BODY PART CLASSIFICATION FROM X-RAY IMAGES

A PROJECT REPORT

Submitted by

RHUTHVIK DENDUKURI [CB.EN.U4CSE19614]
KRITHIKA M A [CB.EN.U4CSE19629]
LEEPAAKSHI G [CB.EN.U4CSE19630]
S SAI CHANDRA KAUSHIK [CB.EN.U4CSE19651]

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

JUNE 2023

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, COIMBATORE – 641 112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **BODY PART CLASSIFICATION FROM X-RAY IMAGES** submitted by Rhuthvik Dendukuri [CB.EN.U4CSE19614], Krithika M A [CB.EN.U4CSE19629], Leepaakshi G [CB.EN.U4CSE19630], S Sai Chandra Kaushik [CB.EN.U4CSE19651] in partial fulfillment of the requirements for the award of Degree **Bachelor of Technology** in Computer Science and Engineering is a bonafide record of the work carried out under our guidance and supervision at the Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore.

Dr. Raghesh Krishnan K
Assistant professor
Department of CSE

Dr.Vidhya Balasubramanian
Chairperson
Department of CSE

Evaluated on:

INTERNAL EXAMINER

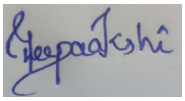
EXTERNAL EXAMINER

DECLARATION

We, the undersigned solemnly declare that the project report **BODY PART CLASSIFICATION FROM X-RAY IMAGES** is based on our own work carried out during the course of our study under the supervision of Dr. Raghesh Krishnan K, Assistant professor, Computer Science and Engineering, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgement has been made wherever the findings of others has been cited.

Rhuthvik Dendukuri[CB.EN.U4CSE19614]- 

Krithika M A [CB.EN.U4CSE19629]- 

Leepaakshi G[CB.EN.U4CSE19630]- 

S Sai Chandra Kaushik[CB.EN.U4CSE19651]-

ABSTRACT

The automatic analysis and classification of X-ray pictures has become critical in the area of medical imaging to optimise clinical practise. The capacity to appropriately classify X-ray images based on the bodily sections displayed can greatly enhance diagnosis efficiency, ultimately leading to improved patient outcomes. In this paper, we report our research on X-ray image classification using the UNIFESP dataset, which has 22 classes and around 1738 images. It is necessary to note that this dataset is quite imbalanced, making accurate classification challenging.

Our research began by by exploring traditional machine learning algorithms to solve the X-ray image classification challenge. Multiple experiments with several image enhancing techniques were performed and then employed popular feature extraction techniques to improve the overall quality and details of the input images. Finally, various machine learning approaches were explored. Despite several trials, the earliest models had very modest accuracy, averaging approximately 50%. To boost classification accuracy, deep learning models were taken into consideration, which can automatically learn and retrieve high-level characteristics from input data. To improve the performance, data augmentation techniques with Deep Convolutional Generative Adversarial Networks (DCGAN) were employed to create more training examples and increase the model's generalisation. Following that, we used Gabor filters, which are well-known for their ability to capture texture and edge information, to improve the discriminative ability of the input images. Finally, different Convolutional Neural Network (CNN) architectures were assessed. Among the models, EfficientNetB3 outperformed the others, reaching an accuracy of 86.94%.

The importance of this research is in its contribution to tackling the issues related to X-ray image classification in a practical clinical situation. We wanted to construct a reliable and accurate system for automatic X-ray image categorization by utilising the

UNIFESP dataset, which comprises a varied spectrum of body parts and medical disorders. The uneven structure of the dataset added difficulties that demanded the use of novel approaches and deep learning models to attain better outcomes. Our results illustrate the effectiveness of using advanced deep learning models, notably EfficientNetB3, to solve the classification challenge with exceptional accuracy.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our beloved Satguru **Sri Mata Amritanandamayi Devi** for providing the bright academic climate at this university, which has made this entire task appreciable. This acknowledgment is intended to thank all those people involved directly or indirectly with our project. We would like to thank our Pro Chancellor **Swami Abhayamritananda Puri**, Vice Chancellor **Dr.Venkat Rangan.P** and **Dr.Bharat Jayaraman**, Dean, Faculty of AI & Computing(Computing), Amrita Vishwa Vidyapeetham for providing us with the necessary infrastructure required for the completion of the project. We express our thanks to **Dr.Vidhya Balasubramanian**, Chairperson, Department of Computer Science Engineering and Principal, School of Computing, Amrita Vishwa Vidyapeetham, **Dr.C.Shunmuga Velayutham and Dr.N.Lalithamani,** Vice Chairpersons of the Department of Computer Science and Engineering for their valuable help and support during our study. We express our gratitude to our guide, **Dr. Raghesh Krishnan K** for their guidance, support and supervision. We feel extremely grateful to **Dr. Padmavati S, Mrs. Sujee R, Mrs. Anupa Vijay, Mrs. Aarti R** for their feedback and encouragement which helped us to complete the project. We would also like to thank the entire fraternity of the Department of Computer Science and Engineering. We would like to extend our sincere thanks to our family and friends for helping and motivating us during the course of the project. Finally, we would like to thank all those who have helped, guided and encouraged us directly or indirectly during the project work. Last but not the least, we thank God for his blessings which made our project a success.

Rhuthvik Dendukuri[CB.EN.U4CSE19614]

Krithika M A[CB.EN.U4CSE19629]

Leepaakshi G[CB.EN.U4CSE19630]

S Sai Chandra Kaushik[CB.EN.U4CSE19651]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

BRIEF Binary Robust Independent Elementary Feature

CLAHE Contrast Limited Adaptive Histogram Equalisation

CNN Convolutional Neural Network

DOG Difference Of Gaussian Kernel

DICOM Digital Imaging and Communications in Medicine

FAST Features From Accelerated Segment Test

FAIR Facebook AI Research

HFE High Frequency Estimation

NAS Neural Architecture Search

ORB Oriented FAST and Rotated BRIEF

PNG Portable Network Graphics

SGD Stochastic Gradient Descent

SOP Service-Object Pair

SURF Speeded-Up Robust Features

SVC Support Vector Classifier

TVH Television Homomorphic

UID Unique Identification

XGBoost Extreme Gradient Boost

Chapter 1

INTRODUCTION

Modern hospitals collect a wide variety of imaging data for diagnosis, planning of care, and monitoring of therapeutic response. Together with other image sources (such as research literature and clinical manuals), these sizable image collections offer fresh possibilities for using enormous image data to create computerised tools for image-based diagnosis, instruction, and biomedical research. These applications are based on the discovery, retrieval, and categorization of patient data that indicate comparable clinical outcomes, such as images that correlate to the same diagnosis. Although there is a rush to employ all available picture sources for these applications, not all image sources provide the proper labels. In cases where appropriate labels are absent, identifying the image's modality is a primary step because the content and semantics of an image can vary depending on its modality. There are also variations in the appearance of images based on the individual diseases depicted. X-ray images are one of the most common performed and well-understood diagnostic imaging techniques. However, classification of X-ray images is challenging due to irregular brightness, contrast, and artifacts caused by prostheses and other implants. There are also high intra-class variability and inter-class similarity among classes.

1.1 Background

X-ray imaging has served as an important diagnostic technique, providing vital insights into the internal structures of the human body. It is critical in determining the presence of a wide range of medical disorders, from fractures and infections to tumours and lung ailments. Physicians can visualise anatomical features and identify anomalies or pathologies using X-ray scans, which are collected as gray-scale images. However, analysing and interpreting X-ray pictures can be a difficult and time-consuming operation. Traditionally, this approach has relied significantly on radiologists' and physicians' skills, who manually evaluate the images to identify and classify the individual

bodily parts represented. This manual approach is not only time-consuming, but also prone to human error and subjectivity, resulting in potential diagnostic discrepancies.

1.2 Problem Definition

To overcome the limitations of manual analysis and interpretation of X-ray images and improve clinical practise, automatic analysis and classification algorithms that can reliably detect and label X-ray images based on the body parts displayed require immediate attention. The primary objective of the research is to build a reliable and efficient model that can automatically classify X-ray images into discrete groups corresponding to different parts of the body. The emphasis is on using powerful machine learning and deep learning models to process massive datasets of X-ray images, such as the UNIFESP dataset, and accurately cluster the images based on the precise body sections depicted. By automating the initial screening and classification of X-ray images, healthcare personnel can devote more time to analysing specific abnormalities or anomalies present in recognised body parts, thereby enhancing the accuracy and efficacy of medical diagnoses.

1.3 Objectives

- Build an automated system for classifying X-ray images based on the bodily parts represented.
- Address the problem of dataset imbalance in the UNIFESP X-ray dataset in order to achieve precise classification.
- Boost classification accuracy by improving the quality of X-ray images with image enhancement techniques.
- Assess the system's effectiveness and its possible impact on clinical practise.

The remainder of this research paper is structured as follows: in Section 2, overview of related works in the field of X-ray image classification are provided. Section 3 & 4 details the methodology and experimental setup, outlining the data pre-processing steps, ML models, and deep learning architectures utilized. Section 5 presents our experimental results and performance evaluation and discussion of the findings.

Finally, in Section 6, conclusion of the paper has been given, highlighting the implications of our research and outlining future directions for further improvements in X-ray image classification.

Chapter 2

LITERATURE SURVEY

In the past decade, a variety of issues, including categorization and diagnosis, have been addressed using artificial intelligence approaches like deep learning, machine learning, fuzzy logic, artificial neural networks, genetic programming, and regression techniques. In this section, we provide a quick overview of the research on the use of deep learning and image processing in the medical industry.

2.1 Literature Review

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks Applied on Medical Image Classification and Retrieval CNNs are a state-of-the-art image classification technique that learns the optimal image features for a given classification task. Different CNN architectures learn different levels of semantic image representation and thus an ensemble of CNNs will enable higher quality features to be extracted.[6]Deep convolutional neural networks (CNNs), with transferable knowledge, have been employed as a solution to limited annotated data through: 1) fine-tuning generic knowledge with a relatively smaller amount of labelled medical imaging data, and 2) learning image representation that is invariant to different domains. These approaches, however, are still reliant on labelled medical image data.

2.1.2 Hierarchical unsupervised feature extractor

Use a new hierarchical unsupervised feature extractor to reduce reliance on annotated training data. Unsupervised approach uses a multi-layer zero-bias convolutional auto-encoder that constrains the transformation of generic features from a pre-trained CNN (for natural images) to non-redundant and locally relevant features for the medical image data.[5]

2.1.3 Bag-of-Visual-Words Applied on Medical Image Classification and Retrieval

BoW is used as the representation of endomicroscopic images and achieves high accuracy in the tasks of classifying the images into neoplastic (pathological) and benign.[2] In an application to texture representation for mammography tissue classification and segmentation is presented. Displayed the best medical annotation results using the BoW approach, where the features were local patches of different sizes taken at every position and scaled to a common size.[7]

2.1.4 Data augmentation

The size of the training data that is available has a significant impact on how well the supervised machine learning models perform because they are very data-dependent. It might be challenging to produce training datasets that are suitably large in many situations. This also holds true for relevant data and images that are currently available. Of course, there are a variety of solutions to this issue, including the usage of pre-trained networks and data augmentation techniques. Create fresh data based on GANs and conventional data augmentation approaches, then utilise the generated datasets as the preprocessing block's input data. [1]

2.2 Summary

Our research started by exploring conventional machine learning algorithms to take on the categorization challenge for X-ray images. Different image enhancing techniques were explored to improve the quality and details of the input images, and then some well-known feature extraction algorithms were used. And finally, a variety of machine learning methods were tried. Despite our best efforts, the earliest models produced only modest accuracy, with an average of about 50%.

Arriving at deep learning models, several models were put to work to increase classification accuracy. First, in order to increase the model's generalization, we used data augmentation techniques with Deep Convolutional Generative Adversarial Networks (DCGAN) to provide more training samples. The discriminative capability of the input

images was then further improved by applying Gabor filters, which are renowned for their efficiency in capturing texture and edge information. Convolutional neural network (CNN) architectures were studied in detail last. EfficientNetB3 outperformed the other models, demonstrating greater performance with an accuracy of 86.94%.

2.3 Dataset

In this research, UNIFESP X-ray dataset was taken as the source of images. It contains 1738 annotated DICOM images, each associated with a unique SOP Instance ID. It comprises of 22 classes, as well as a specialised "others class" that includes body parts that are not covered by the specific classes. Chest X-rays are highest in number in the sample, accounting for almost 700 images, while less often classified body areas such as the clavicle or skull have a lesser number of images. The dataset's prominent feature is its high level of class imbalance, which presents significant difficulty in efficient classification.

Several reasons influenced the choice of the UNIFESP X-ray dataset. First of all, the dataset includes an ample number of body part classes, allowing for in-depth analysis and evaluation of the X-ray image classification problem. This research intends to construct a classification system capable of handling real-world scenarios and a wide range of diagnostic cases by focusing on a dataset that includes frequently encountered and less frequently classified body parts. Furthermore, the dataset's extremely unequal distribution of images across classes provides an opportunity to examine and address class imbalance concerns. The research seeks to give insights and solutions applicable to real-world situations where class imbalances are frequent by working with an imbalanced dataset.

Chapter 3

PROPOSED SYSTEM

3.1 Modules

3.1.1 Preprocessing

Preprocessing module often being the first step in the pipeline plays a crucial role in transforming the image data into a format suitable for each specific model. It involves a variety of techniques to make images noise-free and to enhance essential image qualities such as contrast, brightness, texture etc.

3.1.2 Data augmentation

Data augmentation techniques are applied to increase the size and diversity of the training dataset and address class imbalance concerns. Common augmentation techniques include random rotations, translations, flips and resizes. It could also be done by using GAN networks to generate new images from the dataset distribution. this step helps the model generalize better by exposing it to a wider range of variations.

3.1.3 Feature extraction and description

This step involves in detection and extraction of relevant, distinctive and meaningful features, called keypoints, from the raw images and finding a suitable format that can robustly and descriptively represent the select keypoints to aid the models in classification.

3.1.4 Classification

Appropriate classification models, both machine learning and deep learning based, are chosen and trained on the data to help the models learn to identify the labels of the images with a satisfying accuracy. Multiple evaluation metrics are used to calculate the total effectiveness, accuracy and efficiency of the model.

3.2 Algorithms and Methodologies

3.2.1 Image Enhancement

- **Gabor**

A Gabor filter is a combination of a Gaussian function and a sinusoidal plane wave. The Gabor filter's structure allows it to capture both the spatial and frequency characteristics of an image. A gabor filter is a 2-dimensional kernel which, when convolved on an input image, the filtered image is obtained. Although, there are several parameters that affect the kernel's behaviour. They are as follows:

- **Wavelength(λ):**

  The wavelength deals with the width of the Gabor function, i.e., the more the wavelength, the more the width of the strip.

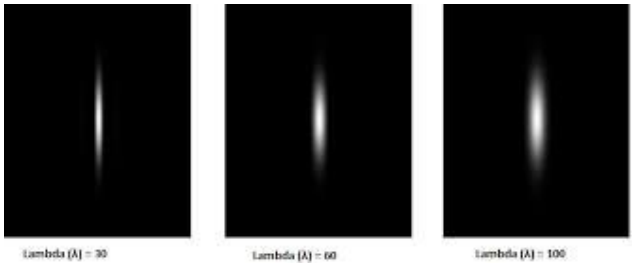


Figure 3.1: Gabor Filter Wavelength Parameter

- **Orientation(θ):**

  The orientation manages the direction of the Gabor function., i.e., when given 0°, the orientation is vertical.

Figure 3.2: Gabor Filter Orientation Parameter

– **Aspect ratio(γ):**

The aspect ratio deals with the height of the gabor function, i.e., more the aspect ration, lesser is the height.

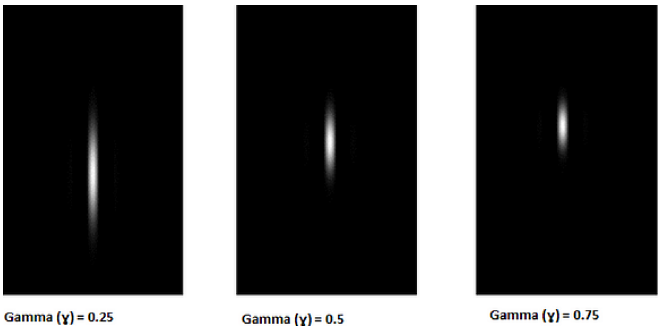


Figure 3.3: Gabor Filter Aspect Ratio Parameter

– **Bandwidth(σ):**

The bandwidth takes over the size of the gabor envelope, i.e., more the bandwidth, the more stripes the function has.

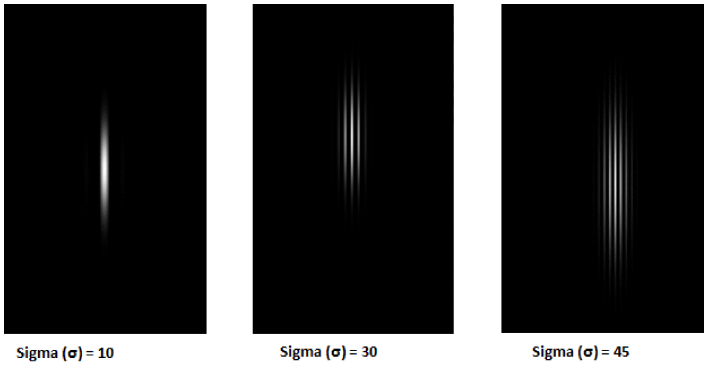


Figure 3.4: Gabor Filter Bandwidth Parameter

- **Offset(ψ):**

  It is the phase offset of the sinusoid.

- **Contrast Limited Adaptive Histogram Equalisation(CLAHE)**

Contrast Limited Adaptive Histogram Equalization (CLAHE) is an image enhancement technique commonly used to improve the visibility and quality of images by enhancing local contrast while avoiding over-amplification of noise which was a notable drawback of Adaptive histogram Equalization (AHE) upon which CLAHE is modelled. CLAHE involves two main steps. Firstly, the image is divided into multiple regions of nearly equal sizes, without overlapping. Secondly, the histogram is computed for each region. A clip limit is then determined to constrain the histograms. Subsequently, the histograms are adjusted to ensure that their heights do not exceed the clip limit.

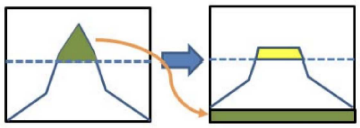


Figure 3.5: CLAHE

- **Total Variation (TV) Homomorphic filters**

TV homomorphic filters aims at enhancing images by simultaneously preserving edges and reducing noise. The image represented as product of reflectance and illumination is transformed logarithmically to linearly separate the two components.

$$m(x, y) = i(x, y) \cdot r(x, y) \tag{3.1}$$

Where;

- m = image,
- i = illumination,

– r = reflectance

Applying "log" on both sides,

$$ln(m(x,y)) = ln(i(x,y)) + ln(r(x,y)) \tag{3.2}$$

The multiplicative noise in image is eliminated by reducing uneven illumination by using total variation denoising to filter the low frequency components. The filtered low frequency component is combined with he original high frequency element to obtain denoised image with edges preserved.

3.2.2 Feature description

- **Scale Invariant Feature Transform(SIFT)**

SIFT helps represent keypoints invariant to scaling, rotation, and viewpoint, partially invariant to change in illumination. It involves four main steps.

1. **Scale-space Extrema Detection:**

   SIFT builds a scale-space representation of an image by creating a series of blurred and scaled versions of the original image using Gaussian blur. It identifies keypoints as local extrema in this scale-space pyramid, which correspond to stable features across different scales. The scale space of an image is defined as a function, L(x, y, σ), that is produced from the convolution of a variable-scale Gaussian, G(x, y, σ), with an input image, I(x, y):

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \qquad \text{(Gaussian Convolution)}$$

Where G is;

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \qquad \text{(Gaussian Operator)}$$

2. **Keypoint Localization:**

   SIFT refines the detected keypoints by eliminating unstable points using a difference-of-Gaussian (DoG) function. It removes low-contrast keypoints and those located on edges or corners.

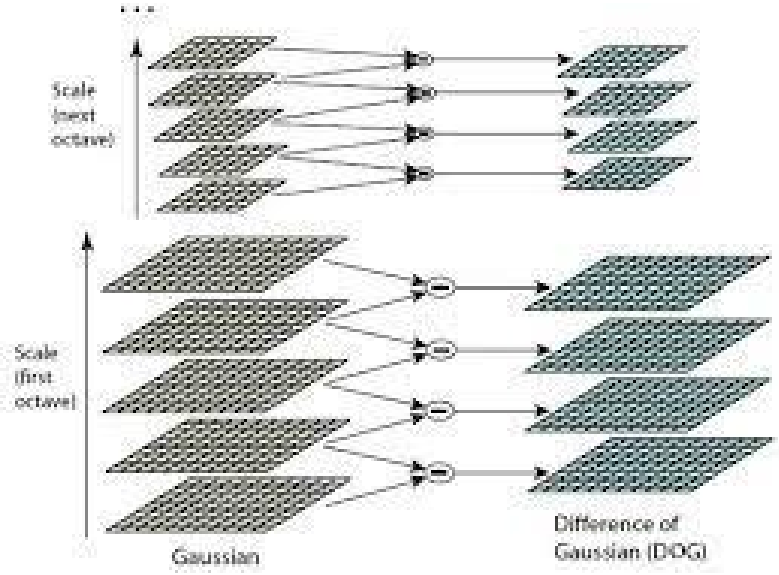


Figure 3.6: Difference of Gaussian

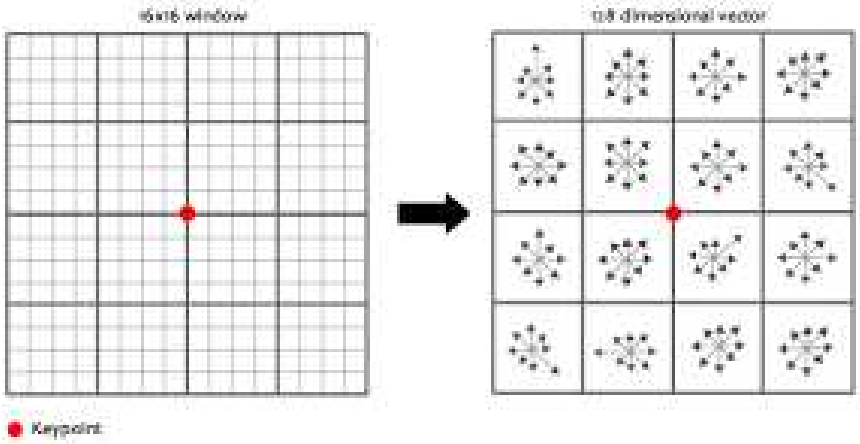
3. **Orientation Assignment:**

   SIFT assigns a consistent orientation to each keypoint to achieve in-variance to image rotation. It calculates gradient magnitudes and orientations in the local neighborhood of each keypoint and assigns a dominant orientation based on the orientations' histogram.

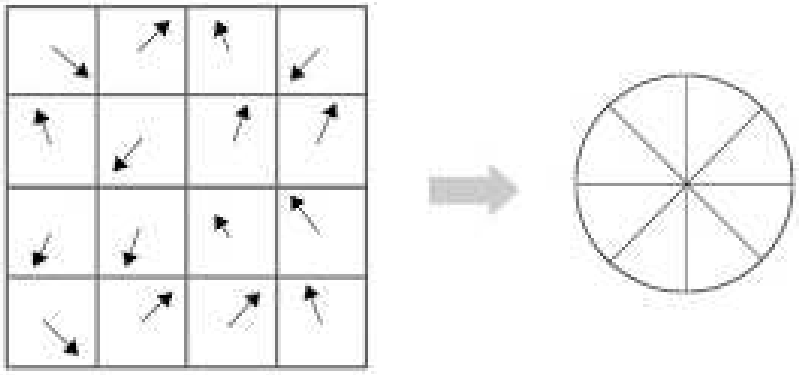
4. **Keypoint Description:**

   Computing a descriptor for a local image region about each keypoint that highly unique and in-variate to any changes in viewpoint or illumination.

A 16x16 block around the selected keypoint is selected and divide it into 16 sub-blocks of 4x4 size.



For each sub-block, 8-bin orientation is computed



4x4x8 directions give 128 bin values. It is represented as a feature vector to form keypoint descriptor.

- **Speed-Up Robust Features(SURF)**

A quick and reliable approach for local, similarity-invariant representation and comparison of pictures is the SURF method (Speeded Up Robust Features). The

SURF approach's key appeal is its ability to compute operators quickly using box filters, enabling real-time applications like tracking and object detection. The SURF has two steps:

1. **Feature extraction**
   - Integral Images: A quick method to calculate the sum of pixel values in an image. This allows fast computation of box type convolution filter.

$$I_{\Sigma}(x) = \sum_{i=0}^{i<=x} \sum_{j=0}^{j<=y} I(i,j) \qquad \text{(Integral Images)}$$

   - Hessian Matrix based interest points: Hessian Matrix performs better in terms of computation and accuracy. Due to this, instead of any other measure for location and scale, SURF uses Hessian matrix's determinant

$$H(f(x,y)) = \begin{bmatrix} \frac{d^2 f}{dx^2} & \frac{d^2 f}{dxdy} \\ \frac{d^2 f}{dxdy} & \frac{d^2 f}{dy^2} \end{bmatrix} \qquad \text{(Hessian matrix of f(x,y))}$$

     Now, given a point X=(x,y), Hessian Matrix H(x,σ) in x at scale σ is given by;

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \qquad \text{(Laplacian of Hessian)}$$

     Gaussian convolutions are often cropped and discretized. This leads to loss in repeatability under image-rotations around odd multiples of π/4.

     To calculate Hessian Matrix, convolution with second-order derivative of Gaussian filter is applied. SURF, then, pushes the approximations

even further with the help of box filters.

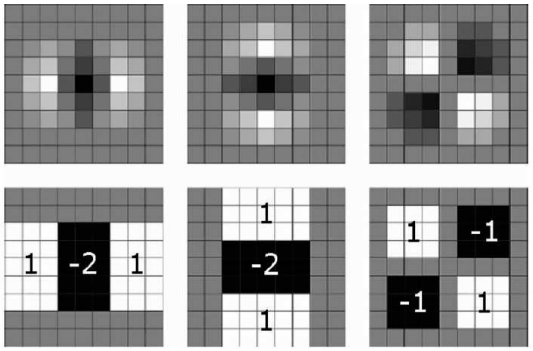


Figure 3.7: Box filters

– Scale-Space Representation: To reach a higher level of the pyramid, the images are sub-sampled after being repeatedly smoothed with a Gaussian. The scale-space is built by up-scaling the filter size and not iterative reduction of the image size, and since multiple filters of varying sizes can be applied on the image concurrently the process is sped up significantly. For each new octave, the filter size is increased doubly to build the scale-space.

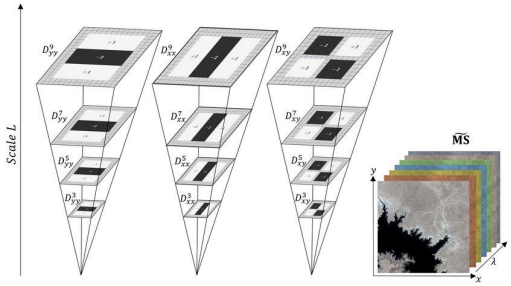


Figure 3.8: Scale Space Representation

2. **Feature Description**

   There are two processes in the construction of the SURF descriptor. Determining a reproducible orientation based on data from a circular neighbourhood of the keypoint. Then a square region is constructed and oriented to obtain the rotation invariant descriptor.

– Orientation assignment: With s being the scale at which the keypoint was identified,the Haar-wavelet responses in the x and y directions in a neighbourhood of 6 s radius around the keypoint is calculated. Additionally, the wavelet responses are calculated at the given scale s, and the sampling step, which is scale dependent, is determined to be s. As a result, wavelets have a large size at high scales. thus, integral images are once more used for quick filtering.Then, the vertical and horizontal wavelet responses in a scanning region is summed, modify the orientation of the scanning region (add $\pi/3$), and recalculate until the orientation with the highest sum value is obtained, which is the principal orientation of the feature descriptor.

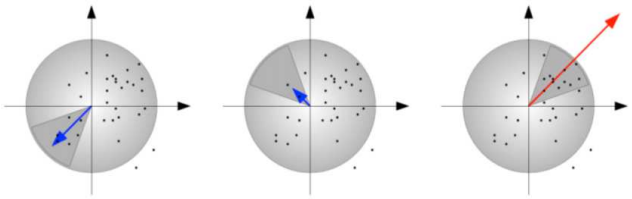


Figure 3.9: Orientation Assignment

– Descriptor commponents: The first stage contains building a square region that is centred on the keypoint and orientated in the direction indicated above. This window is 20s in size. The region is then routinely divided into smaller, 4x4 square sub-regions. At 5x5 evenly spaced sample points, a few basic features for each sub-region are computed. two-size filter. The answers dx and dy are first weighted with a Gaussian (σ = 3.3s) centred at the keypoint to strengthen their resilience against geometric deformations and localization mistakes.

  Each sub-region has a four-dimensional descriptor vector v for its underlying intensity structure V = ($\sum$dx, $\sum$ dy, $\sum$|dx|, $\sum$|dy|). This results in a descriptor vector for all 4×4 sub-regions of length 64.

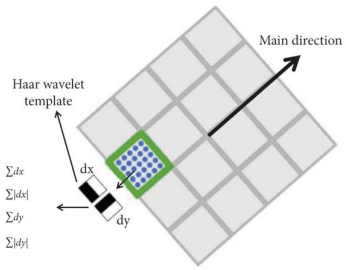
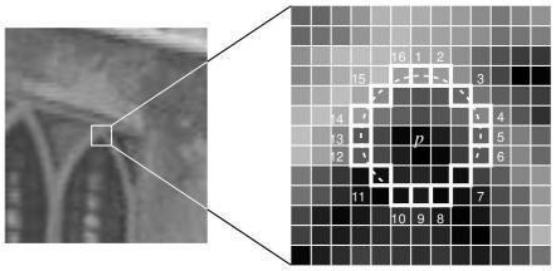


Figure 3.10: Haar wavelet responses



- **Oriented FAST and Rotated BRIEF(ORB)**

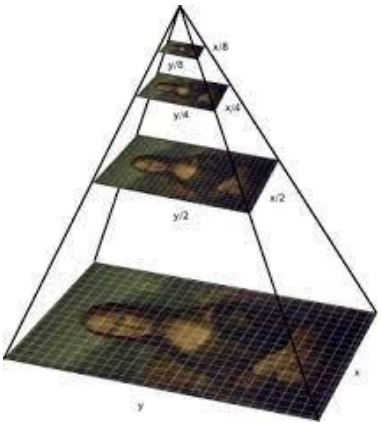
ORB uses a combination of Features from Accelerated Segment Test(FAST) to detect keypoints and Binary Robust Independent Elementary Feature(BRIEF) for orientation.

1. **Features from Accelerated Segment Test(FAST):**

   Considering a pixel from a given image, FAST compares the brightness of the surrounding 16 pixels of the chosen pixel. If more than 8 pixels are brighter than the selected pixel, then the selected pixel is considered a keypoint.

   Since FAST does not have an orientation computation, ORB uses a multi-scale image pyramid of the image, that contains different resolutions of the same image down-sampled at each level. In this way, ORB detects keypoints in each level, or scale. This way, Orb is partially a scale invariant.

   Once the keypoints are located, ORB now assigns an orientation to each keypoint. To detect the intensity changes, ORB uses intensity centroid.



$$m_{pq} = \Sigma_{x,y} x^p y^q I(x,y) \qquad \text{(Moment's Definition of a patch)}$$

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \qquad \text{(Centre of Mass of a patch)}$$

$$\theta = atan2(m_{01}, m_{10}) \qquad \text{(Orientation of the patch)}$$

Once the orientation of the patch is computed, rotation can be performed to compute the descriptor. In this process, rotation invariance is achieved.

2. **Binary Robust Independent Elementary Feature(BRIEF):**

   BRIEF takes the keypoints from the previously processed FAST algorithm and converts it into a binary feature vector. A feature vector can be 128-512 bits string. BRIEF selects a random pair of pixels in a neighborhood around the keypoint. The first pixel is elected from a Gaussian Distribution and another pixel is drawn from a Gaussian Distribution around the first selected pixel.

   If the first pixel is brighter(low intensity) than the second pixel, a value of "1" is assigned to the corresponding bit and "0" is not. This is repeated 128 times to obtain a 128-bit vector.

$$\tau(p;x,y) = \begin{cases} 1 : p(x) < p(y) \\ 0 : p(x) >= p(y) \end{cases} \tag{3.3}$$

Where; p(x) = intensity at pixel "x"

3.2.3 Feature respresentation

Bag Of Visual Words(BOVW)

The Bag of Visual Words (BoVW), inspired by the "Bag of Words" model used in natural language processing. BoVW represents an image as a histogram of visual words or visual features extracted from the image. It involves the following steps

1. **Feature Extraction:**

   The first step in BoVW is to extract local features from the image. Commonly used features include Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), or Oriented FAST and Rotated BRIEF (ORB). These features capture distinct patterns or keypoints in the image.

2. **Feature Encoding:**

   The extracted features are then encoded into a fixed-length vector representation. One commonly used encoding technique is vector quantization, which involves clustering the features into a predefined number of visual words or clusters using methods like k-means clustering. Each feature is assigned to the nearest visual word, and its frequency is counted.

3. **Codebook Generation:**

   The set of visual words obtained from the clustering step forms a codebook or vocabulary. This codebook represents the collection of visual patterns or concepts present in the images.

4. **Histogram Representation:**

   To represent an image, a histogram is constructed based on the frequencies of visual words. Each image is divided into local regions, and the features within each region are quantized to the nearest visual words. The histogram counts the occurrences of each visual word in the image.

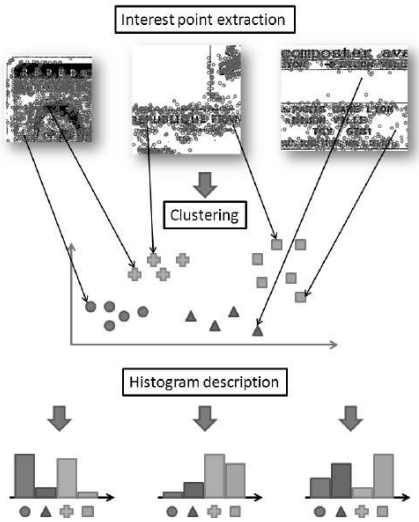


Figure 3.11: BOVW

3.2.4 Image Augmentation

GAN is a method for creating fresh data. The GANs are a collection of machine learning systems that Ian Goodfellow initially created in 2014. This technique allows us to create fake, manufactured data (sounds and images) that are comparable to the input data. In this method, two neural network models—a generator and a discriminator—compete against one another in a game as it is developed in game theory. The role of the generator model is to create content (pictures), while the opponent model's (the discriminator's) role is to distinguish between authentic and fraudulent images. The competing artificial intelligence algorithm can first differentiate the false photographs from the actual ones with ease. However, as time goes on, the generator artificial intelligence model's accuracy and performance improve to the point where it is highly challenging for the competing discriminator model to distinguish between the real and fake material.The architectural details and the layers of each of the two models are as follows:

The generator network consists of 64 filters, 4 ReLU activation functions, 4 batch normalisation layers, and the Tanh activation function. It also has 5 transposed convolutional layers with a window size of 4x4 pixels.

The discriminator network additionally includes 64 filters, 4 Leaky ReLU activation

functions, 3 batch normalisation layers, and the Sigmoid activation function. It also has 5 transposed convolutional layers with a window size of 4x4 pixels.
In this project, this method is used exclusively to increase the number of X-Ray images.

3.2.5 Classification

Machine Learning based classifiers

- **Linear Support Vector Machine:**

  Firstly, extracting features from an image is to be performed in order to classify it using SVM. This algorithm locates the hyperplane in the feature space that divides the various classes. Finding the hyperplane that maximises the distance between the nearest points of the various classes is the main motto.

- **Decision Tree Classifier:**

  A decision tree is a tree structure that resembles a flowchart, where each internal node represents a feature, branches represent rules, and leaf nodes provide the output. It is a supervised machine-learning approach that is applied for classification purposes. Additionally, Random Forest, one of the most potent machine learning algorithms, uses it to train on various subsets of training data

- **Random Forest Classifier:**

  Random Forest is a classifier that uses multiple decision trees on different subsets of the input dataset, descriptors in this case, and averages the results to increase the predicted accuracy. Instead than depending on a single decision tree, the random forest uses forecasts from each tree and predicts the result based on the votes of the majority of predictions.

- **XGBoost Classifier:**

  In boosting, the trees are constructed one after the other with the goal of minimising the errors of the previous tree. Each tree updates the residual errors as a result of learning from its ancestors. The tree that develops next in the sequence will

therefore get knowledge from an updated set of residuals.

The base learners in boosting are weak learners with strong bias and marginally greater predictive power than chance. Each of these weak learners provides some crucial information for prediction, allowing the boosting strategy to successfully combine these weak learners to create a strong learner. Both the bias and the variance are reduced by the final strong learner.

- **Stochastic Gradient Descent(SGD) Classifier:**

  This classifier offers a straightforward stochastic gradient descent learning procedure that supports various classification loss functions and penalties. The decision boundary of an SGDClassifier that was trained using the hinge loss and is comparable to a linear SVM.

- **Gaussian Naive Bayes Classifier:**

  Naive Bayes, a probabilistic classification model for machine learning that is simple but effective, is influenced by the Bayes Theorem.

$$P(y|X) = \frac{P(X|y).P(X)}{P(y)} \tag{3.4}$$

  Where;

  - X = x_1, x_2,.., x_n (list of independent predictors)
  - y = class label
  - P(y|X) is the probability of label y given the predictors X

- **K Neighbors Classifier:**

  The K-NN algorithm places the new case in the category that is most similar to the available categories based on the assumption that the new instance and the data are comparable to the examples that are already accessible.

  The K-NN algorithm saves all the information that is available and categorises fresh data based on similarity computed using a distance metrics such as Euclidean distance, Manhattan distance etc. The KNN algorithm simply saves the information about the cluster centroids during the training phase, and when it receives new data, it categorises it into a category that is minimum in terms of distance to the centroids.

- **Support Vector Machine (SVM) Classifier:**

  The main objective of an SVM classifier is to find an optimal hyperplane that best separates the data points of different classes. A hyperplane is a decision boundary that divides the input space into regions corresponding to each class.

  The SVM training process involves finding the optimal hyperplane by solving an optimization problem. The objective is to minimize the classification error while maximizing the margin. The training algorithm identifies a subset of the data points called support vectors, which are the closest points to the hyperplane.

  To classify new data points, the SVM algorithm calculates their position relative to the learned hyperplane and assigns the class according to the region the data falls in.

Deep Learning based classifiers

- **VGG16**

  VGG16 is a convolutional neural network (CNN) architecture that has achieved significant success in various computer vision tasks, including image classifica-

tion and object recognition.The VGG16 architecture consists of 16 layers, primarily composed of convolutional layers and fully connected layers. The key idea behind VGG16 is to use a series of smaller filters (3x3) with a stride of 1, rather than using a larger receptive field, to improve the representational power of the network. It follows a "convolutional stacking" approach, where multiple convolutional layers with smaller filters are stacked together, followed by max-pooling layers for downsampling.

The architecture consists of 13 convolutional layers, with each followed by a Rectified Linear Unit (ReLU) activation function, and five max-pooling layers. The final part of the network consists of three fully connected layers, which are responsible for the high-level reasoning and decision-making. The last fully connected layer has the same number of units as the number of classes in the classification task, followed by a softmax activation function to obtain the predicted class probabilities.

The precise structure of the VGG-16 network is,

- The first and second convolutional layers are comprised of 64 feature kernel filters and size of the filter is 3×3. As input image (RGB image with depth 3) passed into first and second convolutional layer, dimensions changes to 224x224x64. Then the resulting output is passed to max pooling layer with a stride of 2.
- The third and fourth convolutional layers are of 124 feature kernel filters and size of filter is 3×3. These two layers are followed by a max pooling layer with stride 2 and the resulting output will be reduced to 56x56x128.
- The fifth, sixth and seventh layers are convolutional layers with kernel size 3×3. All three use 256 feature maps. These layers are followed by a max pooling layer with stride 2.
- Eighth to thirteen are two sets of convolutional layers with kernel size 3×3. All these sets of convolutional layers have 512 kernel filters. These layers

are followed by max pooling layer with stride of 1.

– Fourteen and fifteen layers are fully connected hidden layers of 4096 units followed by a softmax output layer (Sixteenth layer) of 1000 units.

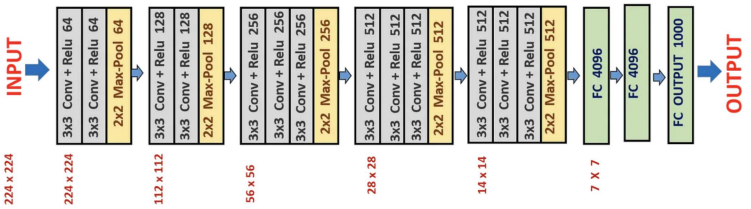


Figure 3.12: VGG16 Architecture

- **DenseNet121**

  DenseNet-121 is a convolutional neural network (CNN) architecture that was introduced by the researchers at Facebook AI Research (FAIR). It is part of the DenseNet family of models, which are known for their dense connections between layers, enabling rich feature reuse and improved gradient flow throughout the network.

  DenseNet-121 gets its name from its total number of layers, which is 121. The architecture is designed to address the vanishing gradient problem and encourage information flow by connecting each layer to every other layer in a feed-forward fashion. This is achieved through a specific type of skip connection called "skip connections with concatenation."

$$x_l = H_l([x_0, x_1, ...x_{l-1}]) \tag{3.5}$$

  The key idea behind DenseNet-121 is the concept of "dense blocks." Each dense block consists of multiple layers that are densely connected to each other. Within a dense block, the feature maps from preceding layers are concatenated and used as inputs for subsequent layers. This dense connectivity pattern helps in leveraging feature reusability, allowing each layer to have direct access to the gradient

information of all preceding layers. It promotes feature propagation and encourages the network to learn more discriminative features.

DenseNet-121 also incorporates transition layers between dense blocks to control the growth of feature maps and reduce the computational complexity. Transition layers consist of 1x1 convolutional layers followed by average pooling, which reduces the spatial dimensions of the feature maps while preserving the essential information.

The final part of DenseNet-121 consists of a global average pooling layer followed by a fully connected layer with a softmax activation function, producing the predicted class probabilities.

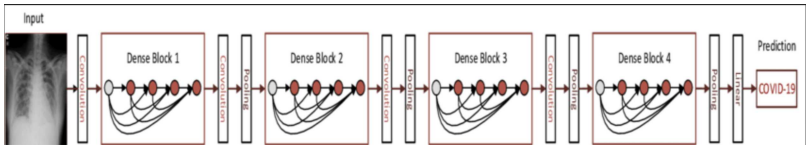


Figure 3.13: DenseNet Architecture

- **ResNet50**

  The architecture is designed to address the problem of vanishing gradients that can occur in deep neural networks by introducing residual connections.

  The key idea behind ResNet-50 is the concept of residual blocks. A residual block consists of a series of convolutional layers followed by a shortcut connection, which directly adds the original input to the output of the block. This allows the network to learn residual functions that approximate the difference between the input and the desired output. The residual connections help alleviate the vanishing gradient problem and enable the network to train deeper architectures more effectively.

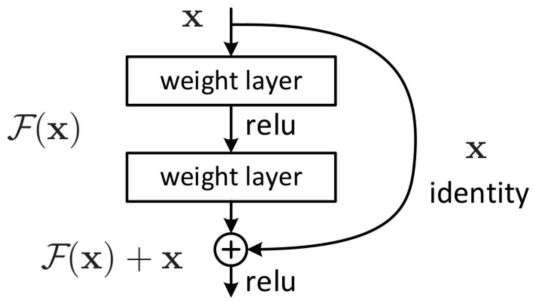


Figure 3.14: Caption

ResNet-50 follows a "bottleneck" architecture, where each residual block consists of three convolutional layers: a 1x1 convolution, a 3x3 convolution, and another 1x1 convolution. The 1x1 convolutions are used to reduce and restore the dimensions of the feature maps, effectively reducing computational complexity.

The architecture also includes downsampling layers with stride 2 to reduce the spatial dimensions of the feature maps and increase the receptive field as the network goes deeper. This downsampling is performed using convolutional layers with larger strides and smaller filter sizes.

The final part of ResNet-50 consists of a global average pooling layer, which aggregates the spatial information into a fixed-length feature vector, and a fully connected layer with a softmax activation function, producing the predicted class probabilities.

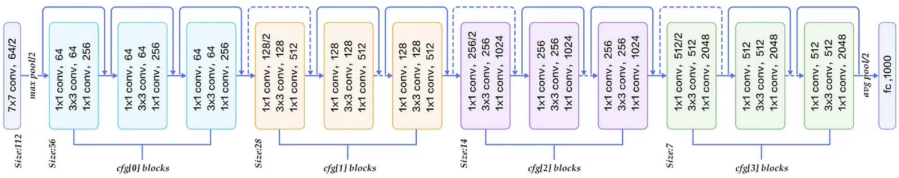


Figure 3.15: ResNet Architecture

- **InceptionV3**

The main innovation of InceptionNet is the use of inception modules, which are multi-branch convolutional blocks. These inception modules allow the network

to capture features at multiple scales and abstraction levels within the same layer. This is achieved by using parallel convolutional operations with different filter sizes (1x1, 3x3, and 5x5) and pooling operations. The outputs of these operations are then concatenated and fed into the next layer.

By using these inception modules, InceptionNet significantly reduces the number of parameters compared to previous architectures. This reduction in parameters helps alleviate overfitting and improves computational efficiency. Additionally, it allows the network to have a larger depth, enabling it to capture more complex patterns and hierarchies of features.

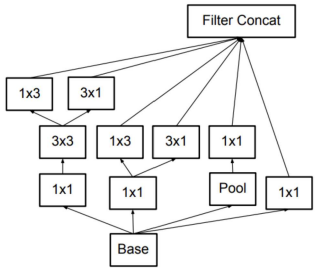


Figure 3.16: Inception Model

InceptionNet also incorporates other techniques to further enhance its performance. One such technique is the use of auxiliary classifiers attached to intermediate layers. These auxiliary classifiers provide additional gradients during training, helping to alleviate the vanishing gradient problem and improve the flow of gradients through the network.

Another important aspect of InceptionNet is its utilization of global average pooling instead of fully connected layers at the end of the network. Global average pooling reduces the number of parameters and eliminates the need for a fixed-sized input, making the network more flexible to handle inputs of different sizes.

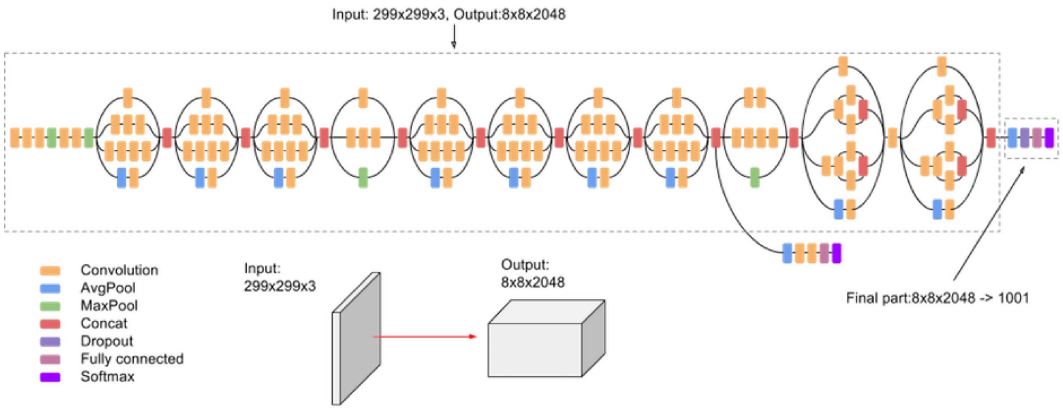


Figure 3.17: InceptionNetV3 Architecture

- **EfficientNetB3**

  The key idea behind EfficientNet is to balance the network's depth, width, and resolution in an optimal manner to achieve better performance without significantly increasing computational cost. The architecture achieves this balance by using a compound scaling method that uniformly scales these dimensions.

  EfficientNet introduces a compound scaling formula that uniformly scales the depth, width, and resolution of the network. The depth refers to the number of layers in the network, the width corresponds to the number of channels in the intermediate layers, and the resolution refers to the input image size. By scaling these dimensions together, EfficientNet ensures that the network remains well-balanced and avoids any individual dimension becoming a bottleneck or limiting factor.

  The compound scaling formula uses a scaling coefficient phi (ϕ) to control the scaling of the network. The value of phi can be chosen to control the trade-off between accuracy and computational efficiency. A larger value of phi results in a larger and more accurate network, while a smaller value reduces the size and computational cost but may sacrifice some accuracy.

  EfficientNet also incorporates a technique called "Mobile Inverted Residual Blocks" (MBConv) that are similar to the inverted residual blocks used in MobileNetV2.

These blocks use depth-wise separable convolutions to reduce computation while maintaining representational power. They also employ a bottleneck structure that reduces the number of parameters and computation in the intermediate layers.

One of the key contributions of EfficientNet is the use of a Neural Architecture Search (NAS) approach to automatically find the optimal combination of network depth, width, and resolution scaling. The authors used a resource-constrained search method to find the best architecture within a given computational budget. This allows EfficientNet to achieve high accuracy while being efficient in terms of memory and computation.

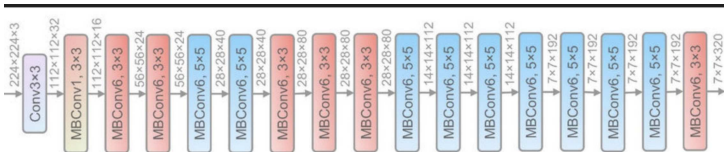


Figure 3.18: EfficientNet Architecture

3.2.6 Architecture diagram

Machine Learning Approach:

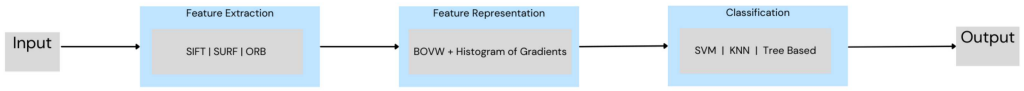


Figure 3.19: Block Diagram Of ML Model without Image Enhancement

Figure 3.20: Block Diagram Of ML Model with Image Enhancement

Deep learning Approach:

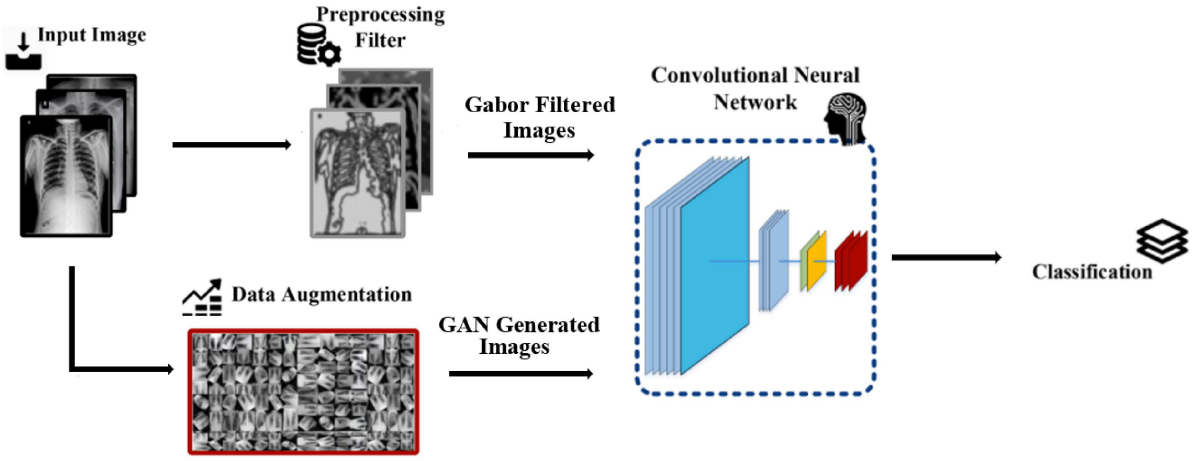


Figure 3.21: Architecture Diagram Of Deep Learning Model

Chapter 4

IMPLEMENTATION

4.1 Pre-processing

- Visualisation of the dataset: The possessed dataset contains multiple images under each label. There are several labels which include combination of multiple body parts. Such labels are treated as those labels had lesser number of images. Due to inadequate images, the further steps would be affected in terms of accuracy. Although, certain combination of labels contain ample images, such labels were retained and the process was continued .

- Dataset Understanding: The provided images in the dataset are in DICOM format. These images are stored in a table along with its location and a unique SOPInstanceUID merged into the image location. An SOPInstanceUID is a unique identifier for each DICOM file. This table has multiple columns, namely, "Filename", "Bits Allocated","SOPInstance". From the data taken, the CSV file contains a target label in the form of an integer that maps to different body parts. These integers are then mapped as labels and allotted to the corresponding image. Now the table has two additional columns, namely, "Target" and "Label".

- Image Format: The raw images present were in DICOM format. For a better computational time, the format of these images was changed to PNG image format. Later, the image path of the transformed images was appended to the SOPInstanceUID and saved as "ImagePath" to the previously created table.

The implementation firstly takes place with respect to Machine Learning Approach and then continued with respect to Deep Learning Methodologies.

4.2 Machine Learning Approach

4.2.1 Methodology

In this approach, implementation of multiple different combinations of workflows with various methodologies are mentioned as follows:

- **Data Split:**
  We use "train_test_split" technique to split our data into "x_train", "x_test", "y_train", "y_test". A parameter "stratify", that makes sure the proportion of values before splitting of the data is maintained even after the data split, is used so there is no bias in the data after splitting. This helps conducting a proper classification of images. Here, as per the data, "x" contains the complete data that is split and "y" contains only the labels of all the images in the dataset.

```
from sklearn.model_selection import train_test_split
y = train.Label
X = train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=50, stratify = X['Label'])
```

Figure 4.1: Split using Stratify

- **Label Encoding:**
  A Label Encoder is used so that a machine learning algorithm can read numerical values better than strings. Since the labels are given as strings, they are converted to numerical ones, starting from "0", and are fed to classify.
  We use fit(), to fit the label encoder, and transform(), to give the encoded labels as output, methods.

- **Scale Invariate Fourier Transform(SIFT):**
  By importing "cv2" library to the Jupyter environment, we can access the SIFT methods to extract the image features. The library "cv2" has a method "SIFT_create()" that takes a parameter as the number of SIFT points to be generated.

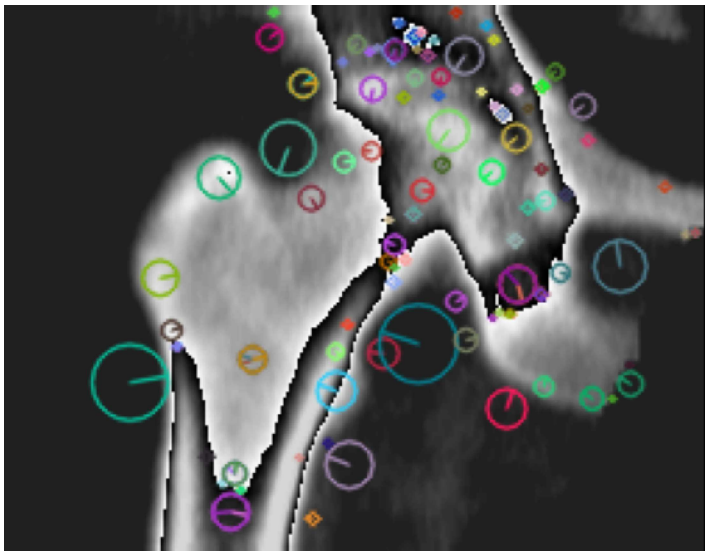


Figure 4.2: Image with 100 SIFT Keypoints

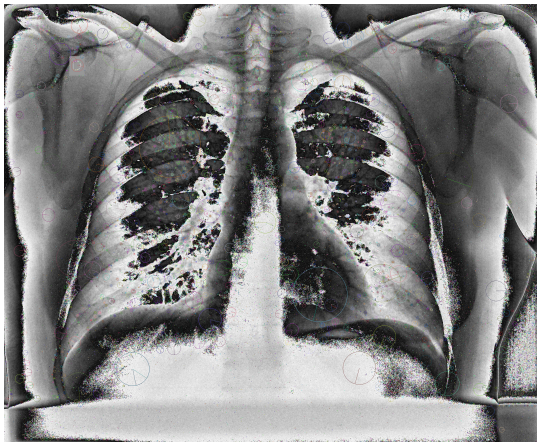


Figure 4.3: SIFT on Chest

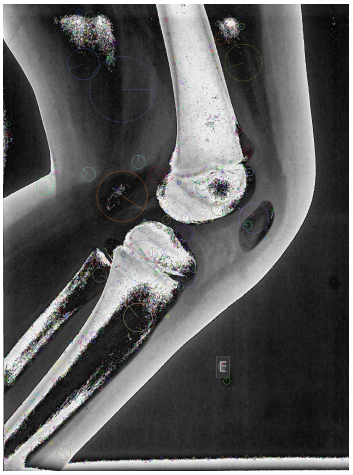


Figure 4.4: SIFT on Knee

The methods that are used to extract the required descriptors from an image are "cv2.SIFT_create(n).detect(image)" and "cv2.SIFT_create(n).compute(image)". The former method, detect(image), detects all the possible or the required keypoints. The latter method, compute(image), take the descriptors from the "detect(image)" method and computes them to store it for further purposes. The keypoints, or descriptors, generated are then stored into a list.

After creating all the SIFT keypoints, 150 best keypoints, the final descriptors list to a size of (227776, 128) has been arrived at.

- **K-Means Clustering:**
  This algorithm is used as an Unsupervised Machine Learning algorithm. Since our descriptors are unlabelled and unclassified data, such an algorithm is implemented to cluster the descriptors vector. In order to make the descriptor points within each group more comparable to one another and distinct from the

descriptor points within the other groups, it divides the set of descriptor points into a number of groups.

After performing the K-Means clustering on the derived descriptors, these clustered descriptor points are fed to the classifiers.

- **Machine Learning Models (with SIFT descriptors):**
  Once all the image features is ready, several machine learning models use those features and classified the images. The following are the Machine Learning Models:

  – ***Linear Support Vector Classification(SVC):***

  *With the help of the imported library, "sklearn", the "LinearSVC()" function was made available from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of **29.59%**.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.14	0.12	0.13	8
4	0.57	0.58	0.58	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.45	0.42	0.43	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.17	0.33	0.22	6
accuracy			0.30	169
macro avg	0.06	0.06	0.06	169
weighted avg	0.29	0.30	0.29	169

Figure 4.5: Linear Support Vector Classification Accuracy

– ***Decision Tree Classifier:***

With the help of the imported library, "sklearn", the "DecisionTreeClassifier()" function was made available from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***27.21%****.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.12	0.12	0.12	8
4	0.56	0.49	0.52	72
5	0.00	0.00	0.00	2
6	0.33	0.14	0.20	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.13	0.20	0.16	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.10	0.20	0.13	5
16	0.56	0.42	0.48	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.17	0.17	0.17	6
accuracy			0.27	169
macro avg	0.09	0.08	0.08	169
weighted avg	0.31	0.27	0.29	169

Figure 4.6: Decision Tree Classifier Accuracy

– ***Random Forest Classifier:***

With the same imported library, "sklearn", the "RandomForestClassifier()" function was used with a parameter as "n_estimators = 100", which creates a 100 decision trees during classification. The descriptors are given to the Random Forest Classifier and resulted with an accuracy of ***46.74%****. Comparing Random Forest Classifier to the Decision Tree*

Classifier, the Random Forest Classifier provided a better result as it uses multiple decision trees.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.45	1.00	0.62	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.70	0.58	0.64	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.47	169
macro avg	0.05	0.07	0.05	169
weighted avg	0.24	0.47	0.31	169

Figure 4.7: Random Forest Classifier Accuracy

– ***XGBoost Classifier:***

*"xgboost" library has been imported that provided us with the used the "XGBClassifier()" function from the imported library. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of **48.52%**. This classifier has given a better result than Decision Tree and Random Forest Classifiers.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.67	0.25	0.36	8
4	0.49	0.97	0.65	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	1.00	0.33	0.50	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.78	0.58	0.67	12
17	1.00	0.14	0.25	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.25	0.17	0.20	6
accuracy			0.49	169
macro avg	0.18	0.11	0.11	169
weighted avg	0.36	0.49	0.37	169

Figure 4.8: XGBoost Classifier Accuracy

– ***Stochastic Gradient Descent(SGD) Classifier:***

With the help of the imported library, "sklearn", the "SGDClassifier()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***36.09%****.*

	precision	recall	f1-score	support
0	0.29	0.25	0.27	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.09	0.12	0.11	8
4	0.56	0.67	0.61	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.47	0.67	0.55	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.33	0.33	0.33	6
accuracy			0.36	169
macro avg	0.08	0.09	0.08	169
weighted avg	0.30	0.36	0.33	169

Figure 4.9: Stochastic Gradient Descent Classifier Accuracy

– ***Gaussian Naive Bayes Classifier:***

The "GaussianNB()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***7.10%****.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.00	0.00	0.00	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.50	0.33	0.40	3
10	0.00	0.00	0.00	4
11	0.11	0.50	0.18	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.04	0.60	0.08	5
16	0.78	0.58	0.67	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.07	169
macro avg	0.06	0.09	0.06	169
weighted avg	0.07	0.07	0.06	169

Figure 4.10: Gaussian Naive Bayes Classifier Accuracy

– ***K Neighbors Classifier:***

We used the "KNeighborsClassifier()" function from the imported library with parameters "n-neighbors=5", implying the presence of 5 neighbors, and "metric = "minkowski", which describes the distance computation with the power "p = 2". The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***42.60%****.*

	precision	recall	f1-score	support
0	0.10	0.12	0.11	8
1	0.00	0.00	0.00	1
2	0.17	0.25	0.20	4
3	0.17	0.25	0.20	8
4	0.48	0.85	0.62	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.78	0.58	0.67	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.43	169
macro avg	0.07	0.09	0.08	169
weighted avg	0.28	0.43	0.33	169

Figure 4.11: K Neighbors Classifier Accuracy

- **Image Enhancement Techniques:**
  Since in medical field, X-rays are supposed to be dealt with utmost accuracy and the prediction is expected to be top-notch. Since the accuracies are not depicting the same, some image enhancement techniques have been explored for a better accuracy in classifying the X-ray images. The techniques are as follows:
  - ***Contrast Limited Adaptive Histogram Equalisation(CLAHE):***
    The original PNG formatted dataset is converted into CL Adaptive Histogram Equalised Images using "cv2" library and "createCLAHE()" function from it. This function converts the PNG images into CLAHE images.

    Further, the newly generated CLAHE images are treated the same. They are split into train and test; label encoded; extracted SIFT features; and clustered. At this point, from the earlier results, certain models have been excluded (K Neighbors Classifier, Gaussian Naive Bayes Classifier, Decision Tree Classifier and Stochastic Gradiest Descent Classifier) and performed classification using only the high accuracy yielding classifiers, i.e., XGBoost and Random Forest Classifiers.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.48	0.97	0.65	72
5	0.00	0.00	0.00	2
6	0.25	0.14	0.18	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.75	0.50	0.60	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.46	169
macro avg	0.06	0.07	0.06	169
weighted avg	0.27	0.46	0.32	169

Figure 4.12: XGBoost Classifier on CLAHE images Accuracy(45.56%)

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.46	1.00	0.63	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.78	0.58	0.67	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.47	169
macro avg	0.05	0.07	0.06	169
weighted avg	0.25	0.47	0.32	169

Figure 4.13: Random Forest Classifier on CLAHE images Accuracy(46.74%)

- ***TV Homomorphic Filtering(TVH Filtering):***
  The PNG images are filtered using the TV Homomorphic Filtering to fix the illumination and reflectance in the image. These images undergo the same process as earlier. At the classification stage, only XGBoost and Random Forest Classifiers were considered.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.33	0.12	0.18	8
4	0.51	0.97	0.67	72
5	0.00	0.00	0.00	2
6	0.33	0.14	0.20	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	1.00	0.33	0.50	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.25	0.20	0.22	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.67	0.50	0.57	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.48	169
macro avg	0.13	0.10	0.10	169
weighted avg	0.33	0.48	0.37	169

Figure 4.14: XGBoost Classifier on TVHF images Accuracy(47.92%)

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.46	1.00	0.63	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.60	0.50	0.55	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.46	169
macro avg	0.05	0.07	0.05	169
weighted avg	0.24	0.46	0.31	169

Figure 4.15: Random Forest Classifier on TVHF images Accuracy(46.15%)

- **Oriented FAST and Rotated BRIEF(ORB):**

  By importing "cv2" library to the Jupyter environment, we can access the ORB methods to extract the image features. The library "cv2" has a method "ORB_create()" that takes a parameter as the number of ORB points to be generated.

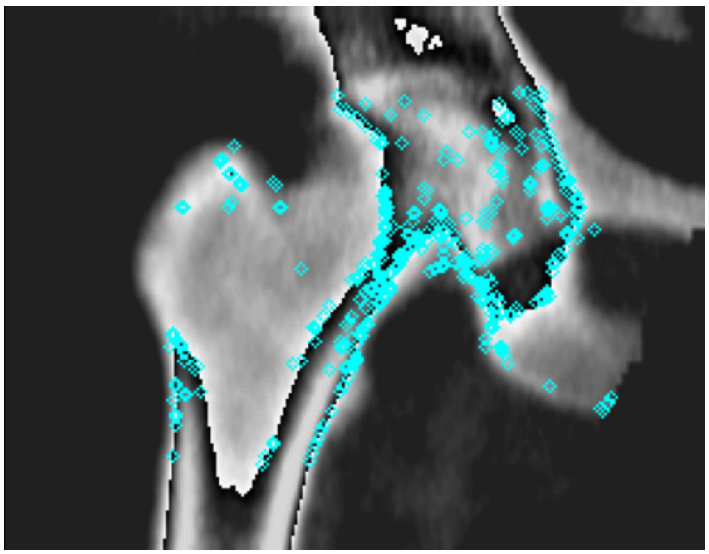


Figure 4.16: Image with 100 ORB keypoints

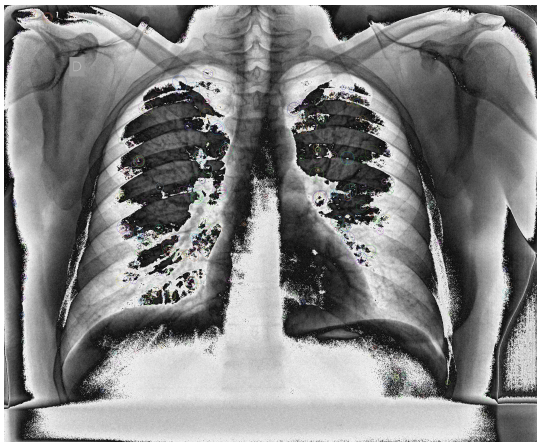


Figure 4.17: ORB on Chest

Figure 4.18: ORB on Knee

The methods that are used to extract the required descriptors from an image are "cv2.ORB_create(n).detect(image)" and "cv2.ORB_create(n).compute(image)". The former method, detect(image), detects all the possible or the required

keypoints. The latter method, compute(image), take the descriptors from the "detect(image)" method and computes them to store it for further purposes. The keypoints, or descriptors, generated are then stored into a list.

After creating all the ORB keypoints, 150 best keypoints, the final descriptors list resulted to a size of (756953, 32).

- **Machine Learning Models(with ORB descriptors):**
  Once clustering is done, the computed descriptors are ready. Once all the image features are derived, several machine learning models used those features and classified the images. The following are the Machine Learning Models:
  - ***Linear Support Vector Classifier:***
    *With the help of the imported library, "sklearn", the "LinearSVC()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of **28.40%**.*

	precision	recall	f1-score	support
0	0.29	0.25	0.27	8
1	0.00	0.00	0.00	1
2	0.20	0.50	0.29	4
3	0.30	0.38	0.33	8
4	0.65	0.56	0.60	72
5	0.00	0.00	0.00	2
6	0.29	0.29	0.29	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.10	0.10	0.10	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.19	0.25	0.21	12
17	0.29	0.29	0.29	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.33	169
macro avg	0.10	0.11	0.10	169
weighted avg	0.35	0.33	0.33	169

Figure 4.19: Linear Support Vector Classifier Accuracy

– ***Decision Tree Classifier:***

With the help of the imported library, "sklearn", the "DecisionTreeClassifier()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***30.17%****.*

	precision	recall	f1-score	support
0	0.12	0.12	0.12	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.09	0.12	0.11	8
4	0.55	0.51	0.53	72
5	0.00	0.00	0.00	2
6	0.33	0.29	0.31	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.14	0.10	0.12	10
13	0.50	0.50	0.50	2
14	0.00	0.00	0.00	4
15	0.33	0.20	0.25	5
16	0.31	0.33	0.32	12
17	0.25	0.29	0.27	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.06	0.17	0.09	6
accuracy			0.30	169
macro avg	0.12	0.11	0.11	169
weighted avg	0.32	0.30	0.31	169

Figure 4.20: Decision Tree Classifier Accuracy

– ***Random Forest Classifier:***

With the same imported library, "sklearn", the "RandomForestClassifier()" function was used with a parameter as "n_estimators = 100", which creates a 100 decision trees during classification. The descriptors are given to the Random Forest Classifier and resulted with an accuracy of ***46.15%****.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.44	1.00	0.62	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.00	0.00	0.00	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.86	0.50	0.63	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.46	169
macro avg	0.06	0.07	0.05	169
weighted avg	0.25	0.46	0.31	169

Figure 4.21: Random Forest Classifier Accuracy

– ***XGBoost Classifier:***

"xgboost" library has been imported that provided us with the used the "XGBClassifier()" function from the imported library. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***47.33%****.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	8
4	0.53	0.96	0.68	72
5	0.00	0.00	0.00	2
6	0.29	0.29	0.29	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.29	0.20	0.24	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.55	0.50	0.52	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	1.00	0.50	0.67	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.47	169
macro avg	0.12	0.11	0.10	169
weighted avg	0.31	0.47	0.36	169

Figure 4.22: XGBoost Classifier Accuracy

– ***Stochastic Gradient Descent Classifier:***

*With the help of the imported library, "sklearn", the "SGDClassifier()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of **38.46%**.*

	precision	recall	f1-score	support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	4
3	0.40	0.50	0.44	8
4	0.55	0.68	0.61	72
5	0.00	0.00	0.00	2
6	0.14	0.14	0.14	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.23	0.30	0.26	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.31	0.33	0.32	12
17	0.29	0.29	0.29	7
18	0.00	0.00	0.00	4
19	1.00	0.50	0.67	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.14	0.17	0.15	6
accuracy			0.38	169
macro avg	0.13	0.13	0.13	169
weighted avg	0.32	0.38	0.35	169

Figure 4.23: Stochastic Gradient Descent Classifier Accuracy

– ***Gaussian Naive Bayes Classifier:***

The "GaussianNB()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***34.91%****.*

	precision	recall	f1-score	support
0	0.06	0.12	0.08	8
1	0.00	0.00	0.00	1
2	0.29	0.50	0.36	4
3	0.24	0.50	0.32	8
4	0.74	0.43	0.54	72
5	0.00	0.00	0.00	2
6	0.17	0.14	0.15	7
7	1.00	0.50	0.67	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.18	0.50	0.27	4
11	0.00	0.00	0.00	2
12	0.29	0.50	0.37	10
13	0.00	0.00	0.00	2
14	0.20	0.25	0.22	4
15	0.00	0.00	0.00	5
16	0.54	0.58	0.56	12
17	0.29	0.29	0.29	7
18	0.00	0.00	0.00	4
19	1.00	1.00	1.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.35	169
macro avg	0.22	0.23	0.21	169
weighted avg	0.44	0.35	0.37	169

Figure 4.24: Gaussian Naive Bayes Classifier Accuracy

– ***K Neighbors Classifier:***

We used the "KNeighborsClassifier()" function from the imported library with parameters "n-neighbors=5", implying the presence of 5 neighbors, and "metric = "minkowski", which describes the distance computation with the power "p = 2". The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***41.42%****.*

	precision	recall	f1-score	support
0	0.06	0.12	0.08	8
1	0.00	0.00	0.00	1
2	0.10	0.25	0.14	4
3	0.12	0.12	0.12	8
4	0.52	0.83	0.64	72
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	7
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	3
10	0.00	0.00	0.00	4
11	0.00	0.00	0.00	2
12	0.67	0.20	0.31	10
13	0.00	0.00	0.00	2
14	0.00	0.00	0.00	4
15	0.00	0.00	0.00	5
16	0.83	0.42	0.56	12
17	0.00	0.00	0.00	7
18	0.00	0.00	0.00	4
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	6
accuracy			0.41	169
macro avg	0.10	0.08	0.08	169
weighted avg	0.33	0.41	0.34	169

Figure 4.25: K Neighbors Classifier Accuracy

- **Speeded-Up Robust Features(SURF):**

  Since the SURF algorithm is patented all throughout, we had to extract the SURF features from MATLAB instead of Jupyter. Although, the SURF is not readily available, we could manage the extraction only after BoVW is performed on the images. This is because, BoVW uses SURF to extract the features from the images and use them to classify. MATLAB has a function "bagOfFeatures" which does the above SURF feature extractions. Importing the SURF features, the images are classified from the machine learning models.

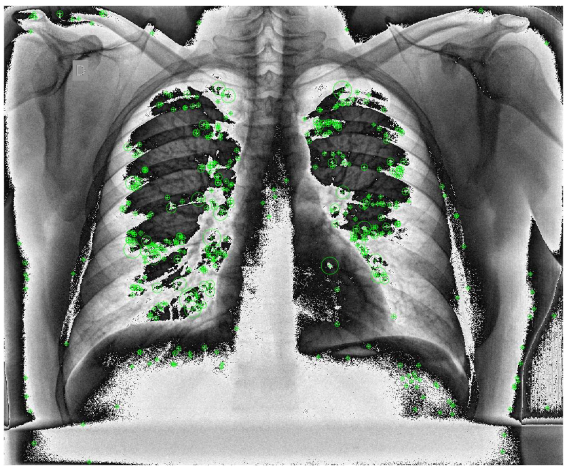


Figure 4.26: SURF on Chest

Figure 4.27: SURF on Knee

Machine Learning Models(with SURF descriptors):

Once clustering is done, the computed descriptors are ready. Once all the image features are derived, several machine learning models used those features and classified the images. The following are the Machine Learning Models:

- ***Linear Support Vector Classifier:***

  With the help of the imported library, "sklearn", the "LinearSVC()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***75.59%****.*

	precision	recall	f1-score	support
0	0.73	0.55	0.63	20
1	0.00	0.00	0.00	3
2	0.89	0.80	0.84	10
3	0.77	0.89	0.83	19
4	0.87	0.98	0.92	181
5	0.60	0.50	0.55	6
6	0.73	0.94	0.82	17
7	0.50	0.25	0.33	4
8	1.00	0.33	0.50	3
9	0.00	0.00	0.00	8
10	0.40	0.60	0.48	10
11	0.00	0.00	0.00	5
12	0.69	0.88	0.77	25
13	1.00	0.20	0.33	5
14	0.33	0.10	0.15	10
15	0.54	0.64	0.58	11
16	0.65	0.50	0.57	30
17	0.69	0.65	0.67	17
18	0.57	0.40	0.47	10
19	0.56	0.83	0.67	6
20	1.00	0.50	0.67	2
21	0.00	0.00	0.00	4
22	0.63	0.75	0.69	16
accuracy			0.76	422
macro avg	0.57	0.49	0.50	422
weighted avg	0.72	0.76	0.73	422

Figure 4.28: Linear Support Vector Classifier Accuracy

– ***Decision Tree Classifier:***

With the help of the imported library, "sklearn", the "DecisionTreeClassifier()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***49.52%****.*

	precision	recall	f1-score	support
0	0.19	0.15	0.17	20
1	0.00	0.00	0.00	3
2	0.08	0.10	0.09	10
3	0.20	0.16	0.18	19
4	0.78	0.81	0.79	181
5	0.14	0.17	0.15	6
6	0.59	0.59	0.59	17
7	0.00	0.00	0.00	4
8	0.00	0.00	0.00	3
9	0.20	0.25	0.22	8
10	0.15	0.20	0.17	10
11	0.00	0.00	0.00	5
12	0.39	0.60	0.48	25
13	0.50	0.20	0.29	5
14	0.12	0.10	0.11	10
15	0.19	0.27	0.22	11
16	0.45	0.33	0.38	30
17	0.31	0.29	0.30	17
18	0.17	0.10	0.12	10
19	0.25	0.17	0.20	6
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	4
22	0.44	0.25	0.32	16
accuracy			0.50	422
macro avg	0.22	0.21	0.21	422
weighted avg	0.49	0.50	0.49	422

Figure 4.29: Decision Tree Classifier Accuracy

– ***Random Forest Classifier:***

With the same imported library, "sklearn", the "RandomForestClassifier()" function was used with a parameter as "n_estimators = 100", which creates a 100 decision trees during classification. The descriptors are given to the Random Forest Classifier and resulted with an accuracy of ***59.95%****.*

	precision	recall	f1-score	support
0	0.33	0.05	0.09	20
1	0.00	0.00	0.00	3
2	0.75	0.30	0.43	10
3	0.17	0.05	0.08	19
4	0.61	0.99	0.75	181
5	0.00	0.00	0.00	6
6	0.75	0.71	0.73	17
7	0.00	0.00	0.00	4
8	0.00	0.00	0.00	3
9	1.00	0.12	0.22	8
10	0.56	0.50	0.53	10
11	0.00	0.00	0.00	5
12	0.57	0.96	0.72	25
13	0.00	0.00	0.00	5
14	0.00	0.00	0.00	10
15	0.67	0.18	0.29	11
16	0.69	0.37	0.48	30
17	0.80	0.24	0.36	17
18	0.00	0.00	0.00	10
19	0.00	0.00	0.00	6
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	4
22	0.45	0.62	0.53	16
accuracy			0.60	422
macro avg	0.32	0.22	0.23	422
weighted avg	0.51	0.60	0.50	422

Figure 4.30: Random Forest Classifier Accuracy

– ***XGBoost Classifier:***

"xgboost" library was imported that provides us with the used the "XGB-Classifier()" function from the imported library. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***69.19%****.*

	precision	recall	f1-score	support
0	0.59	0.50	0.54	20
1	0.00	0.00	0.00	3
2	0.67	0.40	0.50	10
3	0.57	0.42	0.48	19
4	0.81	0.98	0.89	181
5	1.00	0.17	0.29	6
6	0.62	0.88	0.73	17
7	1.00	0.50	0.67	4
8	0.20	0.33	0.25	3
9	0.57	0.50	0.53	8
10	0.50	0.40	0.44	10
11	0.00	0.00	0.00	5
12	0.63	0.88	0.73	25
13	1.00	0.40	0.57	5
14	0.00	0.00	0.00	10
15	0.33	0.36	0.35	11
16	0.57	0.43	0.49	30
17	0.77	0.59	0.67	17
18	0.38	0.30	0.33	10
19	0.50	0.33	0.40	6
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	4
22	0.56	0.62	0.59	16
accuracy			0.69	422
macro avg	0.49	0.39	0.41	422
weighted avg	0.66	0.69	0.66	422

Figure 4.31: XGBoost Classifier Accuracy

– ***Stochastic Gradient Descent Classifier:***

With the help of the imported library, "sklearn", the "SGDClassifier()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***74.40%****.*

	precision	recall	f1-score	support
0	0.33	0.05	0.09	20
1	0.00	0.00	0.00	3
2	0.75	0.30	0.43	10
3	0.17	0.05	0.08	19
4	0.61	0.99	0.75	181
5	0.00	0.00	0.00	6
6	0.75	0.71	0.73	17
7	0.00	0.00	0.00	4
8	0.00	0.00	0.00	3
9	1.00	0.12	0.22	8
10	0.56	0.50	0.53	10
11	0.00	0.00	0.00	5
12	0.57	0.96	0.72	25
13	0.00	0.00	0.00	5
14	0.00	0.00	0.00	10
15	0.67	0.18	0.29	11
16	0.69	0.37	0.48	30
17	0.80	0.24	0.36	17
18	0.00	0.00	0.00	10
19	0.00	0.00	0.00	6
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	4
22	0.45	0.62	0.53	16
accuracy			0.60	422
macro avg	0.32	0.22	0.23	422
weighted avg	0.51	0.60	0.50	422

Figure 4.32: Stochastic Gradient Descent Classifier Accuracy

– ***Gaussian Naive Bayes Classifier:***

The "GaussianNB()" function was used from the imported library itself. The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***53.08%****.*

	precision	recall	f1-score	support
0	0.31	0.55	0.39	20
1	0.00	0.00	0.00	3
2	0.33	0.30	0.32	10
3	0.49	0.89	0.63	19
4	0.94	0.66	0.77	181
5	0.00	0.00	0.00	6
6	0.50	0.65	0.56	17
7	0.00	0.00	0.00	4
8	0.00	0.00	0.00	3
9	0.11	0.12	0.12	8
10	0.29	0.40	0.33	10
11	0.20	0.20	0.20	5
12	0.71	0.48	0.57	25
13	0.00	0.00	0.00	5
14	0.00	0.00	0.00	10
15	0.27	0.55	0.36	11
16	0.44	0.63	0.52	30
17	0.45	0.29	0.36	17
18	0.36	0.40	0.38	10
19	0.10	0.17	0.12	6
20	0.00	0.00	0.00	2
21	0.14	0.25	0.18	4
22	0.53	0.56	0.55	16
accuracy			0.53	422
macro avg	0.27	0.31	0.28	422
weighted avg	0.61	0.53	0.55	422

Figure 4.33: Gaussian Naive Bayes Classifier Accuracy

– ***K Neighbors Classifier:***

We used the "KNeighborsClassifier()" function from the imported library with parameters "n-neighbors=5", implying the presence of 5 neighbors, and "metric = "minkowski", which describes the distance computation with the power "p = 2". The clustered points are given to the Decision Tree Classifier which, in return, gave us an accuracy of ***69.43%****.*

	precision	recall	f1-score	support
0	0.65	0.65	0.65	20
1	0.00	0.00	0.00	3
2	0.35	0.80	0.48	10
3	0.58	0.58	0.58	19
4	0.85	0.97	0.91	181
5	0.00	0.00	0.00	6
6	0.57	0.71	0.63	17
7	0.67	1.00	0.80	4
8	0.00	0.00	0.00	3
9	0.33	0.38	0.35	8
10	0.40	0.40	0.40	10
11	0.00	0.00	0.00	5
12	0.80	0.48	0.60	25
13	0.00	0.00	0.00	5
14	0.29	0.20	0.24	10
15	0.64	0.82	0.72	11
16	0.67	0.53	0.59	30
17	0.64	0.41	0.50	17
18	1.00	0.30	0.46	10
19	0.40	0.33	0.36	6
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	4
22	0.58	0.69	0.63	16
accuracy			0.69	422
macro avg	0.41	0.40	0.39	422
weighted avg	0.67	0.69	0.67	422

Figure 4.34: K Neighbors Classifier Accuracy

4.3 Deep Learning Approach

In this approach, we have implemented Data augmentation techniques for reducing the imbalance between the classes and ran multiple models with different methodologies.

4.3.1 Data Augmentation

In our proposed model, the X-Ray images of the patients are being used. However, due to the unavailability of sufficient datasets for training,the existing dataset is to be increased via the data augmentation techniques.

Traditional Data Augmentation

Applying the conventional data augmentation techniques is the first and simplest way to expand an existing data-set in the field of deep learning. By making small adjustments to the existing images, new ones can be created. Five data augmentation techniques (vertical and horizontal flips, rotation, scaling, translation, and crop) have been used on the X-Ray images in this work. These modifications have been presented using several examples.Total generated a 680 images, but there was loss of data as for some classes traditional augmentation was not optimal as the necessary features were either diminished or ignored for the augmented images therefore there isn't a significant change in results.

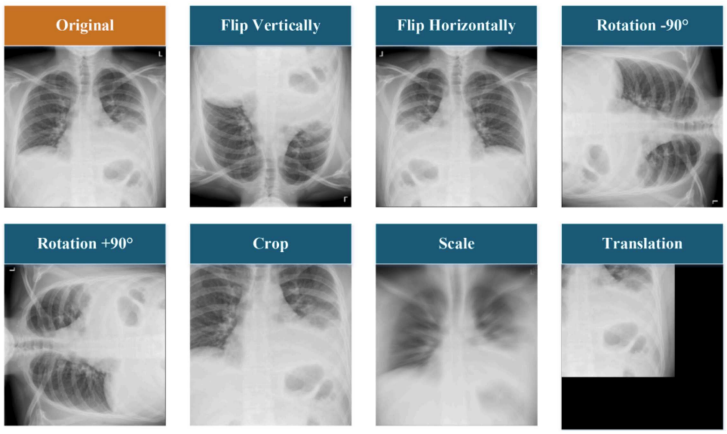


Figure 4.35: Sample modifications made to the traditional data augmentation process.

GAN Data Augmentation

This technique allows us to create fake, manufactured data (sounds and images) that are comparable to the input data. In this method, two neural network models—a generator and a discriminator—compete against one another in a game as it is developed in game theory. The role of the generator model is to create content (pictures), while the opponent model's (the discriminator's) role is to distinguish between authentic and fraudulent images. The details of the designed GAN and the specifications of every existing layer and the values of the hyper parameters considered in each of the generator and the discriminator models have been listed:

Operation	*Kernel*	*Strides*	*Feature maps*	*BN*	*Activation*
Generator					
Transposed Convolution	4×4	1×1	512	✓	ReLU
Transposed Convolution	4×4	2×2	256	✓	ReLU
Transposed Convolution	4×4	2×2	128	✓	ReLU
Transposed Convolution	4×4	2×2	256	✓	ReLU
Transposed Convolution	4×4	2×2	3	–	Tanh
Discriminator					
Convolution	4×4	2×2	64	–	Leaky ReLU
Convolution	4×4	2×2	128	✓	Leaky ReLU
Convolution	4×4	2×2	256	✓	Leaky ReLU
Convolution	4×4	2×2	512	✓	Leaky ReLU
Convolution	4×4	1×1	1	–	Sigmoid
Frame Work	PyTorch				
Batch size for real data	128				
Batch size for generator	128				
Number of worker	2				
Number of epochs	400				
Optimizer	Adam ($\beta_1 = 0.5$)				
Learning rate	0.0002				
Leaky ReLU slope	0.2				
Loss function	Binary Cross Entropy				

Figure 4.36: The hyper parameters of the GAN

As was mentioned before, with this technique, the data volume can be increased considerably, 64 images per class have been generated, excluding 4 classes. That's a total 1088 generated images along with the 1686 images, summing to 2774 images.

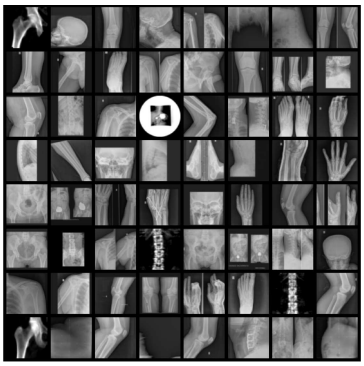


Figure 4.37: Dataset Images

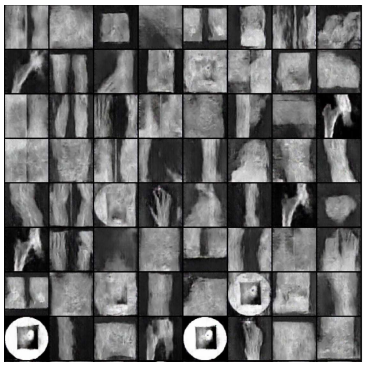


Figure 4.38: GAN Generated Images

4.3.2 Gabor filter

This filter basically checks to see if a specific frequency exists in an image along a certain orientation at a local zone around a point of interest or a region of analysis. To obtain diverse information with regards to different scales and orientations, a set of Gabor filters with 8 orientations and 8 scales were selected in order to extract the image features from the existing X-Ray images. Therefore, the total number of filters used in our experiments is 64. In most cases, it is sufficient to use 8 different orientations to cover the local directions of image features. The values of these parameters have been given below:

Parameters	***Values***
Number of scales	8
Number of orientations	8
Wavelength (λ)	$4\pi; 4\sqrt{2}\pi; 8\pi; 8\sqrt{2}\pi; 16\pi; 16\sqrt{2}\pi; 32\pi; 32\sqrt{2}\pi$
Orientation (θ)	$0; \frac{\pi}{8}; \frac{2\pi}{8}; \frac{3\pi}{8}; \frac{4\pi}{8}; \frac{5\pi}{8}; \frac{6\pi}{8}; \frac{7\pi}{8}$
Maximum frequency	0.250
Frequency (f)	$0.250; 0.177; 0.125; 0.088; 0.062; 0.044; 0.031; 0.022$
Sigma (σ)	$4\sqrt{2}\pi; 8\pi; 8\sqrt{2}\pi; 16\pi; 16\sqrt{2}\pi; 32\pi; 32\sqrt{2}\pi; 64\pi$
Phase offset (ψ)	0.1
Spatial aspect ratio (γ)	$\sqrt{2}$
Envelope	Gaussian

Figure 4.39: The values of the Gabor filter parameters

By exploring all the parameters we finalized on the given parameter combinations: θ**=** π**/4,** σ**=1,** λ**=**π**/2,** γ**=0.05**

$$G(\sigma, \theta, \lambda, \gamma, \psi) = G(1, \pi/4, \pi/2, 0.05, \psi) \qquad \text{(Gabor Filter Parameter)}$$

Figure 4.40: Gabor filtered image with Final Parameters

4.3.3 Convolution Neural Network(CNN) Models

Another strategy for dealing with a lack of data is to use the transfer learning method. A pre-trained model must be tuned as part of the transfer learning process. Therefore, after making the desired adjustments to the data, we can start using a pretrained neural network like EfficientNetB3, InceptionV3, DenseNet121, ResNet50, or VGG16 architectures to train the data.

This outline of the model that is being dealt with has 5 steps:

- **Step 1:**

  The images from the dataset will be preprocessed with necessary action, namely, conversion of the image formats, and creating the necessary files.

- **Step 2:**

  In this step, the images are enhanced using the Gabor image enhancement technique. G(1,$\pi/4$,$pi/2$,0.05,ψ) is the combination of parameters that has been considered for image enhancement. After testing with multiple Gabor parameter combinations, the above combination has been finalised.

- **Step 3:**

  This step is being performed simultaneously with Step 2. The images from the dataset are processed by a Generative Adversial Network(GAN), where the two networks, Generator and Discriminator act against each other. The Generator generates new images from the dataset, while the Discriminator differentiates between the real images from the dataset and the generated images. By default, the number of images generated are 32 per class. It has been changed to 64 later, for a better classification of all the possible images that can be generated.

- **Step 4:**

  The above arrived images, i.e., from GAN and Gabor filter, the total number of images are 1686(from the dataset) and 1088(GAN generated), combining to 2774 images. These images are generated with their respective labels attached. Later, the images are given to CNN models, namely, EfficientNet B3, Inception V3 and DenseNet 121. These are finalised upon from the earlier CNNs' accuracies.

- **Step 5:**

  By using the necessary weights, the 2774 images are trained by the CNNs taken. Here, the concept of early stopping has been put to act, where the epochs are considered arbitrary. Though, they are defined, epochs are stopped by the model where necessary.

  - **EfficientNetB3 accuracy** - 86.94%
  - **InceptionV3 accuracy** - 84.55%
  - **DenseNet121 accuracy** - 81.19%

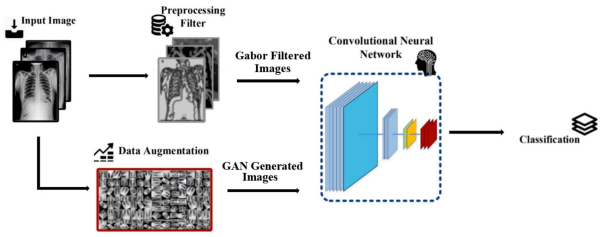


Figure 4.41: Final Model Architecture

Chapter 5

RESULTS AND DISCUSSION

5.1 Machine Learning Models Results Comparison

The extracted features from the ORB, SIFT, SURF are given to the pre trained machine learning models and compared for finding the best performing model.

Classifier	ORB	SIFT	SURF
SVM	19.8	17.2	20.3
Random Forest	46.1	46.7	47.6
XGBoost	47.3	48.5	50.6
SGD	38.5	36.1	31.9
Naive Bayes	34.9	7.1	8.1
KNN	41.4	42.6	33.7
Decision Tree	30.17	27.2	34.4

Table 5.1: Unenhanced Data Accuracy Comparison

The above features are extracted from the raw unenhanced data and by comparing we conclude that XGBoost and Random Forest are better performing. So now we are going to enhance the data using various enhancing techniques and check the performance. The

Classifier	CLAHE	ADAPTIVE-HE	TV HOMOMORPHIC
Random Forest	46.7	47.9	46.2
XGBoost	45.5	47.5	48.0

Table 5.2: Enhanced Data Accuracy Comparison

performance of the models reduced cause of the data loss during the enhancement of the images, hence un-enhanced data is better for feature extraction.

The accuracy's are less due to the highly imbalanced dataset so many classes with low number of images are getting misclassified.Therefore we are proceeding with data augmentation and deep learning techniques.

5.2 Deep Learning Models Results Comparison

Model	Accuracy
EfficientNetB3	66.2
InceptionV3	64.8
DenseNet121	56.9
ResNet50	57.4
VGG16	49.6

Table 5.3: Traditional Augmented Data on pretrained Deep learning Models

Table 5.3 compares the accuracy of the Deep learning pre trained models with raw dataset and the traditional augmented(scaling, cropping, flipping, rotation, translation, etc) images given as input. EfficientNetB3 is the most efficient, if observed. Therefore we will be proceeding with EfficientNetB3, InceptionV3, and DenseNet121.

Model	Traditional Augmentation	GAN Augmentation	GAN + Gabor Filter
EfficientNetB3	66.2	83.47	86.94
InceptionV3	64.82	8.96	84.55
DenseNet121	56.94	81.12	81.19

Table 5.4: Deep learning Models and Data Augmentation Comparision

Table 5.4 states the comparison for a proposed model using basic augmented images with gabor filter and GAN Generated images as inputs to the pre processed models and compared the performance. The maximum classification accuracy was achieved by tweaked EfficientNetB3, which produced results for 22 classes with an overall accuracy of 86.94%.

Chapter 6

CONCLUSION

As observed from the classification results obtained using various classification techniques, it is difficult to achieve high accuracy for individual classes. The issue of significant inter-class similarity and intra-class variability, which are present particularly among the image classes from the same sub-body area, is the cause of this. When working with a sizable medical database, this issue frequently arises. Given the difficulty in accessing medical data and the high expense of labelling data, generative models have a lot to offer the field of medical imaging. Generative adversarial networks provide a promising avenue to generate new images. We can produce X-ray images that are qualitatively realistic and useful for a range of applications.

In order to attain higher classification accuracies for body part X-ray classification than with conventional data augmentation techniques, we combine produced X-ray images with real training images.We proposed a model using basic augmented images with gabor filter and GAN generated images as inputs to the pre processed models and compared the performance. The maximum classification accuracy was achieved by tweaked EfficientNetB3, which produced results for 22 classes with an overall accuracy of 86.94%. It should be noted that every individual class may not achieve this result. Based on the experimental findings over the whole dataset, all of the approaches are competitive.

References

[1] E. Ahn, A. Kumar, M. Fulham, D. Feng, and J. Kim, "Unsupervised domain adaptation to classify medical images using zero-bias convolutional auto-encoders and context-based feature augmentation," *IEEE Transactions on Medical Imaging*, vol. 39, no. 7, pp. 2385–2394, 2020.

[2] E. Ahn, A. Kumar, J. Kim, C. Li, D. Feng, and M. Fulham, "X-ray image classification using domain transferred convolutional neural networks and local sparse spatial pyramid," in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 2016, pp. 855–858.

[3] ——, "X-ray image classification using domain transferred convolutional neural networks and local sparse spatial pyramid," in *2016 IEEE 13th international symposium on biomedical imaging (ISBI)*. IEEE, 2016, pp. 855–858.

[4] U. Avni, H. Greenspan, E. Konen, M. Sharon, and J. Goldberger, "X-ray categorization and retrieval on the organ and pathology level, using patch-based visual words," *IEEE Transactions on Medical Imaging*, vol. 30, no. 3, pp. 733–746, 2011.

[5] A. H. Barshooi and A. Amirkhani, "A novel data augmentation based on gabor filter and convolutional deep learning for improving the classification of covid-19 chest x-ray images," *Biomedical Signal Processing and Control*, vol. 72, p. 103326, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S174680942100923X

[6] S. Choi, "X-ray image body part clustering using deep convolutional neural network: Snumedinfo at imageclef 2015 medical clustering task." in *CLEF (Working Notes)*, 2015.

[7] A. Giełczyk, A. Marciniak, M. Tarczewska, and Z. Lutowski, "Preprocessing methods in chest x-ray image classification," *Plos one*, vol. 17, no. 4, p. e0265949, 2022.

[8] B. C. Ko, S. H. Kim, and J.-Y. Nam, "X-ray image classification using random forests with local wavelet-based cs-local binary patterns," *Journal of digital imaging*, vol. 24, pp. 1141–1151, 2011.

[9] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 31–40, 2017.

[10] A. Madani, M. Moradi, A. Karargyris, and T. Syeda-Mahmood, "Chest x-ray generation and data augmentation for cardiovascular abnormality classification," in *Medical imaging 2018: Image processing*, vol. 10574. SPIE, 2018, pp. 415–420.

[11] A. Mueen, R. Zainuddin, and M. S. Baba, "Automatic multilevel medical image annotation and retrieval," *Journal of digital imaging*, vol. 21, pp. 290–295, 2008.

[12] V. Thamilarasi and R. Roselin, "Automatic classification and accuracy by deep learning using cnn methods in lung chest x-ray images," in *IOP Conference Series: Materials Science and Engineering*, vol. 1055, no. 1. IOP Publishing, 2021, p. 012099.

[13] G. Tian, H. Fu, and D. Dagan Feng, "Automatic medical image categorization and annotation using lbp and mpeg-7 edge histograms," in *2008 International Conference on Information Technology and Applications in Biomedicine*, 2008, pp. 51–53.

[14] M. R. Zare, D. O. Alebiosu, and S. L. Lee, "Comparison of handcrafted features and deep learning in classification of medical x-ray images," in *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*, 2018, pp. 1–5.

[15] M. R. Zare, W. C. Seng, and A. Mueen, "Automatic classification of medical x-ray images," *Malaysian Journal of Computer Science*, vol. 26, no. 1, p. 9–22, Mar. 2013. [Online]. Available: https://mjir.um.edu.my/index.php/MJCS/article/view/6725