MURA Multiclass Classification with Transfer Learning and Explainability Maps

DATA 255: Deep Learning,
Professor Dr. Mohammad Masum

Team 7,

Nikhil Thota
Pavan Kumar Reddy K
Hema Sai Venkata Dumpala
Irfan Shaik

Background & Introduction

- Musculoskeletal disorders are among the leading causes of disability worldwide, affecting millions and leading to significant healthcare challenges.
- The deaths caused by in 2020, due to the Musculoskeletal disorders is around ~90k, and more than ~220k injuries world wide.
- Early and accurate detection of musculoskeletal abnormalities can significantly impact treatment outcomes, reducing the need for invasive procedures and improving quality of life.
- MURA (Multiskeleton RadioGraphs) is a dataset proposed in an open Competition conducted by Stanford ML community to support the deep learning studies on radiographs.
- The dataset was created to foster advancements in artificial intelligence (AI) applications in medical imaging, aiming to enhance the accuracy and efficiency of diagnostics in musculoskeletal radiology.
- XAI aims to make the results of AI systems more transparent and understandable to human experts, which is essential in healthcare settings, and in while making sensitive decisions.

MURA - Musculoskeletal RadioGraphs



- MURA dataset was introduced in the work of Pranav Rajpurkar et al. (2018) titled “MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs”, also co-authored by Andrew NG.
- MURA consists of 14,863 musculoskeletal studies, totaling 40,561 images, each study offering one or more views, primarily focusing on the upper extremity, including the **hand, wrist, forearm, elbow, humerus, finger and shoulder.**
- The training, and validation splits are divided, and the test split is hidden due to the integrity of the test.

```
Training Data Sample:
                                                  path  abnormality body_part  \
0  MURA-v1.1/train/XR_SHOULDER/patient00001/study...            1  SHOULDER
1  MURA-v1.1/train/XR_SHOULDER/patient00002/study...            1  SHOULDER
2  MURA-v1.1/train/XR_SHOULDER/patient00003/study...            1  SHOULDER
3  MURA-v1.1/train/XR_SHOULDER/patient00004/study...            1  SHOULDER
4  MURA-v1.1/train/XR_SHOULDER/patient00005/study...            1  SHOULDER

              label
0  Abnormal-SHOULDER
1  Abnormal-SHOULDER
2  Abnormal-SHOULDER
3  Abnormal-SHOULDER
4  Abnormal-SHOULDER

Validation Data Sample:
                                                  path  abnormality body_part  \
0  MURA-v1.1/valid/XR_WRIST/patient11185/study1_p...            1     WRIST
1  MURA-v1.1/valid/XR_WRIST/patient11186/study1_p...            1     WRIST
2  MURA-v1.1/valid/XR_WRIST/patient11186/study2_p...            1     WRIST
3  MURA-v1.1/valid/XR_WRIST/patient11186/study3_p...            1     WRIST
4  MURA-v1.1/valid/XR_WRIST/patient11187/study1_p...            1     WRIST

            label
0  Abnormal-WRIST
1  Abnormal-WRIST
2  Abnormal-WRIST
3  Abnormal-WRIST
4  Abnormal-WRIST
```

Fig: Metadata of the MURA

Study	Type	Train	Validation	Total	% of total
Elbow	Normal	1094	92	1186	8.09%
Elbow	Abnormal	660	66	726	4.95%
Finger	Normal	1280	92	1372	9.36%
Finger	Abnormal	655	83	738	5.04%
Hand	Normal	1497	101	1598	10.90%
Hand	Abnormal	521	66	587	4.01%
Humerus	Normal	321	68	389	2.65%
Humerus	Abnormal	271	67	338	2.31%
Forearm	Normal	590	69	659	4.50%
Forearm	Abnormal	287	64	351	2.39%
Shoulder	Normal	1364	99	1463	9.98%
Shoulder	Abnormal	1457	95	1552	10.59%
Wrist	Normal	2134	140	2274	15.52%
Wrist	Abnormal	1326	97	1423	9.71%
Total	**Normal**	**8280**	**661**	**8941**	**61.01%**
Total	**Abnormal**	**5177**	**538**	**5715**	**38.99%**
Total	**No. of Studies**	**13457**	**1199**	**14656**	**100.00%**

Fig: Distribution of Studies across the original dataset

Problem Statement & Objectives

The timely and accurate diagnosis of musculoskeletal abnormalities is crucial for effective treatment and improved patient outcomes, yet it remains a complex challenge in medical imaging due to the intricate nature of musculoskeletal injuries and significant anatomical variability among individuals.

This project utilizes the extensive MURA dataset to develop a sophisticated deep learning model for multiclass classification, accurately determining both the specific body part and the presence of abnormalities. By integrating advanced explainable AI techniques, we aim to improve the model's transparency and provide clinicians with clear, interpretable insights for better diagnostic accuracy.

1. Convert the Original MURA dataset into a Multiclass classification compatible dataset, by restructuring the image paths, and re-creating the annotations.
2. Use Transfer Learning, to finetune a Deep Learning Model on the created dataset.
3. Validate the results, along with hyperparameter tuning, and predictions on the test dataset.
4. Add Explanability to let everybody else understand the reason behind the decision.
5. Predict the results, and generate explanations for the results on test dataset.
6. Document the whole Process diligently, to avoid any hiccups in the project execution, and plan.

MURA for MultiClass Classification



- Converting the MURA dataset into a Multiclass classification compatible dataset, meaning creating the new labels - <abnormal or not>_<bodypart> is key to this Project, utilizing metadata files (image paths) for train, and valid.
- Exploded the images into a new folder, by converting the granularity of the problem from Studies to Images, and utilized the metadata files to segregate them into respective train, valid and test datasets, along with the new metadata files for each dataset.

```
  1%|          | 17/3197 [00:00<00:19, 165.39it/s]Full path: valid/XR_WRIST/patient11185/study1_positive/image1.png
Body part extracted: WRIST
Study type segment in path: study1_positive
Moved and renamed: ./valid/XR_WRIST/patient11185/study1_positive/image1.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_1.png
Full path: valid/XR_WRIST/patient11185/study1_positive/image2.png
Body part extracted: WRIST
Study type segment in path: study1_positive
Moved and renamed: ./valid/XR_WRIST/patient11185/study1_positive/image2.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_2.png
Full path: valid/XR_WRIST/patient11185/study1_positive/image3.png
Body part extracted: WRIST
Study type segment in path: study1_positive
Moved and renamed: ./valid/XR_WRIST/patient11185/study1_positive/image3.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_3.png
Full path: valid/XR_WRIST/patient11185/study1_positive/image4.png
Body part extracted: WRIST
Study type segment in path: study1_positive
Moved and renamed: ./valid/XR_WRIST/patient11185/study1_positive/image4.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_4.png
Full path: valid/XR_WRIST/patient11186/study1_positive/image1.png
Body part extracted: WRIST
Study type segment in path: study1_positive
Moved and renamed: ./valid/XR_WRIST/patient11186/study1_positive/image1.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_5.png
Full path: valid/XR_WRIST/patient11186/study1_positive/image2.png
Body part extracted: WRIST
Study type segment in path: study1_positive
Moved and renamed: ./valid/XR_WRIST/patient11186/study1_positive/image2.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_6.png
Full path: valid/XR_WRIST/patient11186/study2_positive/image1.png
Body part extracted: WRIST
Study type segment in path: study2_positive
Moved and renamed: ./valid/XR_WRIST/patient11186/study2_positive/image1.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_7.png
Full path: valid/XR_WRIST/patient11186/study2_positive/image2.png
Body part extracted: WRIST
Study type segment in path: study2_positive
Moved and renamed: ./valid/XR_WRIST/patient11186/study2_positive/image2.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_8.png
Full path: valid/XR_WRIST/patient11186/study2_positive/image3.png
Body part extracted: WRIST
Study type segment in path: study2_positive
Moved and renamed: ./valid/XR_WRIST/patient11186/study2_positive/image3.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_9.png
Full path: valid/XR_WRIST/patient11186/study3_positive/image1.png
Body part extracted: WRIST
Study type segment in path: study3_positive
Moved and renamed: ./valid/XR_WRIST/patient11186/study3_positive/image1.png to multiclass_valid/Abnormal-WRIST/abnormal_wrist_10.png
Full path: valid/XR_WRIST/patient11186/study3_positive/image2.png
Body part extracted: WRIST
```

Fig: Logs while creating the Multiclass dataset

	path	Body_part	abnormality	bodypart_abnormality
0	multiclass_train/Abnormal-ELBOW/abnormal_elbow...	Abnormal	ELBOW	abnormal_elbow
1	multiclass_train/Abnormal-ELBOW/abnormal_elbow...	Abnormal	ELBOW	abnormal_elbow
2	multiclass_train/Abnormal-ELBOW/abnormal_elbow...	Abnormal	ELBOW	abnormal_elbow
3	multiclass_train/Abnormal-ELBOW/abnormal_elbow...	Abnormal	ELBOW	abnormal_elbow
4	multiclass_train/Abnormal-ELBOW/abnormal_elbow...	Abnormal	ELBOW	abnormal_elbow
...	...	...	...	...
31276	multiclass_train/Normal-WRIST/normal_wrist_993...	Normal	WRIST	normal_wrist
31277	multiclass_train/Normal-WRIST/normal_wrist_994...	Normal	WRIST	normal_wrist
31278	multiclass_train/Normal-WRIST/normal_wrist_995...	Normal	WRIST	normal_wrist
31279	multiclass_train/Normal-WRIST/normal_wrist_996...	Normal	WRIST	normal_wrist
31280	multiclass_train/Normal-WRIST/normal_wrist_999...	Normal	WRIST	normal_wrist

31281 rows × 4 columns

Fig: Updated Metadata file for training dataset

MURA for MultiClass Classification

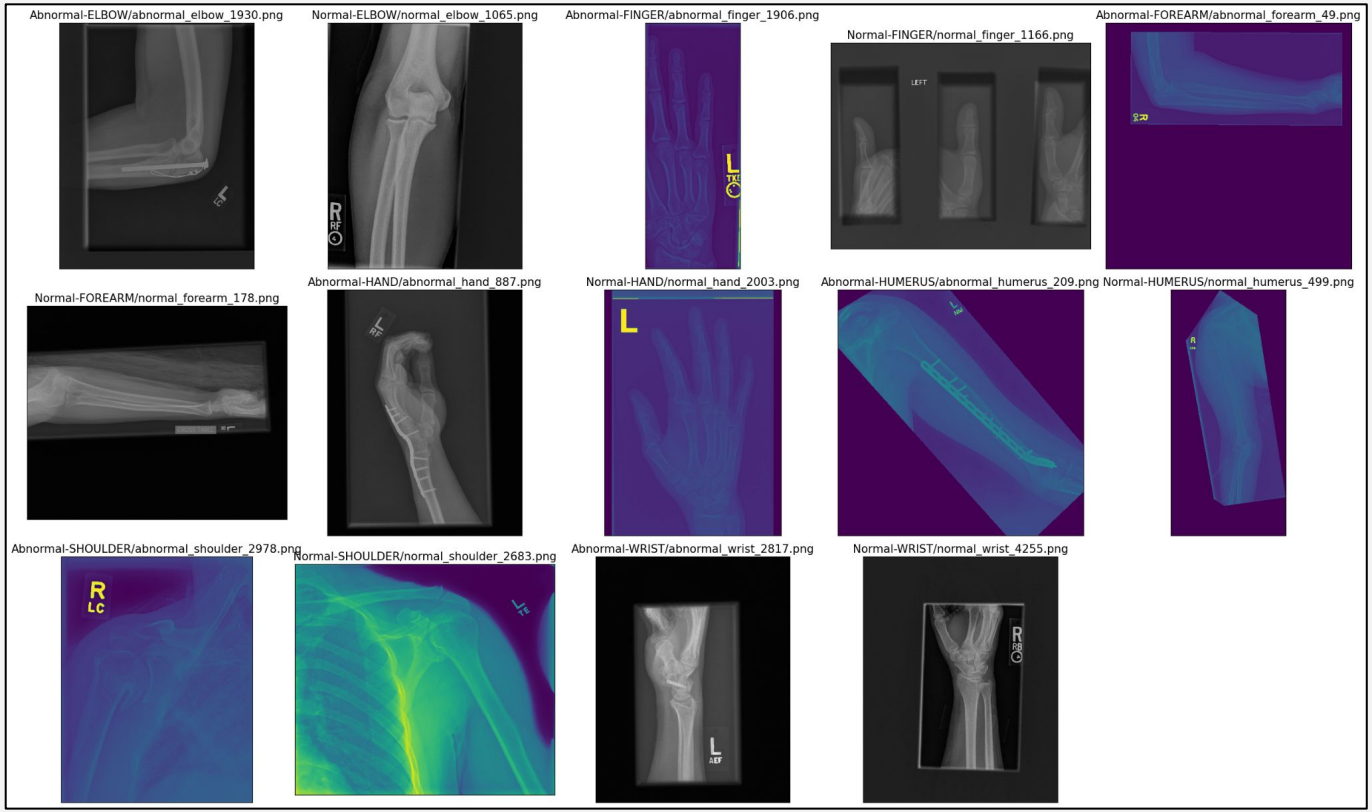
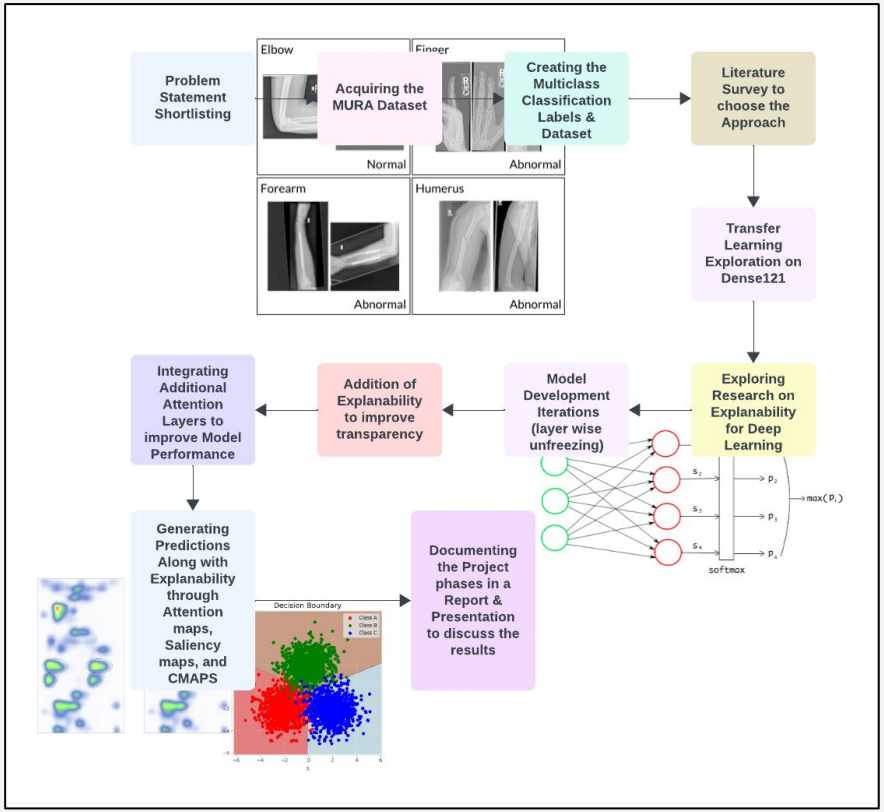
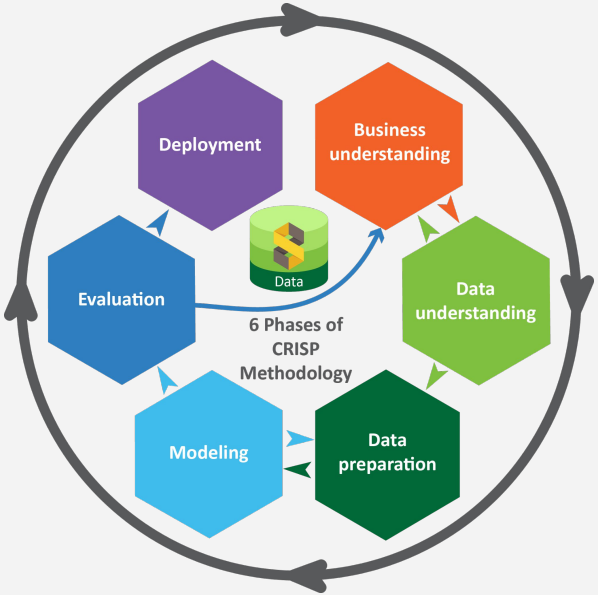


Fig: Examples of Images after creating the MultiClass Classification dataset from the Original MURA Dataset

Methodology





Adapting Crisp-DM to this project, the given project plan has ensured in timely completion of different phases of the Project, while meeting the goals and objectives

Literature Review

S No	Paper Title	Authors	Description
1	MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs	Pranav Rajpurkar, Andrew Y. Ng, et al. (2018)	Introduced the MURA dataset, also created the benchmark of Kappa's statistic of 0.77 on test dataset
2	Exploring Large-scale Public Medical Image Datasets	Luke Oakden-Rayner et al. (2020)	Found that label accuracy and the process of dataset creation critically affect the utility of datasets in training AI models by studying MURA, Chest14
3	Classification of Shoulder X-ray Images with Deep Learning Ensemble Models	Fatih Uysal, Fırat Hardalaç, Ozan Peker, Tolga Tolunay, Nil Tokgöz (2021)	Highlighted the superior performance of ensemble models combining ResNet, DenseNet, and other architectures in achieving high diagnostic accuracy and reliability in fracture detection.
4	Radiography Classification: A Comparison between Eleven Convolutional Neural Networks	Ananda, Cefa Karabağ et al. (2020)	Investigates the classification of normal and abnormal radiographic images using eleven different CNN architectures on the MURA dataset.
5	Utilizing heat maps as explainable artificial intelligence for detecting abnormalities on wrist and elbow radiographs	S. Lysdahlgaard	The paper emphasizes the application of 20 transfer-learning models, highlighting the effectiveness of XAI in improving diagnostic transparency and accuracy.

Data Statistics - I

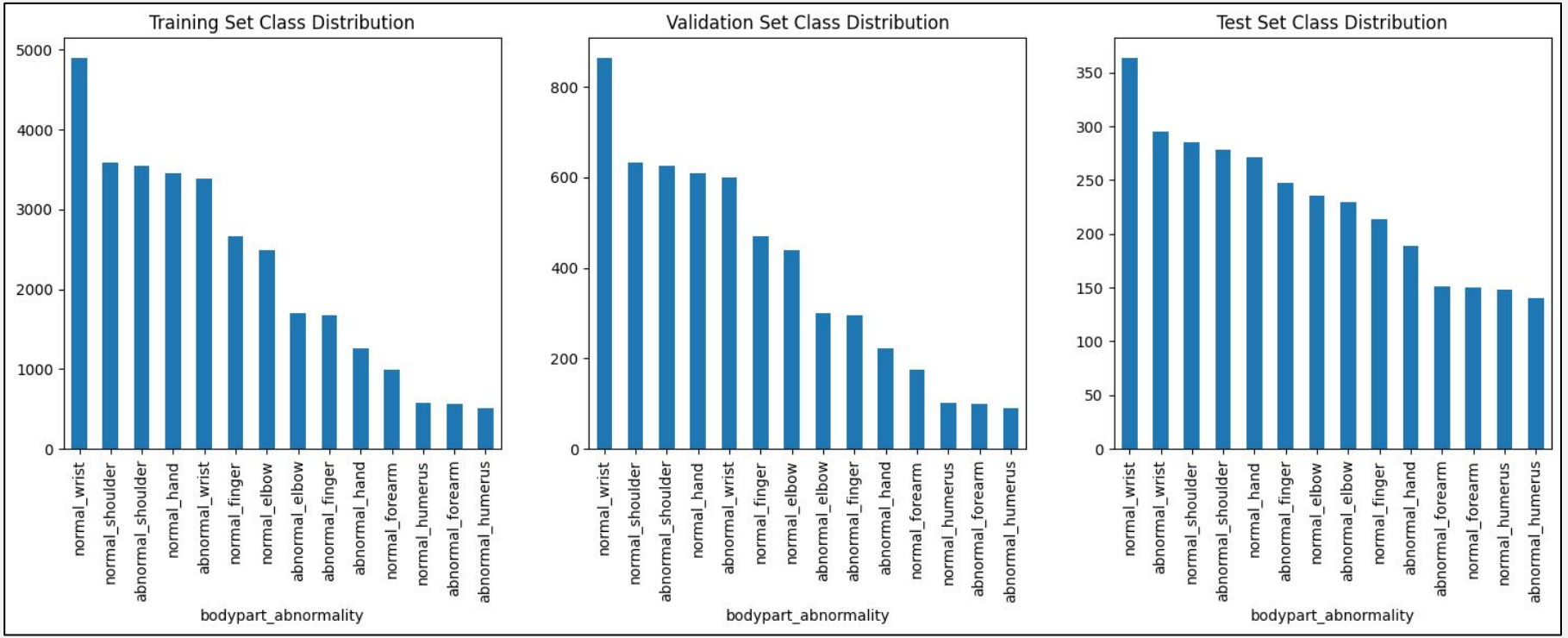
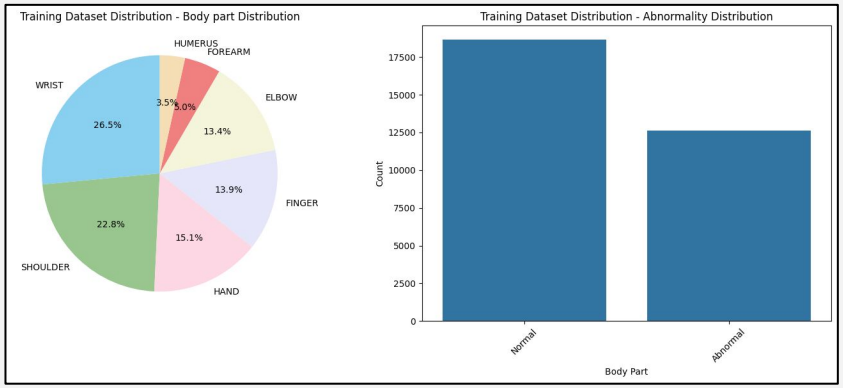
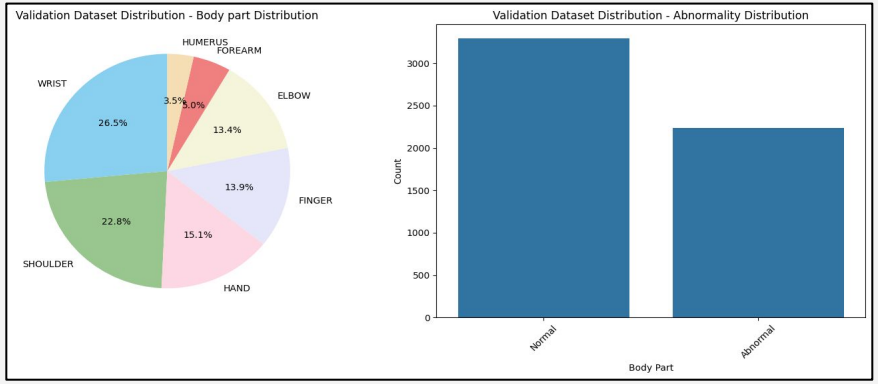
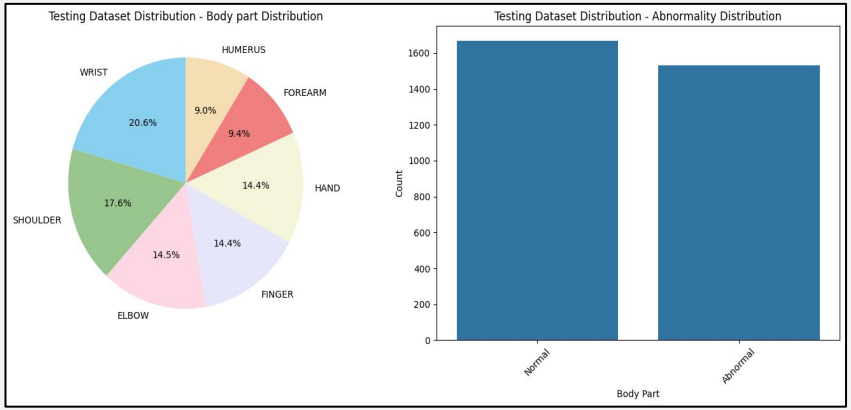


Fig: Stratified Splitting to create the Training, Validation and Test Datasets for Multi Class Classification

Data Statistics - II

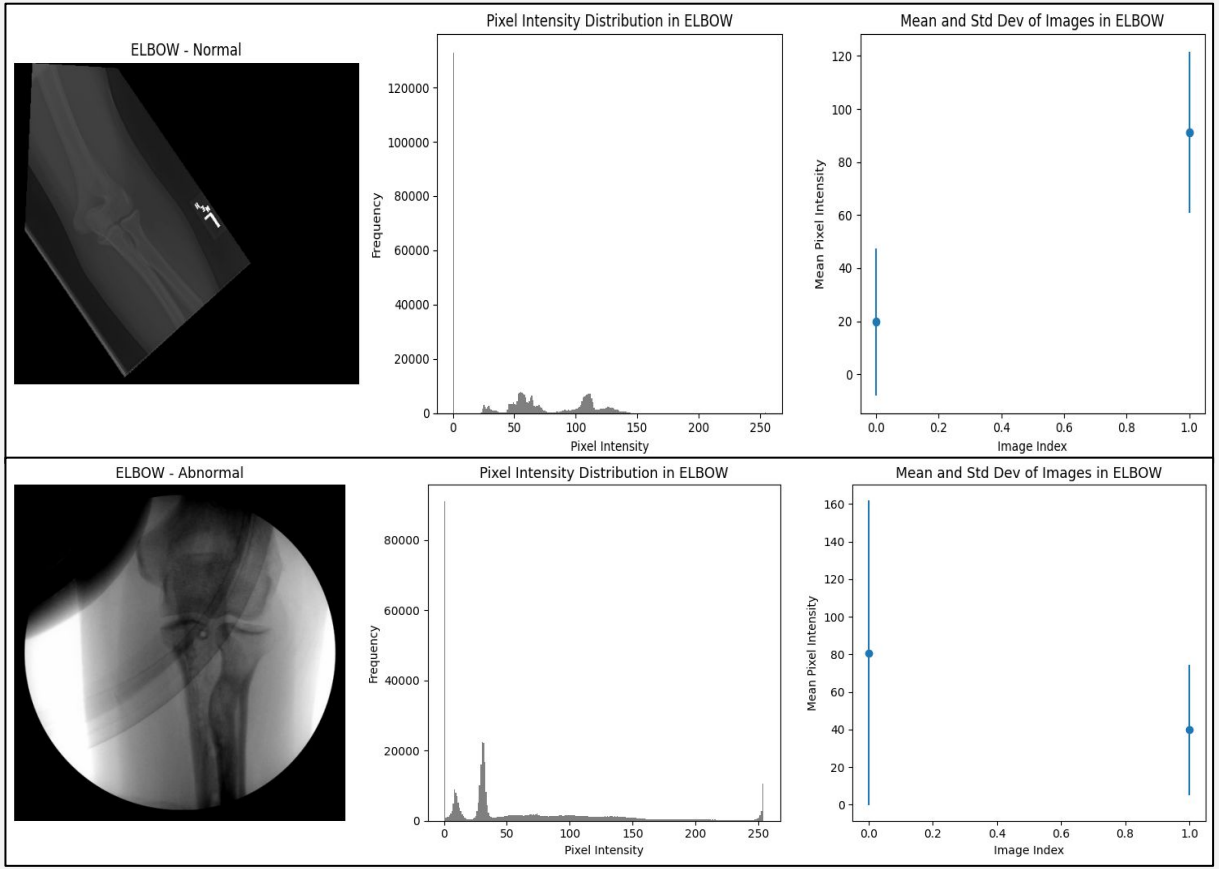






- Due to the Stratified Splitting criteria, which was utilized while splitting the datasets from the reconstructed MURA Multiclass, it is evident that the distribution of the Class Labels is even across all the classes, across all the splits.
- It is observed that the number of samples in HUMERUS, FOREARM are significantly less when compared with the rest of the dataset, SMOTE techniques will come in play, in the training processes

Exploratory Data Analysis - I

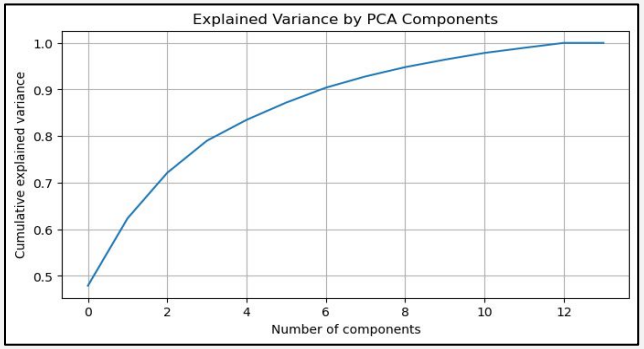


For the abnormal elbow, there are significant peaks at the lower intensities, indicating more dark regions (possibly representing more open space within the image or less dense tissue). The normal elbow, however, shows a more even distribution but still concentrated in lower to mid-range intensities, typical of normal bone images in X-rays.

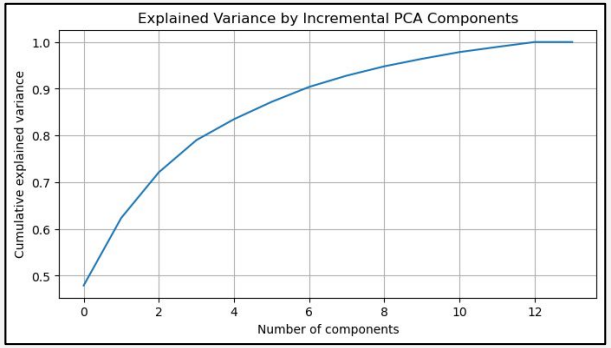
Significant variability in one of the indices for both abnormal and normal categories. This could indicate that some images are much darker or brighter than others, possibly due to different imaging conditions or the severity of the conditions depicted.

Standardizing the image data (e.g., mean subtraction and division by the standard deviation) based on the observed mean and standard deviation can also help in stabilizing the training process.

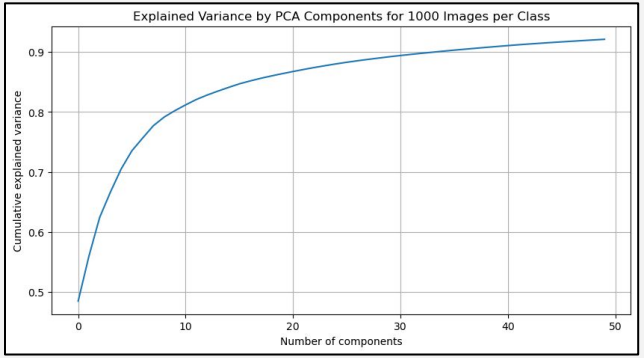
EDA II - Comparison of PCA and Incremental PCA



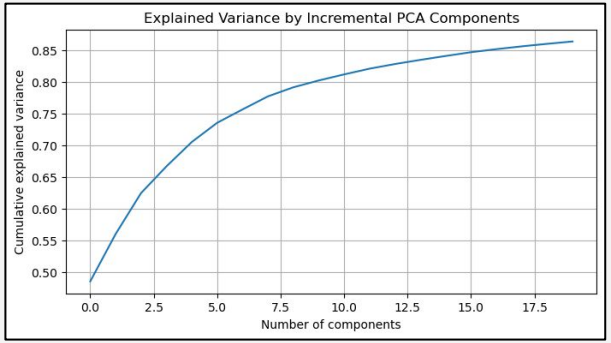
PCA with 10 images from each class - 1400 images, 8 components capture 95% variance



iPCA with 10 images from each class - 1400 images, 11 components capture 95% variance



PCA with 1000 images from each class - 14000 images, 43 components capture 95% variance



iPCA with 1000 images from each class - 14000 images, 19 components capture 95% variance

Neural Networks: DenseNet

- Dense Convolutional Network (DenseNet) is an innovative convolutional neural network (CNN) architecture, introduced by Huang et al. (2018), that is particularly well-suited for image classification tasks, having demonstrated significant improvements over previous architectures such as VGG and ResNet, particularly in terms of efficiency and depth.
- Unlike traditional CNN's where each layer receives input from only the previous layer and sends output to the next layer, DenseNet connects each layer to every other layer in a **feed-forward fashion** (the output of each layer is used as input to all subsequent layers).

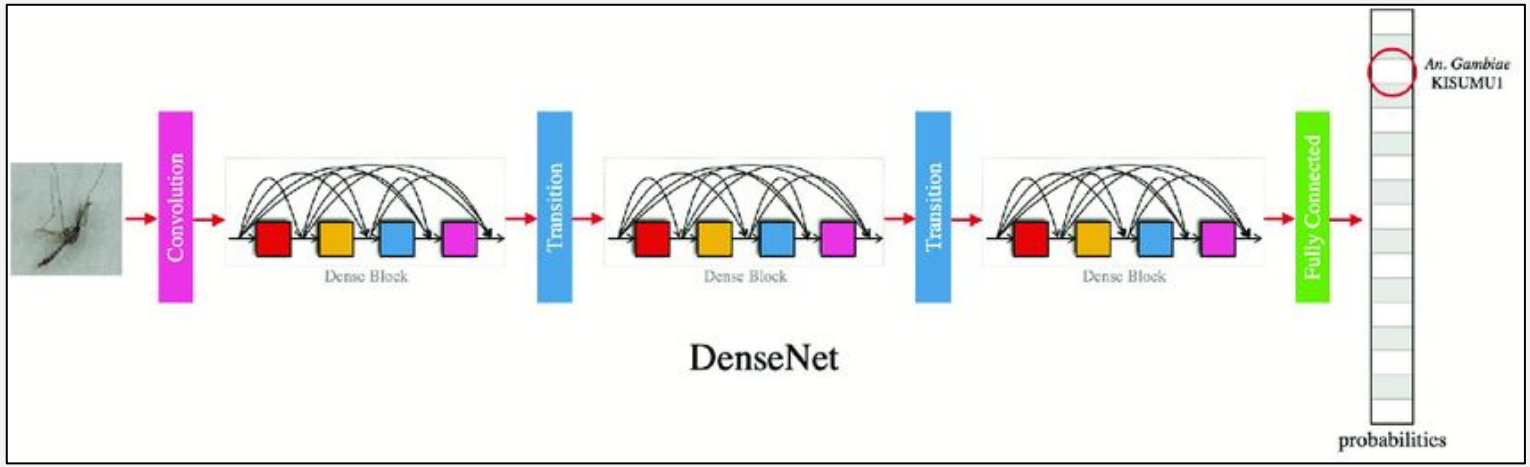
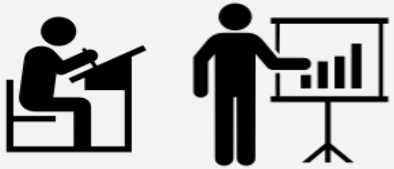


Fig: DenseNet Architecture for Image Classification

Dense Neural Networks (DenseNet)



- Each layer receives inputs from all preceding layers and passes its own feature-maps to all subsequent layers. This ensures maximum information flow between layers in the network.
- DenseNet introduced the concept of a **growth rate**, denoted by 'k', which controls how much **new information** each layer contributes. The network remains compact as each layer adds only a small set of feature-maps to the collective knowledge.
- Implements a **1x1** convolution before a **3x3** convolution to reduce the number of input feature-maps, lessening computational load.
- Situated between dense blocks, these layers use **batch normalization, a 1x1 convolution, and a 2x2 average pooling** to reduce feature-map size and manage network capacity.
- Allows every layer to access feature-maps from all previous layers, reducing redundancy and the number of necessary parameters.

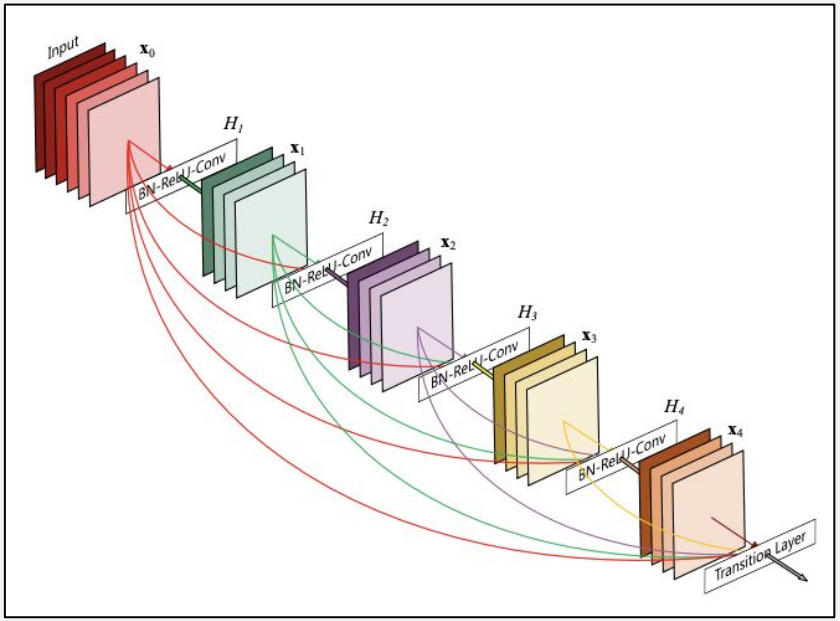
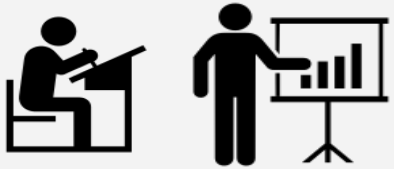


Fig: A 5-layer dense block with a growth rate of k=4. Each layer takes all preceding feature-maps as input

Dense121



- DenseNet-121 utilizes four dense blocks where each layer connects directly to all subsequent layers via feature-map concatenation, enhancing feature reuse and reducing redundancy. These blocks consist of layers with batch normalization (BN), rectified linear unit (ReLU), and a 3x3 convolution (Conv).
- Transition layers between these blocks, comprising BN, a 1x1 convolutional bottleneck, and 2x2 average pooling, help manage capacity and reduce dimensionality.
- Bottleneck layers, positioned before each 3x3 convolution in dense blocks, minimize the input feature-maps for efficiency.
- The network concludes with a global average pooling layer that reduces each feature-map to a single number, followed by a softmax classifier that outputs the probabilities for the target classes.

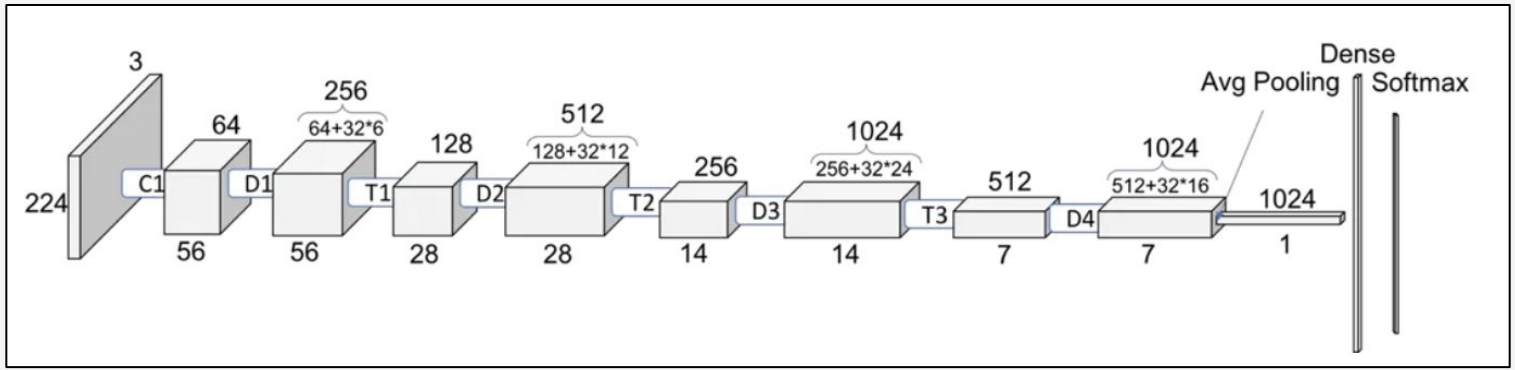
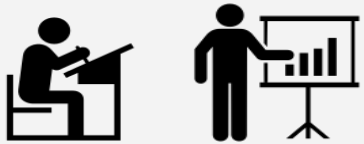


Fig: Dense121 Architecture

Modeling - Data Preparation



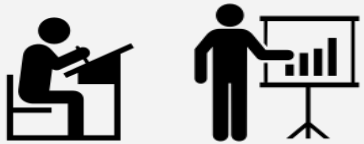
```python
def create_train_generator(data_directory, target_size=(224, 224), batch_size=32):
    """Create a training data generator that fetches images from subdirectories."""
    # Initialize the Image Data Generator with some data augmentation options
    train_datagen = ImageDataGenerator(
        rescale=1./255,  # Normalize the image pixels to [0, 1]
        rotation_range=40,  # Random rotation in the range (-40, 40)
        width_shift_range=0.2,  # Random horizontal shift
        height_shift_range=0.2,  # Random vertical shift
        shear_range=0.2,  # Shear transformations
        zoom_range=0.2,  # Zooming
        horizontal_flip=True,  # Enable horizontal flipping
        fill_mode='nearest'  # Strategy to fill newly created pixels after a transformation
    )

    # Flow images from the specified directory, and automatically label them based on the folder names
    train_generator = train_datagen.flow_from_directory(
        directory=data_directory,  # Path to the target directory
        target_size=target_size,  # Resize all images to this size
        batch_size=batch_size,  # Number of images to return each time
        class_mode='categorical'  # Because we are doing multi-class classification
    )

    return train_generator

# Usage
train_directory = 'multiclass_train'  # Specify the path to your training directory
train_data_generator = create_train_generator(train_directory)
```

```
Found 31281 images belonging to 14 classes.
```

Fig: Data Augmentation of Training Data, and Setting up Data Generators for Modeling of Dense121

- The create_train_generator function utilizes tensorflow ImageDataGenerator to prepare and augment image data for training a model such as DenseNet-121. It normalizes images to a [0, 1] range and applies various augmentations like random rotations, shifts, shear, zoom, and horizontal flips to enhance model robustness.
- The generator loads images from specified directories, **resizing** them to **224x224** pixels, suitable for DenseNet-121 input. It organizes images into batches of 32 and labels them categorically based on directory structure for multi-class classification

Modeling - Configurations



```python
def build_model(num_classes):
    # Load the pre-trained DenseNet model, excluding the top fully connected layer
    base_model = DenseNet121(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

    # Freeze all layers in the base model
    for layer in base_model.layers:
        layer.trainable = False

    # Add new layers on top for our specific task
    x = GlobalAveragePooling2D()(base_model.output)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    # This is the model we will train
    model = Model(inputs=base_model.input, outputs=predictions)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Example usage
model = build_model(num_classes=14)  # num_classes = 14 due to 14 classes in this Multiclass Classification
model.summary()
```

- On top of the DenseNet121 base, the model extends with a GlobalAveragePooling2D layer to reduce spatial dimensions, followed by a dense layer of 1024 neurons with ReLU activation to learn task-specific features.
- Finally, it includes a softmax output layer with a number of neurons corresponding to the number of classes (14 for this multiclass classification).

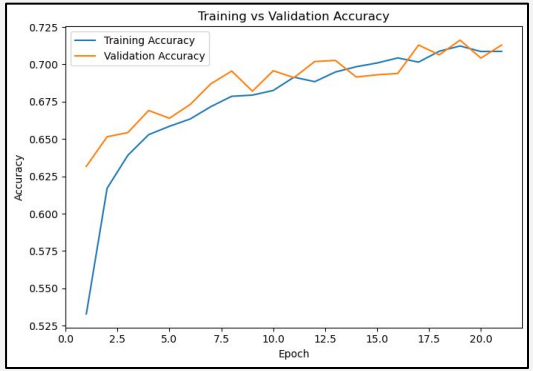
- The function build_model initializes a DenseNet121 as the base model without its top layer (fully connected layers), pre-trained on the ImageNet dataset. It's set to handle inputs of size 224x224x3, which is standard for DenseNet121
- To preserve the learned features of the pre-trained model and prevent them from being updated during training, all layers of the DenseNet121 are frozen, which is common across Transfer Learning methodologies, especially while training the model on much smaller datasets like MURA.

Experiment - 1

Model Specifics: Dense-121, with all layers frozen, additional GAP, Dense (ReLu), Dense (Softmax)

```
conv5_block16_1_bn (BatchN  (None, 7, 7, 128)          512       ['conv5_block16_1_conv[0][0]']
ormalization)

conv5_block16_1_relu (Acti  (None, 7, 7, 128)          0         ['conv5_block16_1_bn[0][0]']
vation)

conv5_block16_2_conv (Conv  (None, 7, 7, 32)           36864     ['conv5_block16_1_relu[0][0]']
2D)

conv5_block16_concat (Conc  (None, 7, 7, 1024)         0         ['conv5_block15_concat[0][0]',
atenate)                                                         'conv5_block16_2_conv[0][0]']

bn (BatchNormalization)     (None, 7, 7, 1024)         4096      ['conv5_block16_concat[0][0]']

relu (Activation)           (None, 7, 7, 1024)         0         ['bn[0][0]']

global_average_pooling2d (  (None, 1024)               0         ['relu[0][0]']
GlobalAveragePooling2D)

dense (Dense)               (None, 1024)               1049600   ['global_average_pooling2d[0][
                                                                 0]']

dense_1 (Dense)             (None, 14)                 14350     ['dense[0][0]']

==================================================================================================
Total params: 8101454 (30.90 MB)
Trainable params: 1063950 (4.06 MB)
Non-trainable params: 7037504 (26.85 MB)
__________________________________________________________________________________________________
```

```
Epoch 19/50
977/977 [==============================] - 265s 271ms/step - loss: 0.7182 - accuracy: 0.7123 - val_loss: 0.7153 - val_accuracy: 0.7162
Epoch 20/50
977/977 [==============================] - 264s 270ms/step - loss: 0.7182 - accuracy: 0.7086 - val_loss: 0.7167 - val_accuracy: 0.7042
```



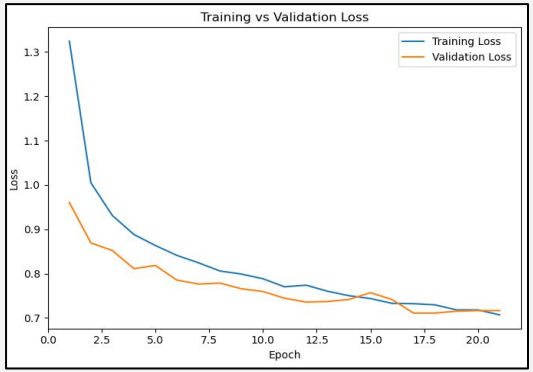


Fig: The above curves warn us about the overfitting, which was expected

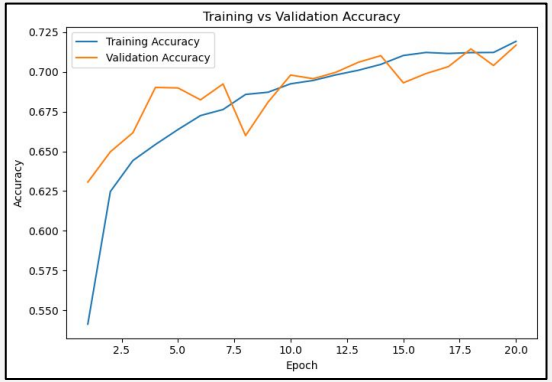
Experiment - 2

Model Specifics: Dense-121, with only last layer unfrozen, GAP, Dense(ReLU), Dense (Softmax)

```
conv5_block16_0_bn (BatchN  (None, 7, 7, 992)           3968      ['conv5_block15_concat[0][0]']
ormalization)

conv5_block16_0_relu (Acti  (None, 7, 7, 992)           0         ['conv5_block16_0_bn[0][0]']
vation)

conv5_block16_1_conv (Conv  (None, 7, 7, 128)           126976    ['conv5_block16_0_relu[0][0]']
2D)

conv5_block16_1_bn (BatchN  (None, 7, 7, 128)           512       ['conv5_block16_1_conv[0][0]']
ormalization)

conv5_block16_1_relu (Acti  (None, 7, 7, 128)           0         ['conv5_block16_1_bn[0][0]']
vation)

conv5_block16_2_conv (Conv  (None, 7, 7, 32)            36864     ['conv5_block16_1_relu[0][0]']
2D)

conv5_block16_concat (Conc  (None, 7, 7, 1024)          0         ['conv5_block15_concat[0][0]',
atenate)                                                           'conv5_block16_2_conv[0][0]']

bn (BatchNormalization)     (None, 7, 7, 1024)          4096      ['conv5_block16_concat[0][0]']

relu (Activation)           (None, 7, 7, 1024)          0         ['bn[0][0]']

global_average_pooling2d_1  (None, 1024)                0         ['relu[0][0]']
 (GlobalAveragePooling2D)

dense_2 (Dense)             (None, 1024)                1049600   ['global_average_pooling2d_1[0
                                                                   ][0]']

dense_3 (Dense)             (None, 14)                  14350     ['dense_2[0][0]']

==================================================================================================
Total params: 8101454 (30.90 MB)
Trainable params: 1100814 (4.20 MB)
Non-trainable params: 7000640 (26.71 MB)
```

Fig: Change in the trainable Parameters compared to Exp. 1

```
Epoch 19/20
977/977 [==============================] - 265s 271ms/step - loss: 0.6930 - accuracy: 0.7122 - val_loss: 0.7246 - val_accuracy: 0.7040
Epoch 20/20
977/977 [==============================] - 264s 270ms/step - loss: 0.6856 - accuracy: 0.7192 - val_loss: 0.6958 - val_accuracy: 0.7168
```



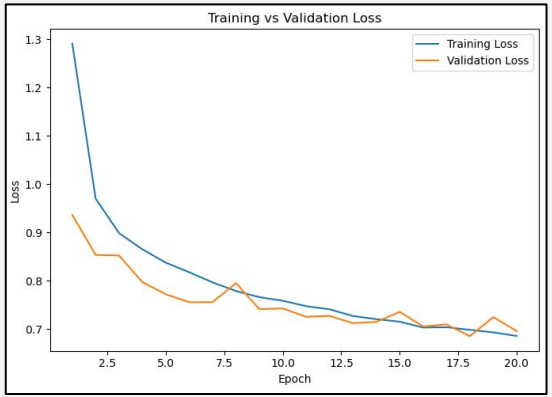


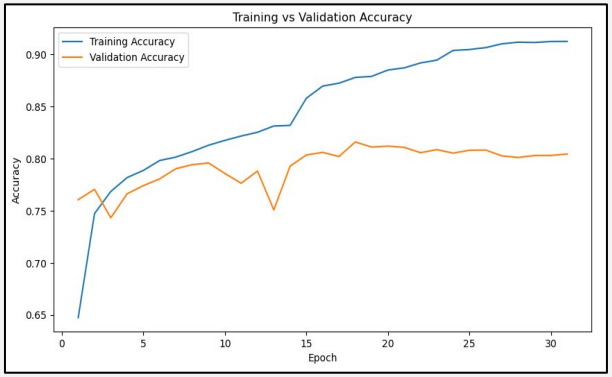
Fig: The above curves warn us about the overfitting, which is expected at this stage of training

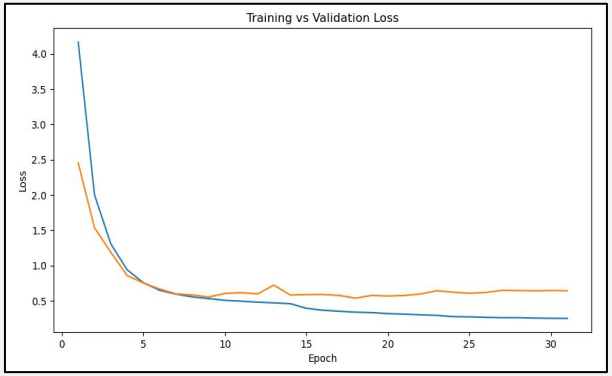
Experiment - 3

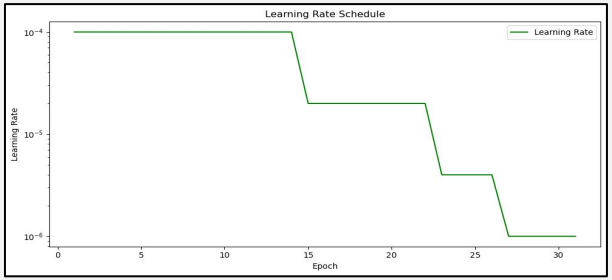
Model Specifics: Dense-121 with Learning rate scheduler, only last 4 layers unfrozen, GAP, Dense(ReLU), Dropout (0.5), and a Dense (Softmax) as the last layer

```python
def build_model(num_classes, unfreeze_last_n_conv=4):
    # Load the pre-trained DenseNet model, excluding the top fully connected layer
    base_model = DenseNet121(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

    # Unfreeze the last n convolutional layers
    for layer in base_model.layers[-unfreeze_last_n_conv:]:
        layer.trainable = True

    # Add new layers on top for our specific task
    x = GlobalAveragePooling2D()(base_model.output)
    x = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x)  # Add L2 regularization
    x = Dropout(0.5)(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    # This is the model we will train
    model = Model(inputs=base_model.input, outputs=predictions)

    # Use a relevant learning rate scheduler
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=1e-6)

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

    return model, reduce_lr

# Unfreeze the last 4 conv layers and build the model
model, reduce_lr = build_model(num_classes=14, unfreeze_last_n_conv=5)
model.summary()
```

```
=================================================================
Total params: 7569486 (28.88 MB)
Trainable params: 7485838 (28.56 MB)
Non-trainable params: 83648 (326.75 KB)
```







Experiment - 4

Model Specifics: Dense-121 with Learning rate scheduler & L2, only last 5 layers unfrozen, GAP, Dense(ReLU), Dropout (0.6) & Dense, and a Dense (Softmax)

conv5_block15_1_conv (Conv 2D)	(None, 7, 7, 128)	122880	['conv5_block15_0_relu[0][0]']
conv5_block15_1_bn (BatchN ormalization)	(None, 7, 7, 128)	512	['conv5_block15_1_conv[0][0]']
conv5_block15_1_relu (Acti vation)	(None, 7, 7, 128)	0	['conv5_block15_1_bn[0][0]']
conv5_block15_2_conv (Conv 2D)	(None, 7, 7, 32)	36864	['conv5_block15_1_relu[0][0]']
conv5_block15_concat (Conc atenate)	(None, 7, 7, 992)	0	['conv5_block14_concat[0][0]', 'conv5_block15_2_conv[0][0]']
conv5_block16_0_bn (BatchN ormalization)	(None, 7, 7, 992)	3968	['conv5_block15_concat[0][0]']
conv5_block16_0_relu (Acti vation)	(None, 7, 7, 992)	0	['conv5_block16_0_bn[0][0]']
conv5_block16_1_conv (Conv 2D)	(None, 7, 7, 128)	126976	['conv5_block16_0_relu[0][0]']
conv5_block16_1_bn (BatchN ormalization)	(None, 7, 7, 128)	512	['conv5_block16_1_conv[0][0]']
conv5_block16_1_relu (Acti vation)	(None, 7, 7, 128)	0	['conv5_block16_1_bn[0][0]']
conv5_block16_2_conv (Conv 2D)	(None, 7, 7, 32)	36864	['conv5_block16_1_relu[0][0]']
conv5_block16_concat (Conc atenate)	(None, 7, 7, 1024)	0	['conv5_block15_concat[0][0]', 'conv5_block16_2_conv[0][0]']
bn (BatchNormalization)	(None, 7, 7, 1024)	4096	['conv5_block16_concat[0][0]']
relu (Activation)	(None, 7, 7, 1024)	0	['bn[0][0]']
global_average_pooling2d_5 (GlobalAveragePooling2D)	(None, 1024)	0	['relu[0][0]']
dense_10 (Dense)	(None, 512)	524800	['global_average_pooling2d_5[0][0]']
dropout_4 (Dropout)	(None, 512)	0	['dense_10[0][0]']
dense_11 (Dense)	(None, 14)	7182	['dropout_4[0][0]']

Total params: 7569486 (28.88 MB)
Trainable params: 7485838 (28.56 MB)
Non-trainable params: 83648 (326.75 KB)

This Model Architecture was trained for ~10 epochs, it was able to get the best training results out of all the experiments.

The training process of this model architecture is a Work in progress, the team believes it can produce better results if trained with higher number of epochs, and finetune some hyperparameters to achieve the ~ SOTA scores

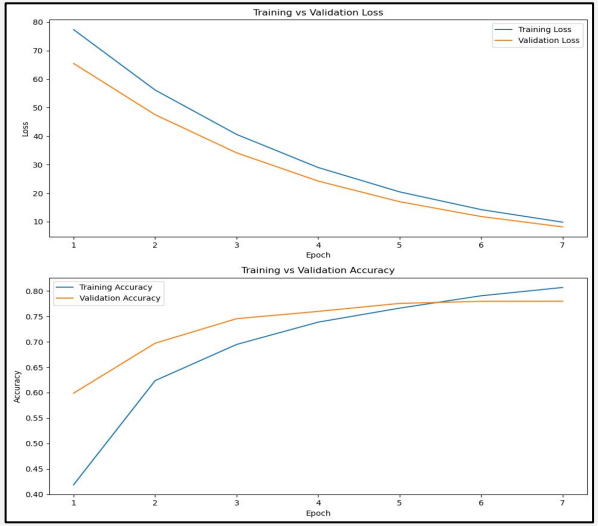
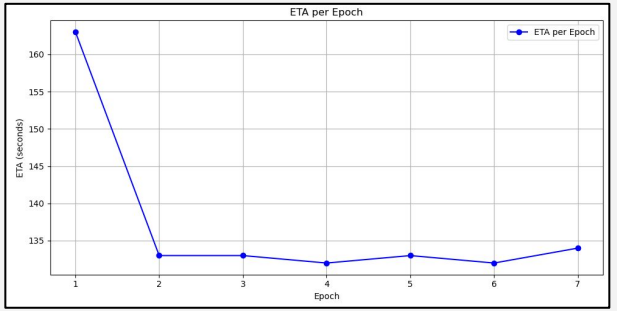
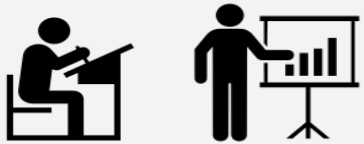


Fig: The above curves warn us about the overfitting, due to the number of attention heads, and high complexity



Experiment - 5

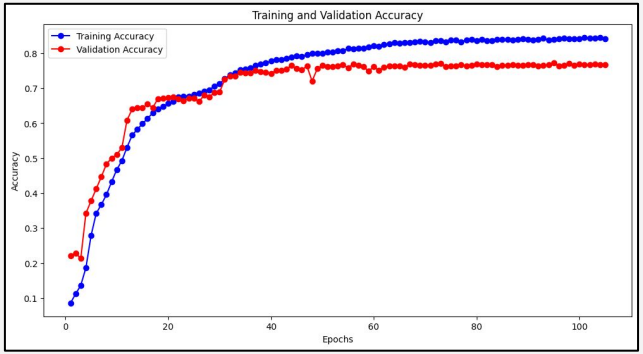


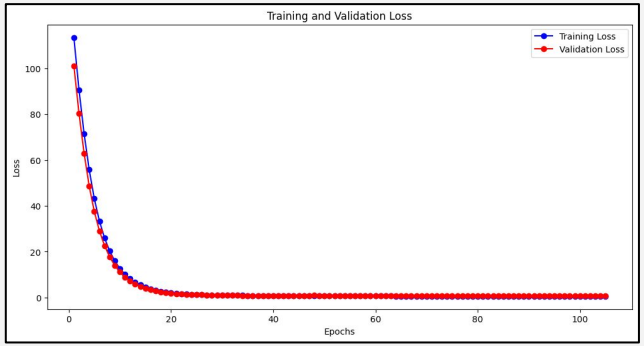
Model Specifics: DenseNet-121 with the last 5 conv. layers unfrozen, incorporates 3 attention heads and uses a concatenation mechanism, features dropout rates of 0.5 before and 0.7 after GAP, includes two fully connected layers with heavy dropout (0.7 and 0.6) and L2 regularization, and utilizes a learning rate scheduler.

conv5_block16_concat (Concatenate)	(None, 7, 7, 1024)	0	['conv5_block15_concat[0][0]', 'conv5_block16_2_conv[0][0]']
bn (BatchNormalization)	(None, 7, 7, 1024)	4096	['conv5_block16_concat[0][0]']
relu (Activation)	(None, 7, 7, 1024)	0	['bn[0][0]']
attention_0 (Conv2D)	(None, 7, 7, 1)	1025	['relu[0][0]']
attention_1 (Conv2D)	(None, 7, 7, 1)	1025	['relu[0][0]']
attention_2 (Conv2D)	(None, 7, 7, 1)	1025	['relu[0][0]']
multiply_3 (Multiply)	(None, 7, 7, 1024)	0	['relu[0][0]', 'attention_0[0][0]']
multiply_4 (Multiply)	(None, 7, 7, 1024)	0	['relu[0][0]', 'attention_1[0][0]']
multiply_5 (Multiply)	(None, 7, 7, 1024)	0	['relu[0][0]', 'attention_2[0][0]']
concatenate_1 (Concatenate)	(None, 7, 7, 3072)	0	['multiply_3[0][0]', 'multiply_4[0][0]', 'multiply_5[0][0]']
dropout_4 (Dropout)	(None, 7, 7, 3072)	0	['concatenate_1[0][0]']
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 3072)	0	['dropout_4[0][0]']
dropout_5 (Dropout)	(None, 3072)	0	['global_average_pooling2d_1[0][0]']
dense_3 (Dense)	(None, 512)	1573376	['dropout_5[0][0]']
dropout_6 (Dropout)	(None, 512)	0	['dense_3[0][0]']
dense_4 (Dense)	(None, 256)	131328	['dropout_6[0][0]']
dropout_7 (Dropout)	(None, 256)	0	['dense_4[0][0]']
dense_5 (Dense)	(None, 14)	3598	['dropout_7[0][0]']

==========

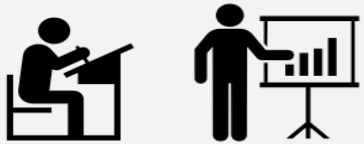
Total params: 8748881 (33.37 MB)
Trainable params: 8665233 (33.06 MB)
Non-trainable params: 83648 (326.75 KB)





The use of attention mechanisms helps the model to focus on more relevant features in the input data, potentially improving training efficiency and accuracy by directing the model's focus to important patterns, as evidenced by the convergence of training and validation loss and accuracy in the plots.

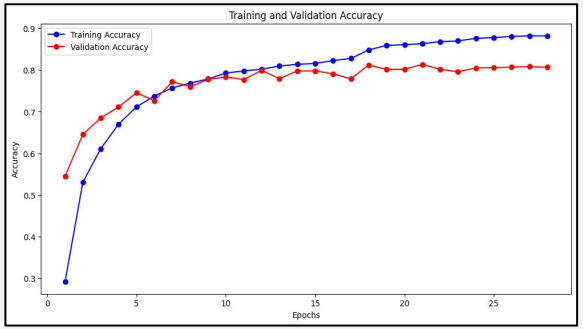
Experiment - 6

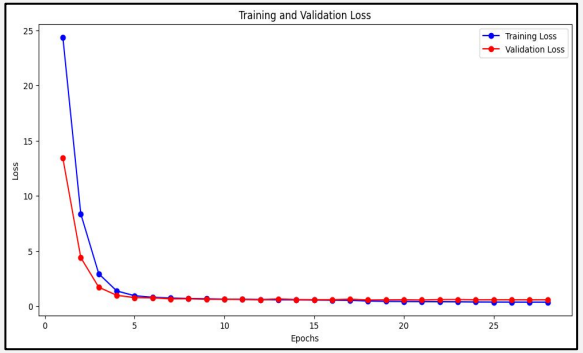


Model Specifics: DenseNet-121 with the last 5 conv. layers unfrozen, incorporates squeeze-and-excite blocks, 3 attention heads and uses a concatenation mechanism, features dropout rates of 0.5 before and 0.7 after GAP, includes two fully connected layers with heavy dropout (0.7 and 0.6) and L2 regularization without Dense layers, and utilizes ReduceLROnPlateau scheduler for learning rate.

multiply_1 (Multiply)	(None, 7, 7, 1024)	0	['multiply[0][0]', 'attention_0[0][0]']
multiply_2 (Multiply)	(None, 7, 7, 1024)	0	['multiply[0][0]', 'attention_1[0][0]']
multiply_3 (Multiply)	(None, 7, 7, 1024)	0	['multiply[0][0]', 'attention_2[0][0]']
concatenate (Concatenate)	(None, 7, 7, 3072)	0	['multiply_1[0][0]', 'multiply_2[0][0]', 'multiply_3[0][0]']
dropout (Dropout)	(None, 7, 7, 3072)	0	['concatenate[0][0]']
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 3072)	0	['dropout[0][0]']
dropout_1 (Dropout)	(None, 3072)	0	['global_average_pooling2d_1[0][0]']
dense_2 (Dense)	(None, 512)	1573376	['dropout_1[0][0]']
batch_normalization (BatchNormalization)	(None, 512)	2048	['dense_2[0][0]']
dropout_2 (Dropout)	(None, 512)	0	['batch_normalization[0][0]']
dense_3 (Dense)	(None, 512)	262656	['dropout_2[0][0]']
dense_4 (Dense)	(None, 512)	1572864	['dropout_1[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 512)	2048	['dense_3[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 512)	2048	['dense_4[0][0]']
add (Add)	(None, 512)	0	['batch_normalization_1[0][0]', 'batch_normalization_2[0][0]']
activation (Activation)	(None, 512)	0	['add[0][0]']
dropout_3 (Dropout)	(None, 512)	0	['activation[0][0]']
dense_5 (Dense)	(None, 256)	131328	['dropout_3[0][0]']
dropout_4 (Dropout)	(None, 256)	0	['dense_5[0][0]']
dense_6 (Dense)	(None, 14)	3598	['dropout_4[0][0]']

Total params: 10722705 (40.90 MB)
Trainable params: 10635985 (40.57 MB)
Non-trainable params: 86720 (338.75 KB)

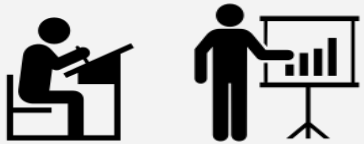




SE blocks perform a form of feature recalibration through which they adaptively adjust the weights of different feature channels. This is achieved by first squeezing global spatial information into a channel descriptor using global average pooling. This step summarizes the spatial features for each channel, condensing the information into a compact form.

Following the squeeze operation, an **excitation operation** applies a simple gating mechanism (typically two fully-connected layers with a relu activation between them, followed by a sigmoid activation) to capture channel-wise dependencies. This gating mechanism can learn to use the global information to emphasize informative features and suppress less useful ones dynamically.

Training Summary

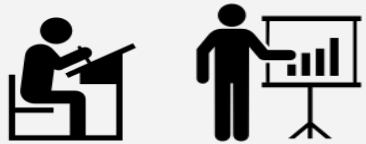


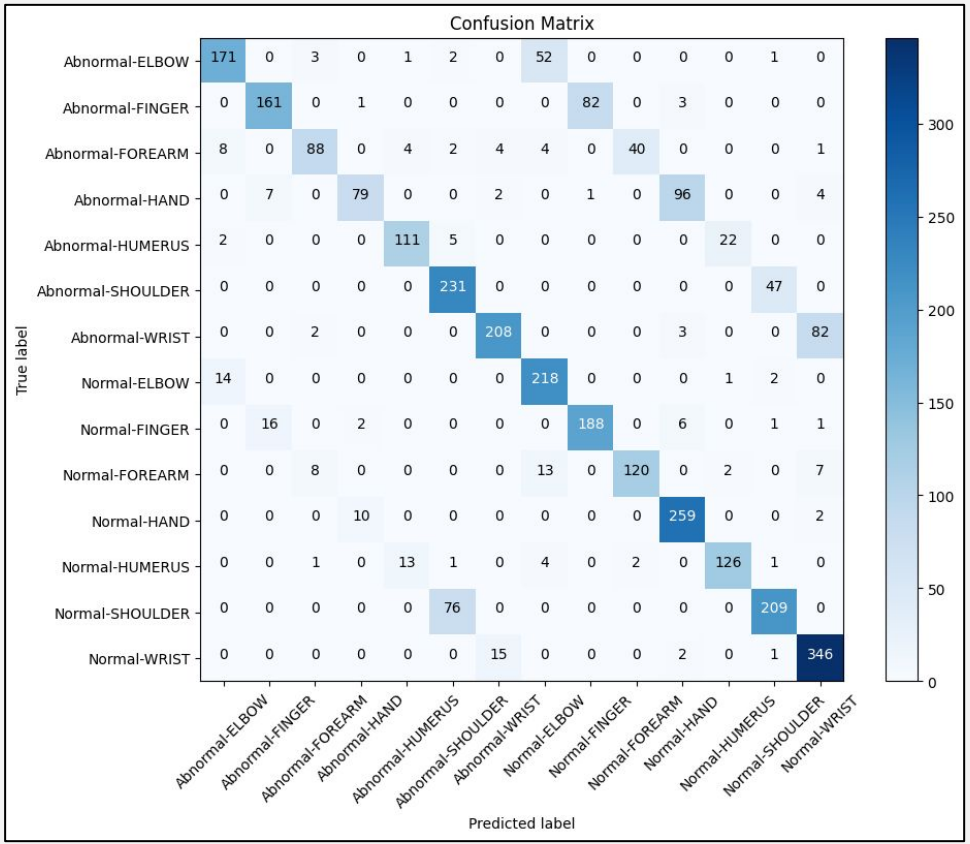
		Training		Validation	
# Exp	**Model Details**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
Exp 1	Dense121- No layers Unfrozen	0.72	0.71	0.72	0.70
Exp 2	Dense 121 - Last Layer Unfrozen	0.68	0.72	0.69	0.72
Exp 3	Dense 121 - Last 4 layers Unfrozen, Strong L1, L2 and learning rate scheduler	0.25	0.91	0.64	0.80
Exp 4	Dense121 - Last 5 layers, L1, L2, learning rate scheduler	0.41	0.82	0.55	0.81
Exp 5	Dense121 - Last 5 layers, Attention heads = 3, strong L1, L2, learning rate scheduler with additional dropout	0.61	0.84	0.79	0.77
Exp 6	Dense121 - SE feature extraction, Last 5 layers, Attention heads = 3, strong L1, L2, learning rate scheduler with additional dropout	**0.37**	**0.88**	**0.59**	**0.81**

Parameter	Value
Base Model	DenseNet121
Optimizer	Adam
Initial Learning Rate	0.00001 (Assumed from previous messages)
Loss Function	categorical_crossentropy (Assumed)
Batch Size	64
Epochs	150
Target Size	224 x 224
Data Augmentation	Yes (rotation, width/height shift, shear, zoom, horizontal flip)
Rescale	1/255
Callbacks	ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
ModelCheckpoint Mode	min
EarlyStopping Patience	10
EarlyStopping Mode	min

The above Hyperparameters yielded the best results achieved during the training, and evaluation phase, the primary focus of the experimentation was mostly on the placement of different layers like GAP, Dropout, and Attention to yield better performance

Testing & Inference - I

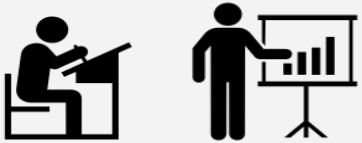


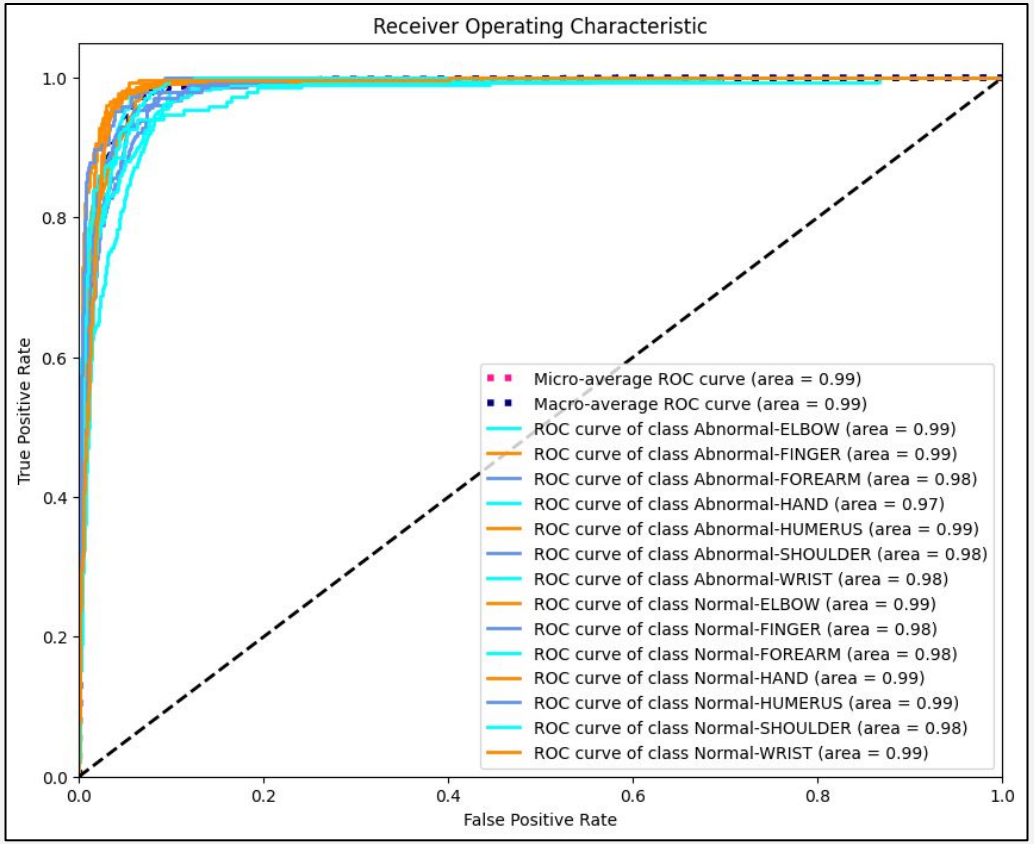


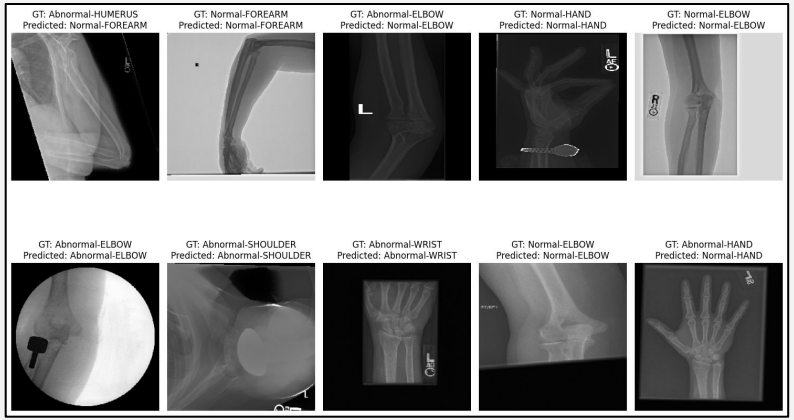

```
Found 3197 images belonging to 14 classes.
50/50 [==============================] - 16s 303ms/step
Classification Report
                    precision    recall  f1-score   support

     Abnormal-ELBOW       0.88      0.74      0.80       230
    Abnormal-FINGER       0.88      0.65      0.75       247
   Abnormal-FOREARM       0.86      0.58      0.70       151
      Abnormal-HAND       0.86      0.42      0.56       189
   Abnormal-HUMERUS       0.86      0.79      0.83       140
  Abnormal-SHOULDER       0.73      0.83      0.78       278
     Abnormal-WRIST       0.91      0.71      0.79       295
      Normal-ELBOW       0.75      0.93      0.83       235
     Normal-FINGER       0.69      0.88      0.78       214
    Normal-FOREARM       0.74      0.80      0.77       150
       Normal-HAND       0.70      0.96      0.81       271
    Normal-HUMERUS       0.83      0.85      0.84       148
   Normal-SHOULDER       0.80      0.73      0.76       285
      Normal-WRIST       0.78      0.95      0.86       364

          accuracy                           0.79      3197
         macro avg       0.80      0.77      0.78      3197
      weighted avg       0.80      0.79      0.78      3197
```

An overall of **0.78** weighted avg. F1, and an overall accuracy of **0.79**. Abnormal HAND is the class which has modest recall, from the confusion matrix its due to the high number of misclassifications in **ABNORMAL WRIST, ABNORMAL FINGER, ABNORMAL HAND**

Testing & Inference - II



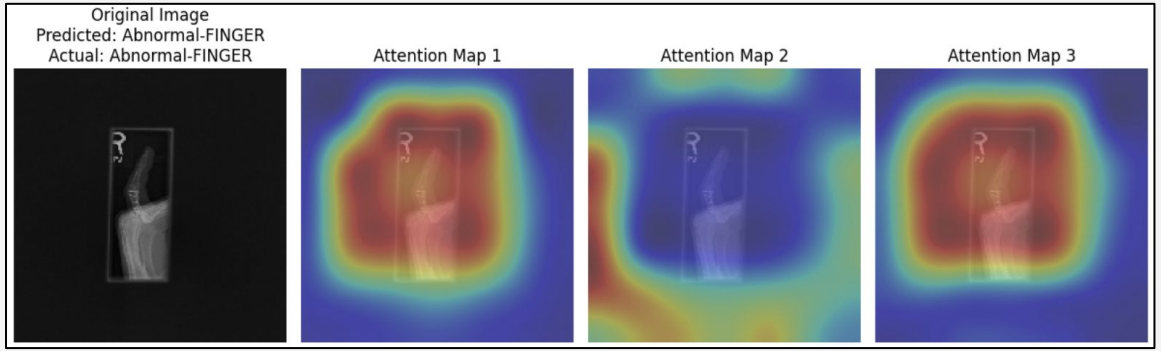


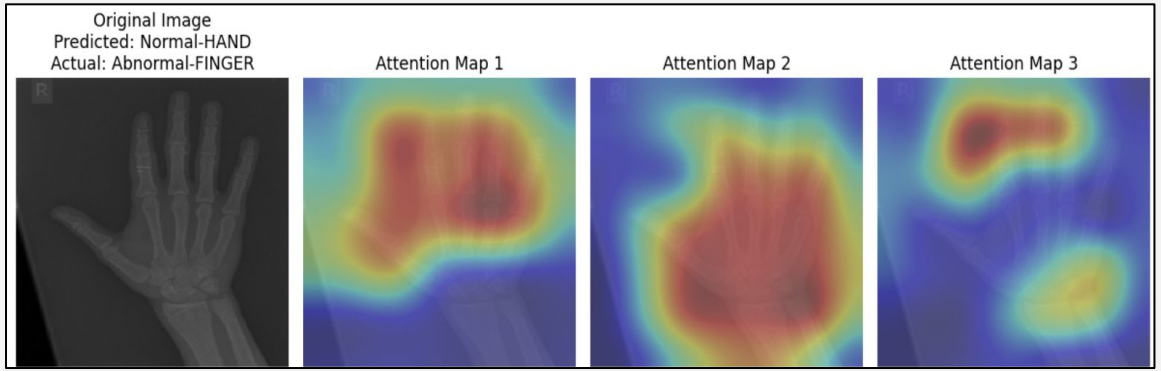


The Receiver Operating Characteristic (ROC) curves for various classes show impressive Area Under Curve (AUC) values close to 1.0, indicating excellent model discrimination capability between positive and negative classes.

The model demonstrates strong precision across various classes, with values such as 0.91 for Abnormal-Wrist and 0.75 for Normal-Elbow. The recall and F1-score are equally robust, peaking at 0.96 for Normal-Hand in recall and 0.86 for Normal-Wrist in F1-score, reflecting a balanced precision-recall trade-off.

X AI - Attention Maps

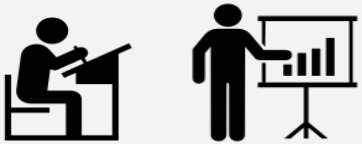


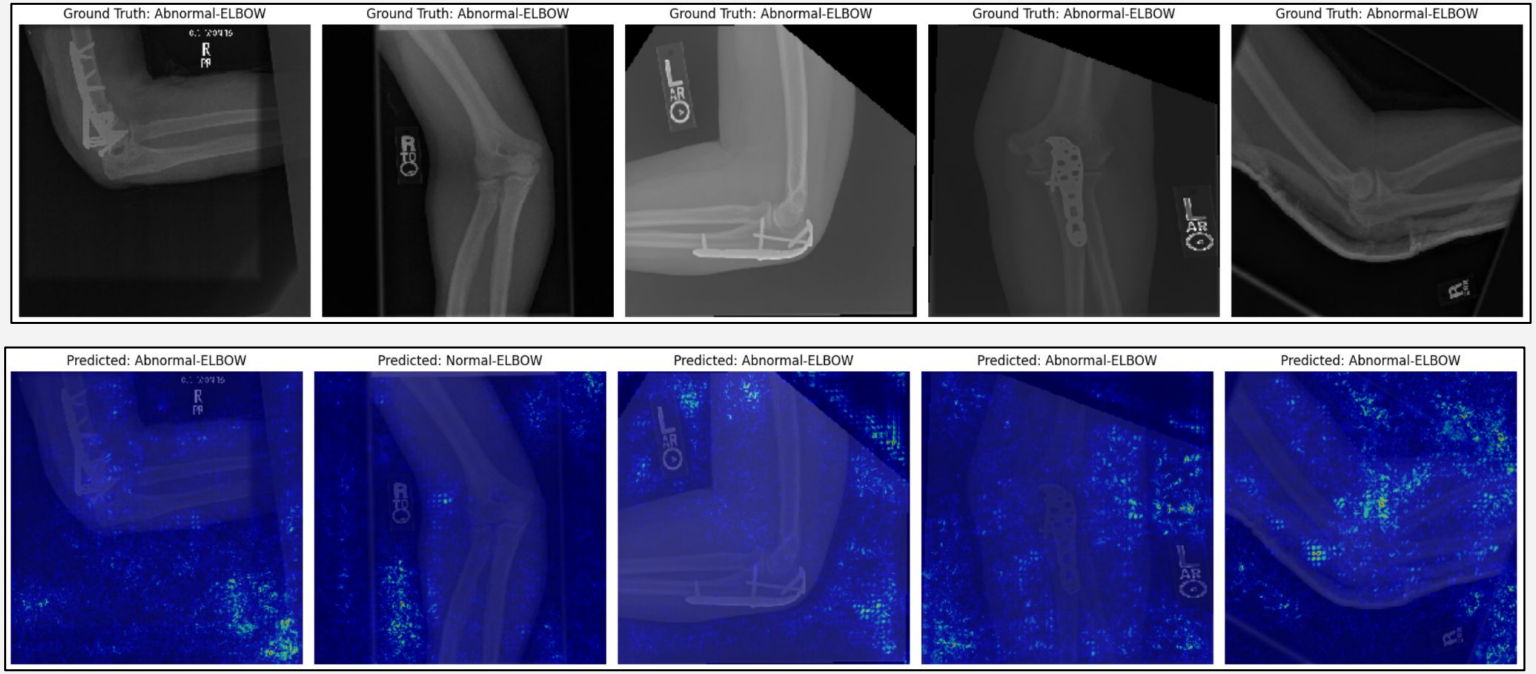


These are visual representations showing which areas of the input image were most influential in the model's decision-making process. Typically, these maps are generated by overlaying a heatmap on the original image. The regions highlighted in brighter colors (often red or yellow) indicate higher attention by the model, meaning these areas had more impact on the output prediction.

By examining the areas highlighted in the attention maps, you can determine whether the model is focusing on anatomically relevant areas when making its predictions. For example, if the model consistently highlights the joint areas or any visible abnormalities in the bones when predicting "Abnormal-FINGER," it likely understands the key features that define an abnormal X-ray.

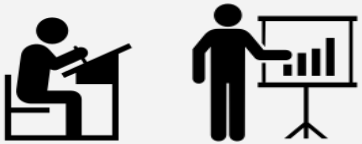
X AI - Saliency Maps

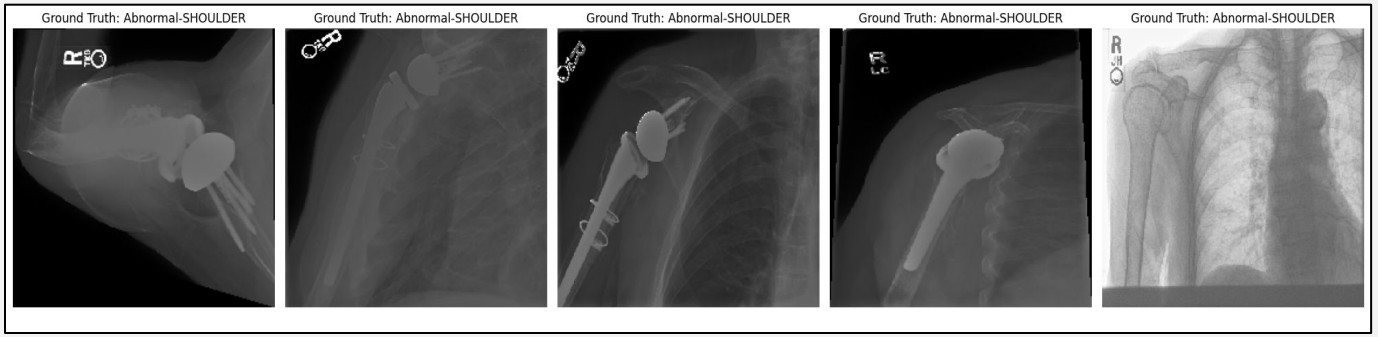




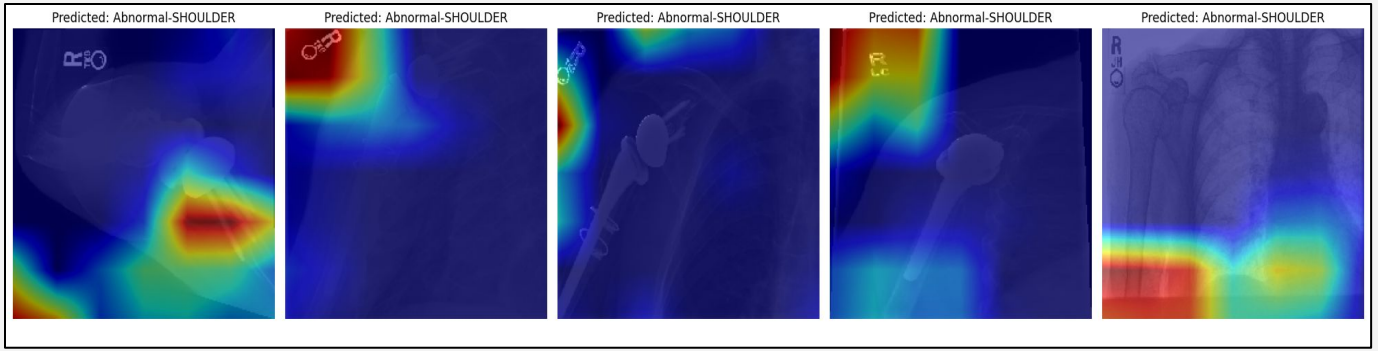
Saliency maps are generated using tf_keras_vis, which is a visualization library designed to work with TensorFlow and Keras models. We selected a specific convolutional layer (conv5_block16_2_conv) to examine, which is a deeper layer in the network and likely to capture high-level features critical for making predictions about elbow abnormalities

X AI - Grad CAM



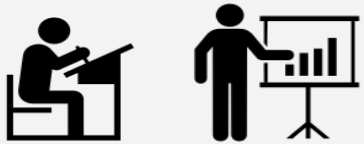


A specific convolutional layer - conv5_block16_2_conv, that captures the high-level features in the image, which are crucial for making predictions. This layer's feature maps are used to compute the Grad-CAM heatmaps.



Grad-CAM uses the gradients of any target concept flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept.

Competition Results and Comparison



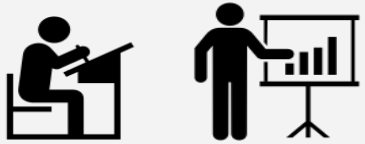
Rank	Model Name	Cohen Kappa
	Best Radiologist Performance Stanford University Rajpurkar & Irvin et al.	0.778
1	base-comb2-xuan-v3(ensemble) jzhang Availink	0.843
2	base-comb2-xuan(ensemble) jtzhang Availink	0.834
3	muti_type (ensemble model) SCU_MILAB	0.833
4	base-comb4(ensemble) jtzhang Availink	0.824
5	base-comb2-jun2(ensemble)	0.814
	dl-team7-dense121	0.612
65	Bhaukali_v1.0 (single model) IIT BHU, Varanasi	0.589
68	Densenet169-lite(single model) Tang	0.560
69	ensemble1 ensemble	0.534
70	DenseNet (single model) Zhou	0.518

Kappa offers a normalized score that can be directly used to compare different models or different configurations of the same model on the same task.

In clinical settings, it's common for different radiologists to have varying opinions on a case. Kappa's emphasis on agreement helps in assessing how consistently a model performs in comparison to human experts, as highlighted in the leaderboard rankings where models are compared to the best radiologist performance.

A Kappa between 0.81 and 1 is considered almost perfect agreement. A Kappa between 0.61 and 0.80 shows substantial agreement.

Conclusion & Future Scope



The results of the Project explain how a pre-trained CNN architecture like DenseNet can be utilized to further finetune on specific tasks like this. It details about the different strategies that can be employed - Layer by layer unfreezing and finetuning, adding relevant layers to the existing architecture, and others.

The X AI techniques like Grad CAM, Attention Maps, Saliency Maps - The explainability is directly proportional to how well the model classifies, so improving the differentiating capabilities of the model will directly translate to better explainability.

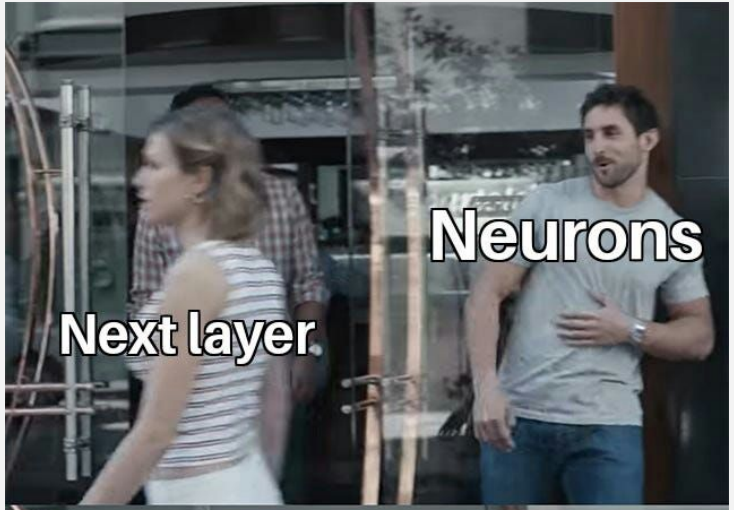
There is a lot of scope for improvement in the performance metrics, by unfreezing significant number of layers, and re-training them with the specific dataset - this is a current WORK IN PROGRESS

The Future Scope for this project is huge:

- Employing further training strategies (experimenting with the # layers)
- Creating an Ensemble of Neural networks or DenseNet like architectures to boost the performance of the model comprehensively
- To implement ViT with Transfer learning, and understand the explainability on the inference
- Creating Edge efficient Models which are capable of predicting on edge devices
- Exploring other X AI techniques for better comprehension of the results of a Neural network
- There's a huge scope for research in this domain, ONE SUCH TECHNIQUE WOULD BE CREATING THE WEIGHTS FOR RADIOGRAPHY IMAGES

Thank You!


dont worry about it if you don't
understand


Next layer
Neurons


Dropout
Neurons

References

1. Bone X-Ray Deep Learning Competition - here
2. MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs - here
3. Utilizing heat maps as explainable artificial intelligence for detecting abnormalities on wrist and elbow radiographs - here
4. Evaluating Explainable Artificial Intelligence for X-ray Image Analysis - here
5. Trustworthy deep learning framework for the detection of abnormalities in X-ray shoulder images - here
6. Explainable AI-Based Humerus Fracture Detection and Classification from X-Ray Images - here
7. Transparency of deep neural networks for medical image analysis: A review of interpretability methods - here
8. Fully Convolutional Architectures for Multiclass Segmentation in Chest Radiographs - here