Research on Musculoskeletal Abnormality Detection using Deep Learning Techniques

A report submitted in partial fulfilment of the requirements

for the award of the degree of

B.Tech Computer Science and Engineering

by

Surya Teja Regalla
(Roll No: 120CS0021)

Shiva Prasad Kudikala
(Roll No: 120CS0016)

Under the Guidance of

Dr. K. Nagaraju

Assistant Professor

Dept. of CSE, IIITDM Kurnool

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DESIGN AND MANUFACTURING KURNOOL

November 2023

Evaluation Sheet

Title of the Project: Research on Musculoskeletal Abnormality Detection using Deep Learning Techniques

Name of the Student(s): Surya Teja Regalla (120CS0021)
Shiva Prasad Kudikala (120CS0016)

Examiner(s):

Supervisor(s):

Head of the Department:

Date:

Place:

Certificate

We, **Surya Teja Regalla(120cs0021) and Shiva Prasad Kudikala(120cs0016)**, hereby declare that the material presented in the Project Report titled **Research on Musculoskeletal Abnormality Detection using Deep Learning Techniques** represents original work carried out by us in the **Department of Computer Science and Engineering** at the **Indian Institute of Information Technology Design and Manufacturing Kurnool** during the years **2023 - 2024**. With oue signature, we certify that:

- We have not manipulated any of the data or results.
- We have not committed any plagiarism of intellectual property. We have clearly indicated and referenced the contributions of others.
- We have explicitly acknowledged all collaborative research and discussions.
- We have understood that any false claim will result in severe disciplinary action.
- We have understood that the work may be screened for any form of academic misconduct.

Date: Student's Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this Report is carried out under my supervision, and is worthy of consideration for the requirements of B.Tech. Project work.

Advisor's Name: Advisor's Signature

Abstract

Using finger data from the MURA dataset as its primary emphasis, this proposed work offers a unique method for musculoskeletal anomaly diagnosis. Our model integrates deep learning approaches, such as DenseNet and ResNet architectures with Siamese networks. Contrast Limited Adaptive Histogram Equalization (CLAHE) preprocessing is used to improve radiograph quality. Furthermore, Triplet Focal Loss is used in the suggested model to enhance training and performance. The experimental findings reveal that our method may effectively identify musculoskeletal anomalies in finger radiographs, indicating the possibility of improved diagnostic precision in clinical contexts.

Acknowledgements

We would like to express our sincere gratitude to Dr. K. Nagaraju, Assistant professor, Department of Computer Science and Engineering for motivating and supervising us all through this project. We would like to thank for his timely guidance and support in completion of the project.

The opportunity to join the IIITDM Kurnool is an achievement for us. We are thankful to beloved god and our parents for the grace and motivation they gave to join the college and complete the degree. Sincerest gratitude to all the faculty members of IIITDM Kurnool those who taught us academics and life lessons.

We are grateful to our institute, Indian Institute of Information Technology Design and Manufacturing Kurnool.

Contents

List of Figures

List of Tables

Abbreviations

MURA	**MU**sculosketal **R**adiographs **A**bnormality
PACS	**P**icture **A**rchive and **C**ommunication **S**ystem
CNN	**C**onvolutional **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etworks
CLAHE	**C**ontrast **L**imited **A**daptive **H**istogram **E**qualization
AHE	**A**daptive **H**istogram **E**qualization
MRI	**M**agnetic **R**esonance **I**maging
RGB	**R**ed **G**reen **B**lue
ResNet	**Res**idual **Net**works
GAP	**G**lobal **A**verage **P**ooling
ReLU	**Re**ctified **L**inear **U**nit
CAM	**C**lass **A**ctivation **M**ap
Grad-CAM	**Grad**ient-Weighted **C**lass **A**ctivation **M**ap
ER	**E**mergency **R**oom
BHTL	**B**atch **H**ard **T**riplet **L**oss
TFL	**T**riplet **F**ocal **L**oss

Symbols

κ	Cohen's kappa statistic
σ	Constant
$\mathcal{L}$	Curly L or Script L

Dedicated to all the IIITDM professors, and students who guided and helped us during the course of this project ...

Chapter 1

Introduction

1.1 What is Musculoskeletal Abnormality Detection?

The process of finding and assessing abnormalities or disorders related to the musculoskeletal system through the use of diagnostic imaging and other medical imaging techniques is known as "musculoskeletal abnormality detection." The musculoskeletal system of the human body, which is made up of bones, joints, muscles, tendons, ligaments, and other connective tissues, provides the body with structure, motion, and support.
It is crucial for the diagnosis, treatment, and oversight of diseases and injuries affecting the musculoskeletal system. Tumors, osteoarthritis, rheumatoid arthritis, ligament injuries, muscle tears, and fractures are routinely diagnosed by musculoskeletal imaging.

1.1.1 Why Musculoskeletal Abnormality Detection is needed?

One of the most important tasks for radiologists is to determine whether a study is abnormal. If the study is normal, the patient is clear of disease and does not need further diagnostic testing; if an anomaly is detected with a reasonable degree of confidence, the patient might need further assessment.
Since musculoskeletal illnesses impact over 1.7 billion people worldwide and account for over 30 million ER visits annually, they are particularly important to diagnose since they cause the majority of disability and severe, chronic pain.

1.2 Origin of the Project

Globally, radiologists are reading more and more cases in the medical imaging field; this trend is made harder by the extra challenges faced by those who work in areas of poverty. Making the crucial distinction between normal and abnormal investigations is one of their primary duties; this distinction has a significant impact on patient care and outcomes.
The increasing workload of radiologists worldwide has prompted the creation of novel tools intended to streamline image interpretation. Among these innovations is a tool that can be used to highlight issues in medical images and quickly bring them to the attention of a radiologist. This has the potential to reduce errors and speed up the image interpretation process. Furthermore, by visualizing a scenario in which cases exhibiting abnormalities are identified and given priority, this type of tool introduces the concept of worklist prioritization. By making sure that the sickest patients in the most severe cases receive prompt diagnoses, a strategic approach can be used to maximize patient care and outcomes.

1.3 Proposed Design

The main goal of this research is to use deep learning technologies to improve the diagnostic capabilities and accuracy of musculoskeletal abnormality detection. DenseNet-169 and ResNet50 architectures are integrated into the Siamese network architecture used in the project. Finger radiographs data is preprocessed using Contrast Limited Adaptive Histogram Equalization (CLAHE) techniques and specialized loss functions (batch hard triplet loss and triplet focal loss) are used.

1.4 Literature Survey

MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs [1]

In this paper, they used a sizable radiograph dataset MURA that included 14,863 upper extremity musculoskeletal investigations for this work. One or more perspectives for an upper extremity study are fed into the model. Their convolutional neural network forecasts the likelihood of anomaly on each view. By taking the arithmetic mean of the abnormality probabilities that the network produces for each image, we are able to calculate the overall

likelihood of abnormality for the research. If the study's probability of abnormality is higher than 0.5, the model predicts to be abnormal. They made the optimization of deep networks tractable by using a 169-layer Dense Convolutional Network architecture, which connects each layer to every other layer in a feed-forward manner. They adjusted the weighted binary cross entropy loss by swapping out the last fully connected layer for one with a single output. Next, they applied a sigmoid nonlinearity to every image X of study type T in the training set. Three of their gold standard radiologists were selected at random, and their performance on the test set was evaluated for both the radiologists and their model. The Cohen's kappa statistic (k), which indicates how well each radiologist and model agrees, and was used to compare radiologists and their models. As shown in Figure 1.1 The radiologists' best results were obtained from humerus or wrist investigations, whereas their worst results came from finger studies. Additionally, the model performed best on tests involving the wrist and worst on research involving the fingers.

	Radiologist 1	Radiologist 2	Radiologist 3	Model
Elbow	0.850 (0.830, 0.871)	0.710 (0.674, 0.745)	0.719 (0.685, 0.752)	0.710 (0.674, 0.745)
Finger	0.304 (0.249, 0.358)	0.403 (0.339, 0.467)	0.410 (0.358, 0.463)	0.389 (0.332, 0.446)
Forearm	0.796 (0.772, 0.821)	0.802 (0.779, 0.825)	0.798 (0.774, 0.822)	0.737 (0.707, 0.766)
Hand	0.661 (0.623, 0.698)	0.927 (0.917, 0.937)	0.789 (0.762, 0.815)	0.851 (0.830, 0.871)
Humerus	0.867 (0.850, 0.883)	0.733 (0.703, 0.764)	0.933 (0.925, 0.942)	0.600 (0.558, 0.642)
Shoulder	0.864 (0.847, 0.881)	0.791 (0.765, 0.816)	0.864 (0.847, 0.881)	0.729 (0.697, 0.760)
Wrist	0.791 (0.766, 0.817)	0.931 (0.922, 0.940)	0.931 (0.922, 0.940)	0.931 (0.922, 0.940)
Overall	0.731 (0.726, 0.735)	0.763 (0.759, 0.767)	0.778 (0.774, 0.782)	0.705 (0.700, 0.710)

FIGURE 1.1: Comparing radiologists and their model on the Cohen's kappa statistic. They highlight the best (green) and worst (red) performances on each of the study types and in aggregate.

Person Re-Identification With Triplet Focal Loss [2]

Researchers have discovered that the triplet loss's success depends on the hard triplets' mining. In this study, they propose the triplet focal loss for person ReID, inspired by the focal loss created for the classification model. By simply projecting the initial distance from the Euclidean space to an exponential kernel space, triplet focal loss can adaptively up-weight the training samples of the hard triplets and comparably down-weight the easy triplets. One popular verification model that is used to train CNN as an embedding function [3] is the triplet loss, proposed by Weinberger and Saul [4]. Its goal is to optimize the embedding space so that samples with the same identity are separated by a significantly smaller distance than samples with different identities. They created a deep CNN model based on the "Inception" module and developed the Deep Metric Learning

(DML) technique, which uses the Siamese architecture to learn pair-wise similarity metrics. They have employed triplet loss, which uses an input image triplet which is anchor sample, positive sample, and negative sample.

Afterwards, they turned to batch hard triplet loss, whose main idea is to mine the hard triplet samples within each mini-batch by combining the triplet production stage with the training process. Batch Hard Triplet Loss selects the hardest positive and negative sample from the mini-batch, treating each image as an anchor sample in turn. Subsequently, Triplet Focal Loss was proposed, which has the ability to adaptively up-weight training samples of hard triplets and comparably down-weight training samples of easy triplets. The Euclidean distance is mapped to an exponential kernel space, penalizing the hard triplets significantly more than the easy ones.

Hard negative examples are hard, but useful [5]

Triplet losses generally concentrate on choosing the most helpful triplets of images to take into account, using algorithms that choose similar instances from separate classes or dissimilar examples from the same class. Accordingly, they discovered in their earlier research that using the hardest negative instances for optimization results in bad training behavior. Hard negative triplets are those that are not in the correct configuration, where the anchor-positive similarity is less than the anchor-negative similarity. In this paper, they have deduced why triplet loss training fails in the presence of hard negatives. Furthermore, they have provided a correction for the loss function that makes it possible to optimize using hard negative instances. This repair leads to more broadly applicable features and picture retrieval results that beat the state of the art for datasets with substantial intra-class variance.

Alternative strategies, such as semi-hard triplet mining [6], which concentrates on triplets with negative instances that are nearly as close to the anchor as positive examples, have been devised by authors to avoid this bad effect on training. The most similar negative example that is less similar than the equivalent positive example is chosen for an anchor using the semi-hard negative triplets triplet selection approach.The issue that the optimization frequently fails for these triplets can be fixed by excluding triplets with hard negative examples. But hard negative examples are important. The hardest negative examples are literally the cases where the distance metric fails to capture semantic similarity. The loss function they have used is, as shown in equation below

$$\mathcal{L}(S_{\text{ap}}, S_{\text{an}}) = -\log\left(\frac{e^{S_{\text{ap}}}}{e^{S_{\text{ap}}} + e^{S_{\text{an}}}}\right)$$

where S_{ap} is a similarity metric of anchor-positive pair, and S_{an} of anchor-negative pair.

The solution for the challenge with hard negative triplets is to decouple them into anchor-positive pairs and anchor-negative pairs, and ignore the anchor-positive pairs, and introduce a contrastive loss that penalizes the anchor-negative similarity. They named this Selectively Contrastive Triplet loss L_{SC}, and defined this as follows:

$$L_{SC}(S_{\text{ap}}, S_{\text{an}}) = \begin{cases} \lambda S_{\text{an}}, & \text{if } S_{\text{an}} > S_{\text{ap}} \\ L(S_{\text{ap}}, S_{\text{an}}), & \text{otherwise} \end{cases}$$

1.5 Division of Work

Name	Roll No.	Role
K. Shiva Prasad	120cs0016	1. Literature Study 2. ResNet50 Implementation 3. Triplet Loss 4. Triplet Focal Loss
R. Surya Teja	120cs0021	1. Literature Study 2. DenseNet-169 Implementation 3. CLAHE Preprocessing 4. Batch Hard Triplet Loss 5. Hard Examples Training

Chapter 2

Datasets, Libraries and Modules

2.1 Dataset

The Picture Archive and Communication System (PACS) of Stanford Hospital provided HIPAA-compliant images that were used to compile the Stanford MURA dataset [1]. 40,561 multi-view radiographs from 14,863 studies of 12,173 patients between 2001 and 2012 make up the dataset. There are seven common radiographic study types for the upper extremities: elbow, finger, forearm, hand, humerus, shoulder, and wrist. Each image falls into one of these categories. When interpreting and diagnosing each study, board-certified radiologists manually labeled it as normal or abnormal. Table 2.1 illustrates this. The dataset is divided into three sets: validation (783 patients, 1,199 studies, 3,197 images), test (206 patients, 207 studies, 556 images), and training (11,184 patients, 13,457 studies, 36,808 images).

Study	Train		Validation		Total
	Normal	Abnormal	Normal	Abnormal	
Elbow	1,094	660	92	66	1,912
Finger	1,280	655	92	83	2,110
Hand	1,497	521	101	66	2,185
Humerus	321	271	68	67	727
Forearm	590	287	69	64	1,010
Shoulder	1,364	1,457	99	95	3,015
Wrist	2,134	1,326	140	97	3,697
Total number of studies	8,280	5,177	661	538	14,656

TABLE 2.1: Distribution of studies in the training and validation sets for the MURA dataset

Since the MURA dataset test set isn't publicly available, the study reports the findings from the validation set instead.

2.2 Libraries and Modules

TensorFlow

Popular machine learning library TensorFlow provides an adaptable and strong framework for building and training deep learning models, such as CNN and RNN. It is widely used for many different deep learning applications in both academic and industrial settings.

Keras

The open-source neural network library Keras provides an easy-to-use interface for creating, honing, and implementing deep learning models. It is clear that a lot of data science, machine learning, and artificial intelligence applications use it to quickly experiment and prototype neural networks.

cv2

The OpenCV open-source computer vision library provides a wide range of tools and algorithms for processing images and videos. The cv2 module is a Python interface for OpenCV. OpenCV offers functions for capturing, processing, analyzing, and manipulating images and videos. Many features are available, including machine learning, object recognition, feature detection, and image filtering and transformation.

Numpy

A well-known Python package for scientific data analysis and numerical computing is called NumPy. It provides many numerical functions to manipulate arrays and a multi-dimensional array data structure that is both powerful and highly efficient. NumPy is a vital library in the scientific Python community, widely used in data science, machine learning, and scientific computation because of its versatility.

Pandas

Pandas is a Python library that provides robust and versatile data manipulation and analysis capabilities. It offers a flexible and potent data structure known as DataFrame. Pandas is a popular tool for working with structured data in fields like data science and machine learning.

Matplotlib

A well-known Python package called Matplotlib is used to generate a wide range of visualizations, such as scatterplots, histograms, and static, animated, and interactive plots. It offers a full suite of tools that make making 2D and 3D visualizations easier. A popular library in data science, machine learning, and scientific research is called Matplotlib.

Chapter 3

Dataset Preprocessing

3.1 Dataset Preprocessing

3.1.1 Dataset Preparation:

The MuRA Dataset, which consists of radiographs of seven distinct body parts (Hand, Wrist, Forearm, Humerus, Shoulder, Finger, Elbow), was used to prepare the dataset. There are training and validation sets in the dataset. We extracted radiographs and their labels specifically for the Fingers in order to narrow down our focus. By concentrating on the finger, we hope to increase the model's precision and efficacy for this specific anatomical area.

3.1.2 Frame extraction:

To iterate over each row in the CSV file, use the pandas DataFrame's iterrows() method. The loop variable data represents a row for each iteration, and data['Path'] pulls the value containing the file path to the image from the row's 'Path' column.

Next, using the OpenCV library, the cv2.imread(data['Path']) function reads the image from the given file path. This makes it possible to load the picture into the variable image and process it further. After reading the associated image file path for each row in the DataFrame, the loop processes each image individually and applies the necessary operations to each one. A progress bar is provided by the tqdm function to monitor the processing of images during the loop.

3.1.3 Frame resizing:

The image to be resized (image in this case) and the desired size specified as a tuple (size, size), are the two arguments passed to the cv2.resize function. The target dimensions for the image's height and width are most likely represented by the size variable. To ensure uniformity in input dimensions for neural networks, the images in this particular case are resized to the specified dimensions (size, size) before being processed further.

3.1.4 CLAHE Pre-processing:

An enhanced form of Adaptive Histogram Equalization (AHE) [7] is Contrast Limited Adaptive Histogram Equalization (CLAHE) [8]; by restricting contrast enhancement in homogeneous regions, AHE's drawbacks are lessened. Using CLAHE, one can adjust an image's intensity values to increase contrast and improve detail visibility. In contrast to conventional histogram equalization, CLAHE divides the image into discrete areas known as tiles by adapting its contrast enhancement locally [9]. The histogram is equalized within each tile, but a clip limit is applied to avoid noise amplification. Because of this local adaptation, CLAHE can improve contrast without over-amplifying noise in both bright and dark areas of an image. It works especially well in medical imaging, where minute details in MRIs or X-rays can be very important.

Procedure:

3.1.4.1 Convert to LAB Color Space:

The input image is first converted from the RGB color space to the LAB (lightness, green to magenta, and blue to yellow) by the function. using cv2.cvtColor(image, cv2.COLOR_BGR2LAB) as the color space..

3.1.4.2 Split LAB Channels:

Next, using cv2.split(lab), the LAB image is divided into its three channels, producing distinct L, A, and B channels.

3.1.4.3 Apply CLAHE to the L Channel:

The L channel, which expresses the image's lightness, is the only area on which CLAHE is applied.Using parameters like clipLimit (to limit the contrast) and tileGridSize (to define the size of the contextual region for contrast enhancement), cv2.createCLAHE is used to create a CLAHE object. Next, clahe.apply(l) is used to apply the CLAHE object to the L channel.

3.1.4.4 Merge Channels:

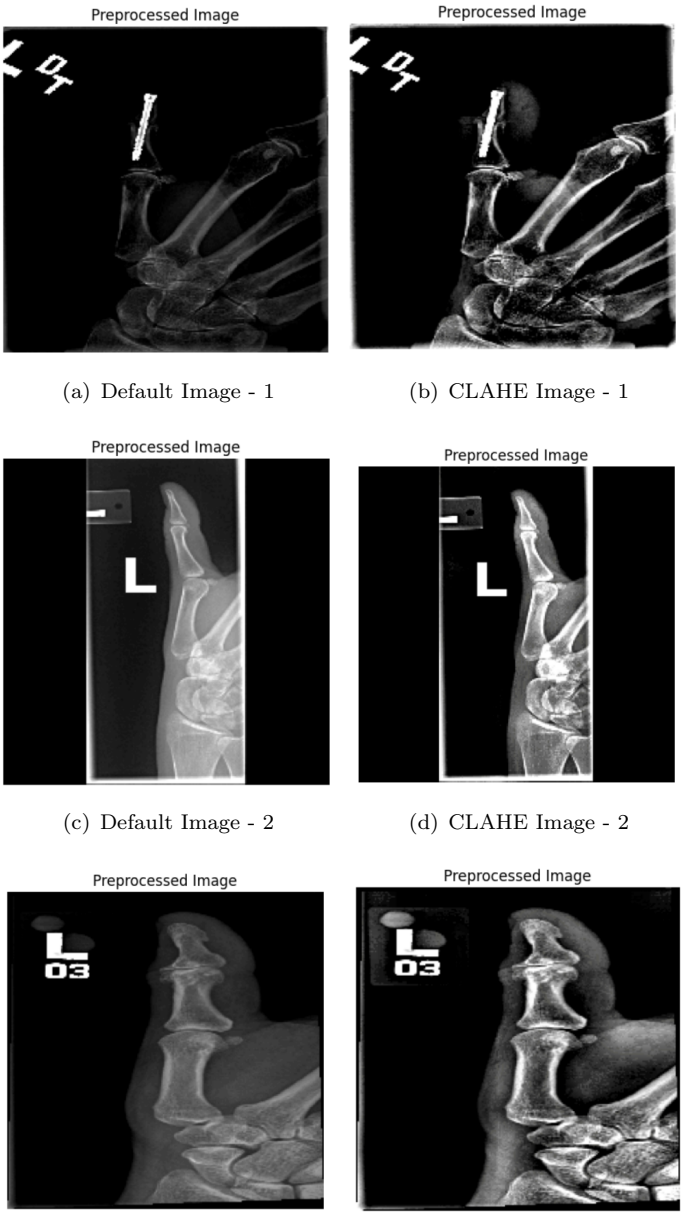
Using cv2.merge((L, A, B)), the processed L channel is merged back with the original A and B channels.

3.1.4.5 Convert Back to BGR Color Space:

After that, cv2.cvtColor(processed_image, cv2.COLOR_LAB2BGR) is used to convert the resulting LAB image with enhanced contrast back to the RGB color space.

The processed image is finally returned with enhanced local contrast. By applying CLAHE to the LAB color space's L channel, adaptive contrast enhancement is made possible while maintaining color information.

In the visual representation Figure 3.1 below, we aim to highlight the distinction between the default preprocessing of dataset images and those processed using the Contrast Limited Adaptive Histogram Equalization (CLAHE) technique. The images on the left showcase the dataset in its original form, subjected to standard preprocessing methods. On the right, the same set of images undergoes preprocessing using the CLAHE technique, emphasizing enhancements in contrast and improved visibility of subtle details.



(a) Default Image - 1 (b) CLAHE Image - 1

(c) Default Image - 2 (d) CLAHE Image - 2

(e) Default Image - 3 (f) CLAHE Image - 3

FIGURE 3.1: Comparision of images preprocessed with and without CLAHE technique

Chapter 4

Methodology

4.1 Proposed Design:

4.1.1 CNN-DenseNet Architechture

- **Convolutional Neural Network (CNN):** CNNs are a particular kind of neural network that are used to extract spatial features from images. They usually consist of several convolutional layers, which are followed by pooling layers to minimize the output's spatial dimensions. A collection of feature maps, each of which represents a distinct component of the input image, make up the CNN's output.
- **DenseNet Architecture:** DenseNet is renowned for its dense connectivity pattern, in which all layers are feedforwardly connected to all other layers as shown in the Figure 4.1. Improved gradient flow during training, parameter efficiency, and feature reuse are several advantages of this connectivity pattern.

The basic building block of DenseNet is called a "dense block," and it consists of a series of layers.

- **Input:** An image or other raw input data is sent to the input layer.
- **Initial Convolutional Layer:** To extract the initial features, a single convolutional layer with a small kernel size is commonly used.
- **Dense Blocks:** The main component of DenseNet is its dense blocks. Multiple densely connected layers make up each dense block. The output of every layer

within a dense block is concatenated with the inputs of every layer that follows it. Both feature reuse and information flow are aided by this dense connectivity.

- **Transition Layers:** Transition layers are used to reduce spatial dimensions and control the number of feature maps between dense blocks. A 1x1 convolutional layer for dimensionality reduction, average pooling, and batch normalization is frequently used in transition layers.
- **Output Layer:** The network's prediction is produced in the final output layer. This layer often uses softmax activation function for multi-class classification in classification tasks.

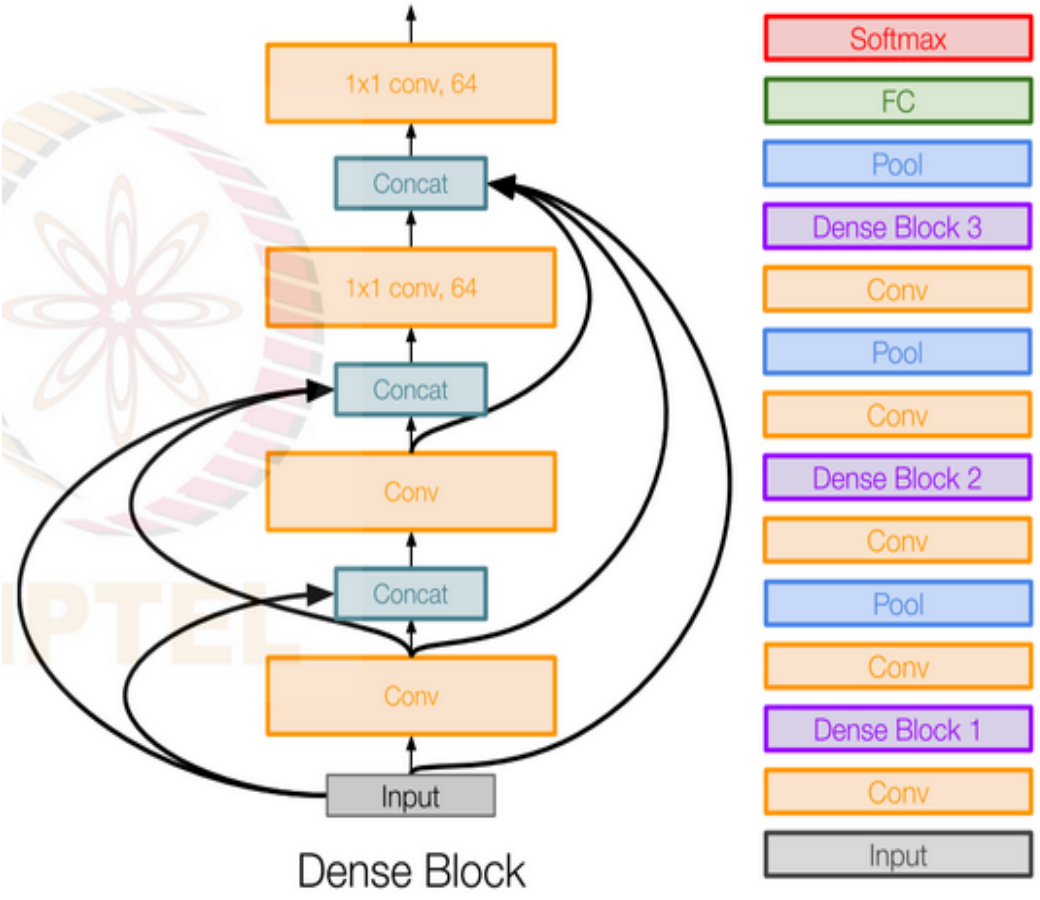


FIGURE 4.1: DenseNet Architecture

4.1.2 CNN-ResNet Architechture

- **ResNet Architecture:** [10] ResNet is well-known for using residual blocks, which solve the issue of vanishing gradients and make it easier to train very deep neural networks.
- The residual block serves as the fundamental building block of ResNet, and multiple layers of these blocks are stacked to create the overall architecture.
- One important breakthrough in ResNet is the use of skip connections, which enable the training of very deep networks with higher performance. The skip connections as shown in the Figure 4.2 and 4.3 make it easier for the gradient to flow through the network during both forward and backward passes. [11]

Here's a simplified representation of the ResNet architecture with respect to the number of layers as shown in the figure 4.4:

- **Input:** An image or other raw input data is sent to the input layer.
- **Initial Convolutional Layer:** The input image's initial convolution is handled by this layer.
- **Residual Stages:** ResNet is divided into stages, with several residual blocks in each stage. Each stage may contain a different number of blocks.
  - **Residual Block:** Two convolutional layers with batch normalization and ReLU activation functions form the basic residual block. By adding the input to the output of convolutional layers, a "shortcut" or "skip connection" is formed.
- **Global Average Pooling (GAP):** The final phase usually consists of global average pooling, which reduces the spatial dimensions of the feature maps to 1x1.
- **Fully Connected Layer:** After flattening the global average-pooled feature maps, a fully connected layer is often used for the final classification.

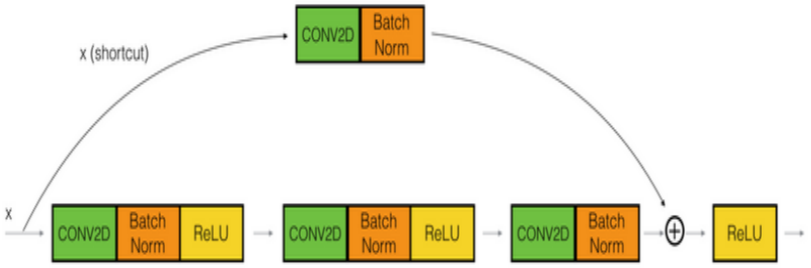


FIGURE 4.2: Convolutional Block

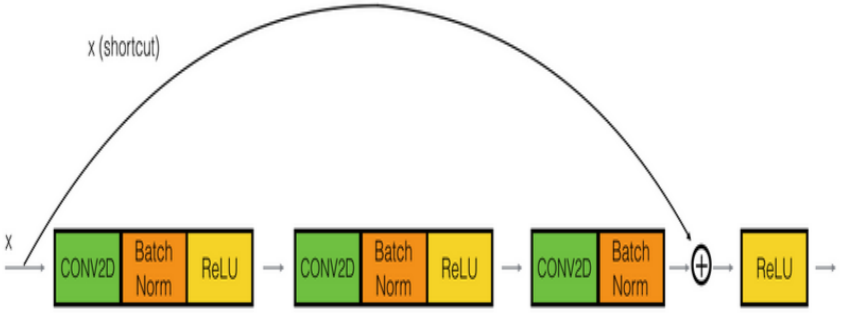


FIGURE 4.3: Identity Block, Skip connection "skips over" 3 layers

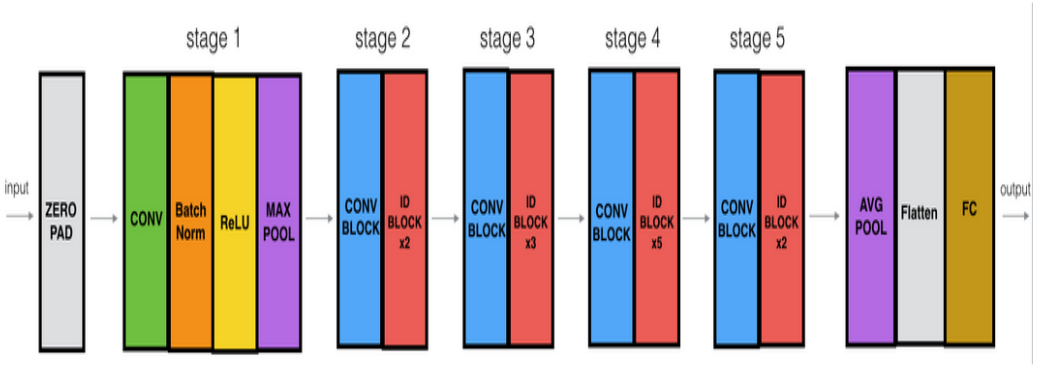


FIGURE 4.4: ResNet50 Architecture

4.1.3 CNN-Siamese Networks

- **Siamese Neural Networks :** A Siamese network is an artificial neural network that contains two identical sub-networks i.e. they have the same configuration with the same parameters and weights as shown in Figure 4.5.
- We will measure the distance between these two vectors and if the distance between these is small then the vectors are similar or of the same classes and if the distance between is larger then the vectors are different from one another, based on the score.
- In a Siamese triplet network, we suppose that x1 is the anchor input, x2 is a positive sample, and x3 is a negative sample. In similarity comparison, x1 and x2 are from the same category, while x3 belongs to a different category. We named x1 and x2 a positive pair, while x1 and x3 a negative pair. By using both positive pairs and negative pairs for training simultaneously, Siamese triplet network is able to learn discriminative features better.
- By connecting the output embeddings to a classifier layer, the Siamese network becomes versatile, enabling not only similarity evaluations but also effective classification.

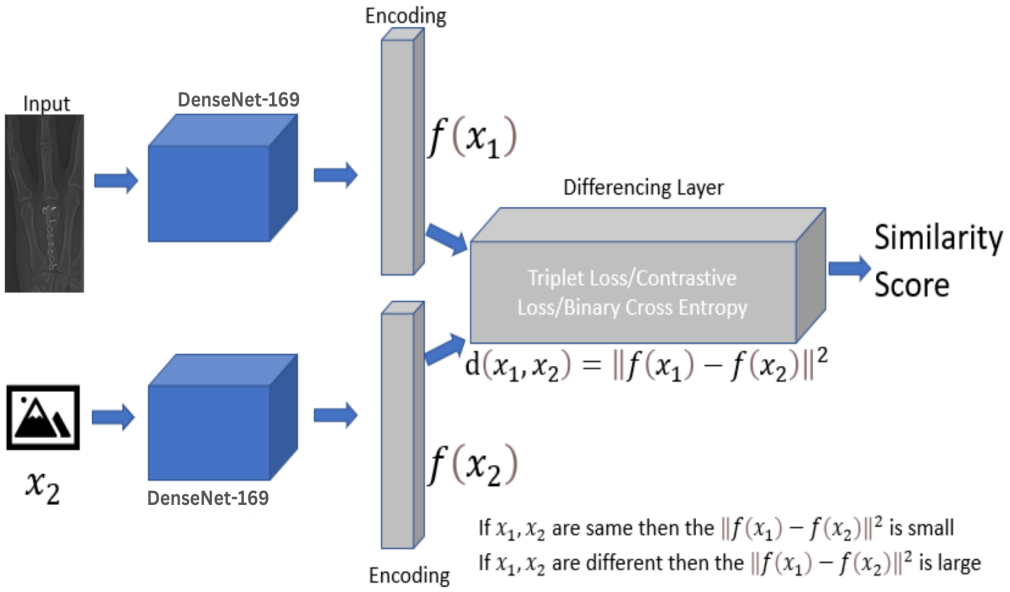


FIGURE 4.5: Siamese Network Architecture

The CNN-Siamese Network architecture for image classification can be summarized as follows:

- **Input:** This layer takes pairs of input images, representing the two images to be compared.
- **CNN Layers:**
  - **Description:** Using DenseNet-169/ResNet-50. These layers are responsible for extracting hierarchical features from the input images.
  - **Shared Weights:** The weights of these convolutional layers are shared between the twin networks, enforcing a degree of symmetry.
  - **Activation Function:** Common activation functions like ReLU (Rectified Linear Unit) are applied to introduce non-linearity.
- **Pooling Layers:** Pooling layers reduce the spatial dimensions of the feature maps, focusing on the most important features.
  - **Description:** The weights of these convolutional layers are shared between the twin networks, enforcing a degree of symmetry.
  - **Pooling Types:** Max pooling or average pooling is commonly used.
- **Flatten Layer:**
  - **Description:** This layer flattens the output from the convolutional and pooling layers into a one-dimensional vector.
  - **Output Shape:** The flattened vector serves as the input for the subsequent fully connected layers.
- **Fully Connected Layers:**
  - **Description:** These layers further process the features extracted by the convolutional layers.
  - **Activation Function:** ReLU or similar activation functions are often used.
- **Distance Calculation Layer:**
  - **Description:** This layer computes the distance or similarity score between the feature vectors produced by the twin networks.
  - **Euclidean Distance:** Commonly used for computing the distance between the feature vectors.

- **Output Layer:**
  - **Description:** The output layer produces a binary classification, indicating whether the input image pair is similar or dissimilar.
  - **Activation Function:** Sigmoid activation function is commonly used for binary classification.
- **Triplet Loss Function:** Triplet Loss takes an image triplet as input, which is called anchor sample, positive sample and negative sample respectively.

$$\mathcal{L}_{\text{Tri}}(\theta) = \sum_{a,p,n} \max(0, D_{a,p} - D_{a,n} + m) \quad \text{where } y_a = y_p \neq y_n$$

  When given an anchor sample x_a, above Eqn tries to make the distance in the embedding space between the anchor sample x_a and positive sample x_p which belong to the same identity closer than that of the anchor sample x_a and negative sample x_n which belong to different identities, by at least margin m.

4.1.4 Loss Functions

In traditional Triplet Loss, the initial step is to sample the training dataset and generate training triplets.[2]

- **BATCH HARD TRIPLET LOSS:**
  - The core idea of the Batch Hard Triplet Loss is to combine the triplet generation step with the training process and to mine the hard triplet samples within each mini-batch.
  - The mini-batch is organized as follows: each mini-batch is filled by randomly sampling P persons and then K instances was sampled for each identity, i.e. each min-batch is filled with P $\times$ K images.
  - Each image in the mini-batch is in turn treated as an anchor sample, Batch Hard Triplet Loss tries to choose the hardest positive and negative sample in the mini-batch.

$$\mathcal{L}_{\text{BHTL}}(\theta) = \sum_{i=1}^{P} \sum_{a=1}^{K} \max(0, D^*_{a,p} - D^*_{a,n} + m)$$

- **TRIPLET FOCAL LOSS:**
  - Focal Loss automatically down-weight the contribution of easy examples during training and rapidly focus the model on hard examples.
  - Triplet Focal Loss which can up-weight the hard triplets training samples and relatively down-weight the easy triplets adaptively
  - By mapping the original distance in the Euclidean space to an exponential kernel space, the hard triplets are penalized much more than the easy ones.
  - The formulation of Triplet Focal Loss based on the Batch Hard Triplet Loss is as follows,

$$\mathcal{L}_{\mathrm{TFL}}(\theta) = \sum_{i=1}^{P}\sum_{a=1}^{K} \max\left(0, \exp\left(\frac{D^*_{a,p}}{\sigma}\right) - \exp\left(\frac{D^*_{a,n}}{\sigma}\right) + m\right)$$

    where $D^*_{a,p}$ and $D^*_{a,n}$ denotes the hardest positive and negative sample corresponding to anchor sample x_a .

Chapter 5

Results and Discussion

5.1 Comparison between the models:

In our comparative analysis, we evaluate the performance of two widely used convolutional neural network architectures: DenseNet169 and ResNet50. The models are assessed using two different configurations, one with a 128-output layer and another with a 256-output layer. The primary metric under consideration include loss. Through experimentation and training, we present a comprehensive Table 5.1 detailing the achieved loss and validation loss values for both models and configurations.

Output-Layer	**ResNet50**		**DenseNet169**	
	Loss	**Val-loss**	**Loss**	**Val-loss**
256-Layer	0.206	0.201	0.0777	0.2072
128-Layer	0.1937	0.1858	0.0750	0.1911

TABLE 5.1: Loss of embedding models

In our analysis as shown in the Table 5.1, the DenseNet169 model exhibits lower loss compared to ResNet50, showcasing its superior performance in capturing intricate patterns within the data. Having successfully implemented the embedding model of a Siamese network, our next objective is to construct the classifier model.Utilize the calculated embeddings as input to a classifier model. Design a classifier architecture (e.g., fully connected layers) on top of the embeddings to perform the final classification task. Specifically, we aim to evaluate the classification accuracies of the DenseNet model from both 128 and 256 output layers.

5.1.1 Evaluation of Classifier Model Performance

Feed the above calculated embeddings into a classifier model, typically consisting of fully connected layers, for the final classification task. Assess the classifier model's performance by computing metrics like accuracy as shown in the Table 5.2.

Output-Layer	**ResNet50**		**DenseNet169**	
	Accuracy	**Val-acc**	**Accuracy**	**Val-acc**
256-Layer	0.635	0.496	0.929	0.722
128-Layer	0.678	0.668	0.946	0.744

TABLE 5.2: Accuracy of classifier models

Error:

The error rate is computed as the sum of non-equal elements divided by the total number of labels in the training and validation set. This is a common metric used in machine learning to assess the performance of a model, providing a measure of how well the model's predictions match the actual labels. The error rate is as shown in the Table 5.3

Output-Layer	**ResNet50**		**DenseNet169**	
	Train-error	**Valid-error**	**Train-error**	**Valid-error**
256-Layer	0.364	0.503	0.069	0.277
128-Layer	0.362	0.331	0.053	0.255

TABLE 5.3: Error in predictions

Cohen Kappa Score:

The Cohen's Kappa score is a statistical measure that assesses the level of agreement between predicted and true labels. The score ranges from -1 to 1, where 1 represents perfect agreement, 0 indicates agreement expected by chance, and negative values suggest less agreement than expected by chance. The formula for Cohen's Kappa involves the observed agreement and expected agreement, correcting for chance agreement. It's particularly useful in scenarios where the class distribution is imbalanced. The cohen kappa scores for the following models is calculated and is as shown in the Table 5.4

Output-Layer	**ResNet50**		**DenseNet169**	
	Train-score	**Valid-score**	**Train-score**	**Valid-score**
256-Layer	0.053	0.082	0.851	0.449
128-Layer	0.093	0.333	0.887	0.491

TABLE 5.4: Cohen Kappa Scores

5.1.2 CLAHE Pre-processing

We evaluated the Densenet169 model and observed enhanced performance with CLAHE preprocessing. The models are assessed using two different configurations, one with a 128-output layer and another with a 256-output layer. The losses are calculated and showed as in Table 5.5.

Output-Layer	**DenseNet169**		**DenseNet169-CLAHE**	
	Loss	**Val-loss**	**Loss**	**Val-loss**
256-Layer	0.0777	0.207	0.087	0.208
128-Layer	0.0750	0.1911	0.066	0.188

TABLE 5.5: Loss of embedding models

Classifier Model:

Feeding the above calculated embeddings into a classifier model, for the final classification task. Assess the classifier model's performance by computing metrics like accuracy as shown in the Table 5.6.

Output-Layer	**DenseNet169**		**DenseNet169-CLAHE**	
	Accuracy	**Val-acc**	**Accuracy**	**Val-acc**
256-Layer	0.929	0.722	0.9375	0.780
128-Layer	0.946	0.744	0.9508	0.752

TABLE 5.6: Accuracy of classifier models

Error:

The error rate is computed as the sum of non-equal elements divided by the total number of labels in the training and validation set. The error rate is as shown in the Table 5.7.

Output-Layer	**DenseNet169**		**DenseNet169-CLAHE**	
	Train-error	**Valid-error**	**Train-error**	**Valid-error**
256-Layer	0.069	0.277	0.062	0.249
128-Layer	0.053	0.255	0.048	0.264

TABLE 5.7: Error in predictions

Cohen Kappa Score:

The cohen kappa scores for the following models is calculated and is as shown in the Table 5.8.

Output-Layer	**DenseNet169**		**DenseNet169-CLAHE**	
	Train-score	**Valid-score**	**Train-score**	**Valid-score**
256-Layer	0.851	0.449	0.868	0.504
128-Layer	0.887	0.491	0.896	0.476

TABLE 5.8: Cohen Kappa Scores

The figures below visually represent the distribution of features within the learned embeddings, showcasing potential groupings or clusters that emerge from the dataset. This analysis provides insights into the effectiveness of CLAHE in enhancing the images. The figures are plotted showing two different configurations, one with a 128-output layer in the figure 5.2 and another with a 256-output layer in the Figure 5.1

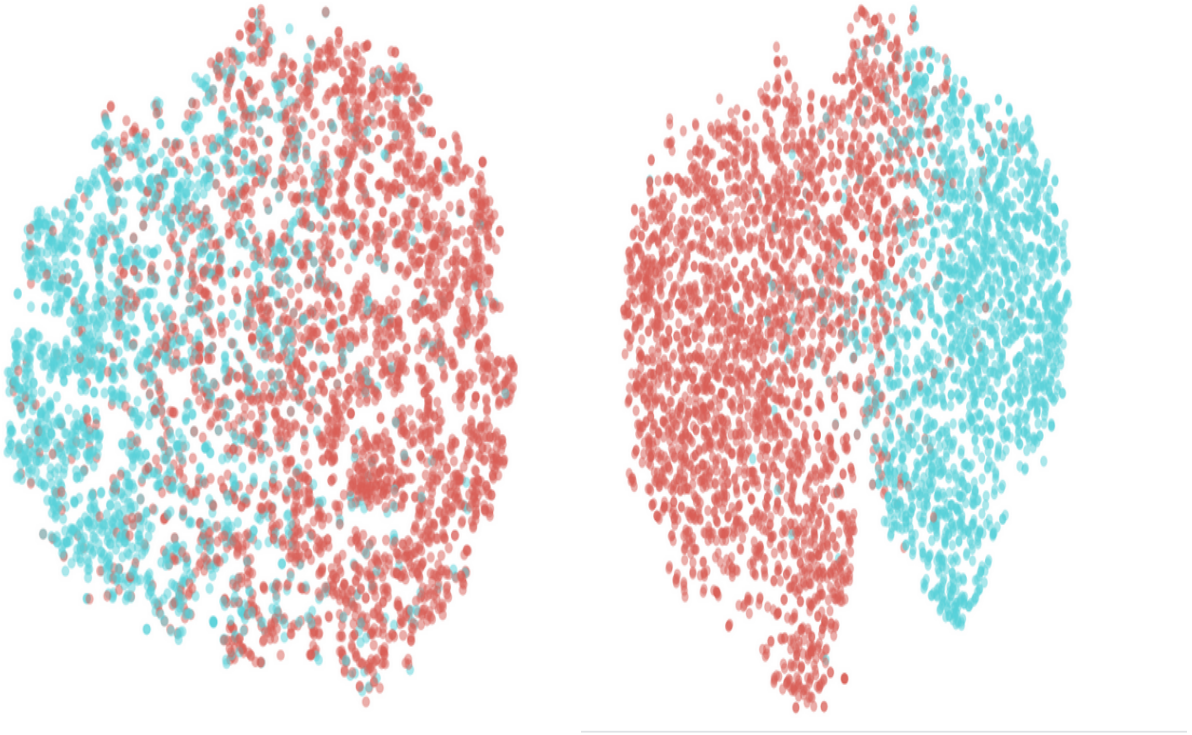


FIGURE 5.1: 256 output-layer DenseNet169 without and with CLAHE

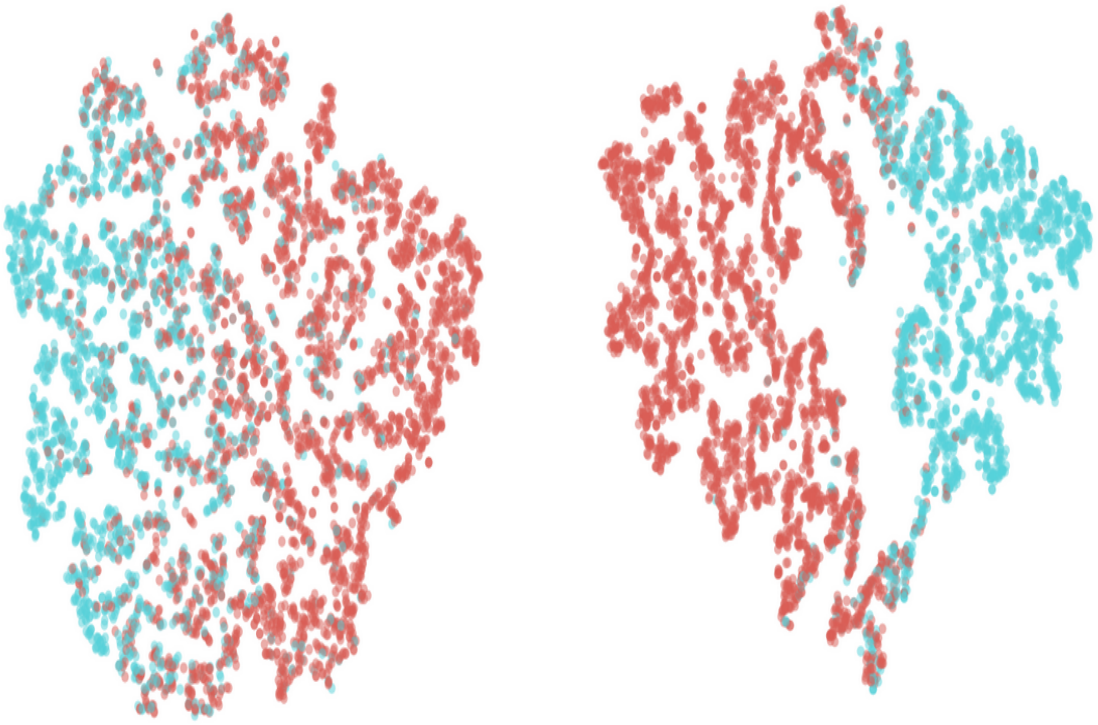


FIGURE 5.2: 128 output-layer DenseNet169 without and with CLAHE

5.1.3 Focal Triplet Loss

We evaluated the Densenet169 model using triplet loss and observed the performance when used focal triplet loss function. The models are assessed using two different configurations, one with a 128-output layer and another with a 256-output layer. The losses are calculated and showed as in Table 5.9.

Output-Layer	**DenseNet169**		**DenseNet169-Focal**	
	Loss	**Val-loss**	**Loss**	**Val-loss**
256-Layer	0.0777	0.207	0.020	0.0401
128-Layer	0.0750	0.1911	0.024	0.0402

TABLE 5.9: Loss of embedding models

Classifier Model:

Feeding the above calculated embeddings into a classifier model, for the final classification task. Assess the classifier model's performance by computing metrics like accuracy as shown in the Table 5.10.

Output-Layer	**DenseNet169**		**DenseNet169-Focal**	
	Accuracy	**Val-acc**	**Accuracy**	**Val-acc**
256-Layer	0.929	0.722	0.746	0.679
128-Layer	0.946	0.744	0.719	0.692

TABLE 5.10: Accuracy of classifier models

Error:

The error rate is computed as the sum of non-equal elements divided by the total number of labels in the training and validation set. The error rate is as shown in the Table 5.11.

Output-Layer	**DenseNet169**		**DenseNet169-Focal**	
	Train-error	**Valid-error**	**Train-error**	**Valid-error**
256-Layer	0.069	0.277	0.264	0.321
128-Layer	0.053	0.255	0.274	0.3210

TABLE 5.11: Error in predictions

Cohen Kappa Score:

The cohen kappa scores for the following models is calculated and is as shown in the Table 5.12.

Output-Layer	**DenseNet169**		**DenseNet169-Focal**	
	Train-score	**Valid-score**	**Train-score**	**Valid-score**
256-Layer	0.851	0.449	0.403	0.374
128-Layer	0.887	0.491	0.408	0.374

TABLE 5.12: Cohen Kappa Scores

5.2 Test Case - 01

In this test case, the image upload functionality is tested for a medical image analysis model that aims to classify images as either positive (abnormal) or negative (normal). The model provides a predictive score indicating the likelihood of abnormality in the uploaded image. Additionally, the test evaluates the generation of a Heatmap using Smooth Grad-CAM (Gradient-weighted Class Activation Mapping). The Heatmap visually highlights the regions of the image that significantly influence the model's decision, offering transparency and interpretability to the end-users by revealing the areas contributing to the positive or negative classification. In the below Figure 5.3, we can see that the image have been predicted as abnormal i.e positive and has shown the score and Heatmap highlighting the region of abnormality. And in the figure 5.4 the image given was predicted as normal with scores shown.

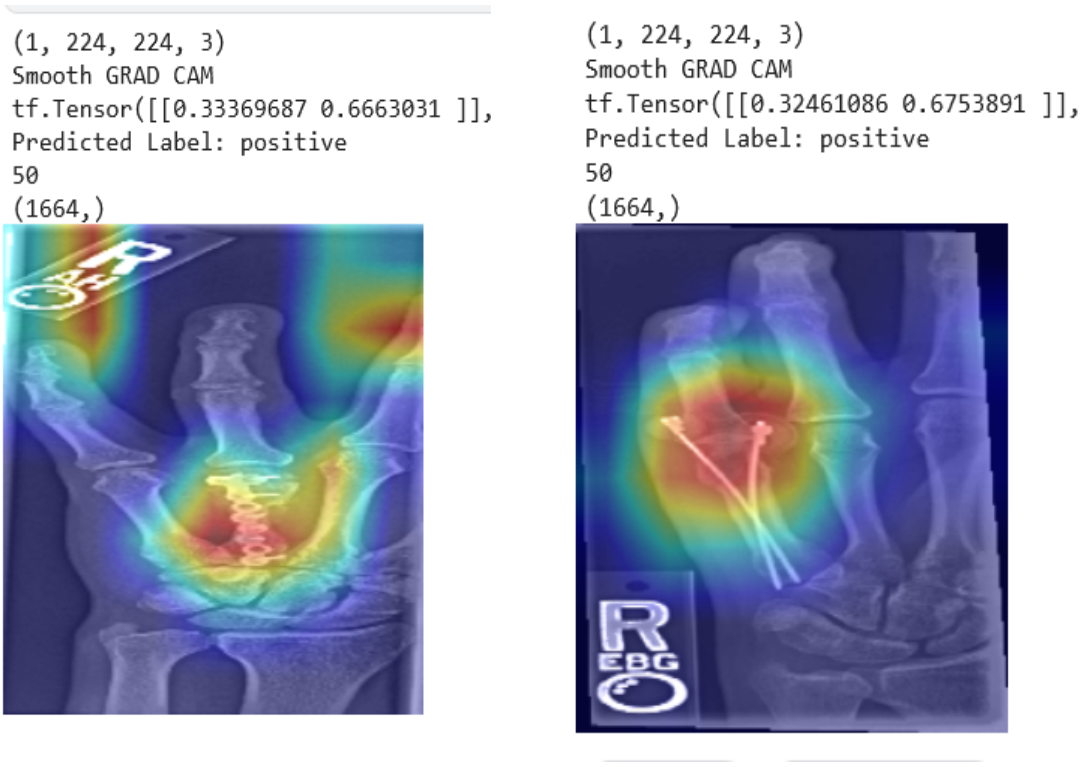


FIGURE 5.3: Test case images of class positive

FIGURE 5.4: Test case images of class negative

Chapter 6

Summary

The detection of musculoskeletal abnormalities is especially crucial because musculoskeletal disorders account for the majority of disabilities and severe, chronic pain, affecting over 1.7 billion people globally and resulting in over 30 million ER visits each year.

Finding abnormalities in musculoskeletal radiographs has significant therapeutic implications. Worklist prioritization could be achieved by using an abnormality detection model. In this case, the abnormally detected studies could be advanced in the workflow of image interpretation, enabling prompt diagnosis and treatment for the sickest patients.

This Project has been a great learning experience. This was for the first time we worked on Image Analysis. It helped us gain knowledge and bridge the gap between theoretical knowledge and practical knowledge.

Bibliography

[1] P. Rajpurkar, J. Irvin, A. Bagul, D. Ding, T. Duan, H. Mehta, B. Yang, K. Zhu, D. Laird, R. L. Ball, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, "Mura: Large dataset for abnormality detection in musculoskeletal radiographs," 2018.

[2] S. Zhang, Q. Zhang, X. Wei, Y. Zhang, and Y. Xia, "Person re-identification with triplet focal loss," *IEEE Access*, vol. 6, pp. 78 092–78 099, 2018.

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[4] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification." *Journal of machine learning research*, vol. 10, no. 2, 2009.

[5] H. Xuan, A. Stylianou, X. Liu, and R. Pless, "Hard negative examples are hard, but useful," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer, 2020, pp. 126–142.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[7] W. Yussof, M. S. Hitam, E. A. Awalludin, and Z. Bachok, "Performing contrast limited adaptive histogram equalization technique on combined color models for underwater image enhancement," *International Journal of Interactive Digital Media*, vol. 1, no. 1, pp. 1–6, 2013.

[8] R. D. P. Olvera, E. M. Zeron, J. C. P. Ortega, J. M. R. Arreguin, and E. G. Hurtado, "A feature extraction using sift with a preprocessing by adding clahe algorithm to enhance image histograms," in *2014 International Conference on Mechatronics, Electronics and Automotive Engineering*. IEEE, 2014, pp. 20–25.

[9] T. P. H. Nguyen, Z. Cai, K. Nguyen, S. Keth, N. Shen, and M. Park, "Pre-processing image using brightening, clahe and retinex," *arXiv preprint arXiv:2003.10822*, 2020.

[10] A. Dharma, J. Sitorus, and A. Hatigoran, "Comparison of residual network-50 and convolutional neural network conventional architecture for fruit image classification," *SinkrOn*, vol. 8, pp. 1863–1874, 07 2023.

[11] Y. Liu, C. Chaojun, Y. Che, R. Ke, Y. Ma, and Y. Ma, "Detection of maize tassels from uav rgb imagery with faster r-cnn," *Remote Sensing*, vol. 12, p. 338, 01 2020.

[12] G. Mehr, "Automating abnormality detection in musculoskeletal radiographs through deep learning," 2020.

[13] M. He, X. Wang, and Y. Zhao, "A calibrated deep learning ensemble for abnormality detection in musculoskeletal radiographs," *Scientific Reports*, vol. 11, no. 1, p. 9097, 2021.