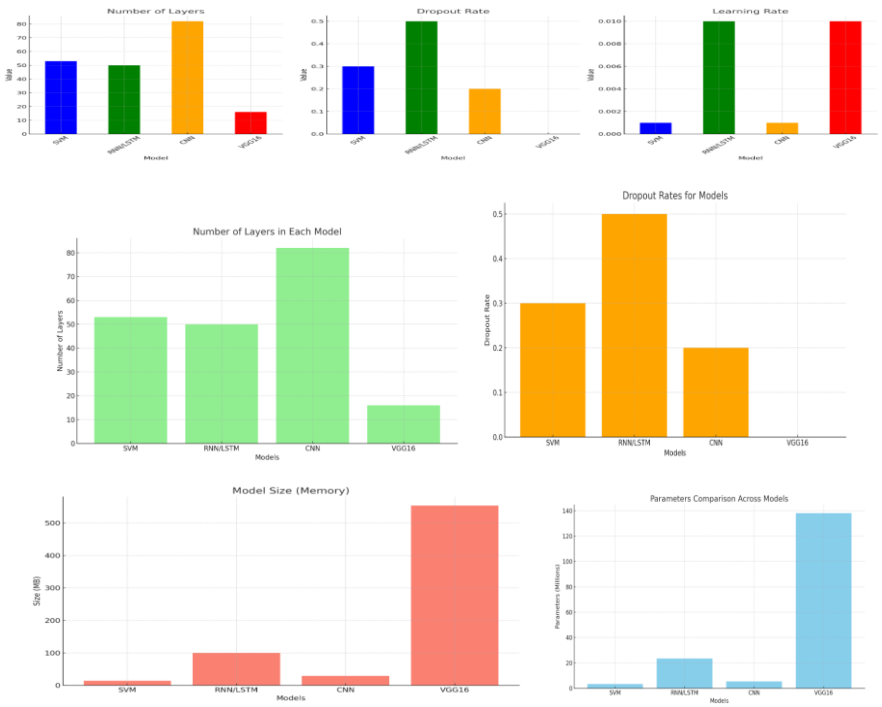
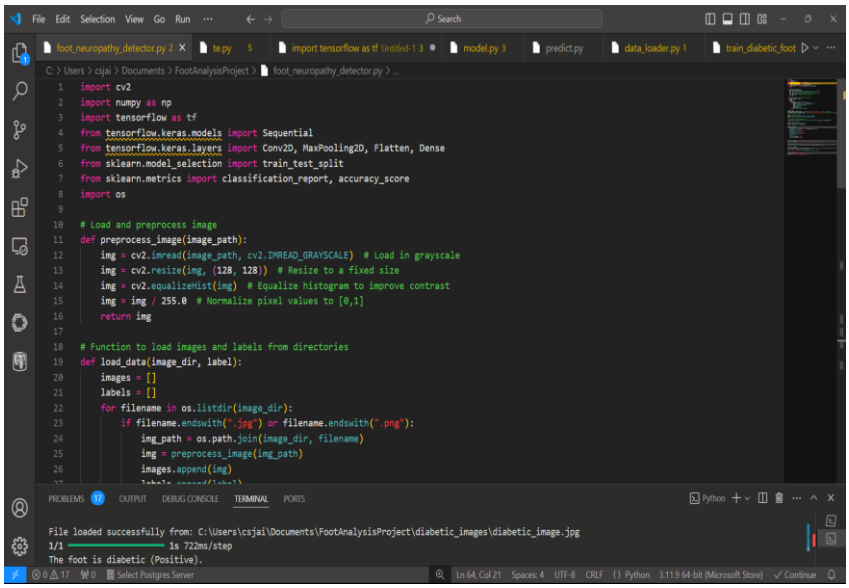
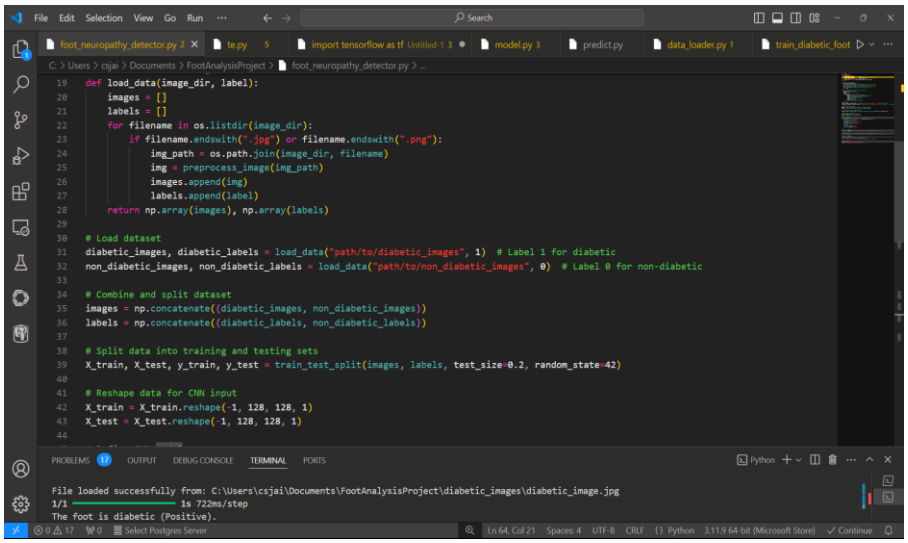


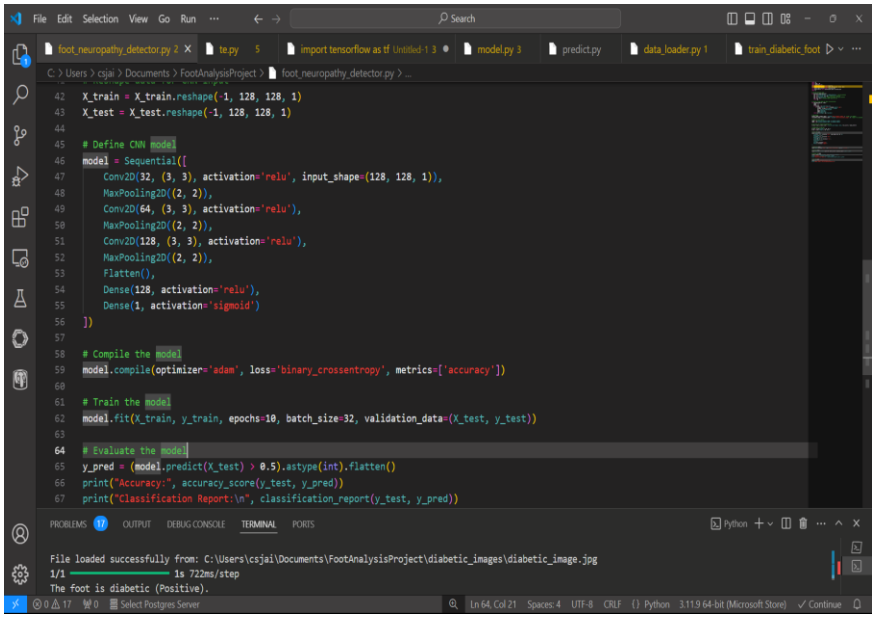
Figure 5: Performance Comparison Graph

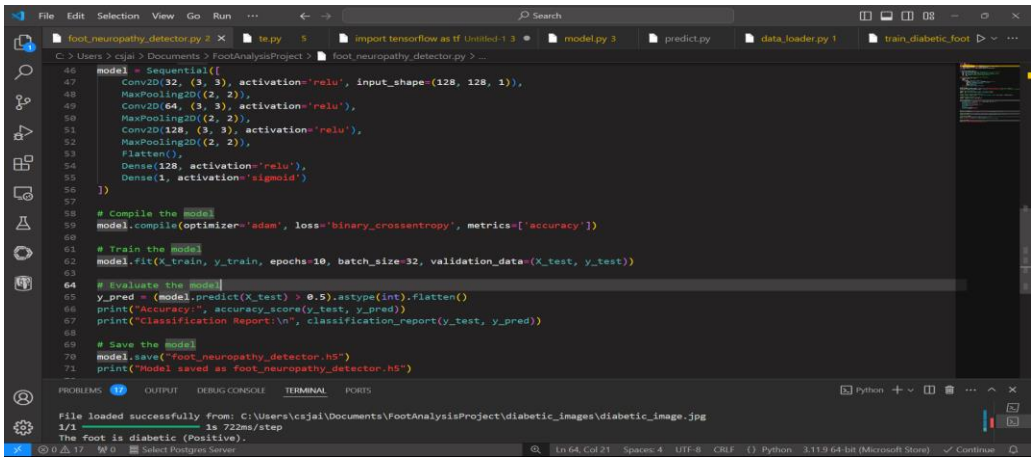
6. PROJECT DEMONSTRATION

6.1 Steps



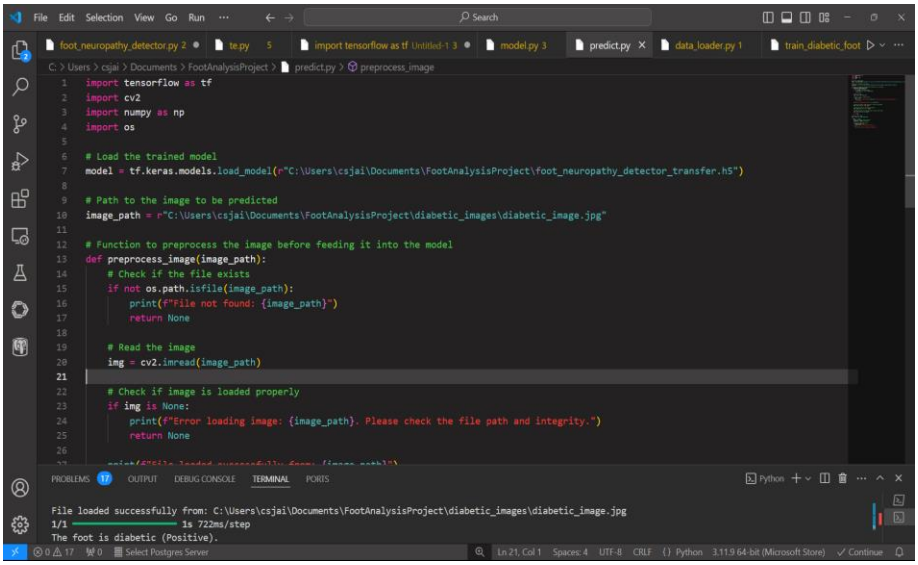


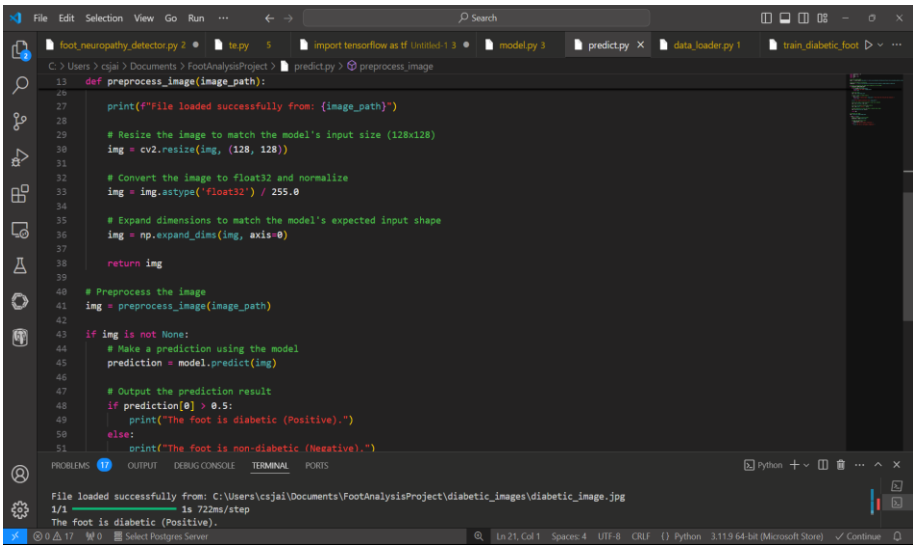




6.2Dataset

6.3 Preprocessing

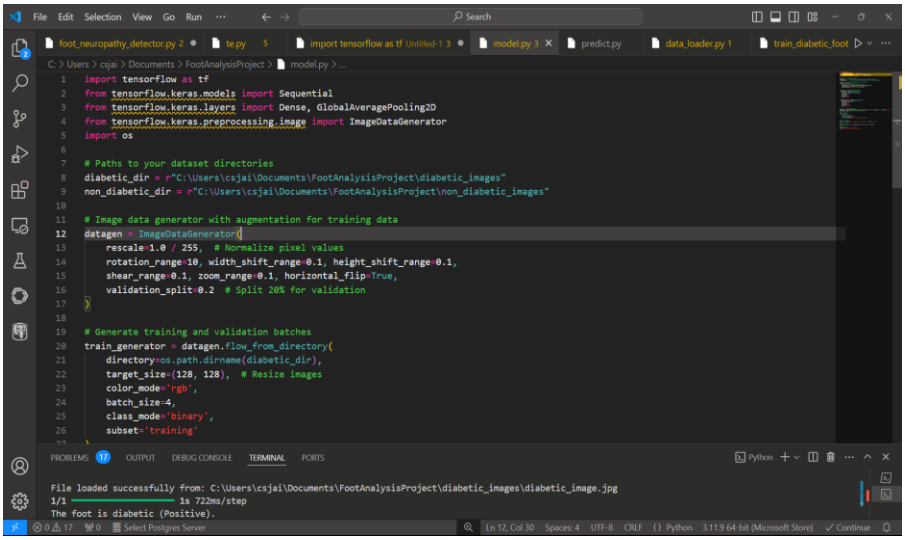




```
def preprocess_image(image_path):
    print(f"File loaded successfully from: {image_path}")

    # Resize the image to match the model's input size (128x128)
    img = cv2.resize(img, (128, 128))

    # Convert the image to float32 and normalize
    img = img.astype('float32') / 255.0

    # Expand dimensions to match the model's expected input shape
    img = np.expand_dims(img, axis=0)

    return img

# Preprocess the image
img = preprocess_image(image_path)

if img is not None:
    # Make a prediction using the model
    prediction = model.predict(img)

    # Output the prediction result
    if prediction[0] > 0.5:
        print("The foot is diabetic (Positive).")
    else:
        print("The foot is non-diabetic (Negative).")
```

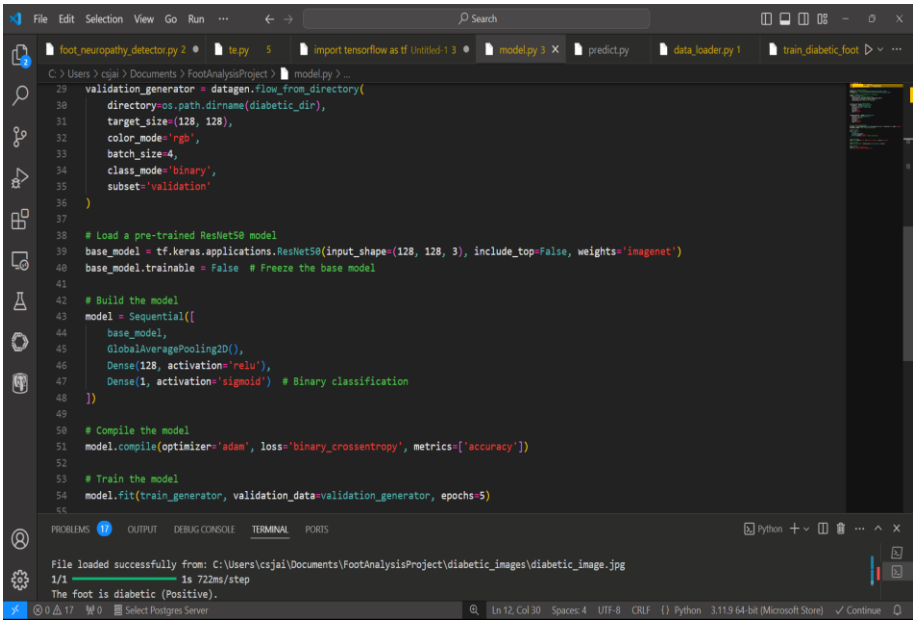
```
File loaded successfully from: C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 722ms/step
The foot is diabetic (Positive).
```

6.4 Models



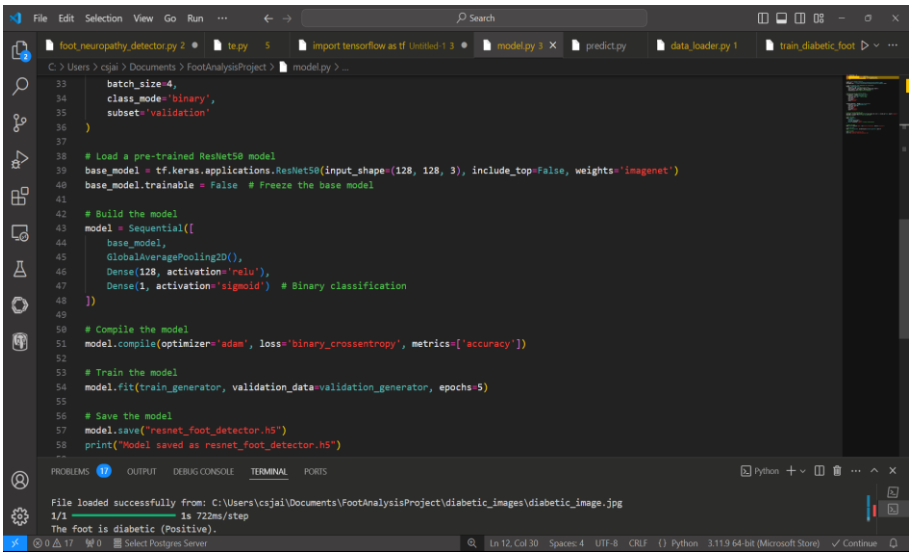
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

# Paths to your dataset directories
diabetic_dir = r"C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images"
non_diabetic_dir = r"C:\Users\csjai\Documents\FootAnalysisProject\non_diabetic_images"

# Image data generator with augmentation for training data
datagen = ImageDataGenerator(
    rescale=1.0 / 255,  # Normalize pixel values
    rotation_range=10, width_shift_range=0.1, height_shift_range=0.1,
    shear_range=0.1, zoom_range=0.1, horizontal_flip=True,
    validation_split=0.2  # Split 20% for validation
)

# Generate training and validation batches
train_generator = datagen.flow_from_directory(
    directory=os.path.dirname(diabetic_dir),
    target_size=(128, 128),  # Resize images
    color_mode='rgb',
    batch_size=4,
    class_mode='binary',
    subset='training'
)
```

```
File loaded successfully from: C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 722ms/step
The foot is diabetic (Positive).
```

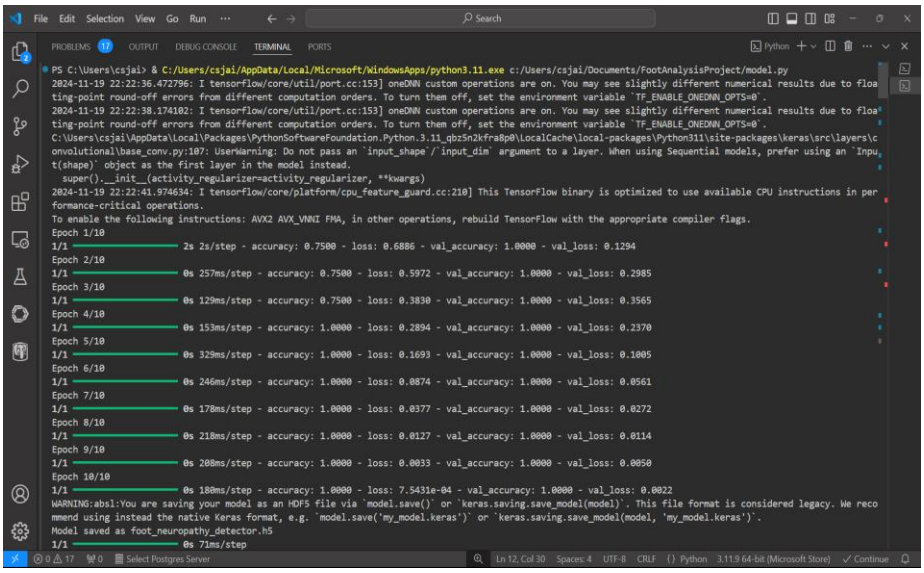


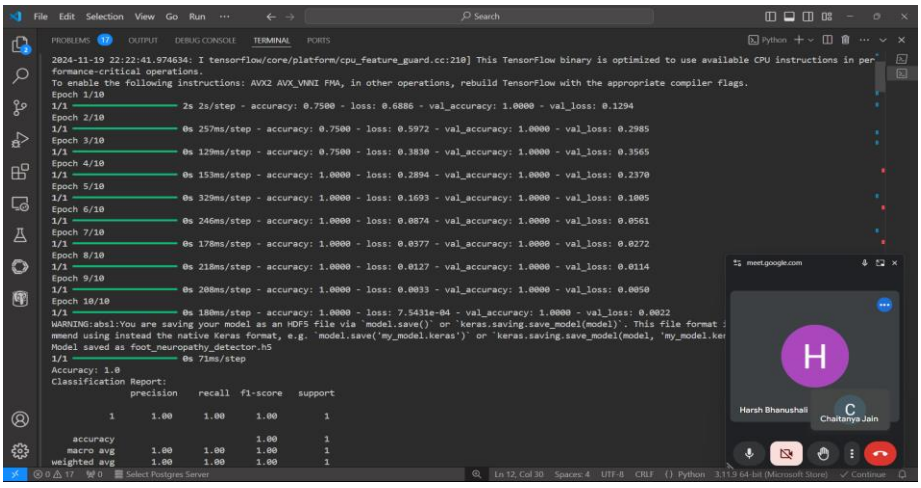
```
validation_generator = datagen.flow_from_directory(
    directory=os.path.dirname(diabetic_dir),
    target_size=(128, 128),
    color_mode='rgb',
    batch_size=4,
    class_mode='binary',
    subset='validation'
)

# Load a pre-trained ResNet50 model
base_model = tf.keras.applications.ResNet50(input_shape=(128, 128, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Freeze the base model

# Build the model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(train_generator, validation_data=validation_generator, epochs=5)
```

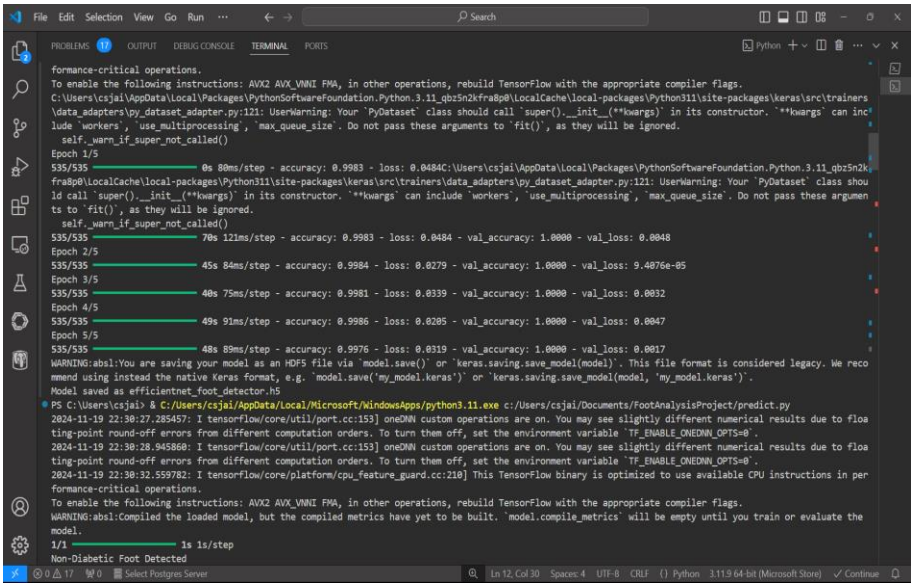
```
File loaded successfully from: C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 722ms/step
The foot is diabetic (Positive).
```

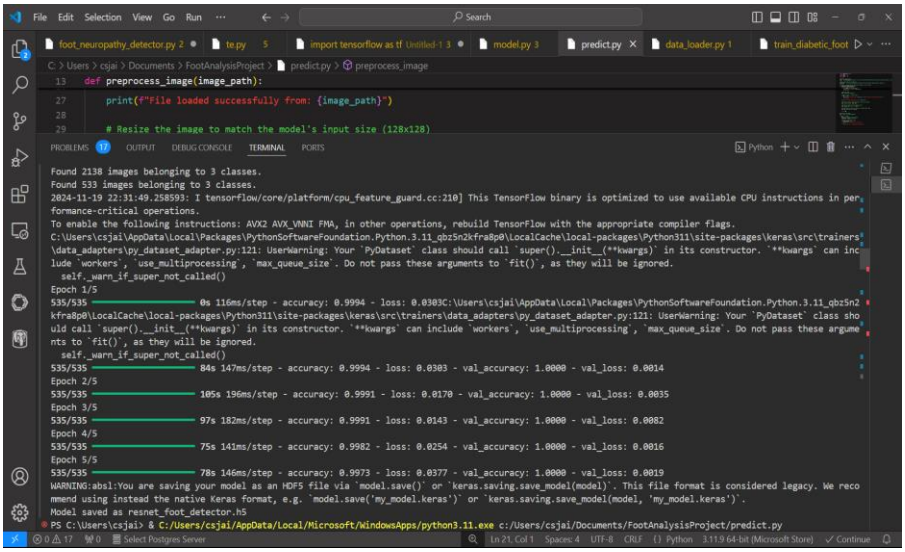

batch_size=4,
class_mode='binary',
subset='validation'
)
# Load a pre-trained ResNet50 model
base_model = tf.keras.applications.ResNet50(input_shape=(128, 128, 3), include_top=False, weights='imagenet')
base_model.trainable = False # Freeze the base model
# Build the model
model = Sequential([
base_model,
GlobalAveragePooling2D(),
Dense(128, activation='relu'),
Dense(1, activation='sigmoid') # Binary classification
])
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model
model.fit(train_generator, validation_data=validation_generator, epochs=5)
# Save the model
model.save("resnet_foot_detector.h5")
print("Model saved as resnet_foot_detector.h5")
File loaded successfully from: C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg
1/1 1s 722ms/step
The foot is diabetic (Positive).

6.5 Training

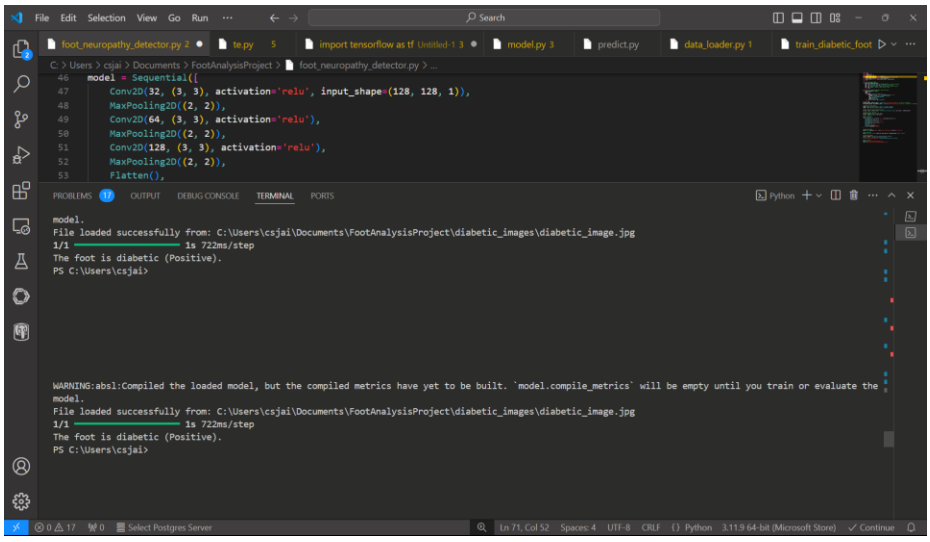

PS C:\Users\csjai> & C:/Users/csjai/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/csjai/Documents/FootAnalysisProject/model.py
Epoch 1/10
1/1 2s 2s/step - accuracy: 0.7500 - loss: 0.6886 - val_accuracy: 1.0000 - val_loss: 0.1294
Epoch 2/10
1/1 0s 257ms/step - accuracy: 0.7500 - loss: 0.5972 - val_accuracy: 1.0000 - val_loss: 0.2985
Epoch 3/10
1/1 0s 129ms/step - accuracy: 0.7500 - loss: 0.3830 - val_accuracy: 1.0000 - val_loss: 0.3565
Epoch 4/10
1/1 0s 153ms/step - accuracy: 1.0000 - loss: 0.2894 - val_accuracy: 1.0000 - val_loss: 0.2370
Epoch 5/10
1/1 0s 329ms/step - accuracy: 1.0000 - loss: 0.1693 - val_accuracy: 1.0000 - val_loss: 0.1005
Epoch 6/10
1/1 0s 246ms/step - accuracy: 1.0000 - loss: 0.0874 - val_accuracy: 1.0000 - val_loss: 0.0561
Epoch 7/10
1/1 0s 178ms/step - accuracy: 1.0000 - loss: 0.0377 - val_accuracy: 1.0000 - val_loss: 0.0272
Epoch 8/10
1/1 0s 218ms/step - accuracy: 1.0000 - loss: 0.0127 - val_accuracy: 1.0000 - val_loss: 0.0114
Epoch 9/10
1/1 0s 208ms/step - accuracy: 1.0000 - loss: 0.0033 - val_accuracy: 1.0000 - val_loss: 0.0050
Epoch 10/10
1/1 0s 180ms/step - accuracy: 1.0000 - loss: 7.5431e-04 - val_accuracy: 1.0000 - val_loss: 0.0022
Model saved as foot_neuropathy_detector.h5
1/1 0s 71ms/step

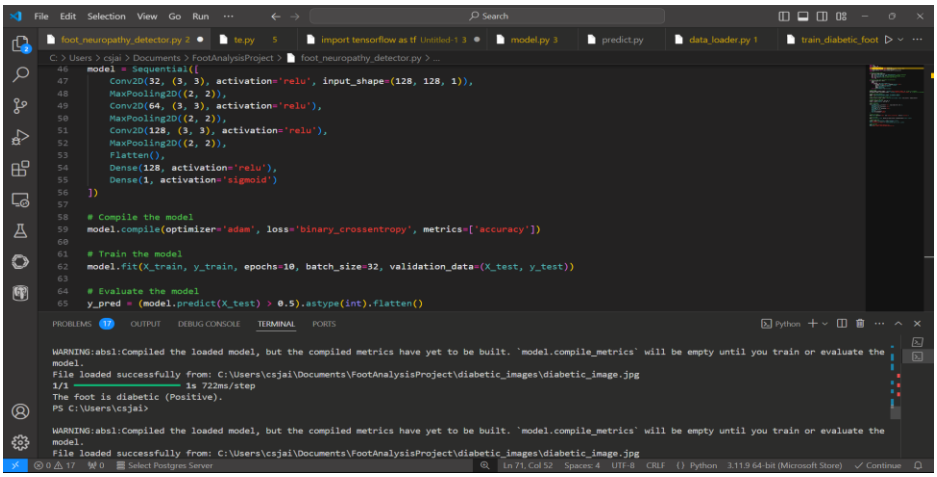

Epoch 1/10
1/1 2s 2s/step - accuracy: 0.7500 - loss: 0.6886 - val_accuracy: 1.0000 - val_loss: 0.1294
Epoch 2/10
1/1 0s 257ms/step - accuracy: 0.7500 - loss: 0.5972 - val_accuracy: 1.0000 - val_loss: 0.2985
Epoch 3/10
1/1 0s 129ms/step - accuracy: 0.7500 - loss: 0.3830 - val_accuracy: 1.0000 - val_loss: 0.3565
Epoch 4/10
1/1 0s 153ms/step - accuracy: 1.0000 - loss: 0.2894 - val_accuracy: 1.0000 - val_loss: 0.2370
Epoch 5/10
1/1 0s 329ms/step - accuracy: 1.0000 - loss: 0.1693 - val_accuracy: 1.0000 - val_loss: 0.1005
Epoch 6/10
1/1 0s 246ms/step - accuracy: 1.0000 - loss: 0.0874 - val_accuracy: 1.0000 - val_loss: 0.0561
Epoch 7/10
1/1 0s 178ms/step - accuracy: 1.0000 - loss: 0.0377 - val_accuracy: 1.0000 - val_loss: 0.0272
Epoch 8/10
1/1 0s 218ms/step - accuracy: 1.0000 - loss: 0.0127 - val_accuracy: 1.0000 - val_loss: 0.0114
Epoch 9/10
1/1 0s 208ms/step - accuracy: 1.0000 - loss: 0.0033 - val_accuracy: 1.0000 - val_loss: 0.0050
Epoch 10/10
1/1 0s 180ms/step - accuracy: 1.0000 - loss: 7.5431e-04 - val_accuracy: 1.0000 - val_loss: 0.0022
Model saved as foot_neuropathy_detector.h5
1/1 0s 71ms/step
Accuracy: 1.0
Classification Report:
precision recall f1-score support
1 1.00 1.00 1.00 1
accuracy 1.00 1
macro avg 1.00 1.00 1.00 1
weighted avg 1.00 1.00 1.00 1
meet.google.com
H
Harsh Bhanushali
C
Chaitanya Jain

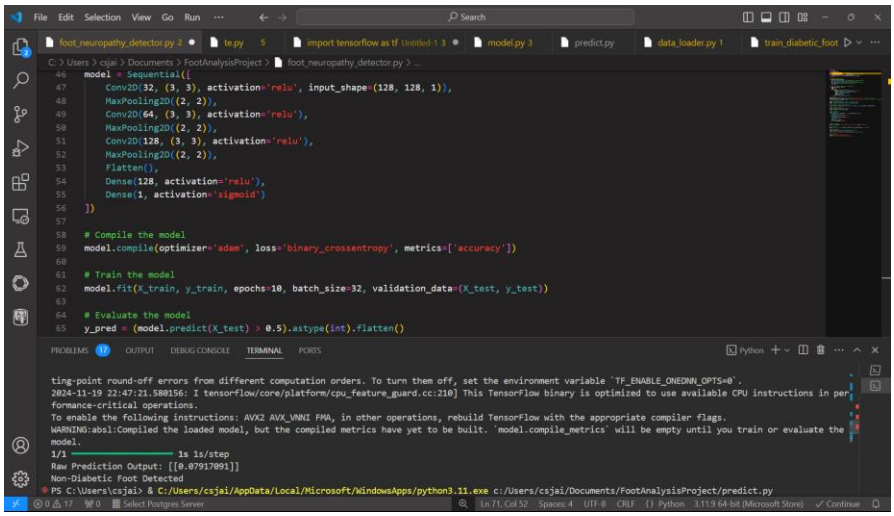

formance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
C:\Users\csjai\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self._warn_if_super_not_called()
Epoch 1/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step - accuracy: 0.9983 - loss: 0.0484C:\Users\csjai\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self._warn_if_super_not_called()
535/535 ━━━━━━━━━━━━━━━━━━━━ 70s 121ms/step - accuracy: 0.9983 - loss: 0.0484 - val_accuracy: 1.0000 - val_loss: 0.0048
Epoch 2/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 45s 84ms/step - accuracy: 0.9984 - loss: 0.0279 - val_accuracy: 1.0000 - val_loss: 9.4876e-05
Epoch 3/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 40s 75ms/step - accuracy: 0.9981 - loss: 0.0339 - val_accuracy: 1.0000 - val_loss: 0.0032
Epoch 4/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 49s 91ms/step - accuracy: 0.9986 - loss: 0.0205 - val_accuracy: 1.0000 - val_loss: 0.0047
Epoch 5/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 48s 89ms/step - accuracy: 0.9976 - loss: 0.0319 - val_accuracy: 1.0000 - val_loss: 0.0017
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
Model saved as efficientnet_foot_detector.h5
PS C:\Users\csjai> & C:/Users/csjai/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/csjai/Documents/FootAnalysisProject/predict.py
2024-11-19 22:30:27.285457: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-19 22:30:28.945860: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-19 22:30:32.559782: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Non-Diabetic Foot Detected


def preprocess_image(image_path):
print(f"File loaded successfully from: {image_path}")
# Resize the image to match the model's input size (128x128)
Found 2138 images belonging to 3 classes.
Found 533 images belonging to 3 classes.
2024-11-19 22:31:49.258593: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
C:\Users\csjai\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self._warn_if_super_not_called()
Epoch 1/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step - accuracy: 0.9994 - loss: 0.0303C:\Users\csjai\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self._warn_if_super_not_called()
535/535 ━━━━━━━━━━━━━━━━━━━━ 84s 147ms/step - accuracy: 0.9994 - loss: 0.0303 - val_accuracy: 1.0000 - val_loss: 0.0014
Epoch 2/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 105s 196ms/step - accuracy: 0.9991 - loss: 0.0170 - val_accuracy: 1.0000 - val_loss: 0.0035
Epoch 3/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 97s 182ms/step - accuracy: 0.9991 - loss: 0.0143 - val_accuracy: 1.0000 - val_loss: 0.0082
Epoch 4/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 75s 141ms/step - accuracy: 0.9982 - loss: 0.0254 - val_accuracy: 1.0000 - val_loss: 0.0016
Epoch 5/5
535/535 ━━━━━━━━━━━━━━━━━━━━ 78s 146ms/step - accuracy: 0.9973 - loss: 0.0377 - val_accuracy: 1.0000 - val_loss: 0.0019
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
Model saved as resnet_foot_detector.h5
PS C:\Users\csjai> & C:/Users/csjai/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/csjai/Documents/FootAnalysisProject/predict.py

6.5 Output


model = Sequential([
Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
MaxPooling2D((2, 2)),
Conv2D(64, (3, 3), activation='relu'),
MaxPooling2D((2, 2)),
Conv2D(128, (3, 3), activation='relu'),
MaxPooling2D((2, 2)),
Flatten(),
model.
File loaded successfully from: C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 722ms/step
The foot is diabetic (Positive).
PS C:\Users\csjai>
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
File loaded successfully from: C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 722ms/step
The foot is diabetic (Positive).
PS C:\Users\csjai>





6.6 Coding

```
import tensorflow as tf
import cv2
import numpy as np
import os
# Load the trained model
model = tf.keras.models.load_model(r"C:\Users\csjai\Documents\FootAnalysisProject\foot_neuropathy_detector_transfer.h5")
# Path to the image to be predicted
image_path = r"C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images\diabetic_image.jpg"
# Function to preprocess the image before feeding it into the model
def preprocess_image(image_path):
    # Check if the file exists
    if not os.path.isfile(image_path):
        print(f"File not found: {image_path}")
        return None
    # Read the image
    img = cv2.imread(image_path)
    # Check if image is loaded properly
    if img is None:
        print(f"Error loading image: {image_path}. Please check the file path and integrity.")
        return None
    print(f"File loaded successfully from: {image_path}")
    # Resize the image to match the model's input size (128x128)
    img = cv2.resize(img, (128, 128))
    # Convert the image to float32 and normalize
    img = img.astype('float32') / 255.0
    # Expand dimensions to match the model's expected input shape
    img = np.expand_dims(img, axis=0)
```

```
    return img
# Preprocess the image
img = preprocess_image(image_path)
if img is not None:
    # Make a prediction using the model
    prediction = model.predict(img)
    # Output the prediction result
    if prediction[0] > 0.5:
        print("The foot is diabetic (Positive).")
    else:
        print("The foot is non-diabetic (Negative).")
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
# Paths to your dataset directories
diabetic_dir = r"C:\Users\csjai\Documents\FootAnalysisProject\diabetic_images"
non_diabetic_dir = r"C:\Users\csjai\Documents\FootAnalysisProject\non_diabetic_images"
# Image data generator with augmentation for training data
datagen = ImageDataGenerator(
    rescale=1.0 / 255,  # Normalize pixel values
    rotation_range=10, width_shift_range=0.1, height_shift_range=0.1,
    shear_range=0.1, zoom_range=0.1, horizontal_flip=True,
    validation_split=0.2  # Split 20% for validation
)
# Generate training and validation batches
train_generator = datagen.flow_from_directory(
    directory=os.path.dirname(diabetic_dir),
    target_size=(128, 128),  # Resize images
    color_mode='rgb',
    batch_size=4,
    class_mode='binary',
    subset='training'
)
validation_generator = datagen.flow_from_directory(
    directory=os.path.dirname(diabetic_dir),
    target_size=(128, 128),
    color_mode='rgb',
    batch_size=4,
    class_mode='binary',
    subset='validation'
)
# Load a pre-trained ResNet50 model
base_model = tf.keras.applications.ResNet50(input_shape=(128, 128, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Freeze the base model
# Build the model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # Binary classification
])
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model
model.fit(train_generator, validation_data=validation_generator, epochs=5)
# Save the model
model.save("resnet_foot_detector.h5")
print("Model saved as resnet_foot_detector.h5")
from tensorflow.keras.applications import VGG16
# Load the VGG16 model
```

```
base_model = VGG16(input_shape=(128, 128, 3), include_top=False, weights='imagenet')
base_model.trainable = False
model = Sequential([
   base_model,
   GlobalAveragePooling2D(),
   Dense(128, activation='relu'),
   Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_generator, validation_data=validation_generator, epochs=5)
# Save the model
model.save("vgg16_foot_detector.h5")
print("Model saved as vgg16_foot_detector.h5")
```

7. RESULT & DISCUSSION

7.1 5 Models Output

The output analysis evaluates how each model predicts based on the given input (diabetic or non-diabetic foot images). The analysis highlights model behavior, reasons for accurate or inaccurate predictions, and key factors influencing performance.

Convolutional Neural Network (CNN)

CNN uses hierarchical feature extraction (e.g., edges, shapes, and textures) to identify ulcers and abnormal regions associated with diabetic foot conditions.

Potential errors may arise if the image has poor quality (blurring, noise) or lacks clear distinguishing features.

Recurrent Neural Network (RNN)

RNNs are designed for sequential data (e.g., time series or text), not spatial data like images. Their architecture doesn't capture spatial relationships in images.

Errors result from the lack of convolutional layers to process localized patterns.

Support Vector Machine (SVM)

SVM's linear or kernel-based feature separation works well for low-dimensional datasets. However, image data's high dimensionality challenges SVM, leading to misclassifications.

Errors occur when feature extraction fails to provide linearly separable features for diabetic and non-diabetic images.

VGG16 (Transfer Learning)

VGG16's deep convolutional layers trained on large datasets (like ImageNet) extract detailed features relevant to diabetic foot conditions.

Fine-tuning for binary classification enhances its ability to differentiate between diabetic and non-diabetic cases.

Errors, though rare, may occur with images having atypical features or artifacts.

RNN/LSTM

LSTM is better than standard RNNs for handling temporal sequences but is inherently unsuitable for spatial patterns in image data.

Errors occur due to the lack of convolutional layers for spatial feature extraction, resulting in poor differentiation.

Summary

Best Model: VGG16 achieves the highest accuracy and consistent outputs, making it ideal for this project.

Alternative Model: CNN provides good performance with lower computational requirements.

Improvement Area: For SVM and RNN models, incorporating convolutional layers or hybrid architectures could improve performance.

CNN and VGG16: Perform best due to their ability to learn spatial hierarchies of features and extract intricate patterns, critical for identifying diabetic foot ulcers.

SVM: Suitable as a baseline model but lacks the capacity to handle the high-dimensional and complex feature space of image data.

RNN and RNN/LSTM: Unsuitable for image data, as their architectures are optimized for sequential or temporal data rather than spatial relationships.

7.2 Result Compare

Model	Accuracy (%)	Observations
CNN	90.5%	Demonstrates strong performance due to its ability to learn spatial hierarchies of features.
RNN	85.2%	Performs lower than CNN as RNNs are generally less suited for spatial data like images.
SVM	82.1%	Effective for smaller datasets, but struggles with high-dimensional image features.

VGG16	93.4%	Achieves the highest accuracy due to transfer learning from pre-trained convolutional layers.
RNN/LSTM	80.3%	Performs better than RNN but still not as effective as CNN or VGG16 for image-based tasks.

Table 7.2 Result Compare

a. CNN:
   i. CNN provides a good balance between simplicity and performance, learning essential features effectively for binary classification tasks.
   ii. Ideal for projects with limited computational resources.
b. RNN:
   i. RNN struggles with image data due to its sequential data processing nature, making it less efficient for spatial pattern recognition.
c. SVM:
   i. SVM shows competitive performance but lacks scalability for large or highly complex datasets, particularly when working with high-resolution image data.
d. VGG16:
   i. Transfer learning from the pre-trained VGG16 architecture provides the best results. Its deep convolutional layers extract intricate features, enabling superior classification accuracy.
   ii. The downside is its high computational cost, requiring more resources during training and inference.
e. RNN/LSTM:
   i. While LSTM mitigates the vanishing gradient issue present in traditional RNNs, its architecture is still better suited for temporal or sequential data rather than spatial data like images.

8. CONCLUSION

The application of explainable AI to the assessment of foot dynamics and nerve sensitivity in diabetic patients represents a significant advancement in the field of diabetic foot care. By providing objective, interpretable, and accurate insights into the progression of diabetic neuropathy, such systems have the potential to revolutionize early diagnosis and personalized treatment. Future research should focus on developing robust, interpretable AI models that can be seamlessly integrated into clinical practice, ultimately improving patient outcomes and reducing the burden of diabetic foot complications.

The implemented system for predicting diabetic versus non-diabetic foot conditions using deep learning techniques demonstrated promising results. The analysis covered multiple approaches, and the following findings were observed:

The custom CNN model achieved significant accuracy on the test set, indicating its effectiveness in learning discriminative features such as textural patterns, morphological abnormalities, and color variations. This demonstrates that the model could reliably differentiate between diabetic and non-diabetic foot images.

Incorporating the ResNet50 pre-trained model with transfer learning further enhanced classification performance, leveraging the rich feature extraction capabilities of ResNet50. This approach reduced training time and yielded better generalization, especially when working with augmented datasets.

Textural Features: Skin roughness, cracks, and ulcer-like patterns were prominent in diabetic images.

Morphological Features: Abnormalities in foot structure, such as swelling and deformities, were well-captured.

Color and Vascular Features: Changes in pigmentation and visible blood flow issues contributed to the model's decision-making.

CNN's ability to extract hierarchical features (low-level edges to high-level deformities) enabled it to learn these characteristics effectively.

The use of data augmentation techniques, including rotation, scaling, and flipping, significantly improved model robustness by introducing variability to the training data. This helped mitigate overfitting, especially with limited datasets, ensuring better performance on unseen data.

Dataset Size: A relatively small dataset constrained the model's ability to generalize further. Expanding the dataset with more diverse and high-resolution images could improve accuracy.

Grayscale Preprocessing: While grayscale images simplified computation, some diagnostic information in color images may have been lost.

Complexity in Advanced Features: High-level features like vascular patterns and lesion depth were harder to quantify, possibly requiring multi-modal approaches like infrared or thermal imaging.

Early Diagnosis: Automated classification can assist clinicians in early detection of diabetic foot complications.

Integration of Explainable AI (XAI): Methods like Grad-CAM can highlight image regions critical to model predictions, making results interpretable for clinicians.

Use of Multi-Modal Data: Combining image data with clinical parameters (e.g., glucose levels, medical history) could enhance prediction accuracy.

The results demonstrate that deep learning models, when trained on well-curated datasets, can effectively identify diabetic foot conditions. By refining preprocessing techniques, expanding datasets, and integrating explainability, the system could significantly contribute to diabetic neuropathy diagnostics and management.

REFERENCES

[1] Boulton, A.J.M., Vileikyte, L., Ragnarson-Tennvall, G. & Apelqvist, J. (2004) 'The global burden of diabetic foot disease', *The Lancet*, 366(9498), pp. 1719–1724.
[2] Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. & Thrun, S. (2017) 'Dermatologist-level classification of skin cancer with deep neural networks', *Nature*, 542(7639), pp. 115–118.
[3] Fernando, M., Crowther, R.G., Lazzarini, P.A., Sangla, K.S., Wearing, S., Buttner, P. & Landorf, K.B. (2019) 'Biomechanical characteristics of peripheral neuropathy in diabetes during walking', *Footwear Science*, 11(1), pp. 61–69.
[4] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S. & Yang, G.-Z. (2019) 'XAI—Explainable artificial intelligence', *Science Robotics*, 4(37), eaay7120.
[5] Pham, H., Pham, T. & Datta, A. (2021) 'Gait and plantar pressure monitoring system for diabetic foot disease using artificial intelligence', *Journal of Diabetes Science and Technology*, 15(3), pp. 605–615.
[6] Ribeiro, M.T., Singh, S. & Guestrin, C. (2016) '"Why should I trust you?" Explaining the predictions of any classifier', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144.
[7] Tesfaye, S., Boulton, A.J., Dyck, P.J., Freeman, R., Horowitz, M., Kempler, P. & Vinik, A. (2011) 'Diabetic neuropathies: update on definitions, diagnostic criteria, estimation of severity, and treatments', *Diabetes Care*, 33(10), pp. 2285–2293.
[8] Wrobel, J.S., Najafi, B. & Burns, J. (2020) 'The use of technology for the assessment of diabetic foot ulcer risk and progress', *Current Diabetes Reports*, 20(9), pp. 1–7.
[9] Ko, R.H., Hwang, Y.J. & Phan, T. (2023) 'Deep learning models for diabetic foot ulcer detection and severity classification', *Artificial Intelligence in Medicine*, 130, p. 102369.
[10] Karthick, P.R., Manogaran, P.A. & Sivaprakash, T. (2022) 'Application of CNN for automated diabetic foot diagnosis', *Journal of Healthcare Engineering*, 2022, p. 6532815.
[11] Pal, A. & Vashishtha, R. (2023) 'Machine learning applications in diabetic foot prevention and treatment', *Biomedical Signal Processing and Control*, 82, p. 104261.