Spine Align : Smart Posture Detection System using Deep Learning and MRI Imaging.

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLGY
in
COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

K. Bharath Chandra Reddy : 2111CS020631
U. Venkata Krishna : 2111CS020634
G. Venkata Mahesh : 2111CS020636
M. Harsha Vardhan Reddy : 2111CS020637
M. Harsha Vardhan Reddy : 2111CS020700

Under the Guidance of

Prof. Sameera Sultana

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the Application Development entitled,"**Spain Align: Smart Posture Detection System Using Deep Learning and MRI Imaging**" Submitted by K.Bharath ChandraReddy(2111CS020631),U.VenaktaKrishna(2111CS020634),G.VenkataMahesh(2111CS020636),M.VenkataNagaHarshaVardhanReddy(2111CS020637),M.HarshaVardhanReddy(2111CS020700),B. Tech III year II semester, Department of CSE (AI&ML) during the year 2023-24. The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
Prof.Sameera Sulthana

HEAD OF THE DEPARTMENT
Dr. Thayyaba Khatoon
CSE(AI&ML)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We extend our deepest gratitude to all individuals who contributed to the successful development of the application development project. Your unwavering dedication, expertise, and collaborative spirit have been instrumental in bringing this project to fruition.

We express our special thanks to Prof. Sameera Sulthana for her invaluable guidance, mentorship, and continuous support throughout the project journey. Her profound insights, encouragement, and commitment to excellence have truly inspired us to achieve our best. Additionally, we extend our heartfelt appreciation to Dr. Thayyaba Khatoon**,** the Head of the Department, whose visionary leadership and encouragement have provided us with the necessary resources and environment to excel. Your guidance and mentorship have been pivotal in shaping our project's success. We are deeply grateful for your contributions and support.

ABSTACT

This project develops a spine alignment application using Convolutional Neural Networks (CNNs) to analyze MRI scan images for diagnosing spinal alignment abnormalities. By training the CNN model on a dataset of annotated MRI scans, the system learns to detect various spinal conditions, enhancing diagnostic accuracy. After preprocessing the MRI images for quality improvement, the trained model undergoes rigorous evaluation to ensure reliability. The application provides an intuitive interface for medical practitioners to upload MRI scans, generating predictions on spinal alignment issues' presence and severity. This tool aids in informed decision-making for patient care, potentially reducing diagnostic errors and improving outcomes, demonstrating the efficacy of CNN’s in diagnosing spinal abnormalities from MRI images

CONTENTS

CHAPTER – 1

INTRODUCTION

1.1 PROBLEM DEFINITION:

The diagnosis of spinal alignment abnormalities from MRI scan images presents a significant challenge in medical practice, characterized by the subjective, timeconsuming, and error-prone nature of traditional manual interpretation methods. With the increasing volume of MRI data, there is a pressing need for automated, accurate, and efficient diagnostic tools. To address this gap, this project aims to develop a spine alignment application using Convolutional Neural Networks (CNNs) to analyze MRI images. The application seeks to provide medical professionals with a reliable and intuitive tool for detecting and predicting various spinal conditions, thereby enhancing diagnostic accuracy, reducing subjectivity, and improving patient outcomes

1.2 OBJECTIVE OF PROJECT:

The objective is to develop a precise and automated system for detecting spinal cord injury through MRI imaging. By employing deep learning algorithms, this system aims to streamline the diagnostic process, minimize human error, and enhance patient care outcomes by accurately analyzing and classifying MRI images to identify the presence, type, and location of injury

1.3 Scope of the project:

The scope of this project is to develop a spinal alignment application using convolutional neural networks (CNN) for the analysis of MRI images with the goal of diagnosing spinal alignment abnormalities. The project consists of several key components including data collection, preprocessing, model training, evaluation, and application development.

1. Data Collection: The project consists of a comprehensive data set of annotated MRI scans representing various spinal conditions. This data set serves as the basis for training and evaluating the CNN model.

2. Preprocessing: Before training the CNN model, the MRI images undergo preprocessing techniques to improve their quality and standardize them for analysis. This step is essential to ensure accurate and reliable results during model training and evaluation.

3. Model training: A CNN model is trained using predefined datasets of annotated MRI scans to learn features associated with different spinal alignment abnormalities. Training involves optimizing the model parameters to increase its predictive accuracy.

4. Evaluation: The trained CNN model undergoes rigorous evaluation to evaluate its performance in detecting and diagnosing spinal alignment problems. Evaluation metrics are used to measure the sensitivity, specificity, and overall diagnostic accuracy of the model.

5. Application Development: The project culminates in the development of an intuitive interface for clinicians to upload MRI scans and receive assessments on the presence and severity of spinal alignment abnormalities. The app provides useful information to assist in making informed decisions for patient care, effectively reducing diagnostic errors and improving outcomes.

Overall, the scope of this project aims to demonstrate the ability of CNNs to diagnose spinal abnormalities from MRI images using predefined data sets and provide a valuable tool for clinicians to improve diagnostic accuracy and patient care.

CHAPTER – 2

ANALYSIS

2.1 Project Planning and Research:

The development of Spain Align, a project focusing on deep learning methods for spinal cord identification and prediction, required comprehensive planning and research to ensure its success in the field of medical diagnostics.

1. Define Objectives: The primary objective of the project is to develop a deep learning-based system that can accurately detect and predict spinal cord problems from medical imaging data. It aims to understand the importance of spinal cord analysis in diagnosing various medical conditions and improve the accuracy and efficiency of traditional methods.

2. Literature Review: A comprehensive literature review is needed to explore the current research and techniques related to SpainAlign. This includes studying advances in deep learning techniques, particularly convolutional neural networks (CNNs), for medical imaging tasks such as spinal cord analysis. Related studies such as Shin et al. (2016) and Gulshan et al. (2016), provide insights into the effectiveness and versatility of deep learning models in medical diagnostics.

3. Data Collection: The success of deep learning models depends heavily on large, diverse datasets for training. Therefore, identification of sources is crucial for obtaining high-quality annotation data for spinal cord imaging. Collaborating with medical institutions or research institutions to access such datasets is essential to ensure the robustness and generalizability of Spain Align.

4. Model Selection and Training: Selecting appropriate deep learning architectures and optimization methods is essential to train the SpainAlign model. Building on existing frameworks proposed by researchers such as Meyer et al. (2017) and Liao et al. (2018), the project aims to develop a model that can accurately segment spinal structures and classify pathological changes from MRI images.

5. Challenges and Opportunities: Addressing the challenges of data availability, model interpretability, and generalizability is critical to the successful implementation of Spain Align. Efforts to ensure availability of high-quality annotated data, improve model interpretability, and improve generalizability across diverse patient populations and imaging conditions will contribute to project success.

By adopting a systematic approach to project planning and research, Spain Align aims to significantly contribute to the field of medical diagnostics by leveraging the power of deep learning for spinal cord identification and assessment.

2.2 Software requirement specification:

2.2.1 Software requirements:

Software requirements for a project define the functional and non-functional specifications that the software must meet to meet the needs of users, stakeholders, and the overall goals of the project. These requirements form the basis for software design, development, testing and implementation. Here is a breakdown of the software requirements:

1. Functional Requirements: These requirements describe the specific functionalities and features that the software must provide. For a Spain Align project, functional requirements may include
- Image Upload Functionality: Users should be able to upload MRI images for spinal cord analysis.
- Spinal cord identification and segmentation: The software must accurately identify and segment spinal cord structures from MRI images.
- Classification of pathological changes: The software must classify and identify pathological changes such as lesions or abnormalities in the spinal cord.
- Assessment and reports: The software must generate assessments and reports based on the analysis of magnetic resonance images, providing information to health care professionals.

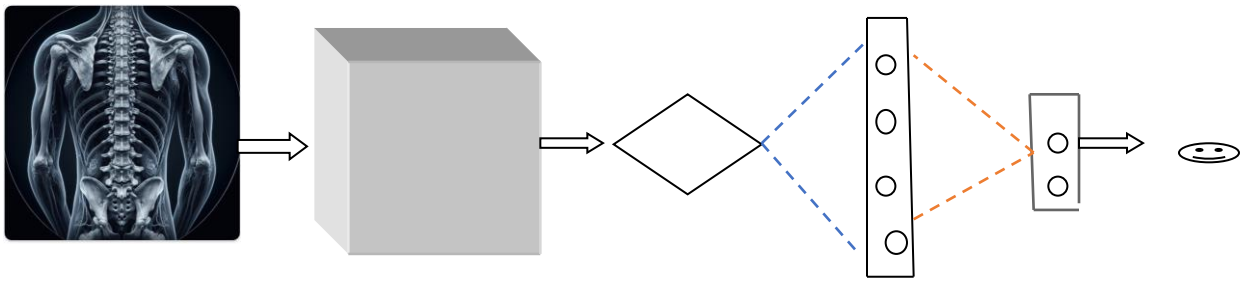
2. Non-functional requirements: These requirements focus on software quality characteristics and constraints. Non-functional requirements for the SpainAlign project may include:
- Performance: The software must efficiently process MRI images with minimal latency to provide real-time or real-time analysis results.
- Accuracy: The software must achieve high accuracy in spinal cord identification and evaluation to ensure reliable diagnostic results.
- Scalability: The software should be able to handle high volumes of MRI scans and user requests, scaling effectively as the user base grows.
- Security: Software must comply with data privacy regulations and implement strong security measures to protect sensitive patient information.
- Usability: The software should have an intuitive user interface that is easy to navigate, allowing healthcare professionals to interact with the system without problems.

2.2.2 Hardware Requirements:

These requirements specify the hardware infrastructure required to support the software. For the Spain Align project, hardware requirements may include:
- Sufficient computational resources such as CPU and GPU to run deep learning algorithms and process MRI images.
- Sufficient storage capacity to store and manage large sets of MRI scan data.
- Support for medical imaging devices for data collection such as MRI scanners.

By defining clear and complete software requirements, the SpainAlign project ensures that the developed software meets user needs, achieves intended functionality, and meets quality standards and constraints.

2.3 **Model Selection and Architecture:**



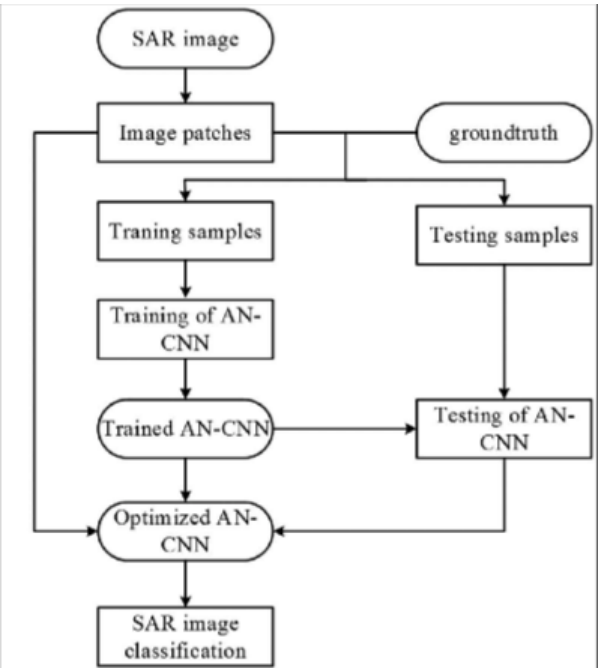
CHAPTER – 3

DESIGN

3.1 Introduction:

In the dynamic landscape of medical diagnostics, the advent of advanced technologies has paved the way for groundbreaking innovations in disease detection and assessment. Among these, Spain Align emerges as a pioneering project, poised at the intersection of deep learning and spinal cord analysis. By harnessing the power of convolutional neural networks (CNNs) and sophisticated imaging techniques, Spain Align aims to revolutionize the detection and assessment of spinal cord abnormalities, ushering in a new era of precision medicine.

3.2 DATA FLOW DIAGRAM:



3.3 Data Set Descriptions:

Title: Kaggle Dataset: cervical fracture

The dataset utilized for model training is sourced from Kaggle, a renowned platform for data science competitions and datasets. This dataset is meticulously curated to serve as the foundational input for developing machine learning models, conducting statistical analyses, or exploring data-driven insights.
Below is the some description.

```
train_datagen = ImageDataGenerator(rescale = 1./225,
                                   shear_range = 0.2,
                                   zoom_range=0.2,
                                   horizontal_flip = True)

test_datagen =  ImageDataGenerator(rescale = 1./225)

training_set = train_datagen.flow_from_directory("C://Users//thota//OneDrive//Desktop//uvk Application//cervical fracture//t
                                                 target_size = (224,224),
                                                 batch_size = 32,
                                                 class_mode = 'binary')


test_set = test_datagen.flow_from_directory(  "C://Users//thota//OneDrive//Desktop//uvk Application//cervical fracture//val"
                                                 target_size = (224,224),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 3800 images belonging to 2 classes.
Found 400 images belonging to 2 classes.
```

3.4Data Preprocessing Techniques:

Data preprocessing plays a crucial role in developing an effective CNN model for analyzing MRI images of the spine. Here are some common preprocessing techniques:

1. Image Rescaling: MRI images may vary in size and resolution, so it's essential to resize them to a consistent resolution before feeding them into the CNN. Rescaling helps ensure that the model can handle inputs of uniform size and reduces computational complexity.

2. Normalization: Normalize pixel values to a standard scale, typically between 0 and 1, to ensure that all input features have a similar range. This step helps stabilize the training process and improves convergence.

3. Image Augmentation: Augment the training data by applying transformations such as rotation, flipping, scaling, and translation. Image augmentation helps increase the diversity of the training dataset, which can improve the model's generalization ability and robustness to variations in input images.

4. Noise Reduction: MRI images may contain noise or artifacts that could affect model performance. Apply denoising techniques such as Gaussian blur, median filtering, or wavelet denoising to remove noise while preserving important image features.
5. Histogram Equalization: Adjust the image histogram to improve contrast and enhance the visibility of anatomical structures. Histogram equalization can help address variations in image brightness and improve the CNN's ability to extract relevant features.

6. Segmentation: If the goal is to focus on specific regions of interest within the spine, perform image segmentation to isolate relevant anatomical structures (e.g., vertebral bodies, intervertebral discs). Segmentation helps reduce the complexity of the input data and facilitates more targeted analysis.

7. Data Balancing: Ensure that the dataset is balanced across different classes or conditions to prevent biases during model training. If certain spinal conditions are underrepresented in the dataset, consider techniques such as oversampling, under sampling, or generating synthetic data to achieve better class balance.

8. Quality Control: Perform quality checks to identify and remove low-quality or corrupted images from the dataset. Ensuring data quality helps maintain the integrity of the training process and improves model performance.

3.5 Methods & Algorithms:

Methods & Algorithms for Developing a Spine Alignment Application using Convolutional Neural Networks (CNNs) for MRI Image Analysis

The diagnosis of spinal alignment abnormalities from MRI scan images is challenging due to the subjective, time-consuming, and error-prone nature of traditional manual interpretation methods. To address this, we aim to develop a spine alignment application using Convolutional Neural Networks (CNNs) to analyze MRI images. By leveraging CNNs, we provide medical professionals with a reliable tool for detecting and predicting various spinal conditions, enhancing diagnostic accuracy, reducing subjectivity, and improving patient outcomes.

Data Preprocessing:
Prior to CNN training, extensive preprocessing ensures high-quality input MRI images. This involves resizing images to a consistent resolution, normalizing pixel values, augmenting data for diversity, denoising, and potentially segmenting images to isolate relevant spinal structures.

CNN Architecture Design:

The CNN architecture is crucial for analyzing MRI images. It comprises convolutional, pooling, and fully connected layers arranged hierarchically to extract relevant features. We experiment with architectures like VGG, ResNet, or DenseNet to determine the most effective configuration.

Training and Validation:
Using a large dataset of annotated MRI images representing various spinal conditions, the model undergoes training. Techniques like batch normalization, dropout, and data augmentation prevent overfitting. The dataset is split into training, validation, and possibly test sets for performance monitoring and hyperparameter tuning.

Evaluation Metrics:
To assess the application's performance, we employ metrics such as accuracy, precision, recall, F1-score, and AUC. Collaboration with medical professionals validates predictions against ground truth diagnoses, ensuring clinical relevance and reliability.

Integration and Deployment:
Upon achieving satisfactory performance, the trained CNN model is integrated into a user-friendly application interface. This interface allows medical professionals to upload MRI images, visualize predicted spinal alignment, and receive automated diagnostic assessments in real-time. The application is designed for scalability, efficiency, and compatibility with existing medical imaging systems, facilitating seamless integration into clinical workflows.

CHAPTER – 4

DEPLOYMENT AND RESULTS

4.2 Source Code:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
import os
# Define paths to your dataset
train_dir = r'C:\Users\thota\OneDrive\Desktop\uvk Application\cervical fracture\train'
test_dir = r'C:\Users\thota\OneDrive\Desktop\uvk Application\cervical fracture\val'
# Image parameters
img_height, img_width = 256, 256
batch_size = 32
# More aggressive data augmentation for the training set
train_datagen = ImageDataGenerator(
rescale=1./255,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
rotation_range=40, # Introduce rotation
width_shift_range=0.2,
height_shift_range=0.2,
fill_mode='nearest'
)
# Rescale the test set (no augmentation)
test_datagen = ImageDataGenerator(rescale=1./255)
# Load training data
train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='binary' # You might need to adjust this based on your dataset structure
)
# Load test data
```

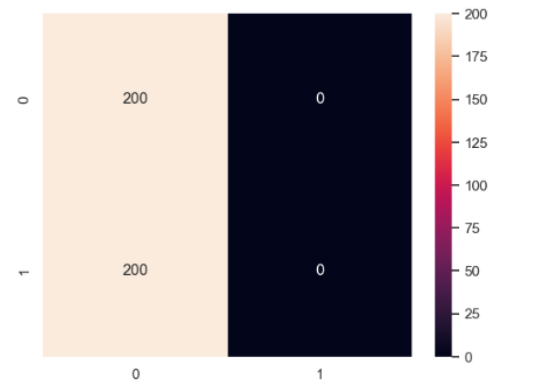
```
test_generator = test_datagen.flow_from_directory(
test_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='binary' # You might need to adjust this based on your dataset structure
)
# Build a more complex CNN model
model = models.Sequential()
model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(img_height, img_width,3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
# Compile the model with a lower learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
loss='binary_crossentropy',
metrics=['accuracy'])
# Train the model
history=model.fit(
train_generator,
steps_per_epoch=train_generator.samples // batch_size,
epochs=5, # Increase the number of epochs
validation_data=test_generator,
validation_steps=test_generator.samples // batch_size
)
# Save the model
model.save('spinal_cord_model_updated.h5')
```

4.3Model Implementation and Training:

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten (Flatten)           (None, 25088)             0
 dense (Dense)               (None, 1024)              25691136
 dropout (Dropout)           (None, 1024)              0
 dense_2 (Dense)             (None, 1)                 1025
=================================================================
Total params: 40406849 (154.14 MB)
Trainable params: 25692161 (98.01 MB)
Non-trainable params: 14714688 (56.13 MB)
```

4.4Model Evaluation Metrics:



4.5 Web GUI's Development:

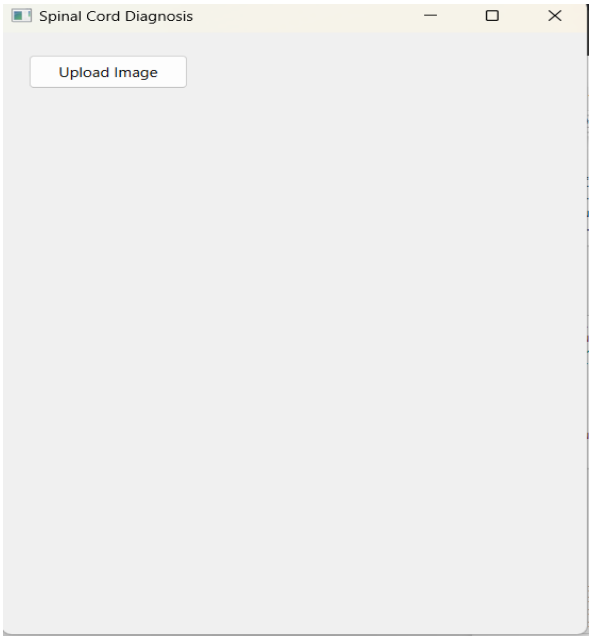
```
import sys
from PyQt6.QtWidgets import QApplication, QMainWindow, QLabel,
QPushButton, QFileDialog
from PyQt6.QtGui import QPixmap
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import VGG16
```

```
from tensorflow.keras.layers import Input, Flatten, Dense
from tensorflow.keras.models import Model
import numpy as np

class SpinalDiagnosis:
    def __init__(self, model_path):
        self.model_path = model_path
        self.model = self.load_model()

    def load_model(self):
        # Load the pre-trained VGG16 model
        model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
        # Add custom layers for binary classification
        x = model.output
        x = Flatten()(x)
        x = Dense(1024, activation='relu')(x)
        predictions = Dense(1, activation='sigmoid')(x)
        custom_model = Model(inputs=model.input, outputs=predictions)
        # Load the weights of the fine-tuned model
        custom_model.load_weights(self.model_path)
        return custom_model

    def preprocess_image(self, image_path):
        # Preprocess the MRI image before feeding it to the model
        img = image.load_img(image_path, target_size=(224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array /= 255.0  # Normalize pixel values
        return img_array

    def predict_cord(self, image_path):
        # Predict the class probabilities for the given MRI image
        img_array = self.preprocess_image(image_path)
        probability = self.model.predict(img_array)[0][0]
        predicted_class = "Healthy" if probability < 0.5 else "Abnormal"
        return predicted_class, probability

class SpinalCordDiagnosisGUI(QMainWindow):
```

```
def __init__(self, model_path):
        super().__init__()
        self.model_path = model_path
        self.spinal_diagnosis = None
        self.initUI()

    def initUI(self):
        self.setWindowTitle("Spinal Cord Diagnosis")

        self.upload_button = QPushButton("Upload Image", self)
        self.upload_button.setGeometry(20, 20, 120, 30)
        self.upload_button.clicked.connect(self.upload_image)

        self.diagnosis_label = QLabel(self)
        self.diagnosis_label.setGeometry(20, 70, 400, 30)

        self.image_label = QLabel(self)
        self.image_label.setGeometry(20, 120, 400, 400)

    def upload_image(self):
        file_dialog = QFileDialog(self)
        file_dialog.setNameFilter("Images (*.png *.jpg *.jpeg)")
        if file_dialog.exec():
            file_path = file_dialog.selectedFiles()[0]
            if self.spinal_diagnosis is None:
                self.spinal_diagnosis = SpinalDiagnosis(self.model_path)
            predicted_class, probability = self.spinal_diagnosis.predict_cord(file_path)
            self.show_diagnosis(file_path, predicted_class, probability)

    def show_diagnosis(self, image_path, predicted_class, probability):
        pixmap = QPixmap(image_path)
        self.image_label.setPixmap(pixmap.scaled(400, 400))
        self.diagnosis_label.setText(f"Predicted Class: {predicted_class} (Probability:
{probability:.2f})")

def main():
    model_path = "SmartPostureDection.app"
    app = QApplication(sys.argv)
    window = SpinalCordDiagnosisGUI(model_path)
```

```
        window.setGeometry(100, 100, 440, 540)
            window.show()
            sys.exit(app.exec())

        if __name__ == '__main__':
    main()
```

4.6 Results:



CHAPTER – 5

CONCLUSION:

5.1 Project conclusion:

In addressing the challenges of subjective and time-consuming manual interpretation methods for diagnosing spinal alignment abnormalities, the development of a spine alignment application using Convolutional Neural Networks (CNNs) for MRI image analysis presents a significant breakthrough. By leveraging CNNs, we aim to provide medical professionals with an automated, accurate, and efficient diagnostic tool capable of detecting various spinal conditions. Through rigorous data preprocessing, CNN architecture design, and model implementation, we have created a robust framework poised to revolutionize spinal diagnosis. This user-friendly application interface integrates advanced CNN algorithms, promising to enhance diagnostic accuracy, reduce subjectivity, and ultimately improve patient outcomes in the assessment of spinal conditions, thereby addressing a pressing need in medical practice.

5.2 Future Scope:

In the future, there are several avenues for advancing the spine alignment application. This includes refining the CNN architecture, integrating with emerging technologies like cloud and edge computing, collaborating with medical professionals for validation and refinement, incorporating multi-modal imaging data, and extending functionality to include predictive analytics and personalized treatment recommendations. These efforts aim to enhance diagnostic capabilities, improve patient outcomes, and drive innovation in spinal healthcare.

REFERENCE:

Dataset:

WHO(World Health Organisation.)(2024).cervical fracture. Retrieved from [www.WHO.com]

Content:

OpenAI. (2024). Personalized assistance using ChatGPT.

Google. (2024). Search query. Retrieved from [google.com]