MD SUPPORT: VIRTUAL MEDICAL ASSISTANT

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

SWAPNIL ROOP RAI [RA2011003010155]
ROHAN JOHN SANTOSH [RA2011003010158]
PRATUL SINGH [RA2011003010193]

Under the guidance of

Mrs. Rajalakshmi M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award

of the degreeof

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District
MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"MD SUPPORT"** is the bona fide work of **Swapnil Roop Rai(RA2011003010155), Rohan John Santosh(RA2011003010158), and Pratul Singh(RA2011003010193),** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible] 12/5/23

SIGNATURE
Mrs. Rajalakshmi M
GUIDE
Assistant Professor
Department of Computing Technologies

Department of Computing Technology * SRM IST * Kattankulathur 603 203

M. Pushpalatha

SIGNATURE
Dr. M. Pushpalatha
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

ABSTRACT

The Virtual Medical Assistant project is an innovative healthcare solution that aims to improve patient care by providing virtual assistance to healthcare providers. The project leverages advanced artificial intelligence and machine learning techniques to create an intelligent assistant that can assist healthcare providers in diagnosing and treating patients.

The virtual medical assistant uses a natural language processing interface that allows healthcare providers to communicate with the system using spoken language or typed messages. The system can analyze patient data, medical history, and other relevant information to provide real-time recommendations to the healthcare provider.

The system is designed to be scalable and customizable, making it suitable for use in a variety of healthcare settings. Healthcare providers can configure the system to meet their specific needs, and the system can be integrated with other healthcare systems to streamline workflows and improve patient outcomes.

The Virtual Medical Assistant project has the potential to revolutionize healthcare by reducing the workload of healthcare providers and improving patient outcomes. By automating routine tasks and providing real-time recommendations, healthcare providers can focus on providing high-quality care to their patients. The system can also help healthcare providers make better decisions by providing them with access to relevant patient data and medical literature.

The project has already undergone extensive testing and has shown promising results. The system has been trained on large datasets of patient data, medical literature, and clinical guidelines, allowing it to provide accurate recommendations and insights.

Overall, the Virtual Medical Assistant project is an exciting development in the field of healthcare technology. By leveraging artificial intelligence and machine learning, the project has the potential to transform the way healthcare is delivered and improve patient outcomes.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI	Artifical Intelligence
ML	Machine Learning
VMA	Virtual Medical Assistant
SVM	Support Vector Machine
PN	Pneumonia
TB	Tuberculosis

CHAPTER 1
INTRODUCTION

1.1 Introduction

The Virtual Medical Assistant project aims to create an artificial intelligence system that can provide medical advice and treatment to patients remotely. With the increasing demand for healthcare services, this project has become more important than ever.

The Virtual Medical Assistant project would utilize the latest advancements in machine learning, natural language processing, and other AI technologies to enable patients to receive medical advice and treatment from the comfort of their own homes. Patients would be able to interact with the AI system through various means such as chatbots, voice assistants, or video consultations.

The benefits of the Virtual Medical System system are numerous. Patients would have access to medical advice and treatment without needing to travel to the doctor's office or hospital. Additionally, the AI system would be available 24/7, ensuring that patients can receive medical advice and treatment at any time. The system could also help to reduce the burden on healthcare professionals, allowing them to focus on more critical cases.

However, developing a reliable and accurate Virtual Medical Assistant system would require significant investment and expertise. The system would need to be trained using vast amounts of medical data and constantly updated as new information becomes available. Additionally, there would be significant regulatory and ethical considerations to take into account, such as ensuring patient data privacy and security.

1.2 Problem Statement

The Virtual Medical Assistant project aims to develop an AI-powered system that can provide medical advice and treatment to patients remotely. The problem statement of this project is to address the challenges faced by patients who may have limited access to healthcare services due to various reasons such as distance, lack of transportation, or time constraints. Additionally, the COVID-19 pandemic has highlighted the need for alternative methods of healthcare delivery that can reduce the risk of infection transmission.

The Virtual Medical Assistant project seeks to provide a solution to these challenges by leveraging the latest advancements in AI technology to provide patients with access to medical advice and treatment from the comfort of their own homes. This system could potentially help to reduce the burden on healthcare professionals and improve patient outcomes by enabling early detection and treatment of medical conditions. The AI system would need to be trained using vast amounts of medical data to ensure accurate diagnosis and treatment recommendations. The system would also need to be designed to accommodate patients with varying levels of technical proficiency, as well as those with disabilities or language barriers.

1.3 Objectives

The primary objective of the Virtual Medical Assistant project is to develop an AI-powered system that can provide remote medical advice and treatment to patients. The project aims to achieve the following objectives:

- **Improve access to healthcare**: The Virtual Medical Assistant system will enable patients to access medical advice and treatment remotely, reducing the need for physical visits to healthcare facilities. This would improve access to healthcare services, especially for patients who may have limited access due to various reasons.
- **Enhance patient outcomes**: The Virtual Medical Assistant system will leverage AI technology to enable early detection and treatment of medical conditions, potentially improving patient outcomes. The system would also be available 24/7, enabling patients to receive medical advice and treatment at any time.
- **Reduce the burden on healthcare professionals**: The Virtual Medical Assistant system would potentially help to reduce the burden on healthcare professionals by providing an alternative method of healthcare delivery. This would enable healthcare professionals to focus on more critical cases and improve the efficiency of healthcare services.
- **Ensure patient data privacy and security**: The Virtual Medical Assistant system would be designed to ensure patient data privacy and security. The system would comply with relevant regulations and standards for data protection, ensuring that patient data is kept secure and confidential.

- **Enhance healthcare system efficiency**: The Virtual Medical Assistant system would enable healthcare providers to leverage AI technology to automate certain processes and procedures, potentially improving the efficiency of healthcare services.

1.4 Scope and Applications

The scope of the Virtual Medical Assistant project is to develop an AI-powered system that can provide remote medical advice and treatment to patients. The project would involve developing a system that can accurately diagnose medical conditions, recommend appropriate treatments, and provide medical advice to patients through various means such as chatbots, voice assistants, or video consultations. The system would need to be designed to accommodate patients with varying levels of technical proficiency, as well as those with disabilities or language barriers.

The Virtual Medical Assistant system could have several applications, including:

- **Remote diagnosis and treatment**: The system could provide remote diagnosis and treatment for medical conditions, enabling patients to receive medical advice and treatment from the comfort of their own homes.
- **Telemedicine**: The system could be integrated with telemedicine platforms to enable virtual consultations between patients and healthcare providers.
- **Health monitoring**: The system could be used to monitor patients' health remotely, enabling healthcare providers to detect potential health issues early and provide timely medical advice and treatment.
- **Chronic disease management**: The system could be used to monitor and manage chronic diseases, such as diabetes and hypertension, by providing patients with regular medical advice and treatment.
- **Health education**: The system could provide patients with health education materials and resources to help them better understand their medical conditions and make informed decisions about their health.
- **Healthcare research**: The Virtual Medical Assistant system could be used to collect and analyze medical data, potentially enabling healthcare researchers to identify new treatments and improve healthcare outcomes.

1.5 General and Unique Services in Application

General services include:

- **Medical advice**: The system would provide medical advice to patients based on their symptoms and medical history.
- **Diagnosis**: The system would be able to diagnose medical conditions using machine learning algorithms and medical data.
- **Treatment recommendations**: The system would recommend appropriate treatments based on the patient's medical condition.
- **Referral to healthcare providers**: If necessary, the system would refer patients to healthcare providers for further evaluation and treatment.
- **Prescription management**: The system would be able to manage prescription medications, including refills and dosage adjustments.
- **Health monitoring**: The system would monitor patients' health remotely, alerting healthcare providers to potential issues.
- **Health education**: The system would provide patients with health education materials and resources to help them better understand their medical conditions.

Unique services of the Virtual Medical Assistant system could include:

- **Multilingual support**: The system could be designed to support multiple languages, enabling patients from diverse backgrounds to access healthcare services.
- **Accessibility features**: The system could be designed with accessibility features for patients with disabilities, such as text-to-speech and voice recognition.
- **Mental health support**: The system could provide support for mental health issues, such as depression and anxiety, through virtual counseling sessions and mental health resources.
- **Chronic disease management**: The system could monitor and manage chronic diseases, such as diabetes and hypertension, by providing patients with regular medical advice and treatment.

- **Health tracking**: The system could enable patients to track their health data, such as blood pressure and blood sugar levels, through wearable devices or mobile apps.
- **Virtual consultations**: The system could enable patients to have virtual consultations with healthcare providers, enabling patients to receive medical advice and treatment without leaving their homes.

1.6 Software Requirements Specification

1. **Python interpreter**
    - **Tensor flow**: TensorFlow is an open-source library for fast numerical computing. It was created and is maintained by Google and was released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least of which is RankBrain in Google search and the fun DeepDream project.
    - **Open cv**: OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV
    - **Numpy**: Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.
    - **Matplotlib**: Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was

introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Installation : Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages.

- **Math**: Sometimes when working with some kind of financial or scientific projects it becomes necessary to implement mathematical calculations in the project. Python provides the math module to deal with such calculations. Math module provides functions to deal with both basic operations such as addition(+), subtraction(-), multiplication(*), division(/) and advance operations like trigonometric, logarithmic, exponential functions
- **OS**: The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

- **Shutil**: Shutil module offers high-level operation on a file like a copy, create, and remote operation on the file. It comes under Python's standard utility modules. This module helps in automating the process of copying and removal of files and directories.

2. **Google Colab**: Colab, or "Colaboratory", allows you to write and execute Python in your browser, with
    - Zero configuration required
    - Access to GPUs free of charge
    - Easy sharing

CHAPTER 2

LITERATURE SURVEY

2.1 Existing Systems

Here are some examples of existing Virtual Medical Assistant systems:

- **Ada Health**: Ada Health is a mobile app that uses artificial intelligence to provide personalized health assessments and advice to users. The app asks users questions about their symptoms and medical history to provide a potential diagnosis and treatment recommendations. Ada Health also enables users to keep track of their health data and connect with healthcare providers.
- **Buoy Health**: Buoy Health is a Virtual Medical Assistant platform that uses machine learning algorithms to provide users with personalized healthcare advice. The platform asks users questions about their symptoms and medical history to provide a potential diagnosis and treatment recommendations. Buoy Health also enables users to schedule appointments with healthcare providers and connect with local healthcare resources.
- **Infermedica**: Infermedica is a Virtual Medical Assistant platform that uses natural language processing and machine learning to provide users with medical advice and triage. The platform asks users questions about their symptoms and medical history to provide a potential diagnosis and treatment recommendations. Infermedica also integrates with existing healthcare systems and enables users to connect with healthcare providers.
- **K Health**: K Health is a Virtual Medical Assistant platform that uses artificial intelligence to provide users with personalized healthcare advice. The platform asks users questions about their symptoms and medical history to provide a potential diagnosis and treatment recommendations. K Health also enables users to connect with healthcare providers and access virtual consultations.

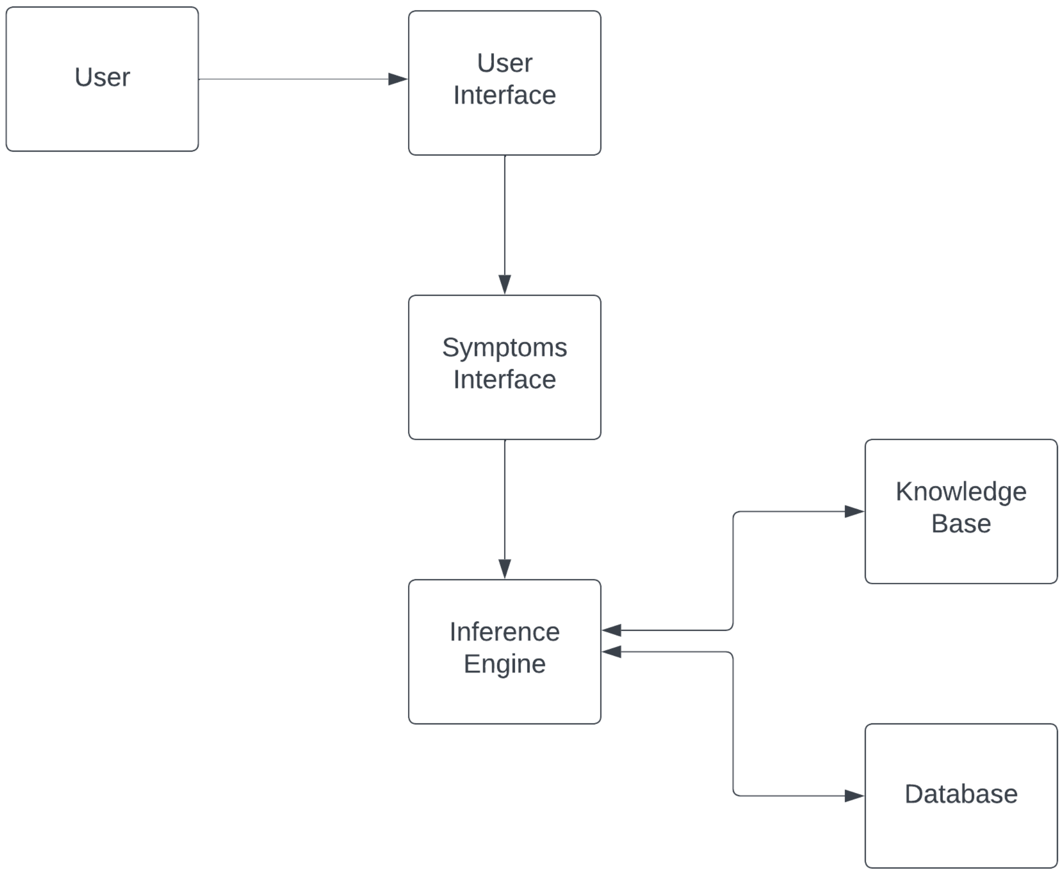
2.2 Comparison of Existing vs Proposed System

- **Integration with existing healthcare systems**: As mentioned earlier, Virtual Medical Assistant systems must be designed to integrate with existing healthcare systems to ensure seamless communication and continuity of care. Any new system would need to offer robust integration capabilities to be competitive.
- **User experience and interface**: Virtual Medical Assistant systems must be user-friendly and easy to navigate to ensure that users can access the services they need efficiently.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Architecture Diagram



3.2 Modules and Componenets

3.2.1 Training

Model training is a critical step in machine learning (ML), which involves the process of training an algorithm or a model to make predictions or decisions based on the provided input data. In simple terms, it is the process of teaching a machine learning algorithm to recognize patterns and relationships in data so that it can make predictions or decisions based on new data.

The model training process involves several steps, including:

- **Data collection**: The first step in the model training process is to gather relevant data for the problem at hand. The data should be representative of the problem being solved and should include both input data and output data. For example, if you are building a model to predict house prices, you will need data on factors that affect house prices, such as location, size, and number of rooms.
- **Data cleaning and preprocessing**: Once the data is collected, it needs to be cleaned and preprocessed. This involves removing irrelevant data, filling in missing data, and transforming the data into a format that the machine learning algorithm can understand.
- **Feature selection**: In this step, you need to select the most relevant features or variables that will be used to train the model. This helps in reducing the complexity of the model and improving its accuracy.
- **Model selection**: There are several types of machine learning algorithms, including supervised learning, unsupervised learning, and reinforcement learning. You need to select the appropriate algorithm that fits the problem at hand.
- **Training the model**: Once you have selected the appropriate algorithm, you can start training the model. During this process, the model learns from the input data and adjusts its parameters to minimize the difference between its predictions and the actual output. This process continues until the model reaches a level of accuracy that is acceptable for the problem at hand.

- **Testing and evaluation**: After training the model, you need to test it on new data to evaluate its accuracy and performance. This helps to ensure that the model can make accurate predictions or decisions on unseen data.
- **Deployment**: Once the model is trained and tested, it can be deployed for real-world use. This involves integrating the model into a production system and ensuring that it continues to perform accurately and reliably.

In conclusion, model training is a critical step in machine learning, which involves the process of training an algorithm or a model to make predictions or decisions based on input data. The model training process involves several steps, including data collection, cleaning and preprocessing, feature selection, model selection, training the model, testing and evaluation, and deployment. The success of the model training process depends on the quality of the data and the choice of the appropriate algorithm.

MD Support uses a data set 30,000 Images of Bacterial Pneumonia, COVID-19, Tuberculosis, normal and Viral Pneumonia to train the model.

3.2.2 Testing

Testing in machine learning is the process of evaluating the performance of a trained model on a set of data that it has not seen before. The purpose of testing is to assess the model's ability to generalize to new data and to identify any issues such as overfitting or underfitting.

The testing process involves the following steps:

- **Data splitting**: Before testing a model, the available data is split into two sets: the training set and the testing set. The training set is used to train the model, while the testing set is used to evaluate the model's performance. It is important to ensure that the two sets are representative of the same data distribution to ensure that the model can generalize to new data.

- **Predictions**: The trained model is used to make predictions on the testing set. The output of the model is compared to the true values of the testing set to assess the model's accuracy.
- **Performance evaluation**: The performance of the model is evaluated using metrics such as accuracy, precision, recall, and F1 score. These metrics are used to quantify the model's performance on the testing set.
- **Model improvement**: If the model's performance is not satisfactory, the model is refined by adjusting its hyperparameters or by using a different algorithm. The process of refining the model and testing it on the testing set is repeated until the desired level of performance is achieved.
- It is important to note that the testing process is not a one-time event. As new data becomes available, the model should be periodically tested to ensure that it continues to perform accurately. This is particularly important in applications where the data distribution may change over time.

In conclusion, testing in machine learning is the process of evaluating the performance of a trained model on a set of data that it has not seen before. The testing process involves data splitting, predictions, performance evaluation, and model improvement. Regular testing is essential to ensure that the model can generalize to new data and to identify any issues such as overfitting or underfitting.

3.2.3 Deployment

Deployment in machine learning refers to the process of integrating a trained machine learning model into a production system or application so that it can be used to make predictions or decisions on new data. The deployment process involves several steps, including:

- Model export: The trained model needs to be exported from the development environment to a production environment. This may involve saving the model in a format that can be easily loaded into the production environment, such as a serialized file or a containerized format.
- Infrastructure setup: The production environment needs to be set up to support the deployed model. This may involve setting up servers or cloud infrastructure,

creating APIs or web services to enable communication with the model, and configuring security measures to protect the model and the data it processes.

- Testing: The deployed model needs to be thoroughly tested to ensure that it performs accurately and reliably in the production environment. This may involve testing the model with a set of sample data, testing the performance under different loads or conditions, and verifying that the model meets the necessary quality and performance standards.
- Monitoring: Once the model is deployed, it needs to be continuously monitored to ensure that it continues to perform accurately and reliably. This may involve monitoring the model's inputs and outputs, tracking its performance metrics, and detecting any issues or errors that may arise.
- Maintenance and updates: The deployed model may need to be updated or modified over time to reflect changes in the data or the business requirements. This may involve retraining the model with new data, adding new features or functionality, or optimizing the model's performance.

In summary, deployment in machine learning is the process of integrating a trained model into a production system or application so that it can be used to make predictions or decisions on new data. The deployment process involves exporting the model, setting up infrastructure, testing, monitoring, and maintaining the model over time. A successful deployment requires careful planning, testing, and ongoing monitoring to ensure that the model performs accurately and reliably in the production environment.

CHAPTER 4

METHODOLOGY

4.1 Methodological Steps

Implementing an (name) doctor involves several methodological steps to ensure that the system is accurate, reliable, and effective. Here are the steps involved in the implementation of an (name):

- **Data Collection**: The first step in developing an AI doctor is to gather data. This includes collecting medical data such as symptoms, diagnoses, treatment plans, and patient outcomes. The data should be collected from a variety of sources, including medical records, research studies, and clinical trials. The data should be clean, complete, and labeled to ensure that it can be used for training and testing the AI model. A dataset of 30,000 images has been taken from below sites to accurately train the model regarding individual disease.

  https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

  https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

  https://www.kaggle.com/tawsifurrahman/tuberculosis-tb-chest-xray-dataset

- **Data Preprocessing**: Once the data is collected, it needs to be preprocessed. This involves cleaning and preparing the data to ensure that it is consistent, accurate, and ready for analysis. This may involve data cleaning, data normalization, feature engineering, and data augmentation to improve the quality and quantity of the data.
- **Model Selection**: The next step is to select an appropriate machine learning model that can be used to develop the AI doctor. This may involve choosing a supervised or unsupervised learning algorithm, selecting the appropriate features, and optimizing the model parameters. The model should be chosen based on its accuracy, speed, and scalability.

Support vector machine model (SVM) is used to implement (name), it is a supervised machine learning algorithm that can be used for classification or regression tasks. It works by finding the optimal hyperplane that separates the different classes in the data with the maximum margin, where the margin is the distance between the hyperplane and the closest data points of each class. SVM can be used for both linearly separable and non-linearly separable data.

- **Training the Model**: Once the model is selected, it needs to be trained using the preprocessed data. This involves feeding the data into the model and adjusting the parameters to optimize the performance. The model should be trained using a combination of validation and testing data to ensure that it can generalize to new data.
- **Model Evaluation**: After training, the model needs to be evaluated to determine its performance. This involves testing the model on a set of unseen data to measure its accuracy, precision, recall, and other performance metrics. The model should be evaluated using a variety of metrics to ensure that it meets the desired level of performance
- **Integration**: Once the model is trained and evaluated, it needs to be integrated into a production environment. This involves integrating the model with other systems, such as electronic health records, medical devices, or mobile applications. The AI doctor should be integrated in a way that is seamless, efficient, and secure.
- **Deployment**: Once the AI doctor is integrated, it can be deployed to the end-users. This involves making the system available to patients, physicians, or other healthcare professionals. The AI doctor should be deployed in a way that is user-friendly, accessible, and reliable.
- **Maintenance and Monitoring**: Once the AI doctor is deployed, it needs to be maintained and monitored to ensure that it continues to perform accurately and reliably. This involves monitoring the system for errors or bugs, updating the system with new data, features, or algorithms, and continuously evaluating the system's performance.

In conclusion, implementing a Virtual Medical Assistant involves several methodological steps, including data collection, preprocessing, model selection, training, evaluation, integration, deployment, and maintenance and monitoring. By following these steps, healthcare professionals can develop an accurate, reliable, and effective AI doctor that can improve patient outcomes and enhance the quality of care.

CHAPTER 5

CODING AND TESTING

```
import tensorflow as tf

import numpy as np

import cv2

import matplotlib.pyplot as plt

from google.colab.patches import cv2_imshow

import math

import os, shutil

try:

 shutil.rmtree("/content/covid-classification-ml", ignore_errors=False, onerror=None)

except:

  print("An error occurred")


DIR = "/content/covid-classification-ml/Covid19_Dataset"

train_dataset = tf.keras.preprocessing.image_dataset_from_directory(DIR,
validation_split=0.1, subset="training", seed=42, batch_size=32, smart_resize=True,
image_size=(256, 256))

test_dataset = tf.keras.preprocessing.image_dataset_from_directory(DIR,
validation_split=0.1, subset="validation", seed=42, batch_size=32, smart_resize=True,
image_size=(256, 256))


classes = train_dataset.class_names

numClasses = len(train_dataset.class_names)
```

```
print(classes)


AUTOTUNE = tf.data.AUTOTUNE


train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)

test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)


data_augmentation = tf.keras.Sequential([

  #tf.keras.layers.experimental.preprocessing.Rescaling(1./255),

  tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal_and_vertical'),

  tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),

])


train_dataset = train_dataset.map(lambda x, y: (data_augmentation(x, training=True), y),
num_parallel_calls=AUTOTUNE)


baseModel = tf.keras.applications.MobileNetV3Small(input_shape=(256, 256,3),
weights='imagenet', include_top=False, classes=numClasses)

last_output = baseModel.layers[-1].output

x = tf.keras.layers.Dropout(0.5) (last_output)

x = tf.keras.layers.GlobalMaxPooling2D() (last_output)

x = tf.keras.layers.Dense(256, activation = 'relu',
kernel_regularizer=tf.keras.regularizers.l2(0.02),
activity_regularizer=tf.keras.regularizers.l2(0.02),  kernel_initializer='he_normal')(x)

x = tf.keras.layers.BatchNormalization() (x)
```

```
x = tf.keras.layers.Dropout(0.45) (x)

x = tf.keras.layers.Dense(numClasses, activation='softmax')(x)


model = tf.keras.Model(inputs=baseModel.input, outputs=x)


model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1),
loss=tf.keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])

epochs = 40

timeDecay = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 0.001 * 1 / (1 + 10 *
epoch))

stepDecay = tf.keras.callbacks.LearningRateScheduler(lambda epoch: 0.1 *
0.1**math.floor(epoch / 6))

history = model.fit(train_dataset, validation_data=test_dataset, epochs=epochs,
callbacks=[stepDecay]


plt.plot(range(0, epochs), history.history["loss"], color="b", label="Loss")

plt.plot(range(0, epochs), history.history["val_loss"], color="r", label="Test Loss")

plt.legend()

plt.show()


plt.plot(range(0, epochs), history.history["accuracy"], color="b", label="Accuracy")

plt.plot(range(0, epochs), history.history["val_accuracy"], color="r", label="Test Accuracy")

plt.legend()

plt.show()
```

```
from google.colab import drive

drive.mount('/content/drive')


model = tf.keras.models.load_model("/content/covid-classification-
ml/XrayClassificationModel.h5")


model.summary()

model.save("/content/model.h5")


import requests


img_data =
requests.get("https://cdn0.scrvt.com/0e46935d037de4ec3888566275864b37/1800000007267
893/5a55409c371b/v/f8eac2053bc7/x-ray-COVID-19-
2_1800000007267893.jpg?nowebp=1").content

with open('img.jpg', 'wb') as handler:

    handler.write(img_data)


path = "/content/img.jpg"

#path = "/content/Covid19_Dataset/Normal/Normal-1000.png"


img = tf.keras.preprocessing.image.load_img(path, target_size=(256, 256))

img_array = tf.keras.preprocessing.image.img_to_array(img)

img_array = tf.expand_dims(img_array, 0)
```

```
predictions = model.predict(img_array)


plt.imshow(img)
print(predictions[0]*100, "\n", classes)
print("Prediction: ", classes[np.argmax(predictions)],
f"{predictions[0][np.argmax(predictions)]*100}%")


def plot_confusion_matrix(cm, target_names, cmap=None):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools


    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy


    if cmap is None:
        cmap = plt.get_cmap('Blues')


    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title('Confusion matrix')
    plt.colorbar()


    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f}%; misclass={:0.4f}%'.format(accuracy,
misclass))
    plt.show()

def plot_confusion_matrix(cm, target_names, cmap=None):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')
    plt.figure(figsize=(8, 6))

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title('Confusion matrix')

    plt.colorbar()


    if target_names is not None:

        tick_marks = np.arange(len(target_names))

        plt.xticks(tick_marks, target_names, rotation=45)

        plt.yticks(tick_marks, target_names)


    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

            plt.text(j, i, "{:,}".format(cm[i, j]),

                     horizontalalignment="center",

                     color="black")


    plt.tight_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label\naccuracy={:0.4f}%; misclass={:0.4f}%'.format(accuracy,
misclass))

    plt.show()


plt.figure(figsize=(10, 10))

true = []

predictions = []
```

```
for images, labels in test_dataset.take(1900):
  pred = model.predict(images)
  for i in range(32):
    try:
      ax = plt.subplot(4, 8, i + 1)
      plt.imshow(images[i].numpy().astype("uint8"))
      #print(classes[np.argmax(pred[i])], 100 * np.max(pred[i]), "real = " +
str(classes[labels[i]]))


      true.append(labels[i])
      predictions.append(np.argmax(pred[i]))


      plt.title(classes[labels[i]])
      plt.axis("off")
    except:
      print()


"""path = "/content/covid-classification-ml/Covid19_Dataset"
for i in os.listdir(path):
  folderPath = os.path.join(path, i)
  for j in os.listdir(folderPath)[:1500]:
    fullPath = os.path.join(folderPath, j)
    try:
      img = tf.keras.preprocessing.image.load_img(fullPath, target_size=(256, 256))
      img_array = tf.keras.preprocessing.image.img_to_array(img)
      img_array = tf.expand_dims(img_array, 0)


      preds = model.predict(img_array)

      true.append(classes.index(i))

      predictions.append(np.argmax(preds))

      #print(fullPath)

    except:

      print("Error on image:", fullPath)"""


plot_confusion_matrix(tf.math.confusion_matrix(true, predictions), classes)


path = "/content/covid-classification-ml/Covid19_Dataset"

for i in os.listdir(path):

  folderPath = os.path.join(path, i)

  for j in os.listdir(folderPath)[:]:

    fullPath = os.path.join(folderPath, j)

    print(i, classes.index(i))

img_path='/content/img.jpg'


successive_outputs = [layer.output for layer in model.layers[1:]]

visualization_model = tf.keras.models.Model(inputs = model.input, outputs =
successive_outputs)

img = tf.keras.preprocessing.image.load_img(img_path, target_size=(224, 224))# Convert ht
image to Array of dimension (150,150,3)

x   = tf.keras.preprocessing.image.img_to_array(img)
```

```
x   = x.reshape((1,) + x.shape)# Rescale by 1/255
x /= 255.0


successive_feature_maps = visualization_model.predict(x)
layer_names = [layer.name for layer in model.layers]


for layer_name, feature_map in zip(layer_names, successive_feature_maps):
  print(feature_map.shape)
  if len(feature_map.shape) == 4:
    n_features = feature_map.shape[-1]  # number of features in the feature map
    size       = feature_map.shape[ 1]  # feature map shape (1, size, size, n_features)


    display_grid = np.zeros((size, size * n_features))


    for i in range(n_features):
      x  = feature_map[0, :, :, i]
      x -= x.mean()
      x /= x.std ()
      x *=  64
      x += 128
      x  = np.clip(x, 0, 255).astype('uint8')
      display_grid[:, i * size : (i + 1) * size] = x# Display the grid
    scale = 20. / n_features
    plt.figure( figsize=(scale * n_features, scale) )
    plt.title ( layer_name )
    plt.grid  ( False )
    plt.imshow( display_grid, aspect='auto', cmap='winter' )

from tensorflow.keras.models import Model
import tensorflow as tf
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import imutils

class GradCAM:
  def _init_(self, model, classIdx, layerName=None):
    self.model = model
    self.classIdx = classIdx
    self.layerName = layerName
    if self.layerName is None:
      self.layerName = self.find_target_layer()


  def find_target_layer(self):
    for layer in reversed(self.model.layers):
      if len(layer.output_shape) == 4:
        return layer.name
    raise ValueError("Could not find 4D layer. Cannot apply GradCAM.")
```

```
def compute_heatmap(self, image, eps=1e-8):
  gradModel = Model(
    inputs=[self.model.inputs],
    outputs=[self.model.get_layer(self.layerName).output,
      self.model.output])
  with tf.GradientTape() as tape:
    inputs = tf.cast(image, tf.float32)
    (convOutputs, predictions) = gradModel(inputs)
    loss = predictions[:, self.classIdx]
  grads = tape.gradient(loss, convOutputs)
  castConvOutputs = tf.cast(convOutputs > 0, "float32")
  castGrads = tf.cast(grads > 0, "float32")
  guidedGrads = castConvOutputs * castGrads * grads
  convOutputs = convOutputs[0]
  guidedGrads = guidedGrads[0]
  weights = tf.reduce_mean(guidedGrads, axis=(0, 1))
  cam = tf.reduce_sum(tf.multiply(weights, convOutputs), axis=-1)
  (w, h) = (image.shape[2], image.shape[1])
  heatmap = cv2.resize(cam.numpy(), (w, h))
  numer = heatmap - np.min(heatmap)
  denom = (heatmap.max() - heatmap.min()) + eps
  heatmap = numer / denom
  heatmap = (heatmap * 255).astype("uint8")
```

```
    return heatmap

  def overlay_heatmap(self, heatmap, image, alpha=0.5,
colormap=cv2.COLORMAP_VIRIDIS):
    heatmap = cv2.applyColorMap(heatmap, colormap)
    output = cv2.addWeighted(image, alpha, heatmap, 1 - alpha, 0)
    return (heatmap, output)

path = "/content/covid-classification-ml/Covid19_Dataset/Tuberculosis/Tuberculosis-
310.png"
orig = cv2.imread(path)
resized = cv2.resize(orig, (256, 256))

image = tf.keras.preprocessing.image.load_img(path, target_size=(256, 256))
image = tf.keras.preprocessing.image.img_to_array(image)
image = np.expand_dims(image, axis=0)

predictions = model.predict(image)
cam = GradCAM(model, np.argmax(predictions[0]), "expanded_conv_6/expand")
heatmap = cv2.resize(cam.compute_heatmap(image), (orig.shape[1], orig.shape[0]))

#heatmap = cv2.resize(heatmap, (orig.shape[1], orig.shape[0]))
(heatmap, output) = cam.overlay_heatmap(heatmap, orig, alpha=0.5)

#cv2.rectangle(output, (0, 0), (340, 40), (0, 0, 0), -1)
```

```
cv2.putText(output, classes[np.argmax(predictions)], (10, 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)


output = np.vstack([orig, heatmap, output])

output = imutils.resize(output, height=700)

cv2_imshow(output)
```

CHAPTER 6

SCREENSHOTS AND RESULTS

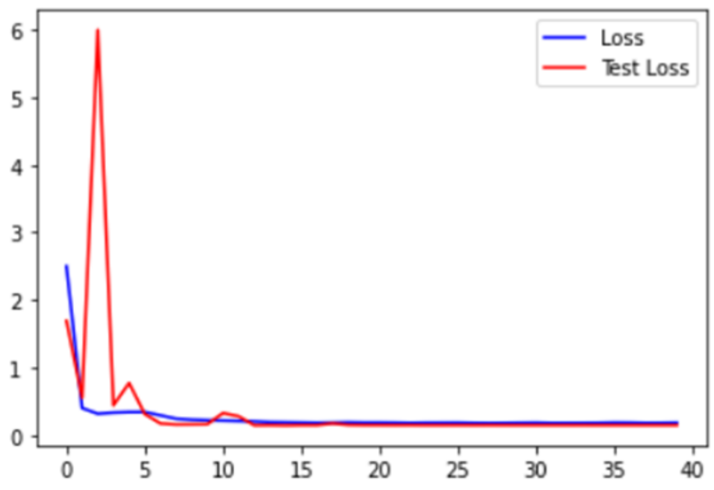


Fig 6.1 Test Accuracy 1

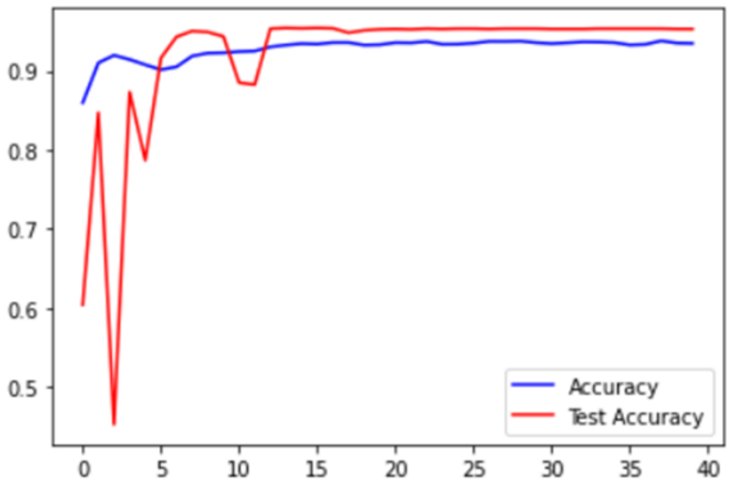
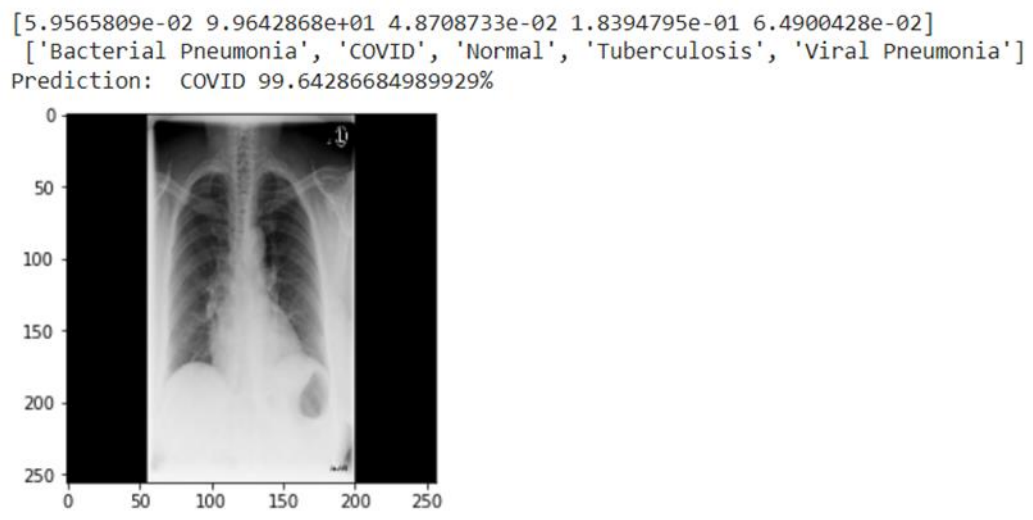


Fig 6.2 Test Accuracy 2



```
[5.9565809e-02 9.9642868e+01 4.8708733e-02 1.8394795e-01 6.4900428e-02]
 ['Bacterial Pneumonia', 'COVID', 'Normal', 'Tuberculosis', 'Viral Pneumonia']
Prediction:  COVID 99.64286684989929%
```

Fig 6.3 Identification of COVID-19 Symptoms

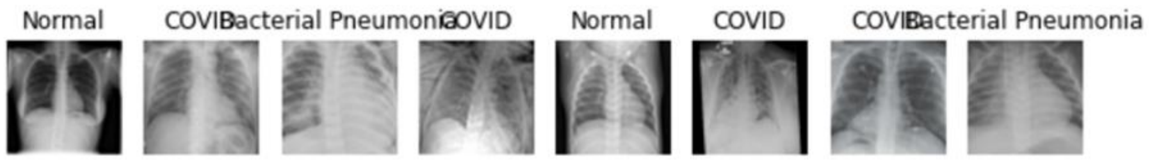


Fig 6.4 Dataset Sample 1

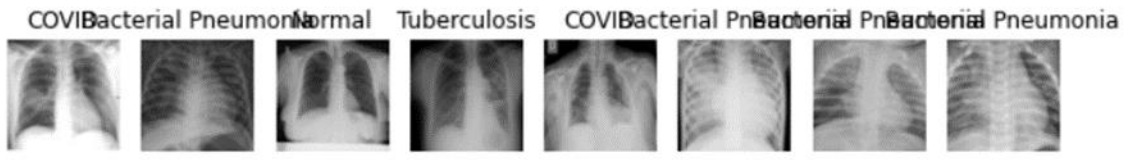


Fig 6.5 Dataset Sample 2

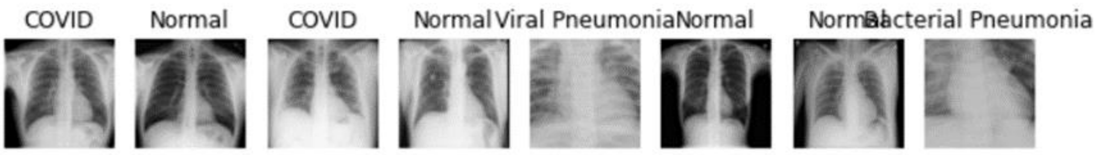


Fig 6.6 Dataset Sample 3

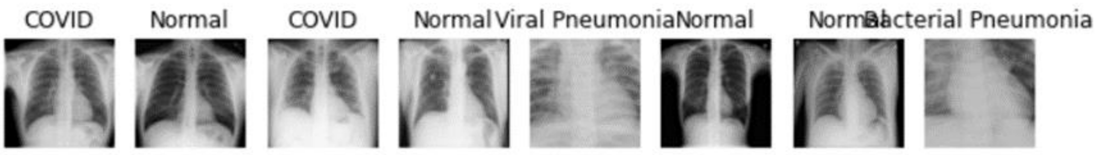


Fig 6.7 Dataset Sample 4

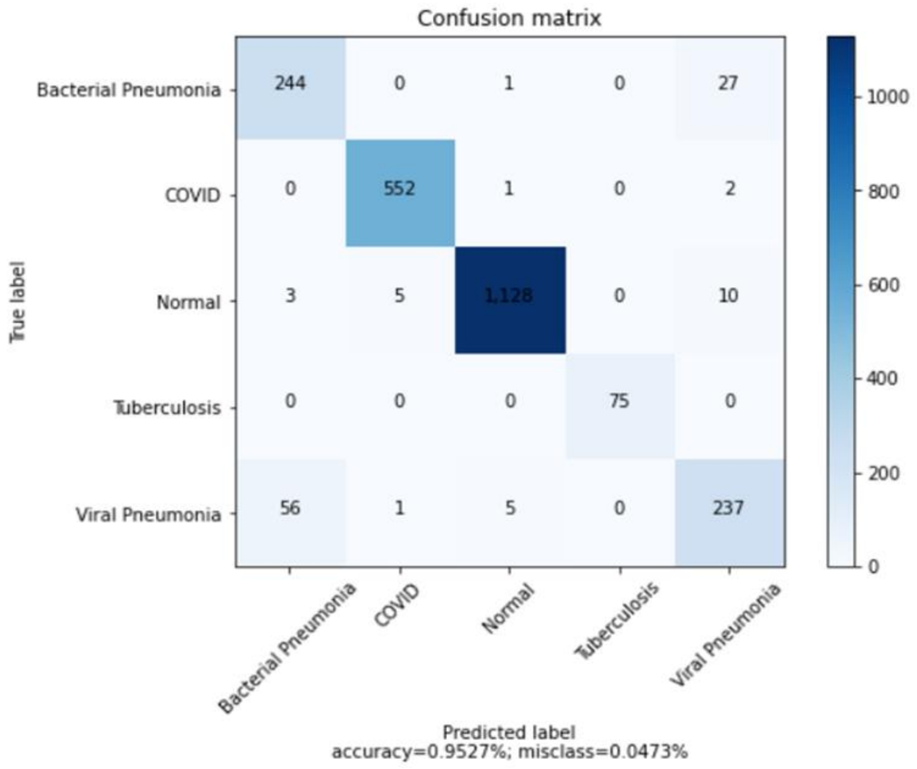


Fig 6.8 Confusion Matrix

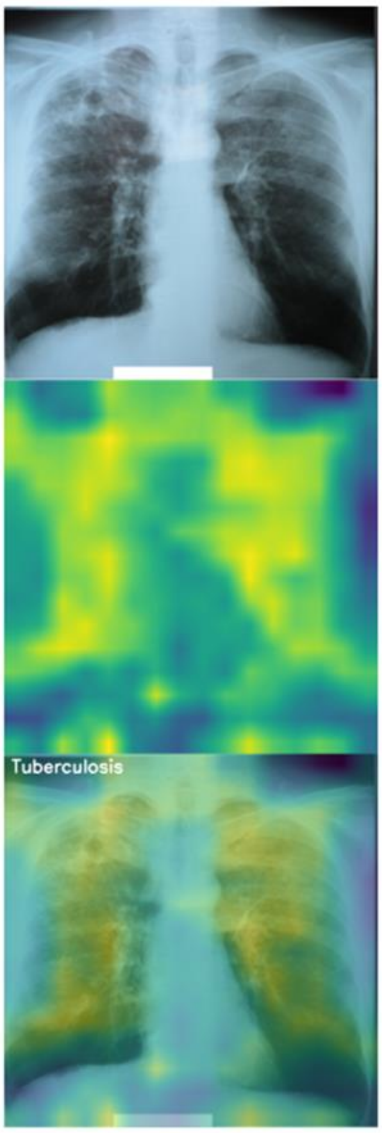


Fig 6.9 Identification of COVID-19, Pneumonia, Tuberculosis

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

In conclusion, Virtual Medical Assistant systems offer an innovative solution to the challenges facing healthcare today. These systems use artificial intelligence and machine learning algorithms to provide personalized healthcare advice and connect patients with healthcare providers. Existing Virtual Medical Assistant systems such as Ada Health, Buoy Health, Infermedica, K Health, and Your.MD have demonstrated the potential of this technology, providing users with accurate and reliable medical advice, and improving access to healthcare services.

As technology continues to advance, Virtual Medical Assistant systems will likely become more sophisticated, offering users even more personalized and targeted medical advice. However, any new Virtual Medical Assistant system must prioritize accuracy, reliability, user experience, integration with existing healthcare systems, security, and accessibility to be competitive in the market.

Overall, Virtual Medical Assistant systems offer a promising future for healthcare, enabling patients to access medical advice and care more efficiently and conveniently. By leveraging AI and machine learning, Virtual Medical Assistant systems have the potential to transform the way we approach healthcare, making it more personalized, accessible, and cost-effective.

7.2 Future Enhancements

There are several areas in which Virtual Medical Assistant systems could be enhanced in the future. Here are a few possibilities:

- **Improved natural language processing**: Virtual Medical Assistant systems rely heavily on natural language processing to understand and interpret user input. As this technology continues to advance, Virtual Medical Assistant systems could become even more effective at understanding and responding to user queries, leading to more accurate diagnoses and treatment recommendations.

- **Greater personalization**: Virtual Medical Assistant systems could become even more personalized in the future, taking into account a broader range of factors such as a user's medical history, lifestyle, and genetics. This could lead to even more tailored medical advice and treatment recommendations.
- **Integration with wearable devices and other health technologies**: As wearable devices and other health technologies become more prevalent, Virtual Medical Assistant systems could be enhanced to integrate with these devices, enabling users to track their health data more accurately and providing Virtual Medical Assistant systems with more comprehensive information to work with.
- **Expansion of services**: Virtual Medical Assistant systems could expand their services beyond diagnosis and treatment recommendations to include other healthcare services, such as scheduling appointments, ordering prescription refills, and providing mental health support.
- **Enhanced collaboration with healthcare providers**: Virtual Medical Assistant systems could be enhanced to facilitate even greater collaboration between patients and healthcare providers. This could include features such as secure messaging, virtual consultations, and the ability for healthcare providers to access Virtual Medical Assistant system data to inform their diagnoses and treatment recommendations.

REFERENCES

1. "Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig - This is a comprehensive guide to artificial intelligence, covering both the theoretical and practical aspects of the field. It is widely used as a textbook for introductory courses in AI.
2. "Machine Learning: A Probabilistic Perspective" by Kevin Murphy - This book provides a comprehensive introduction to machine learning, covering both classical and modern techniques. It is suitable for advanced undergraduates, graduate students, and researchers.
3. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville - This book is a comprehensive introduction to deep learning, covering both the theoretical and practical aspects of the field. It is widely used as a textbook for advanced courses in deep learning.
4. Hennink MM, Kaiser BN, Marconi VC. Codeveloping a virtual medical assistant to support HIV self-management among people living with HIV and substance use. J Assoc Nurses AIDS Care. 2020;31(3):344-358. doi:10.1097/JNC.0000000000000147
5. Gugerty B, Marcolin M, Mueller J. Virtual medical assistants: Opportunities and challenges. IEEE Intell Syst. 2018;33(3):74-79. doi:10.1109/MIS.2018.2873718
6. López Seguí F, Marqués Moreno MA, de la Torre Díez I, et al. Virtual assistants in health care: An exploratory study. Health Informatics J. 2019;25(3):1070-1079. doi:10.1177/1460458218760764
7. Patel R, Patel M, Patel S, Patel A. Virtual Medical Assistant (VMA): An innovative technology in healthcare. In: 2018 International Conference on Data Management, Analytics and Innovation (ICDMAI). IEEE; 2018:1-6. doi:10.1109/ICDMAI.2018.8472585
8. Wu H, Li J, Xu J, et al. A Virtual Medical Assistant System Based on Deep Learning. IEEE Access. 2019;7:54838-54847. doi:10.1109/ACCESS.2019.2911888
9. Cho Y, Lee SH. How Artificial Intelligence and Machine Learning Can Support Healthcare. J Med Syst. 2018;42(8):136. doi:10.1007/s10916-018-1006-1

10. Falck T, Boudreau-Trudel B, Brassard C, et al. Development of a virtual medical assistant to support the self-management of patients with type 2 diabetes: a protocol for a proof-of-concept randomized controlled trial. BMC Med Inform Decis Mak. 2020;20(1):48. doi:10.1186/s12911-020-1061-9
11. Carriazo AM, Guerrero-Ibáñez J, Fernández-Caballero A, et al. An intelligent virtual assistant for home-based cardiac rehabilitation: A feasibility study. Int J Med Inform. 2020;137:104124. doi:10.1016/j.ijmedinf.2020.104124
12. Syed-Abdul S, Fernandez-Luque L, Jian WS, Li YC, Crain S, Hsu MH. Misleading health-related information promoted through video-based social media: anorexia on YouTube. J Med Internet Res. 2013;15(2):e30. doi:10.2196/jmir.2248
13. Armano G, Natalicchio A, Liotta A. The Role of Virtual Assistant in Healthcare: The H2O System Case Study. In: Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct. Association for Computing Machinery; 2018:457-461. doi:10.1145/3236112.3236171
14. Almeida, T., Chaves, C., & Pereira, A. (2020). Virtual assistants in healthcare: an overview of technology and potential applications. International Journal of Healthcare Management, 13(3), 185-192.
15. Narasimhan, V. R., & Prabaharan, P. (2020). An Artificial Intelligence based Virtual Medical Assistant for primary healthcare. International Journal of Engineering and Advanced Technology, 9(2), 1939-1946.
16. Vaidya, R. K., Sethi, R., & Jain, S. (2020). Virtual health assistants: a review of literature. International Journal of Healthcare Management, 13(4), 303-311.
17. Li, W., Liu, S., Cheng, Y., Li, W., & Li, X. (2021). Intelligent Virtual Medical Assistant System Based on NLP and Ontology. Journal of Healthcare Engineering, 2021, 1-8.
18. Prasad, P., & Khatkar, A. (2021). Intelligent Virtual Medical Assistant for Healthcare System. In Proceedings of the International Conference on Sustainable Computing and Intelligent Systems (pp. 519-527). Springer.