HealthCure: An All-in-One Medical Solution

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

Jay Satija: (Roll No. 19UCSE4008)

Mahak Jain: (Roll No. 19UCSE4030)

Under the Guidance of
Dr. Anil Gupta
Professor

Under the Mentorship of
Dr. N.C. Barwar
Head of the Department

Department of Computer Science and Engineering
MBM University, Jodhpur
July, 2022

HealthCure: An All-in-One Medical Solution

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering
in
Computer Science & Engineering

Submitted by

Jay Satija: (Roll No. 19UCSE4008)

Mahak Jain: (Roll No. 19UCSE4030)

Under the Guidance of
Dr. Anil Gupta
Professor

Under the Mentorship of
Dr. N.C. Barwar
Head of the Department

Department of Computer Science and Engineering
M.B.M University, Jodhpur
July, 2022

This page was intentionally left blank.

Department of Computer Science & Engineering

M.B.M. University
Ratanada, Jodhpur, Rajasthan, India –342011

CERTIFICATE

This is to certify that the work contained in this report entitled **"HealthCure: An All-in-One Medical Solution"** is submitted by the group members Mr. Jay Satija (Roll. No: 19UCSE4008) and Ms. Mahak Jain, (Roll. No: 19UCSE4030) to the Department of Computer Science & Engineering, M.B.M. University, Jodhpur, for the partial fulfilment of the requirements for the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

They have carried out their work under my supervision. This work has not been submitted else-where for the award of any other degree or diploma.

The project work in our opinion, has reached the standard fulfilling of the requirements for the degree of Bachelor of Engineering in Computer Science & Engineering in accordance with the regulations of the Institute.

Dr. Anil Gupta
Professor
(Supervisor)
Dept. of Computer Science & Engg.
M.B.M. University, Jodhpur

Dr. N.C. Barwar
(Head)
Dept. of Computer Science & Engg.
M.B.M. University, Jodhpur

This page was intentionally left blank.

DECLARATION

We, ***Jay Satija*** and ***Mahak Jain,*** hereby declare that this project titled **“HealthCure: An All-in-One Medical Solution”** is a record of original work done by us under the supervision and guidance of ***Dr. Anil Gupta***.

We, further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

[illegible]

Jay Satija
8th Semester, CSE
Enroll. – 18R/33039
Roll No. – 19UCSE4008

[illegible]

Mahak Jain
8th Semester, CSE
Enroll. – 18R/06188
Roll No. – 19UCSE4030

This page was intentionally left blank.

ACKNOWLEDGEMENT

First of all, we are indebted to the GOD ALMIGHTY for giving us an opportunity to excel in our efforts to complete this project on time.

We are extremely grateful to Dr. Anil Gupta, Professor, Dept. of Computer Science & Engg., MBM University, Jodhpur for providing all the required resources for the successful completion of our project.

Our heartfelt gratitude to Dr. N.C. Barwar, Head of Dept., Dept. of Computer Science & Engg., MBM University, Jodhpur, for his valuable suggestions and guidance in this project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred to in this project report.

Last but not the least; we are very much thankful to our parents who guided us in every step which we took.

Thanking You

Jay Satija & Mahak Jain

This page was intentionally left blank.

ABSTRACT

AI and machine learning have gained a lot of popularity and acceptance in recent years. With the onset of the Covid-19 pandemic, the situation changed even more. During the crisis, we witnessed a rapid digital transformation and the adoption of disruptive technology across different industries. Healthcare was one of the potential sectors that gained many benefits from deploying disruptive technologies. AI, machine learning, and deep learning have become an imperative part of the sector. Deep learning in healthcare has a huge impact and it has enabled the sector to improve patient monitoring and diagnostics.

In this project, we have tried to detect seven diseases by leveraging AI with just a few clicks at home with a good accuracy and no need to wait for days for the reports. Accordingly, these diseases can be treated quickly. This project will detect first four diseases using CNN (Convolutional Neural Networks) which will take input images, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The rest three diseases will be detected using two most popular classical machine learning algorithms, Random Forest and XGBoost. With time, more datasets will be available which will improve the accuracy of this project. This project can be expanded to any number of diseases in the future as well.

This page was intentionally left blank.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Healthcare organizations of all sizes, types, and specialties are becoming increasingly interested in how artificial intelligence can support better patient care while reducing costs and improving efficiencies. Over a relatively short period of time, the availability and sophistication of AI has exploded, leaving providers, payers, and other stakeholders with a dizzying array of tools, technologies, and strategies to choose from.

Understanding exactly how data is ingested, analyzed, and returned to the end user can have a big impact on expectations for accuracy and reliability, not to mention influencing any investments necessary to whip an organization's data assets into shape. In order to efficiently and effectively choose between vendor products or hire the right data science staff to develop algorithms in-house, healthcare organizations should feel confident that they have a firm grasp on the different flavors of artificial intelligence and how they can apply to specific use cases.

Deep learning is a good place to start. This branch of artificial intelligence has very quickly become transformative for healthcare, offering the ability to analyze data with a speed and precision never seen before. Many of the industry's deep learning headlines are currently related to small-scale pilots or research projects in their pre-commercialized phases. However, deep learning is steadily finding its way into innovative tools that have high-value applications in the real-world clinical environment. Some of the most promising use cases include innovative patient-facing applications as well as a few surprisingly established strategies for improving the health IT user experience. One type of deep learning, known as convolutional neural networks (CNNs), is particularly well-suited to analyzing images, such as MRI results or x-rays.

CNNs are designed with the assumption that they will be processing images, according to computer science experts at Stanford University, allowing the networks to operate more efficiently and handle larger images[1]. As a result, some CNNs are approaching – or even surpassing – the accuracy of human diagnosticians when identifying important features in diagnostic imaging studies.

1.1 Artificial Intelligence

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by animals including humans. AI research has been defined as the field of study of intelligent agents, which refers to any system that perceives its environment and takes actions that maximize its chance of achieving its goals.

The term "artificial intelligence" had previously been used to describe machines that mimic and display "human" cognitive skills that are associated with the human mind, such as "learning" and "problem-solving". This definition has since been rejected by major AI researchers who now describe AI in terms of rationality and acting rationally, which does not limit how intelligence can be articulated.

1.2 Machine Learning

Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.

1.2.1 Random Forest

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

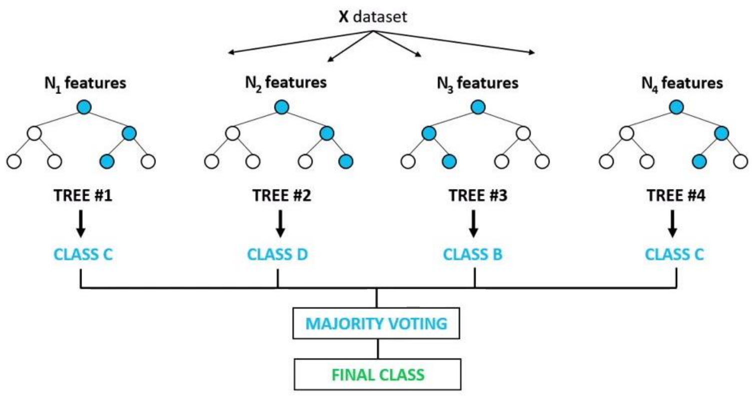


Fig. 1.1: A Random Forest Classifier

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, \ldots, x_n$ with responses $Y = y_1, \ldots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, \ldots, B$:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f}\,)^2}{B-1}}.$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the *out-of-bag error*: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

The above procedure describes the original bagging algorithm for trees. Random forests also include another type of bagging scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. An analysis of how bagging and random subspace projection contribute to accuracy gains under different conditions is given by Ho.

Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split. For regression problems the inventors recommend $p/3$ (rounded down) with a minimum node size of 5 as the default. In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

1.2.2 XGBoost

Boosting is an ensemble modelling, technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor’s error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. XGBoost works as Newton-Raphson in function space unlike gradient boosting that works as gradient descent in function space, a second order Taylor approximation is used in the loss function to make the connection to Newton Raphson method.

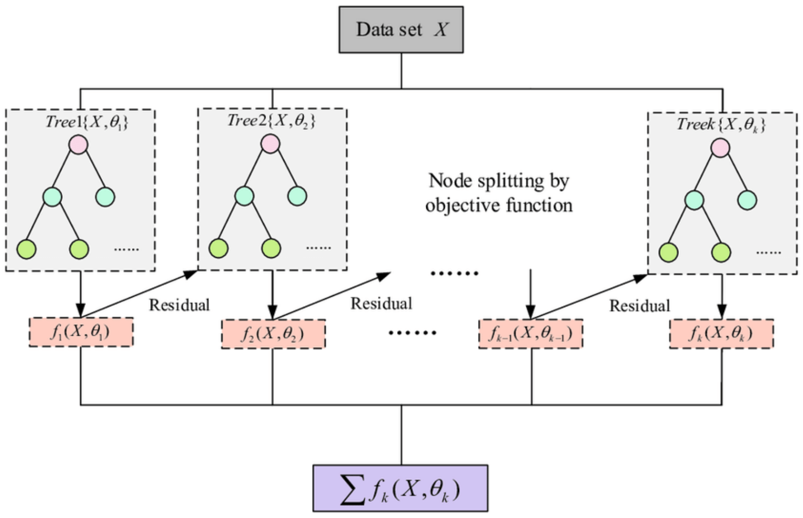
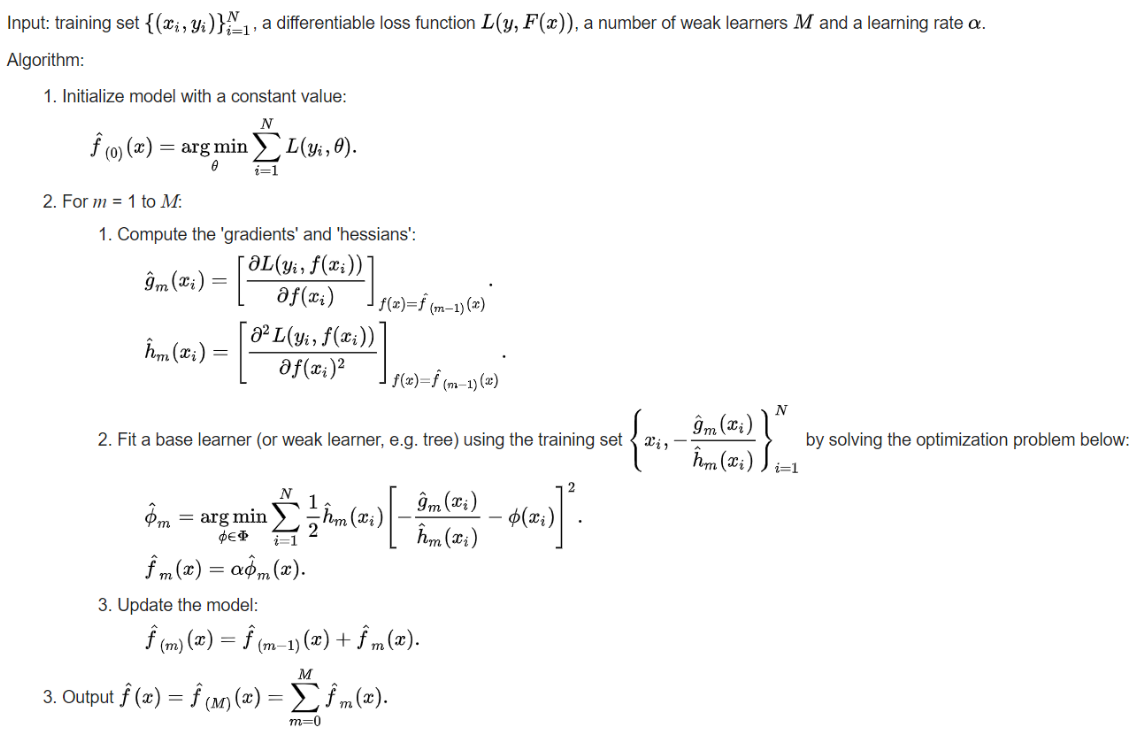


Fig. 1.2: Flow Diagram of XGBoost Algorithm

A generic unregularized XGBoost algorithm is:



Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α.

Algorithm:

```
1. Initialize model with a constant value:
       $\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta).$
2. For $m$ = 1 to $M$:
    1. Compute the 'gradients' and 'hessians':
           $\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$
           $\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}_{(m-1)}(x)}.$
    2. Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right\}_{i=1}^{N}$ by solving the optimization problem below:
           $\hat{\phi}_m = \arg\min_{\phi \in \Phi} \sum_{i=1}^{N} \frac{1}{2}\hat{h}_m(x_i)\left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i)\right]^2.$
           $\hat{f}_m(x) = \alpha\hat{\phi}_m(x).$
    3. Update the model:
           $\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$
3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x).$
```

System Optimization

Regularization: Since the ensembling of decisions, trees can sometimes lead to very complex. XGBoost uses both Lasso and Ridge Regression regularization to penalize the highly complex model.

Parallelization and Cache block: In, XGboost, we cannot train multiple trees parallel, but it can generate the different nodes of tree parallel. For that, data needs to be sorted in order. In order to reduce the cost of sorting, it stores the data in blocks. It stored the data in the compressed column format, with each column sorted by the corresponding feature value. This switch improves algorithmic performance by offsetting any parallelization overheads in computation.

Tree Pruning: XGBoost uses max_depth parameter as specified the stopping criteria for the splitting of the branch, and starts pruning trees backward. This depth-first approach improves computational performance significantly.

Cache-Awareness and Out-of-score computation: This algorithm has been designed to make use of hardware resources efficiently. This is accomplished by cache awareness by allocating internal buffers in each thread to store gradient statistics. Further enhancements such as ‘out-of-core computing optimize available disk space while handling big data-frames that do not fit into memory. In out-of-core computation, Xgboost tries to minimize the dataset by compressing it.

Sparsity Awareness: XGBoost can handle sparse data that may be generated from preprocessing steps or missing values. It uses a special split finding algorithm that is incorporated into it that can handle different types of sparsity patterns.

Weighted Quantile Sketch: XGBoost has in-built the distributed weighted quantile sketch algorithm that makes it easier to effectively find the optimal split points among weighted datasets.

Cross-validation: XGboost implementation comes with a built-in cross-validation method. This helps the algorithm prevents overfitting when the dataset is not that big.

1.3 Deep Learning

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

The adjective "deep" in deep learning refers to the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, but that a network with a nonpolynomial activation function with one hidden layer of unbounded width can. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

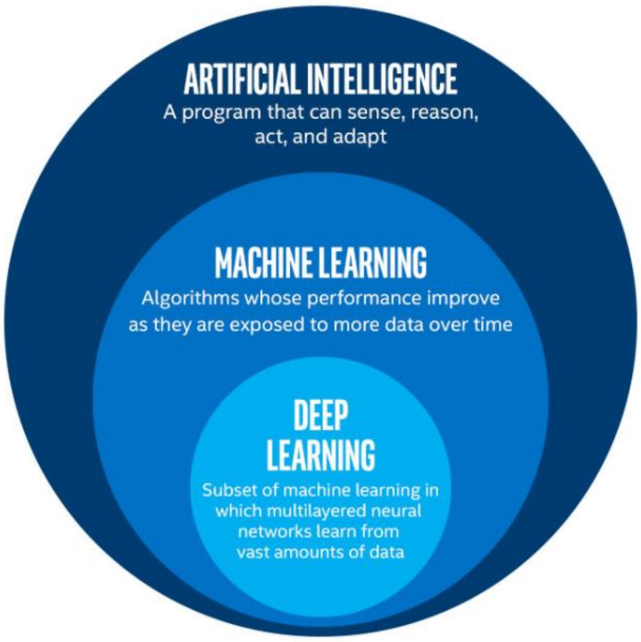

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Fig. 1.3: ML as a subset of AI and DL as a subset of ML

1.3.1 Artificial Neural Networks

Artificial neural networks (ANNs), usually simply called neural networks (NNs) or, more simply yet, neural nets, are computing systems inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives signals then processes them and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

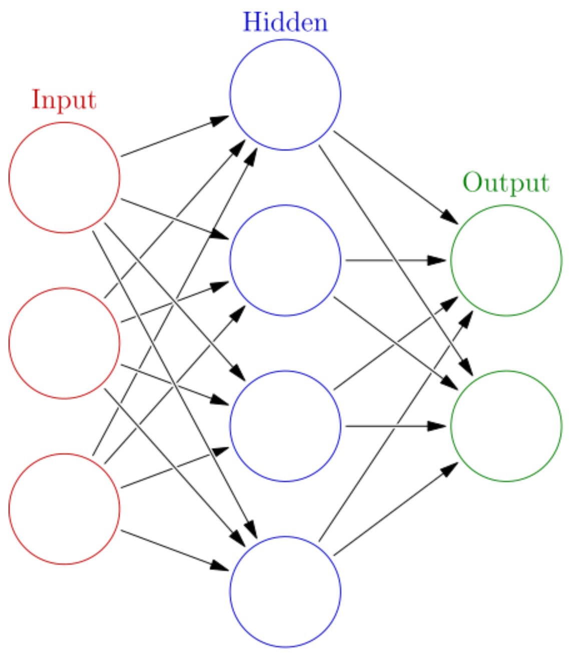


Fig. 1.4: An Artificial Neural Network with layer coloring

1.3.2 Deep Neural Networks

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions. These components functioning similar to the human brains and can be trained like any other ML algorithm.

1.3.3 Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps[2]. Counter-intuitively, most convolutional neural networks are not invariant to translation, due to the downsampling operation they apply to the input[3]. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain–computer interfaces, and financial time series.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

- The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.
- It is the sequential design that gives permission to CNN to learn hierarchical attributes.
- In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
- The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used.

Convolutional layer: The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input[4].

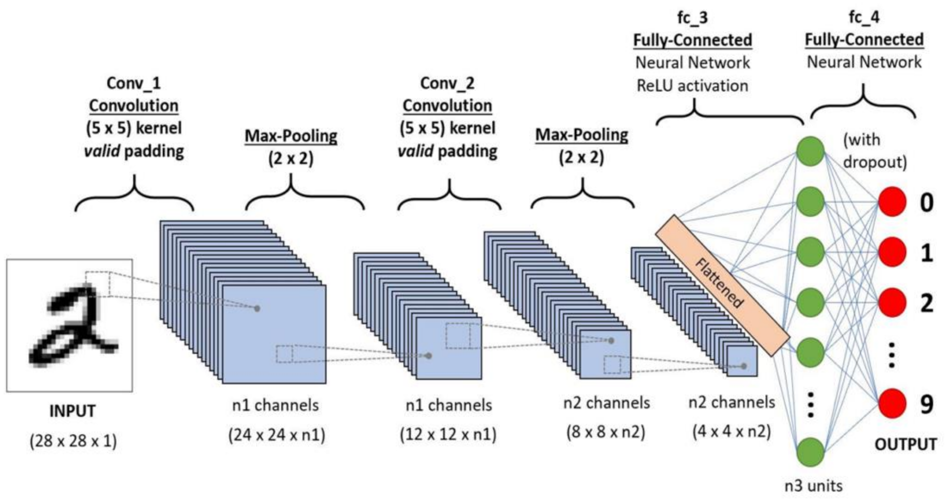


Fig. 1.5: A generic CNN architecture

Pooling layer: Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling, where max pooling is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum.

ReLU layer: ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function. It effectively removes negative values from an activation map by setting them to zero. It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers.

Fully connected layer: After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

Loss layer: The "loss layer", or "loss function", specifies how training penalizes the deviation between the predicted output of the network, and the true data labels (during supervised learning). Various loss functions can be used, depending on the specific task. The Softmax loss function is used for predicting a single class of K mutually exclusive classes.

Hyperparameters are various settings that are used to control the learning process. CNNs use more hyperparameters than a standard multilayer perceptron (MLP).

- Kernel Size
- Padding
- Stride
- Number of filters
- Filter size
- Pooling type and size

1.4 Applications of Deep Learning in Healthcare

The top pathbreaking applications of deep learning in healthcare are as follows:

- Drug Discovery
- Medical Imaging and Diagnostics
- Simplifying Clinical Trials
- Personalized Treatment
- Improved Health Records and Patient Monitoring
- Health Insurance and Fraud Detection

Chapter 2

TOOLS AND TECHNOLOGIES

The following tools and technologies are used:

- **Model Training** - Python (Jupyter Notebook)
- **Backend** – Python (VS Code, Flask Web Framework)
- **Real-Time Storage and Database** - Google Firebase
- **Frontend** - HTML, CSS, and JavaScript
- **Version Control System** - GitHub

Frameworks & Libraries used:

NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Pandas: pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

Flask: Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Following are the features of the Flask Web Framework:

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Complete documentation
- Google App Engine compatibility
- Extensions available to extend functionality

Google Firebase: Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development. It provides different kinds of services that help you to develop high-quality mobile and web applications to grow your business. It is compatible with Web, iOS, Android, and

OS X clients. There are many services which Firebase provides, Most useful of them are:

- Cloud Firestore
- Cloud Functions
- Authentication
- Hosting
- Cloud Storage
- Google Analytics
- Predictions
- Cloud Messaging
- Dynamic Links
- Remote Config

sklearn: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn integrates well with many other Python libraries, such as Matplotlib and plotly for plotting, NumPy for array vectorization, Pandas dataframes, SciPy, and many more.

OpenCV: OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

TensorFlow: TensorFlow was chosen as the right framework for training an state-of-the-art text classification model for this use-case. TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras is a central part of the tightly connected TensorFlow 2 ecosystem, covering every step of the machine learning workflow, from data management to hyperparameter training to deployment solutions. tf.keras.layers.Layer is the base class of all Keras layers, and it inherits from tf.Module.

2.1 Algorithmic Implementation

Following are a few code snippets for the implementation of algorithms in Python programming language using different libraries and frameworks:

Importing Libraries and Frameworks:

```
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score

import xgboost

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras import layers
from tensorflow.keras.models import Model, Sequential, load_model
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.callbacks import EarlyStopping
```

Finding correlation between independent features:

```
corr_mat = df.corr()
top_corr_features = corr_mat.index
```

```
plt.figure(figsize=(15, 15))
g = sns.heatmap(corr_mat[top_corr_features].corr(), annot=True, cmap="Blues")
```

Random Forest Implementation:

```
rfc = RandomForestClassifier(random_state = 10)
rfc.fit(X_train, y_train.ravel())
```

XGBoost Implementation:

```
params = {
    "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30],
    "max_depth"        : [3, 4, 5, 6, 8, 10, 12, 15],
    "min_child_weight" : [1, 3, 5, 7],
    "gamma"            : [0.0, 0.1, 0.2 , 0.3, 0.4],
    "colsample_bytree" : [0.3, 0.4, 0.5, 0.7]
}

clf = xgboost.XGBClassifier()

random_search = RandomizedSearchCV(
    clf,
    param_distributions=params,
    n_iter=5,
    scoring='roc_auc',
    n_jobs=-1,
    cv=5,
    verbose=3
)

random_search.fit(X_train, y_train.ravel())
```

Accuracy calculation:

```
y_preds = classifier.predict(X_test)
print("Accuracy : {:.2f}%".format(accuracy_score(y_test, y_preds)*100))
```

Transfer Learning using VGG16:

```
vgg16_weight_path =
'../models/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5'
base_model = VGG16(
    weights=vgg16_weight_path,
    include_top=False,
    input_shape=IMG_SIZE + (3,)
)

model = Sequential()
model.add(base_model)
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

model.layers[0].trainable = False
model.compile(
    loss='binary_crossentropy',
    optimizer=RMSprop(lr=1e-4),
    metrics=['accuracy']
)

model.summary()

EPOCHS = 30
es = EarlyStopping(
    monitor='val_accuracy',
    mode='max',
    patience=6
)

history = model.fit_generator(
    train_generator,
    steps_per_epoch = 193,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=50,
    callbacks=[es],
)
```

Loss and Accuracy Plotting:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(1, len(history.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')

plt.tight_layout()
plt.show()
```

Generating Model Predictions:

```
predictions = model.predict(X_test_prep)
predictions = [1 if x>0.5 else 0 for x in predictions]

accuracy = accuracy_score(y_test, predictions)
print('Test Accuracy = %.2f' % accuracy)
```

Model Building in Keras:

```
def conv_block(filters, act='relu'):
    """Defining a Convolutional NN block for a Sequential CNN model. """

    block = Sequential()
    block.add(Conv2D(filters, 3, activation=act, padding='same'))
    block.add(Conv2D(filters, 3, activation=act, padding='same'))
    block.add(BatchNormalization())
    block.add(MaxPool2D())

    return block

def dense_block(units, dropout_rate, act='relu'):
    """Defining a Dense NN block for a Sequential CNN model. """

    block = Sequential()
    block.add(Dense(units, activation=act))
    block.add(BatchNormalization())
    block.add(Dropout(dropout_rate))

    return block

def construct_model(act='relu'):
    """Constructing a Sequential CNN architecture for performing the
classification task. """

    model = Sequential([
        Input(shape=(*IMAGE_SIZE, 3)),
        Conv2D(16, 3, activation=act, padding='same'),
        Conv2D(16, 3, activation=act, padding='same'),
        MaxPool2D(),
        conv_block(32),
        conv_block(64),
        conv_block(128),
        Dropout(0.2),
        conv_block(256),
        Dropout(0.2),
        Flatten(),
        dense_block(512, 0.7),
        dense_block(128, 0.5),
        dense_block(64, 0.3),
        Dense(4, activation='softmax')
    ], name = "cnn_model")

    return model
```

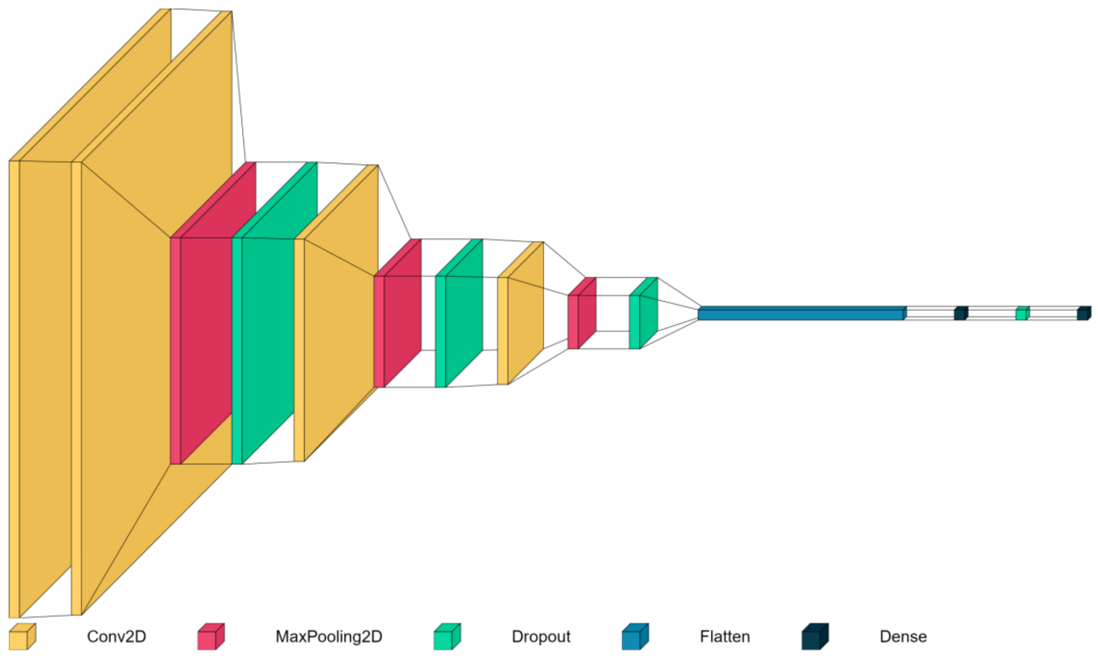
Implementing Callbacks for Early Stopping:

```
class MyCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('val_acc') > 0.99:
            print("\nReached accuracy threshold! Terminating training.")
            self.model.stop_training = True

my_callback = MyCallback()
CALLBACKS = [my_callback]
history = model.fit(train_data, train_labels, validation_data=(val_data,
val_labels), callbacks=CALLBACKS, epochs=EPOCHS)
```

2.2 Model Architecture

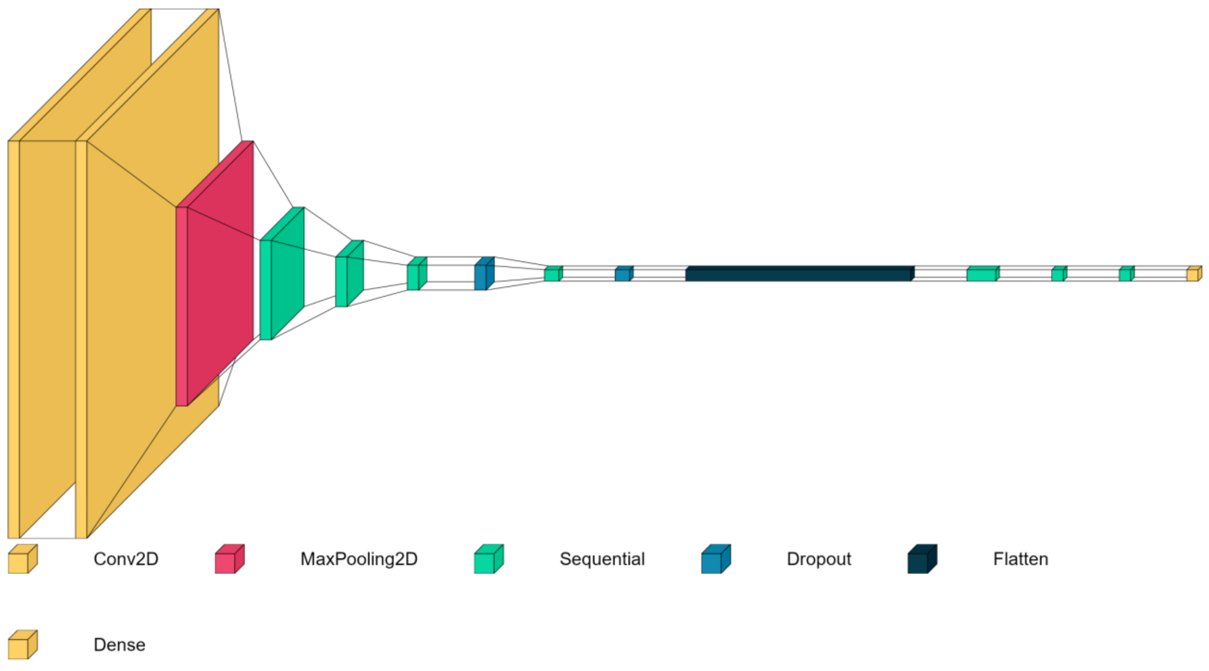
COVID-19 Detection Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 conv2d_1 (Conv2D)           (None, 220, 220, 64)      18496

 max_pooling2d (MaxPooling2D  (None, 110, 110, 64)     0
 )

 dropout (Dropout)           (None, 110, 110, 64)      0

 conv2d_2 (Conv2D)           (None, 108, 108, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 54, 54, 64)        0

 conv2d_3 (Conv2D)           (None, 52, 52, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 dropout_2 (Dropout)         (None, 26, 26, 128)       0

 flatten (Flatten)           (None, 86528)             0

 dense (Dense)               (None, 64)                5537856

 dropout_3 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 5,668,097
Trainable params: 5,668,097
Non-trainable params: 0
_________________________________________________________________

Epochs: 20
Steps per epoch: 8
Train dataset size: 282 images
Test dataset size: 80 images
No hyperparameter tuning
```



Alzheimer Detection Model

```
Model: "cnn_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 176, 176, 16)      448

 conv2d_1 (Conv2D)           (None, 176, 176, 16)      2320

 max_pooling2d (MaxPooling2D  (None, 88, 88, 16)       0
 )

 sequential (Sequential)     (None, 44, 44, 32)        14016

 sequential_1 (Sequential)   (None, 22, 22, 64)        55680

 sequential_2 (Sequential)   (None, 11, 11, 128)       221952

 dropout (Dropout)           (None, 11, 11, 128)       0

 sequential_3 (Sequential)   (None, 5, 5, 256)         886272

 dropout_1 (Dropout)         (None, 5, 5, 256)         0

 flatten (Flatten)           (None, 6400)              0

 sequential_4 (Sequential)   (None, 512)               3279360

 sequential_5 (Sequential)   (None, 128)               66176
 sequential_6 (Sequential)    (None, 64)                8512

 dense_3 (Dense)              (None, 4)                 260

=================================================================
Total params: 4,534,996
Trainable params: 4,532,628
Non-trainable params: 2,368
_________________________________________________________________

Epochs: 50
Optimizer: Adam
Train dataset size: 5121 images
Test dataset size: 20%
Regularization: 99% validation accuracy callback
No hyperparameter tuning
```



Brain Tumor Detection Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten (Flatten)           (None, 25088)             0
 dropout (Dropout)          (None, 25088)            0

 dense (Dense)              (None, 1)                25089

=================================================================
Total params: 14,739,777
Trainable params: 25,089
Non-trainable params: 14,714,688
_________________________________________________________________

Epochs: 30
Optimizer: RMSProp
Steps per epoch: 193
Learning rate: 1e-4
Train dataset size: 193 images
Test dataset size: 10 images
Regularization: Early stopping on validation accuracy with max mode
and patience level 6
No hyperparameter tuning
```

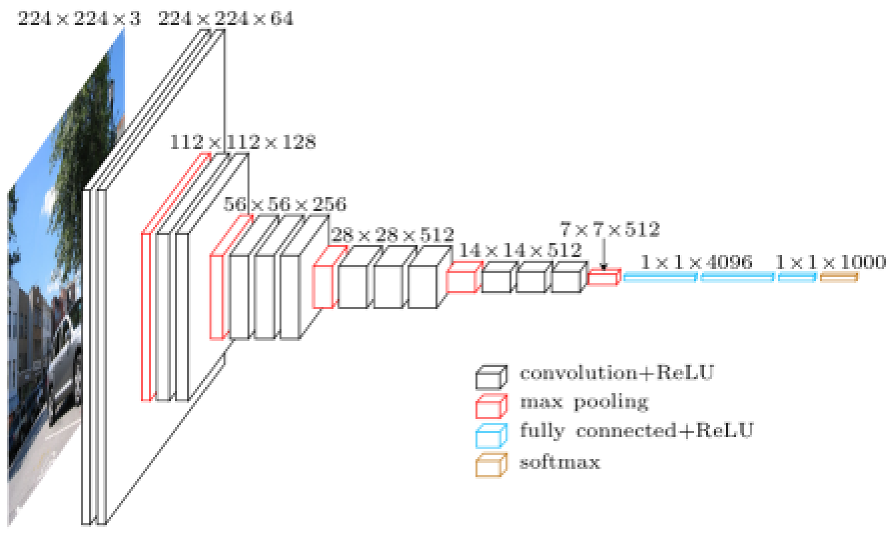
Transfer learning using VGG16 base model –

VGG16 is a convolution neural net (CNN) architecture which was used to win ILSVR(Imagenet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.

- The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.
- VGG16 takes input tensor size as 224, 244 with 3 RGB channel.
- Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with

stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.

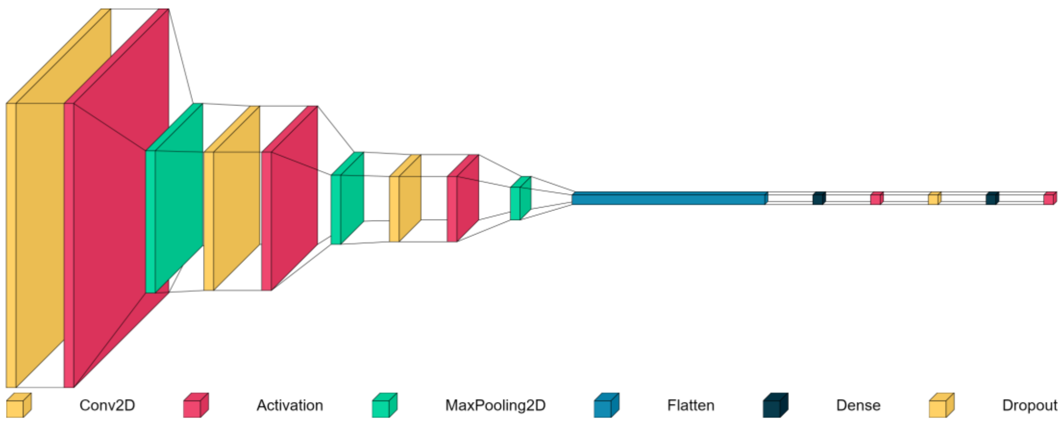
- The convolution and max pool layers are consistently arranged throughout the whole architecture.
- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.



- Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

Pneumonia Detection Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 activation (Activation)     (None, 148, 148, 32)      0

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )
 conv2d_1 (Conv2D)            (None, 72, 72, 32)        9248

 activation_1 (Activation)    (None, 72, 72, 32)        0

 max_pooling2d_1 (MaxPooling   (None, 36, 36, 32)       0
 2D)

 conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496

 activation_2 (Activation)    (None, 34, 34, 64)        0

 max_pooling2d_2 (MaxPooling   (None, 17, 17, 64)       0
 2D)

 flatten (Flatten)            (None, 18496)             0

 dense (Dense)                (None, 64)                1183808

 activation_3 (Activation)    (None, 64)                0

 dropout (Dropout)            (None, 64)                0

 dense_1 (Dense)              (None, 1)                 65

 activation_4 (Activation)    (None, 1)                 0

=================================================================
Total params: 1,212,513
Trainable params: 1,212,513
Non-trainable params: 0
_________________________________________________________________

Epochs: 20
Batch size: 16
Optimizer: RMSProp
Train dataset size: 5216 images
Test dataset size: 624 images
No hyperparameter tuning
```



Diabetes Detection Model

Random Forest Hyperparameters –

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 10,
 'verbose': 0,
 'warm_start': False}
```

(Training Specifications)

```
Dataset size: 768 data points
Test dataset size: 20%
Heavy class imbalance
Feature selection: Embedded method
Hyperparameter tuning: Randomized Search with Cross-Fold Validation
```

Breast Cancer Detection Model

Random Forest Hyperparameters –

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'sqrt',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

(Training Specifications)

```
Dataset size: 569 data points
Test dataset size: 30%
Feature selection: Domain knowledge-based
No hyperparameter tuning
```

Heart Disease Detection Model

XGBoost Hyperparameters –

```
{'objective': 'binary:logistic',
 'use_label_encoder': False,
 'base_score': 0.5,
 'booster': 'gbtree',
 'callbacks': None,
 'colsample_bylevel': 1,
 'colsample_bynode': 1,
 'colsample_bytree': 0.5,
 'early_stopping_rounds': None,
 'enable_categorical': False,
 'eval_metric': None,
 'gamma': 0.4,
 'gpu_id': -1,
 'grow_policy': 'depthwise',
 'importance_type': None,
 'interaction_constraints': '',
 'learning_rate': 0.2,
 'max_bin': 256,
 'max_cat_to_onehot': 4,
 'max_delta_step': 0,
 'max_depth': 5,
 'max_leaves': 0,
 'min_child_weight': 5,
 'missing': nan,
 'monotone_constraints': '()',
 'n_estimators': 100,
 'n_jobs': 0,
 'num_parallel_tree': 1,
 'predictor': 'auto',
 'random_state': 0,
 'reg_alpha': 0,
 'reg_lambda': 1,
 'sampling_method': 'uniform',
 'scale_pos_weight': 1,
 'subsample': 1,
 'tree_method': 'exact',
 'validate_parameters': 1,
 'verbosity': None}
```

(Training Specifications)

```
Dataset size: 303 data points
Test dataset size: 15%
Feature selection: Select K best with Chi-square hypothesis testing
Hyperparameter tuning: Randomized Search with Cross-Fold Validation
```

Chapter 3

PROJECT DETAILS

3.1 Data Collection and Preprocessing

Data is collected from several different sources, like, IEEE GitHub repository, UCSD GitHub repository, Kaggle Datasets, etc. Preparing the dataset for training involved assigning paths, creating categories (labels), resizing images, cropping area of interest from images, splitting into train, test, and validation sets, shuffling training examples, and normalizing images. Data Augmentation was also performed in a few cases.

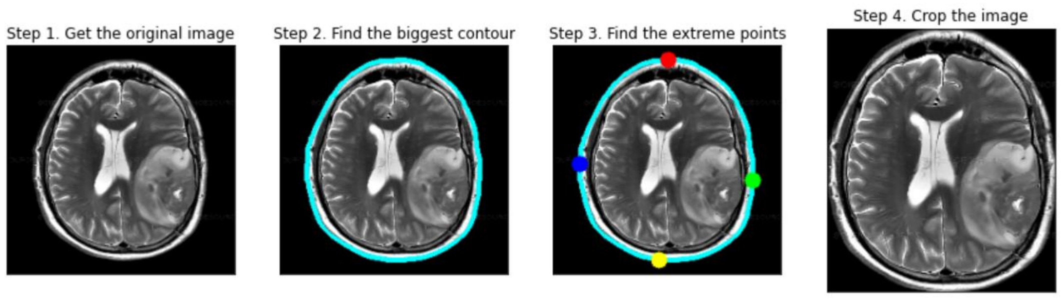
Image size used for training the models are as follows –

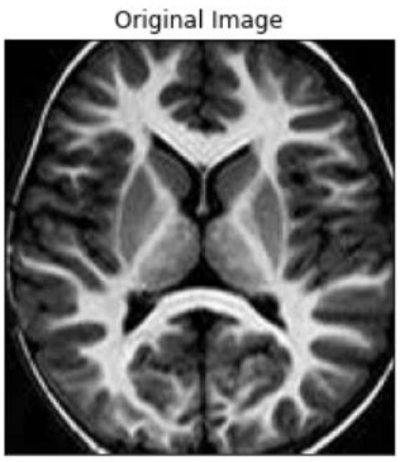
- COVID-19 Detection Model: (224,224)
- Brain Tumor Detection Model: (224,224
- Pneumonia Detection Model: (150,150)
- Alzheimer Detection Model: (176,176)

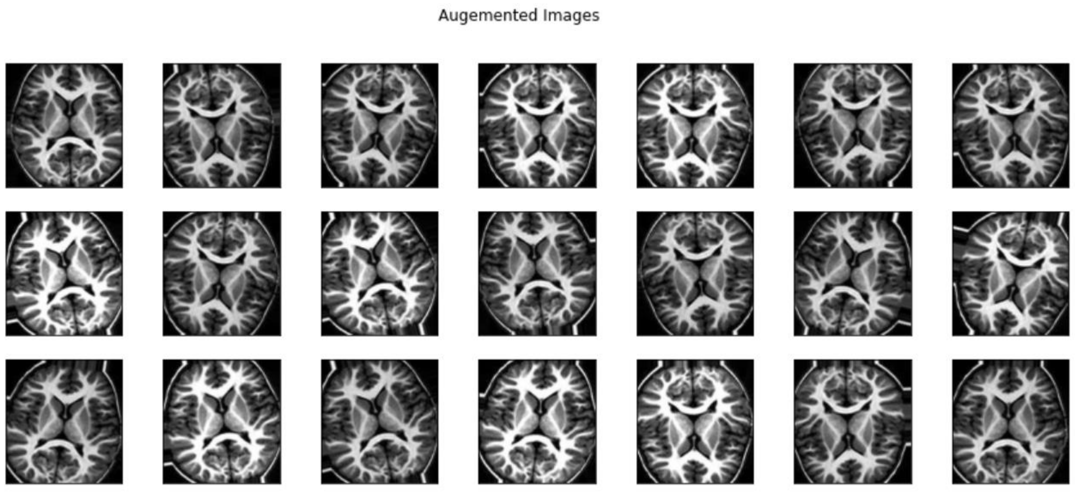
Parameters for data preprocessing using Image Data Generator class in TensorFlow are as follows –

```
rotation_range = 15,
width_shift_range = 0.1,
height_shift_range = 0.1,
zoom_range = 0.2,
shear_range = 0.1,
brightness_range = [0.5, 1.5],
color_mode = 'rgb',
horizontal_flip = True,
vertical_flip = True
```

In brain tumor detection, a few additional steps are followed for data preprocessing to improve the performance of the model. We used OpenCV’s finding extreme points in contours to crop the brain out of the MRI.


Step 1. Get the original image
Step 2. Find the biggest contour
Step 3. Find the extreme points
Step 4. Crop the image





3.2 UI Layer

The user interface is the point at which human users interact with a computer, website or application. The goal of effective UI is to make the user's experience easy and intuitive, requiring minimum effort on the user's part to receive the maximum desired outcome. UI is created in layers of interaction that appeal to the human senses (sight, touch, auditory and more). They include both input devices like a keyboard, mouse, trackpad, microphone, touch screen, fingerprint scanner, e-pen and camera, and output devices like monitors, speakers and printers. Devices that interact with multiple senses are called "multimedia user interfaces." For example, everyday UI uses a combination of tactile input (keyboard and mouse) and a visual and auditory output (monitor and speakers). User interface is important to meet user expectations and support the effective functionality of your site. A well-executed user interface facilitates effective interaction between the user and the program, app or machine through contrasting visuals, clean design and responsiveness.

User Interface of the project is designed using HTML, CSS, and JavaScript. There are total 15 pages, 1 Homepage and 2 pages for each disease (Detection and Results).

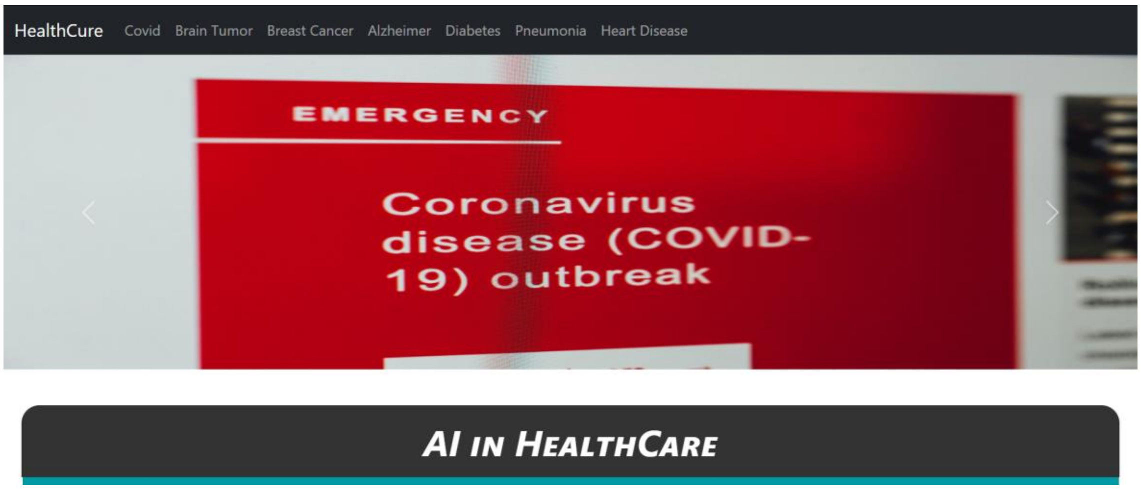


Fig. 3.1: Homepage

COVID-19 - Coronavirus disease 2019 (COVID-19) is a contagious disease caused by a virus, the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The first known case was identified in Wuhan, China, in December 2019. The disease spread worldwide, leading to the COVID-19 pandemic. COVID-19 transmits when people breathe in air contaminated by droplets and small airborne particles containing the virus. The risk of breathing these in is highest when people are in close proximity, but they can be inhaled over longer distances, particularly indoors.

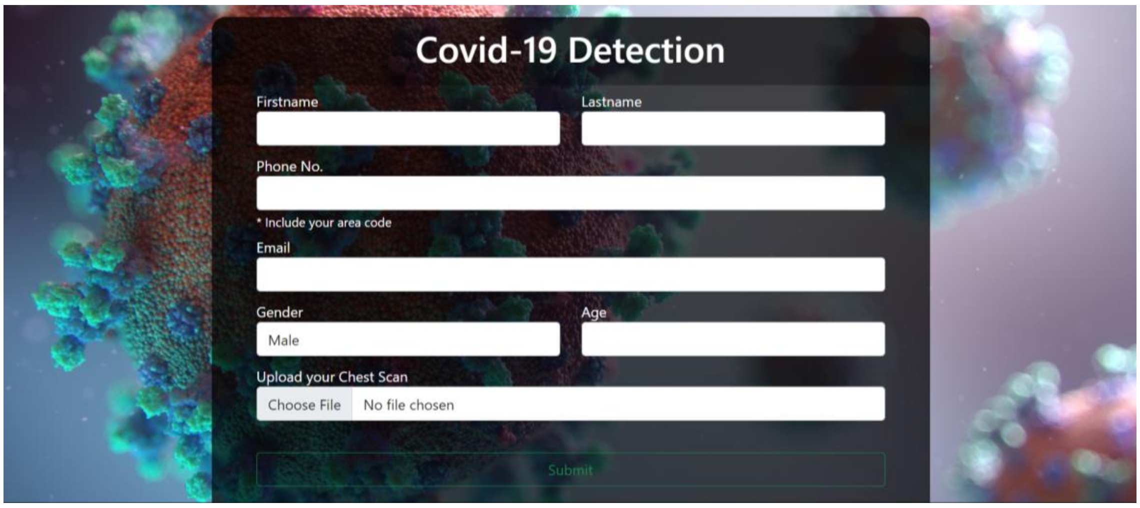
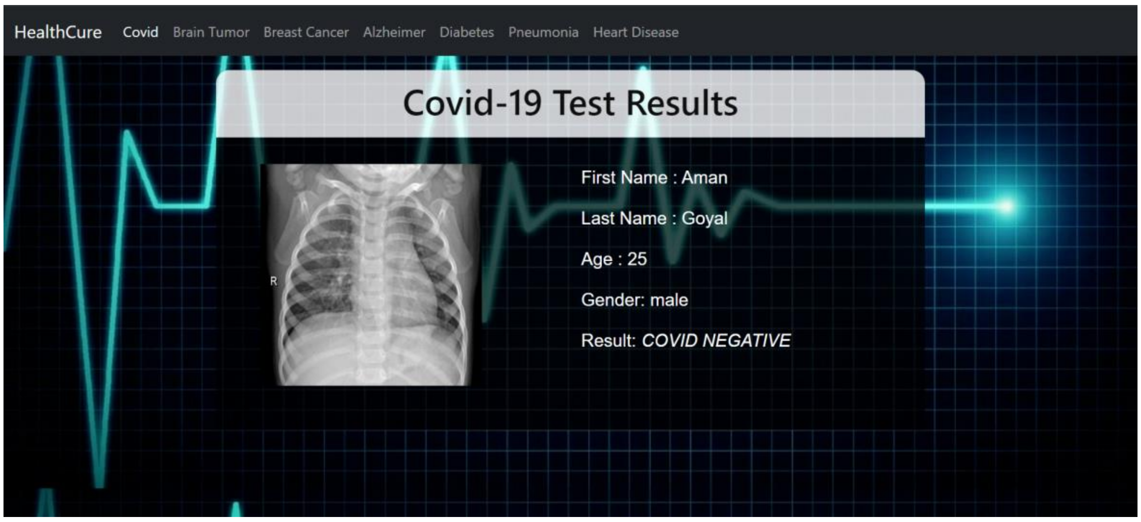


Fig. 3.2: COVID-19 Detection Page



First Name : Aman

Last Name : Goyal

Age : 25

Gender: male

Result: *COVID NEGATIVE*

Fig. 3.3: COVID-19 Results Page

Brain Tumor - A brain tumor occurs when abnormal cells form within the brain. There are two main types of tumors: malignant tumors and benign (non-cancerous) tumors. These can be further classified as primary tumors, which start within the brain, and secondary tumors, which most commonly have spread from tumors located outside the brain, known as brain metastasis tumors.

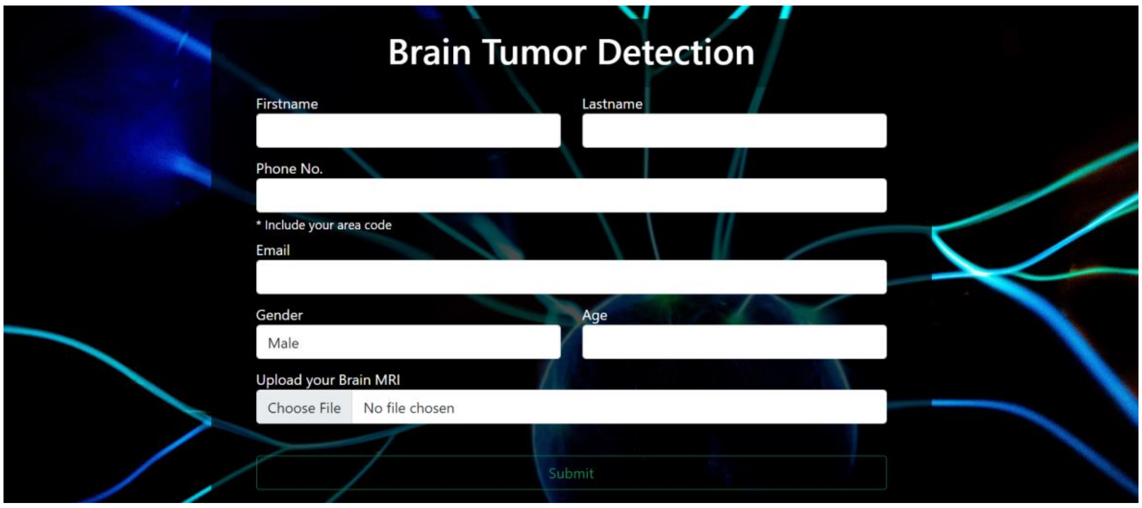
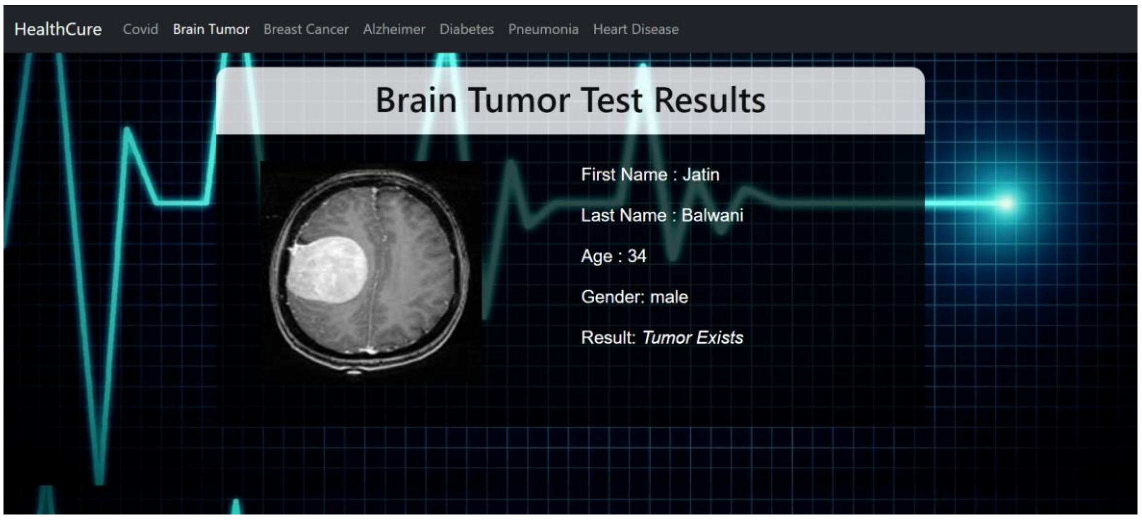


Fig. 3.4: Brain Tumor Detection Page



Brain Tumor Test Results

First Name : Jatin

Last Name : Balwani

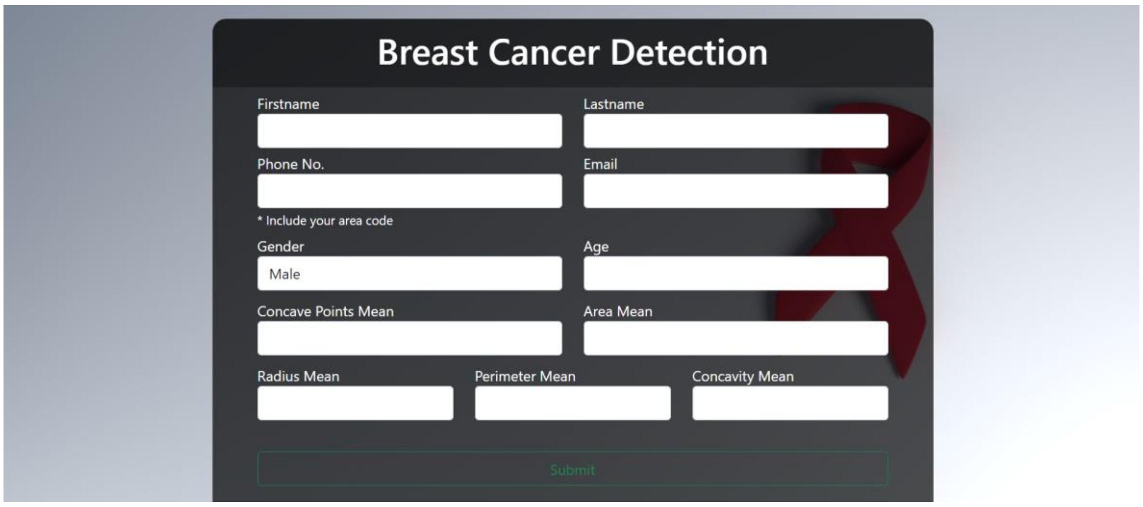
Age : 34

Gender: male

Result: *Tumor Exists*

Fig. 3.5: Brain Tumor Results Page

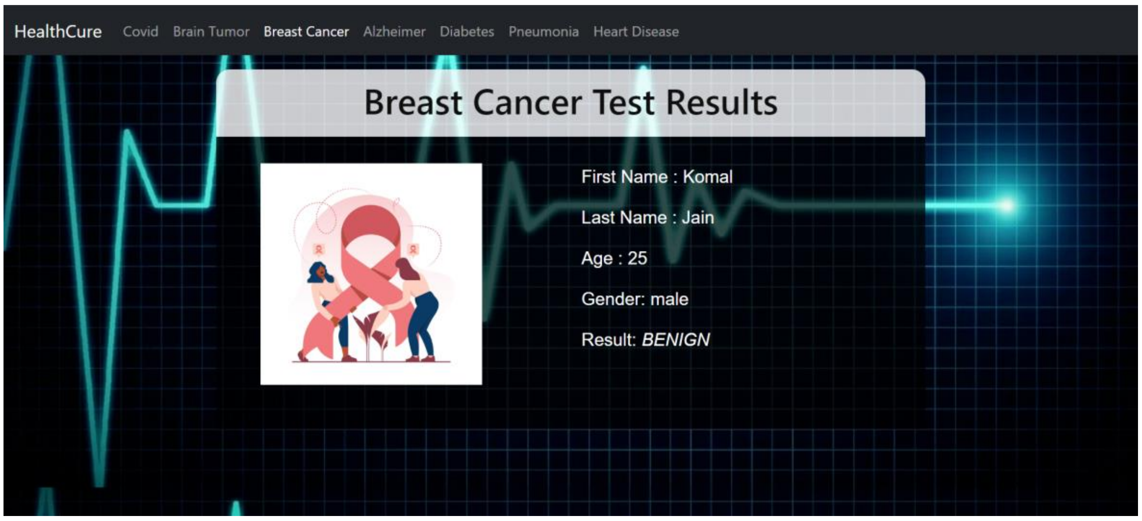
Breat Cancer - Breast cancer is cancer that develops from breast tissue. Signs of breast cancer may include a lump in the breast, a change in breast shape, dimpling of the skin, fluid coming from the nipple, a newly inverted nipple, or a red or scaly patch of skin. In those with distant spread of the disease, there may be bone pain, swollen lymph nodes, shortness of breath, or yellow skin. Breast cancer most commonly develops in cells from the lining of milk ducts and the lobules that supply these ducts with milk. The diagnosis of breast cancer is confirmed by taking a biopsy of the concerning tissue.



Breast Cancer Detection

Firstname	Lastname	
Phone No.	Email	
* Include your area code		
Gender: Male	Age	
Concave Points Mean	Area Mean	
Radius Mean	Perimeter Mean	Concavity Mean

Fig. 3.6: Breast Cancer Detection Page



Breast Cancer Test Results

First Name : Komal

Last Name : Jain

Age : 25

Gender: male

Result: *BENIGN*

Fig. 3.7: Breast Cancer Results Page

Alzheimer - Alzheimer's disease (AD) is a neurodegenerative disease that usually starts slowly and progressively worsens. It is the cause of 60–70% of cases of dementia. The most common early symptom is difficulty in remembering recent events. As the disease advances, symptoms can include problems with language, disorientation (including easily getting lost), mood swings, loss of motivation, self-neglect, and behavioral issues. As a person's condition declines, they often withdraw from family and society.

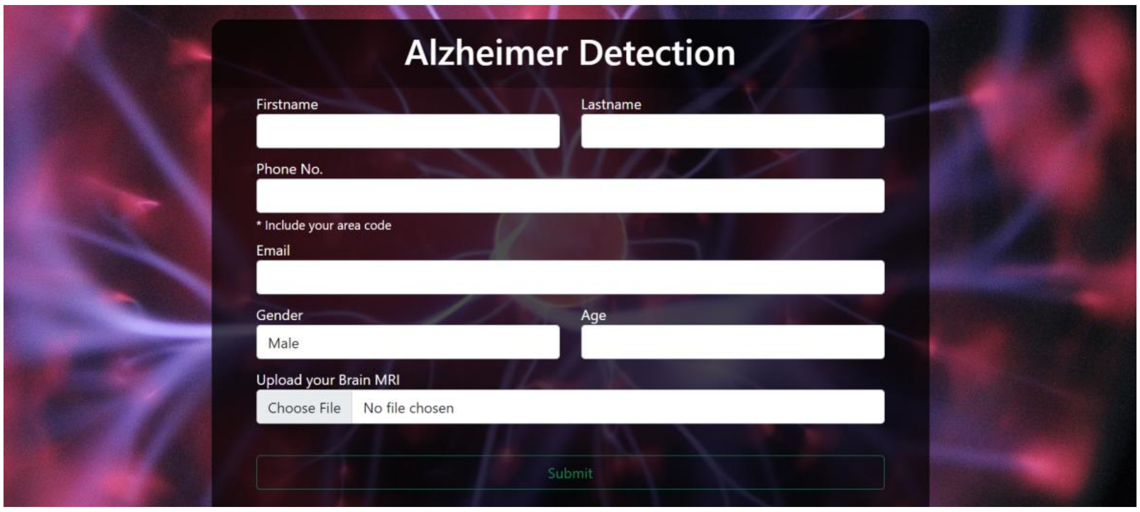
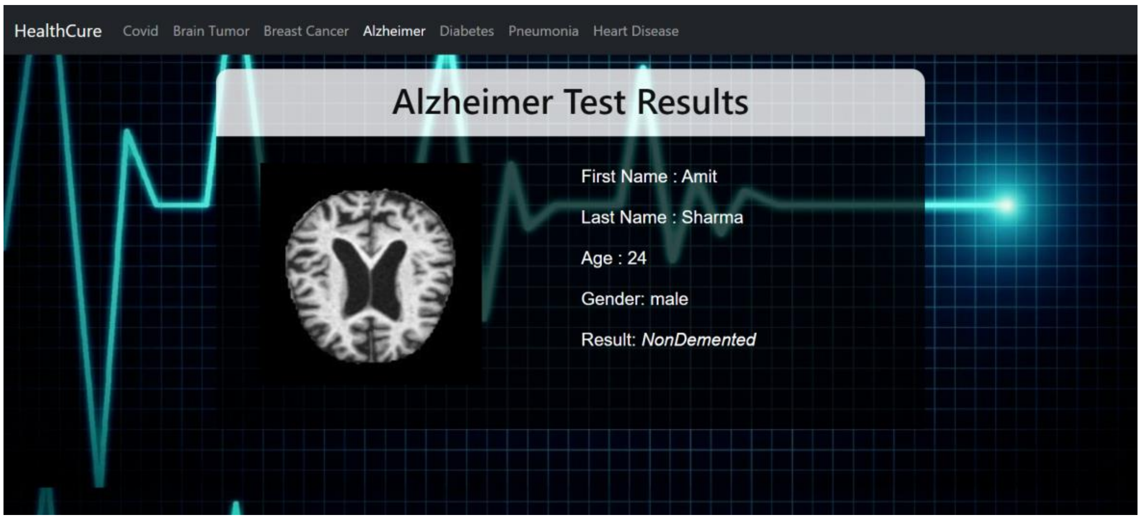


Fig. 3.8: Alzheimer Detection Page



Alzheimer Test Results

First Name : Amit

Last Name : Sharma

Age : 24

Gender: male

Result: *NonDemented*

Fig. 3.9: Alzheimer Results Page

Diabetes - Diabetes mellitus, commonly known as diabetes, is a group of metabolic disorders characterized by a high blood sugar level (hyperglycemia) over a prolonged period of time. Symptoms often include frequent urination, increased thirst and increased appetite. If left untreated, diabetes can cause many health complications. Acute complications can include diabetic ketoacidosis, hyperosmolar hyperglycemic state, or death. Serious long-term complications include cardiovascular disease, stroke, chronic and kidney disease.

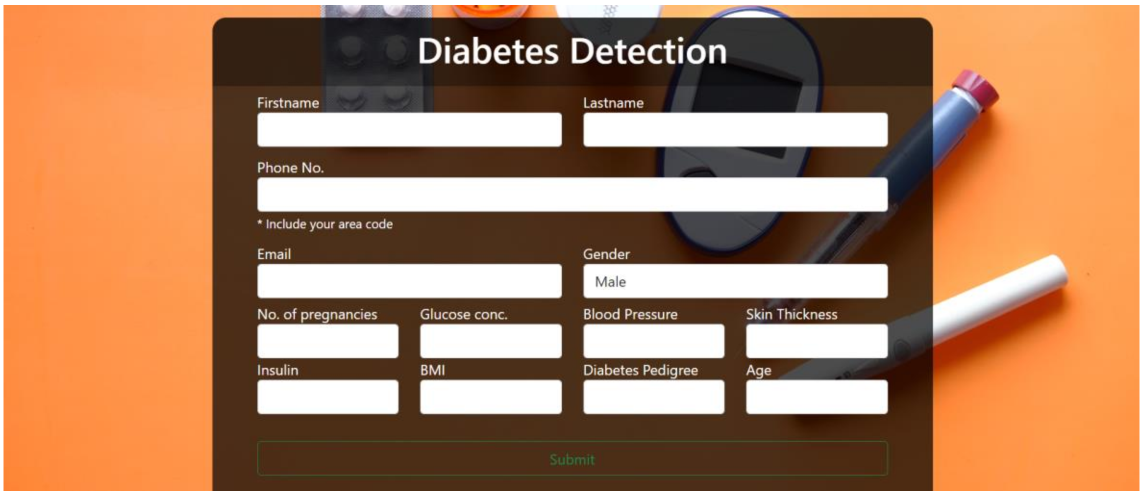
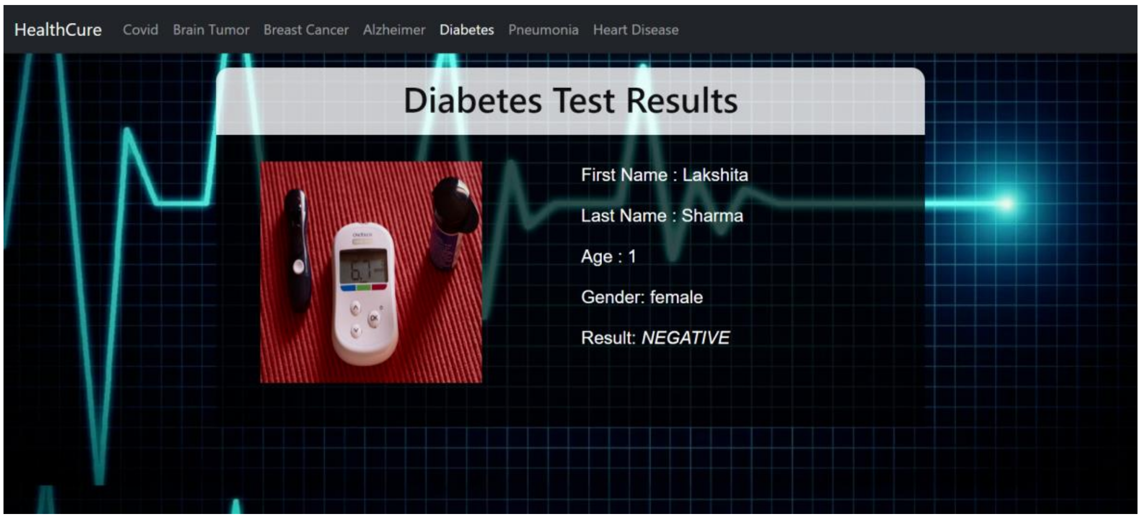


Fig. 3.10: Diabetes Detection Page



Diabetes Test Results

First Name : Lakshita

Last Name : Sharma

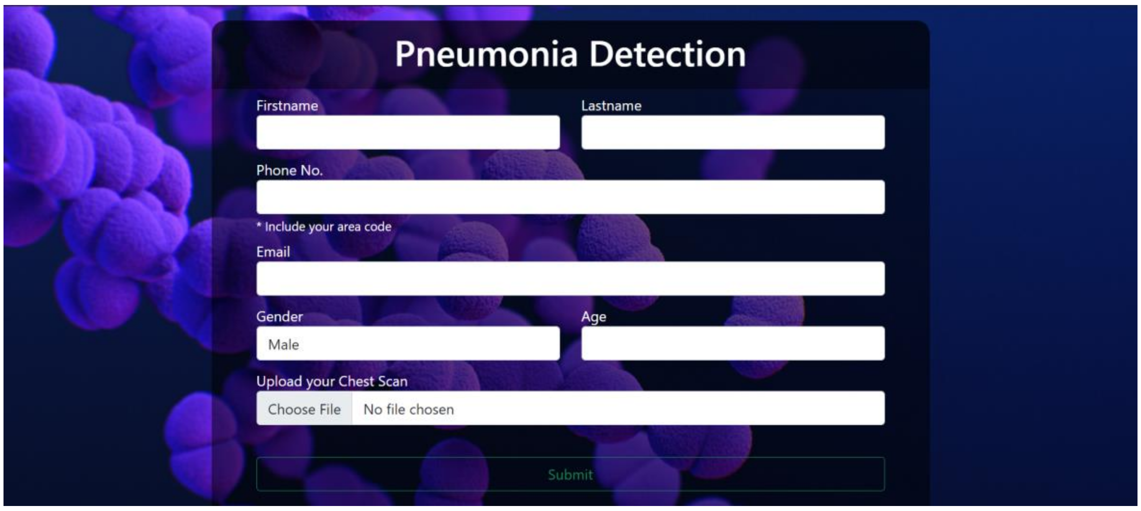
Age : 1

Gender: female

Result: *NEGATIVE*

Fig. 3.11: Diabetes Results Page

Pneumonia - Pneumonia is an inflammatory condition of the lung primarily affecting the small air sacs known as alveoli. Symptoms typically include some combination of productive or dry cough, chest pain, fever, and difficulty breathing. The severity of the condition is variable. Pneumonia is usually caused by infection with viruses or bacteria, and less commonly by other microorganisms.



Pneumonia Detection

Firstname

Lastname

Phone No.

* Include your area code

Email

Gender: Male

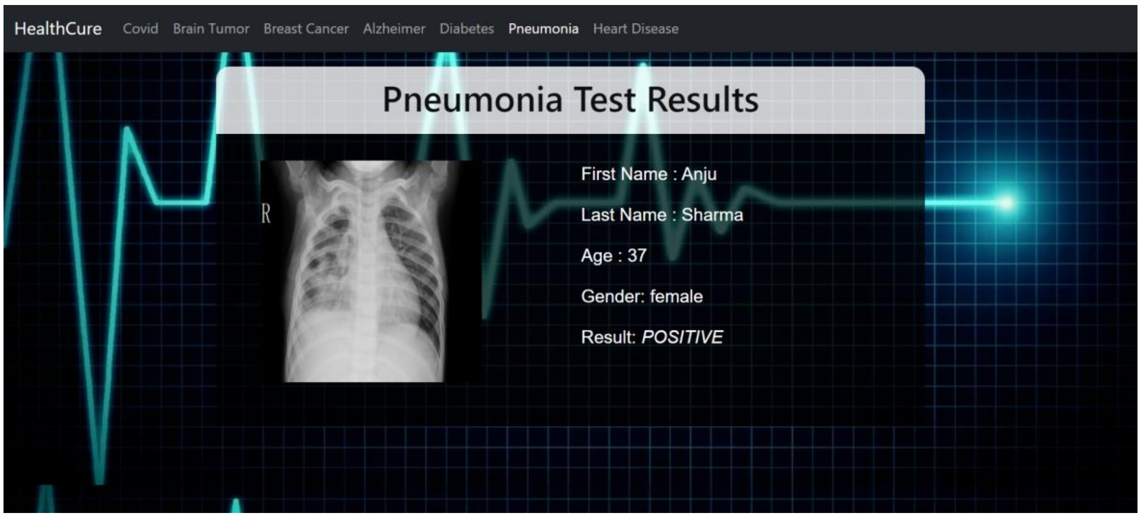
Age

Upload your Chest Scan

Choose File No file chosen

Submit

Fig. 3.12: Pneumonia Detection Page



Pneumonia Test Results

First Name : Anju

Last Name : Sharma

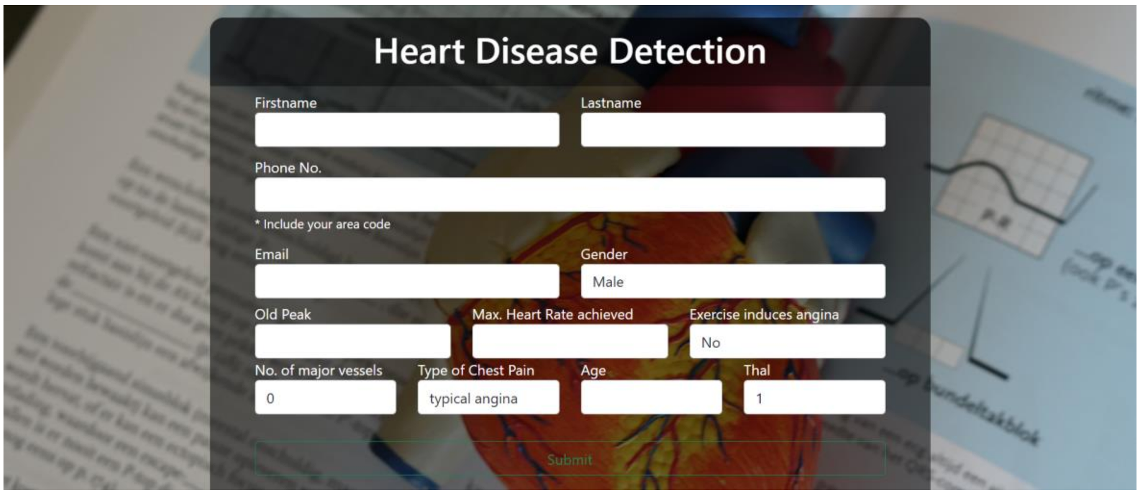
Age : 37

Gender: female

Result: *POSITIVE*

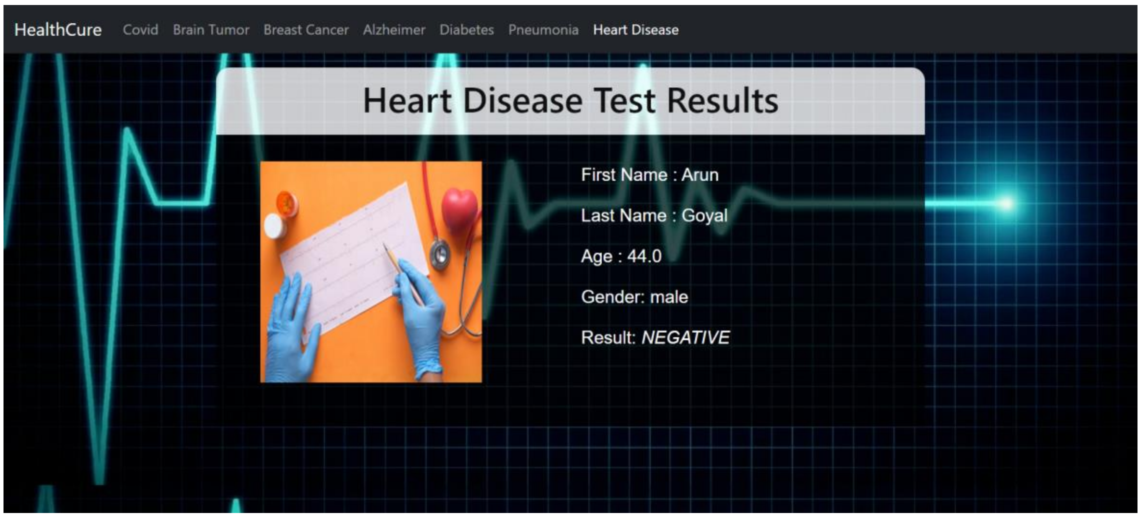
Fig. 3.13: Pneumonia Results Page

Heart Disease - Cardiovascular disease (CVD) is a class of diseases that involve the heart or blood vessels. CVD includes coronary artery diseases (CAD) such as angina and myocardial infarction (commonly known as a heart attack). Other CVDs include stroke, heart failure, hypertensive heart disease, rheumatic heart disease, cardiomyopathy, abnormal heart rhythms, congenital heart disease, valvular heart disease, carditis, aortic aneurysms, peripheral artery disease, thromboembolic disease, and venous thrombosis.


Heart Disease Detection

Firstname	
Lastname	
Phone No.	
* Include your area code	
Email	
Gender	Male
Old Peak	
Max. Heart Rate achieved	
Exercise induces angina	No
No. of major vessels	0
Type of Chest Pain	typical angina
Age	
Thal	1

Fig. 3.14: Heart Disease Detection Page



Heart Disease Test Results

First Name : Arun

Last Name : Goyal

Age : 44.0

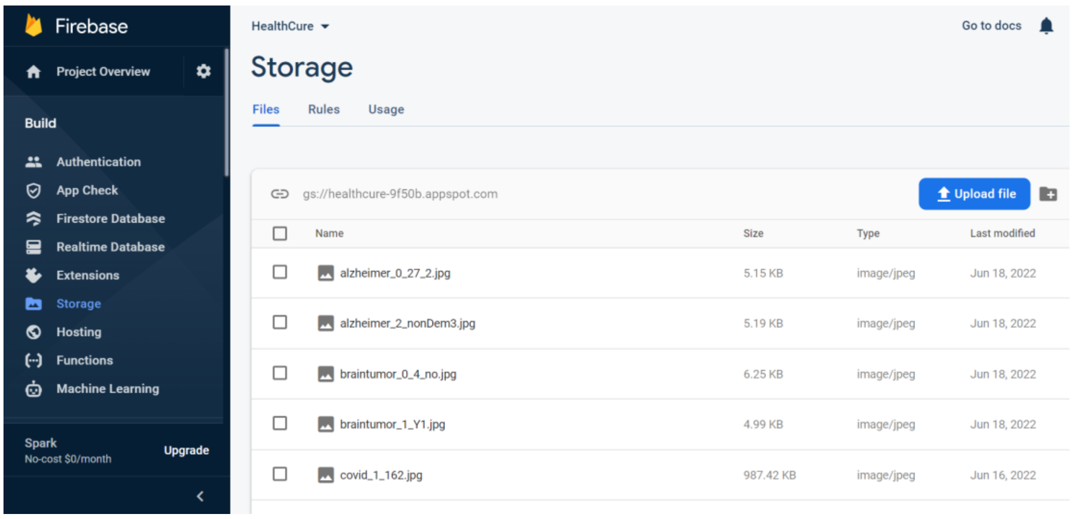
Gender: male

Result: *NEGATIVE*

Fig. 3.15: Heart Disease Results Page

3.3 Storage Layer

Cloud Storage: Cloud Storage for Firebase is built for app developers who need to store and serve user-generated content, such as photos or videos. Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

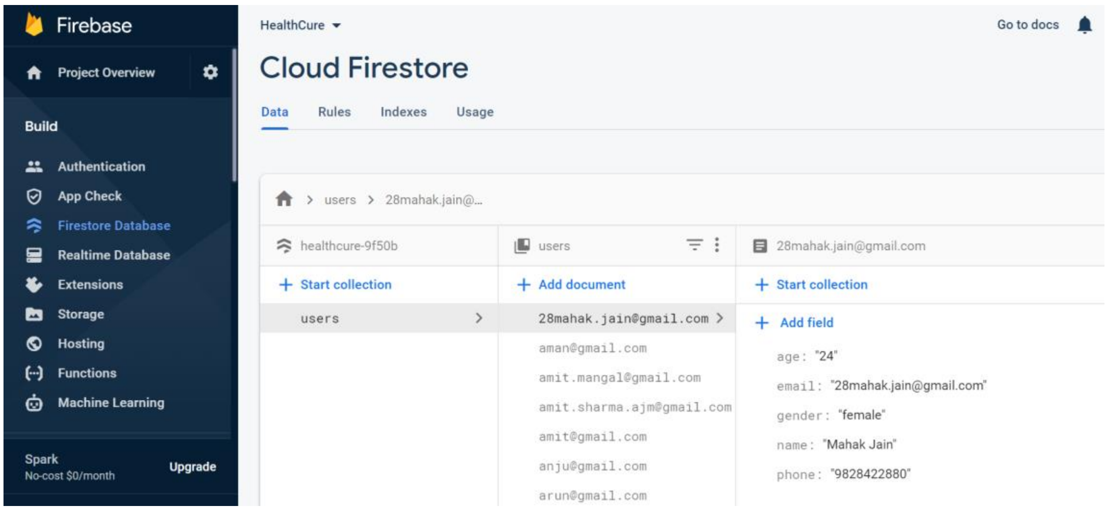


gs://healthcure-9f50b.appspot.com

Name	Size	Type	Last modified
alzheimer_0_27_2.jpg	5.15 KB	image/jpeg	Jun 18, 2022
alzheimer_2_nonDem3.jpg	5.19 KB	image/jpeg	Jun 18, 2022
braintumor_0_4_no.jpg	6.25 KB	image/jpeg	Jun 18, 2022
braintumor_1_Y1.jpg	4.99 KB	image/jpeg	Jun 18, 2022
covid_1_162.jpg	987.42 KB	image/jpeg	Jun 16, 2022

Fig. 3.16: Firebase Cloud Storage Snapshot

Cloud Firestore: Cloud Firestore is a flexible, scalable NoSQL cloud database to store and sync data for client- and server-side development. It is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions. Cloud Firestore is also available in native Node.js, Java, Python, Unity, C++ and Go SDKs, in addition to REST and RPC APIs. Following Cloud Firestore's NoSQL data model, you store data in documents that contain fields mapping to values. These documents are stored in collections, which are containers for your documents that you can use to organize your data and build queries. Documents support many different data types, from simple strings and numbers, to complex, nested objects. You can also create subcollections within documents and build hierarchical data structures that scale as your database grows. The Cloud Firestore data model supports whatever data structure works best for your app.



users > 28mahak.jain@...

healthcure-9f50b	users	28mahak.jain@gmail.com
users	28mahak.jain@gmail.com	age: "24"
	aman@gmail.com	email: "28mahak.jain@gmail.com"
	amit.mangal@gmail.com	gender: "female"
	amit.sharma.ajm@gmail.com	name: "Mahak Jain"
	amit@gmail.com	phone: "9828422880"
	anju@gmail.com	
	arun@gmail.com	

Fig. 3.17: Firebase Cloud Firestore Database Snapshot

3.4 Business Logic Layer

The entire backend logic resides in the app.py (Flask) file. It uses models, static, and templates directory for accessing CNN and Classical ML models trained for disease detection, images for UI, and frontend HTML files respectively. Trained models are stored in different file formats, such as .pkl, .sav, .h5, etc. depending on their complexity. All the python requirements are freezed using the pip tool in the requirements.txt file. The credentials for Firebase Cloud Storage and Firebase Cloud Firestore Database are obtained from the Google Cloud Platform console using service accounts and stored in separate JSON files.
Following are the code snippets for uploading a blob to the Cloud Storage and image cropping respectively.

```python
def upload_blob(bucket_name, source_file_name, destination_blob_name):
    """Uploads a file to the bucket."""
    # bucket_name = "your-bucket-name"
    # source_file_name = "local/path/to/file"
    # destination_blob_name = "storage-object-name"

    storage_client = gcp_storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_name)

    print(
        "File {} uploaded to {}.".format(
            source_file_name, destination_blob_name
        )
    )
```

```python
def preprocess_imgs(set_name, img_size):
    """

    Resize and apply VGG-15 preprocessing
    """

    set_new = []
    for img in set_name:
        img = cv2.resize(img,dsize=img_size,interpolation=cv2.INTER_CUBIC)
        set_new.append(preprocess_input(img))
    return np.array(set_new)


def crop_imgs(set_name, add_pixels_value=0):
    """

    Finds the extreme points on the image and crops the rectangular out of them
    """

    set_new = []
    for img in set_name:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

        # threshold the image, then perform a series of erosions +
        # dilations to remove any small regions of noise
        thresh = cv2.threshold(gray, 45, 255, cv2.THRESH_BINARY)[1]
        thresh = cv2.erode(thresh, None, iterations=2)
        thresh = cv2.dilate(thresh, None, iterations=2)

        # find contours in thresholded image, then grab the largest one
        cnts = cv2.findContours(
            thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        c = max(cnts, key=cv2.contourArea)

        # find the extreme points
        extLeft = tuple(c[c[:, :, 0].argmin()][0])
        extRight = tuple(c[c[:, :, 0].argmax()][0])
        extTop = tuple(c[c[:, :, 1].argmin()][0])
        extBot = tuple(c[c[:, :, 1].argmax()][0])

        ADD_PIXELS = add_pixels_value
        new_img = img[extTop[1]-ADD_PIXELS:extBot[1]+ADD_PIXELS,
                      extLeft[0]-ADD_PIXELS:extRight[0]+ADD_PIXELS].copy()
        set_new.append(new_img)

    return np.array(set_new)
```

Chapter 4

RESULTS

We obtained a decent accuracy for breast cancer and covid-19 detection models. The benchmark of 90% is cleared for both these models. For pneumonia and heart disease detection, we have achieved an accuracy of >80%. The diabetes detection model performed the worst. One possible reason for this performance could be insufficient number of data points leading towards underfitting of the model. The model also suffered from class imbalance in the dataset. For brain tumor detection, we attained 100% accuracy since we tested on 10 images only due to insufficient number of data points. The results for all the models are tabulated below.

Table 4.1 Model Architecture and Accuracy

Disease	**Model Architecture**	**Accuracy**
Alzheimer	CNN	73.54%
Brain Tumor	CNN with pre-trained VGG16 weights	100%
Breast Cancer	Random Forest	91.81%
COVID-19	CNN	93%
Diabetes	Random Forest	66.8%
Pneumonia	CNN	83.17%
Heart Disease	XGBoost	86.96%

The alzheimer detection model didn’t perform good as well. Since there were four classes in the dataset, we need a more complex architecture to train this model or we can use weights from a pre-trained model as we have done in the case of brain tumor detection. VGG16 was used as the base model for transfer learning in brain tumor detection.

Following are a few plots for inferences.

COVID-19 Detection Model

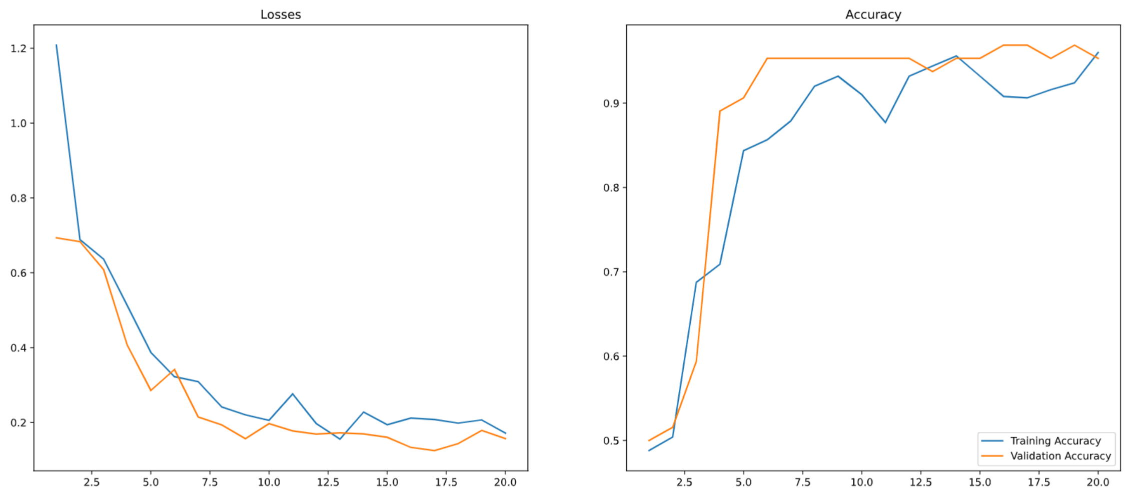


Fig. 4.1: Loss and Accuracy Plot for training of the COVID-19 detection model

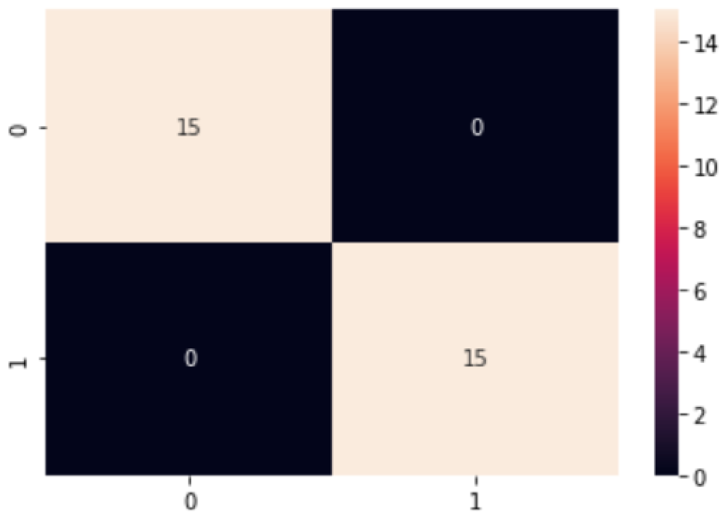


Fig. 4.2: Confusion Matrix for predictions of the COVID-19 detection model

Brain Tumor Detection Model

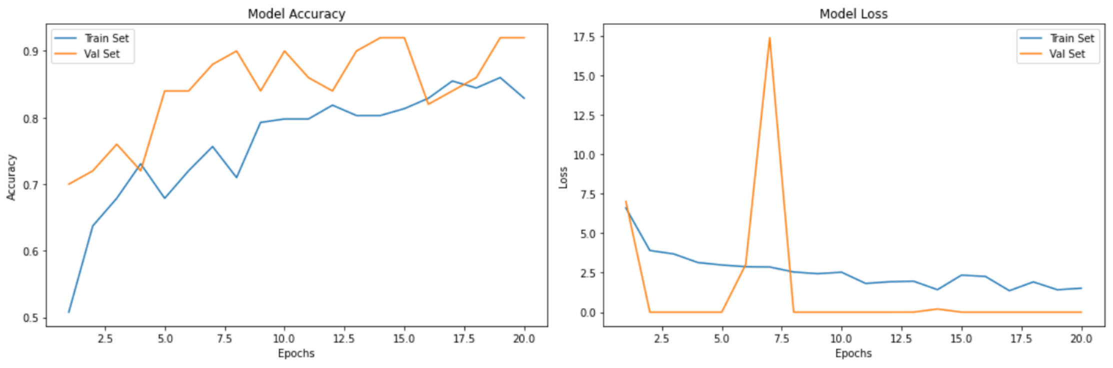


Fig. 4.3: Loss and Accuracy Plot for training of the Brain Tumor detection model

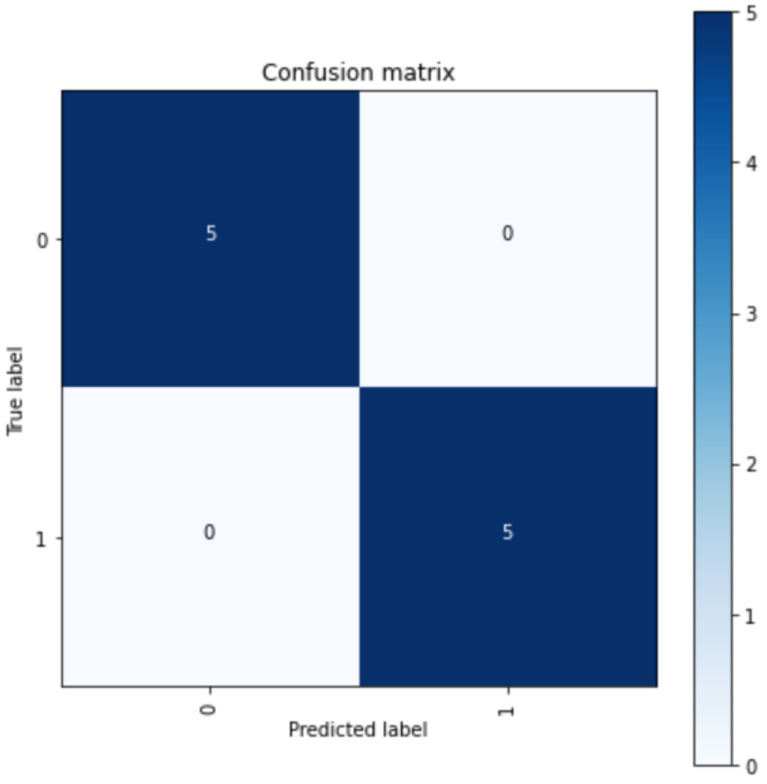


Fig. 4.4: Confusion Matrix for predictions of the Brain Tumor detection model

Alzheimer Detection Model

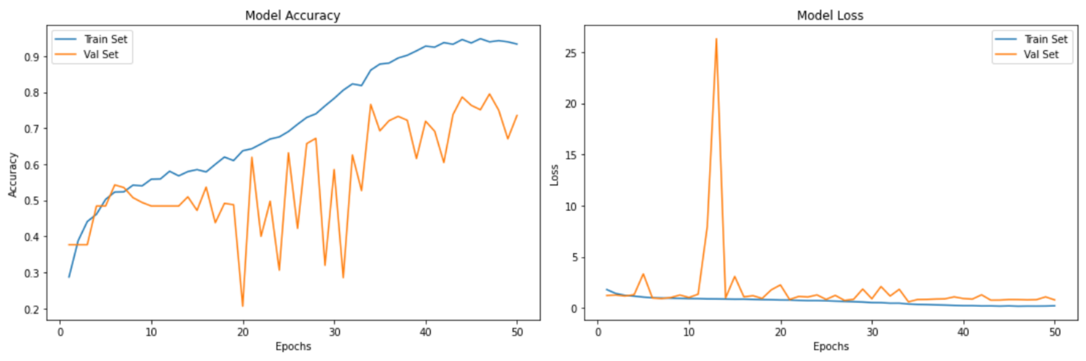


Fig. 4.5: Loss and Accuracy Plot for training of the Alzheimer detection model

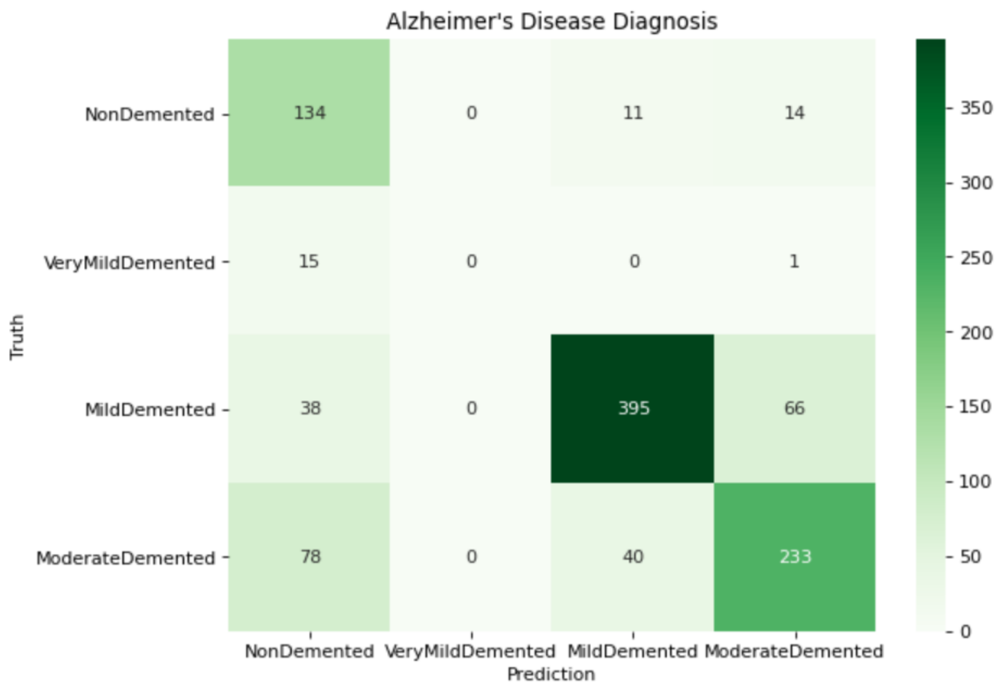


Fig. 4.6: Confusion Matrix for predictions of the Alzheimer detection model

Pneumonia Detection Model

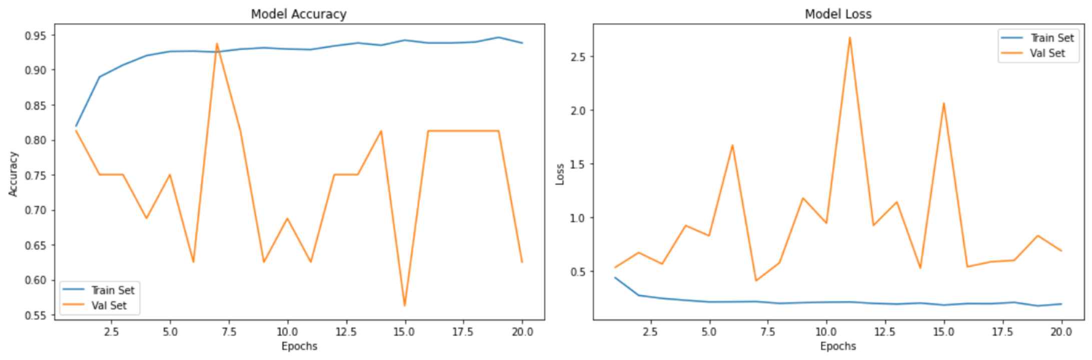


Fig. 4.7: Loss and Accuracy Plot for training of the Pneumonia detection model

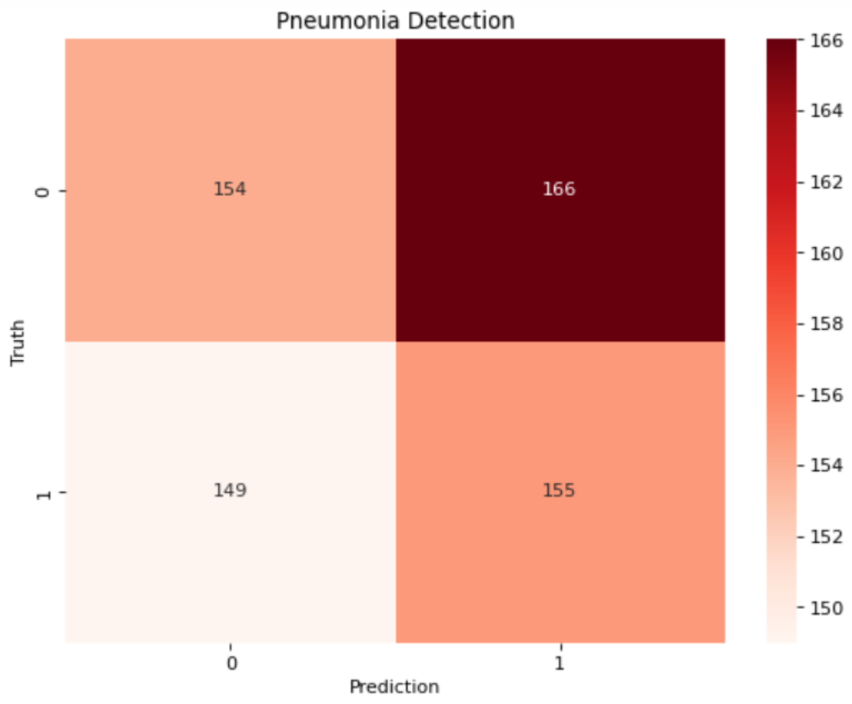


Fig. 4.8: Confusion Matrix for predictions of the Pneumonia detection model

Diabetes Detection Model

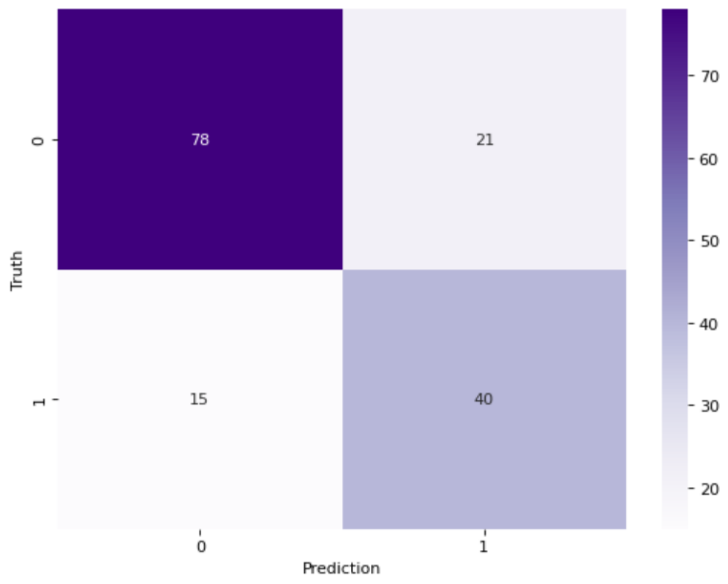


Fig. 4.9: Confusion Matrix for predictions of the Diabetes detection model

Breast Cancer Detection Model

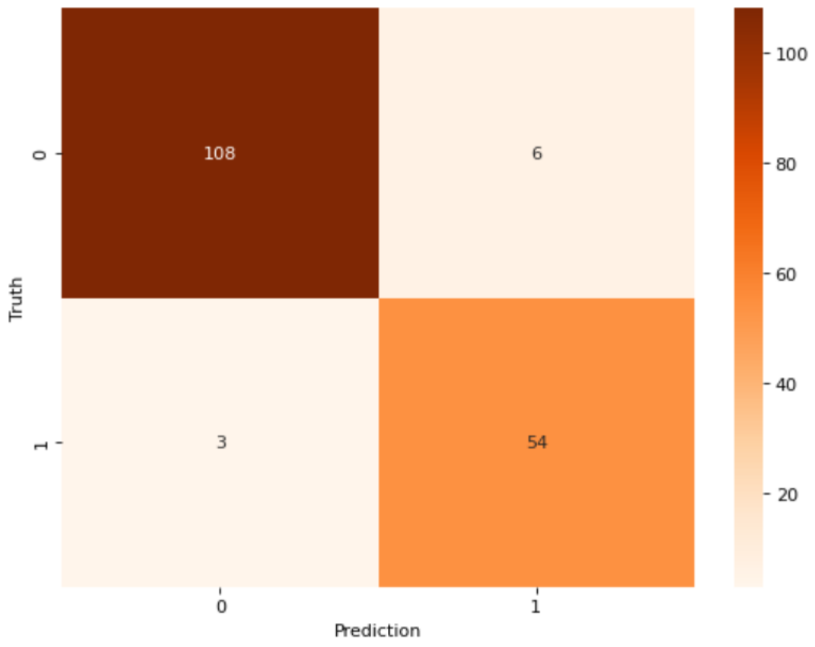


Fig. 4.10: Confusion Matrix for predictions of the Breast Cancer detection model

Heart Disease Detection Model

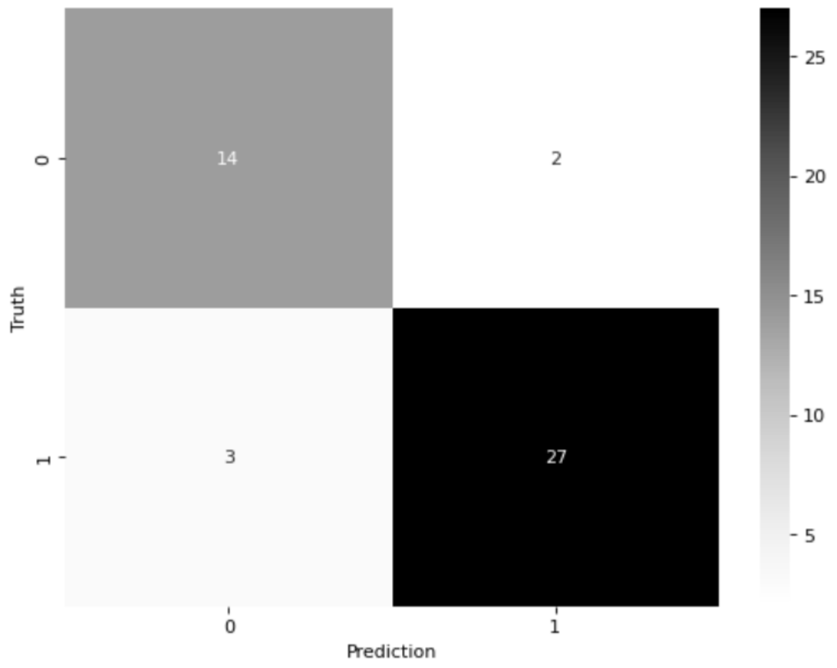


Fig. 4.11: Confusion Matrix for predictions of the Heart Disease detection model

4.1 Comparison with related existing work

The table given below shows the comparison of the accuracy of existing related work published in papers and our model.

Table 4.2 Accuracy comparison with related existing work

S.No.	**Disease Detection**	**Model Architecture**	**Accuracy of existing related work**	**Accuracy of our model**
1.	Alzheimer	CNN	87.1% [17]	73.54%
2.	Brain Tumor	CNN with pre-trained VGG16 weights	99% [18]	100%
3.	Breast Cancer	Random Forest	95.08% [19]	91.81%
4.	COVID-19	CNN	98.03% [20]	93%
5.	Diabetes	Random Forest	84% [21]	74.68%
6.	Pneumonia	CNN	91.2% [22]	83.17%
7.	Heart Disease	XGBoost	87.5% [23]	86.96%

Chapter 5

CONCLUSION & FUTURE WORK

Traditional machine learning methods (such as multilayer perception machines, support vector machines, etc.) mostly use shallow structures to deal with a limited number of samples and computing units. When the target objects have rich meanings, the performance and generalization ability of complex classification problems are obviously insufficient. The convolution neural network (CNN) developed in recent years has been widely used in the field of image processing because it is good at dealing with image classification and recognition problems and has brought great improvement in the accuracy of many machine learning tasks. It has become a powerful and universal deep learning model.

Deep convolution neural networks are used to identify scaling, translation, and other forms of distortion-invariant images. In order to avoid explicit feature extraction, the convolutional network uses feature detection layer to learn from training data implicitly, and because of the weight sharing mechanism, neurons on the same feature mapping surface have the same weight. The ya training network can extract features by W parallel computation, and its parameters and computational complexity are obviously smaller than those of the traditional neural network. Its layout is closer to the actual biological neural network. Weight sharing can greatly reduce the complexity of the network structure. Especially, the multi-dimensional input vector image WDIN can effectively avoid the complexity of data reconstruction in the process of feature extraction and image classification. Deep convolution neural network has incomparable advantages in image feature representation and classification. However, many researchers still regard the deep convolutional neural network as a black box feature extraction model. To explore the connection between each layer of the deep convolutional neural network and the visual nervous system of the human brain, and how to make the deep neural network incremental, as human beings do, to compensate for learning, and to increase understanding of the details of the target object, further research is needed.

References

[1] E. Newman, M. Kilmer, L. Horesh, Image classification using local tensor singular value decompositions (IEEE, international workshop on computational advances in multi-sensor adaptive processing. IEEE, Willemstad, 2018), pp. 1–5. [Accessed 3 June 2022].

[2] X. Wang, C. Chen, Y. Cheng, et al, Zero-shot image classification based on deep feature extraction. United Kingdom: IEEE Transactions on Cognitive & Developmental Systems, 10(2), 1–1 [Accessed 4 June 2022]..

[3] A.A.M. Al-Saffar, H. Tao, M.A. Talab, Review of deep convolution neural network in image classification (International conference on radar, antenna, microwave, electronics, and telecommunications. IEEE, Jakarta, 2018), pp. 26–31. [Accessed 4 June 2022].

[4] Z. Yan, V. Jagadeesh, D. Decoste, et al., HD-CNN: hierarchical deep convolutional neural network for image classification. Eprint Arxiv 4321-4329 [Accessed 5 June 2022].

[5] W. Wiersinga, Joost et al., "Pathophysiology transmission diagnosis and treatment of coronavirus disease 2019 (COVID-19): a review", *Jama*, vol. 324.8, pp. 782-793, 2020. [Accessed 2 July 2022].

[6] U. Akhtar, M. Asad, K. and L. Sungyoung, "Challenges in Managing Real-Time Data in Health Information System (HIS)", *International Conference on Smart Homes and Health Telematics*, [Accessed 3 July 2022].

[7] [online] Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019. [Accessed 3 July 2022].

[8] Yicheng Fang, Huangqi Zhang, Jicheng Xie et al., "Sensitivity of Chest CT for COVID-19: Comparison to RT-PCR", *Radiology*, Feb 2020. [Accessed 4 July 2022].

[9] M. K. Hasan, M. A. Alam, L. Dahal, M. T. E. Elahi, S. Roy, S. R. Wahid, et al., "Challenges of Deep Learning Methods for COVID-19 Detection Using Public Datasets", *medRxiv*, 2020. [Accessed 4 July 2022].

[10] N Zhu, D Zhang, W Wang, X Li, B Yang, J Song et al., "A novel coronavirus from patients with pneumonia in China 2019", *N Engl J Med.*, vol. 382, no. 8, pp. 727-33, 2020, [online] Available: https://doi.org/10.1056/NEJMoa2001017. [Accessed 5 July 2022].

[11] J. McMorran and D.C. Crowther, "Fine needle aspiration cytology (breast)", *General Practice Notebook – a UK medical reference on the world wide web*, [Accessed 5 July 2022].

[12] C. Pena-Reyes and M. Sipper, "A fuzzy approach to breast cancer diagnosis", *Artificial intelligence medicine*, vol. 17, pp. 131-135, [Accessed 6 July 2022].

[13] Rajesh C. Patil and A. S. Bhalchandra, "Brain Tumour Extraction from MRI Images Using MATLAB", *International Journal of Electronics Communication & Soft Computing Science and Engineering*, vol. 2, no. 1, [Accessed 7 July 2022].

[14] GB Karas, P Scheltens, SA Rombouts et al., "Global and local gray matter loss in mild cognitive impairment and Alzheimer's disease", *Neuroimage*, vol. 23(2), pp. 708-716, [Accessed 8 July 2022].

[15] W Burger and Burge MJ, Digital Image Processing, London:Springer-Verlag, [Accessed 8 July 2022].

[16] A. Z. Woldaregay et al., "Data-driven modeling and prediction of blood glucose dynamics: Machine learning applications in type 1 diabetes", *Artif. Intell. Med.*, vol. 98, pp. 109-134, [Accessed 9 July 2022].

[17] H. Fuse, K. Oishi, N. Maikusa, T. Fukami and J. A. D. N. Initiative, "Detection of Alzheimer's Disease with Shape Analysis of MRI Images," 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), 2018, pp. 1031-1034, doi: 10.1109/SCIS-ISIS.2018.00171 [Accessed 9 July 2022].

[18] M. Gurbină, M. Lascu and D. Lascu, "Tumor Detection and Classification of MRI Brain Image using Different Wavelet Transforms and Support Vector Machines," 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019, pp. 505-508, doi: 10.1109/TSP.2019.8769040. [Accessed 9 July 2022].

[19] L. H. A. al-sammarraie and A. A. Ibrahim, "Predicting Breast Cancer in Fine Needle Aspiration Images Using Machine Learning," 2020 4th International

Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2020, pp. 1-4, doi: 10.1109/ISMSIT50672.2020.9254891. [Accessed 9 July 2022].

[20] M. Fradi and M. Machhout, "Real-Time Application for Covid-19 Class Detection based CNN Architecture," 2021 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS), 2021, pp. 1-6, doi: 10.1109/DTS52014.2021.9498055. [Accessed 10 July 2022].

[21] R. Katarya and S. Jain, "Comparison of Different Machine Learning Models for diabetes detection," 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE), 2020, pp. 1-5, doi: 10.1109/ICADEE51157.2020.9368899. [Accessed 10 July 2022].

[22] Z. -Y. Yang and Q. Zhao, "A Multiple Deep Learner Approach for X-Ray Image-Based Pneumonia Detection," 2020 International Conference on Machine Learning and Cybernetics (ICMLC), 2020, pp. 70-75, doi: 10.1109/ICMLC51923.2020.9469043. [Accessed 10 July 2022].

[23] A. Ed-Daoudy and K. Maalmi, "Real-time machine learning for early detection of heart disease using big data approach," 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), 2019, pp. 1-5, doi: 10.1109/WITS.2019.8723839. [Accessed 10 July 2022].