Deep Learning For Abnormalities Detection in Musculoskeletal Radiographs

Kshitij Patel
Department of Computer Science and Engineering, SOT
Pandit Deendayal Energy University
Gandhinagar, India
kshitij.pce18@sot.pdpu.ac.in

Vatsal Patel
Department of Computer Science and Engineering, SOT
Pandit Deendayal Energy University
Gandhinagar, India
vatsal.pce18@sot.pdpu.ac.in

Dr. Samir Patel
Department of Computer Science and Engineering, SOT
Pandit Deendayal Energy University
Gandhinagar, India
Samit.Patel@sot.pdpu.ac.in

Devansh Patel
Surat Municipal Institute of Medical Education and Research
Surat, India
devansh8780@gmail.com

***Abstract*—AI, particularly Deep learning, has made significant progress in perception tasks, allowing machines to better encode and analyze complicated input. Deep learning approaches have recently seen a lot of success in medical image analysis. However, there has been relatively limited research into using deep learning algorithms to classify Fractured bones from X-Ray images. X-Ray images are widely used by doctors for better diagnosis of patients. Like such, X-Ray images of bones are more commonly used by orthopedic surgeons to detect if the bone is abnormal or not. In an emergency room, sometimes missed fractures can result in treatment delays or permanent disabilities. Through this paper, we have presented a novel approach to utilizing the deep learning model to identify fractured bones. Our model aims to detect abnormal bone in Elbow, Finger, Forearm, Hand, Humerus, Shoulder, and Wrist by leveraging the concept of transfer learning on Convolutional Neural Networks.**

Keywords—Artificial Intelligence, Deep Learning, Computer Vision, Convolutional Neural Networks, Medical Imaging

I. Introduction

From infants to the elderly, fractures are very common in our surroundings. Involvement in fall crashes or any other mishap can easily lead to bone fractures. Doctors utilize medical images to determine whether or not bone fractures have occurred. An erudite doctor spends a considerable amount of time analyzing X-Ray images to know more about the gravity of the injury. However, there are times when even an experienced doctor can make a mistake while analyzing the X-Ray images which can often lead to long-term disability in the patient. To aid doctors in bone fracture detection and save time, Computer-aided diagnosis (CAD) has been widely employed in the processing of medical images. The purpose of this paper is to propose a model which can assist doctors in detecting fractures with the help of X-ray images.

There were many options available to approach this purpose like classical computer-vision algorithms of Haar-Cascade [1], which requires extensive labelling of input images. We have used the modern approach of Deep Learning for training a neural network for classification purposes. We have opted for the Efficient-Net [2] neural network model as it has outperformed other neural networks like the DenseNet [3], MobileNet [4], and Nasnet [5]. To reduce training time and lower the computational cost we have used transfer learning techniques [6] to train these models. The detailed work and implementation of our paper is discussed in the following sections.

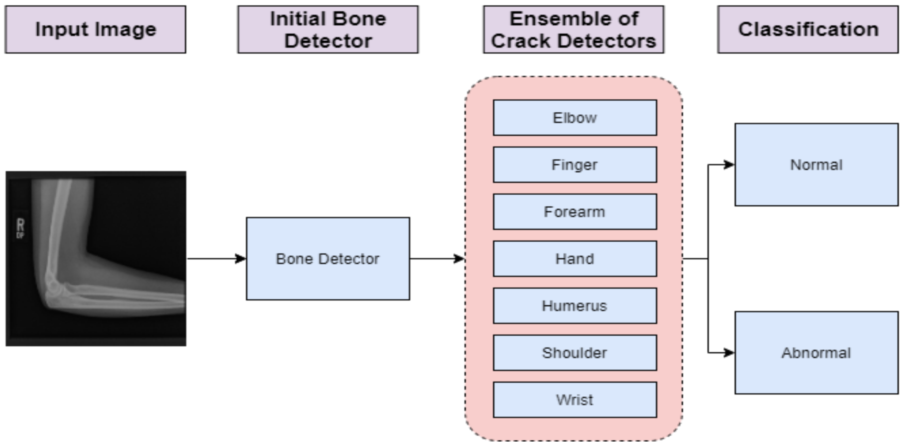


Fig - 1. System Model Diagram

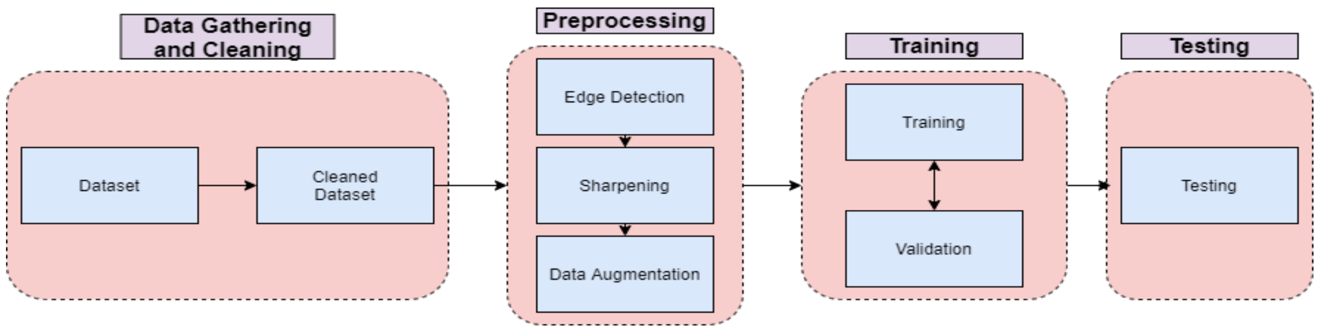


Fig - 2. Workflow of the proposed model

II. Related Works

MURA [7] is a large dataset of X-ray images that are manually labeled by medical professionals. The dataset was prepared by the Machine Learning group of Stanford University. They have trained the DenseNet [3] neural network for the classification of X-rays as either normal or abnormal. They have published this dataset publicly for researchers worldwide to find innovative solutions to this problem.

This paper [8] shows the current state-of-the-art approach for detecting abnormalities in musculoskeletal radiographs. The current state-of-the-art solution uses a two-step process with Faster-RCNN and one other CNN for classification, making the computation expensive. We are proposing to use a single-shot CNN which will reduce the computational costs. The data used by the most accurate solutions is very small as compared to the dataset that we are using. So, we hope to get results which are close to real scenarios.

This article [9] discusses various techniques available for detecting edges in an image. We have used various edge detection techniques such as Sobel [10], Prewitt [11], Roberts [12] and Canny [13] in our work.

III. Methodology

A. *System Model*

The image (Fig – 1) shows the working of our proposed system for detecting abnormalities in the bones of the patients using Musculoskeletal radiographs or X-ray images. The model will take an X-ray image as the input. To find out which type of bone it is, it will pass this image through a Bone Detector which will detect the correct type of bone in the image. After the bone is distinguished it will pass this image to its respective bone model to detect whether there is a fracture in the bone or not. We have trained individual models for fracture detection in each type of bone. As each bone has a different shape, size, orientation, and density, we were required to train a model specifically for each bone type. Various types of bones included - Elbow, Finger, Forearm, Hand, Humerus, Shoulder, and Wrist. These models will classify the input image into two classes - normal and abnormal. If the model is unable to detect a crack in the image, it will classify it as a normal image. On the other hand, if it detects a crack in the image it will classify the image as an abnormal image.

B. *Proposed Work*

The above image (Fig – 2) shows our proposed workflow. The most important aspect of a Deep Learning model is the collection of data and its cleaning. So after carefully reviewing and analyzing various datasets, we have selected the MURA [7] dataset from the Stanford ML group, as it was the most reliable and accurate dataset. As no dataset is perfect, we needed to clean the data and convert it into a format that we can work with. As the dataset contains more than 40000 high-resolution images it takes a huge amount of time to analyze and process the data. We also wrote efficient code that helped in automating the cleaning process of the dataset.

After that, we worked on the preprocessing of images. As X-rays are black and white images, it is difficult to observe subtle cracks in the bones, even for the human eyes. So, to improve the quality of the images we applied various techniques like edge detection, sharpening, and data augmentation which assist during the training process. We have discussed in detail the algorithms we have used for these preprocessing steps in the following chapter about implementation details.

The penultimate step in our proposed workflow is the training of the Deep Learning model using an efficient technique known as transfer learning. Training along with validation of the model was done during this process. As there are many parameters during training, we also tuned them individually to increase the accuracy. We will discuss more about training later. The final step was to test the model on images that it has never seen before.

IV. Implementation

In this section, we will discuss the implementation of our work. We will talk about data collection first and then about data cleaning. After that, we will discuss various data pre-processing techniques that we have used on the images to improve the quality of the images. We will then talk about how we trained various neural network models to detect the type of bone and detect cracks in each type of bone.

A. *Data Collection*

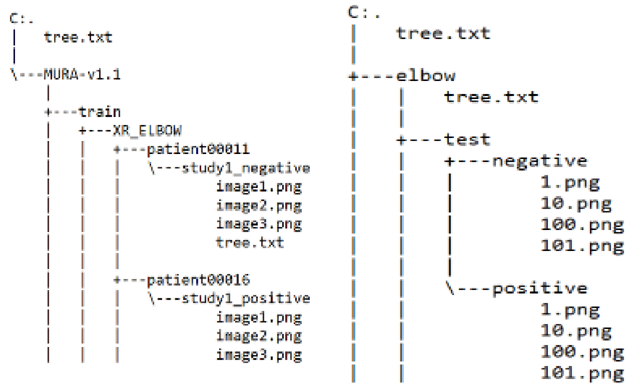
The images were collected from the Musculoskeletal Radiographs (MURA) dataset prepared by the Stanford Machine Learning (ML) group. MURA is a huge dataset of X-ray images of different bones in the human body. It is also one of the largest publicly available datasets. The dataset was collected from 14863 studies, which were carried out on 12173 patients.

Bone	Training	Testing	Total
Elbow	4931	465	5936
Finger	5106	461	5567
Forearm	1825	301	2126
Hand	5543	460	6003
Humerus	1272	288	1560
Shoulder	8379	563	8942
Wrist	9752	659	10411
Total			40005

TABLE 1. Dataset Statistics

The dataset contains more than 40000 X-ray images of different bones in the human body such as the Elbow, Finger, Forearm, Hand, Humerus, Shoulder, and Wrist. Each study was labelled manually as normal or abnormal by certified radiologists from Stanford Hospital. The images are distributed according to [Table 1].

B. Data Cleaning



```
C:.
|   tree.txt
|
\---MURA-v1.1
    |
    +---train
    |   +---XR_ELBOW
    |   |   +---patient00011
    |   |   |   \---study1_negative
    |   |   |           image1.png
    |   |   |           image2.png
    |   |   |           image3.png
    |   |   |           tree.txt
    |   |   |
    |   |   +---patient00016
    |   |   |   \---study1_positive
    |   |   |           image1.png
    |   |   |           image2.png
    |   |   |           image3.png
```

```
C:.
|   tree.txt
|
+---elbow
|   |   tree.txt
|   |
|   +---test
|   |   +---negative
|   |   |       1.png
|   |   |       10.png
|   |   |       100.png
|   |   |       101.png
|   |   |
|   |   \---positive
|   |           1.png
|   |           10.png
|   |           100.png
|   |           101.png
```

Fig - 4. Original vs Cleaned Dataset

Originally the dataset downloaded from MURA consisted of X-Ray images characterized according to unique patient IDs and type of bones which was not feasible to train a machine learning model. So to make it work, we programmed an effective code for data cleaning purposes that yielded X-Ray images of different bones into the train and test folder for each type of bone. Our code separates the images for each category of the bone as either positive or negative. The image (Fig - 3) shows the glimpse of the original dataset vs the cleaned dataset, obtained for the machine learning task.

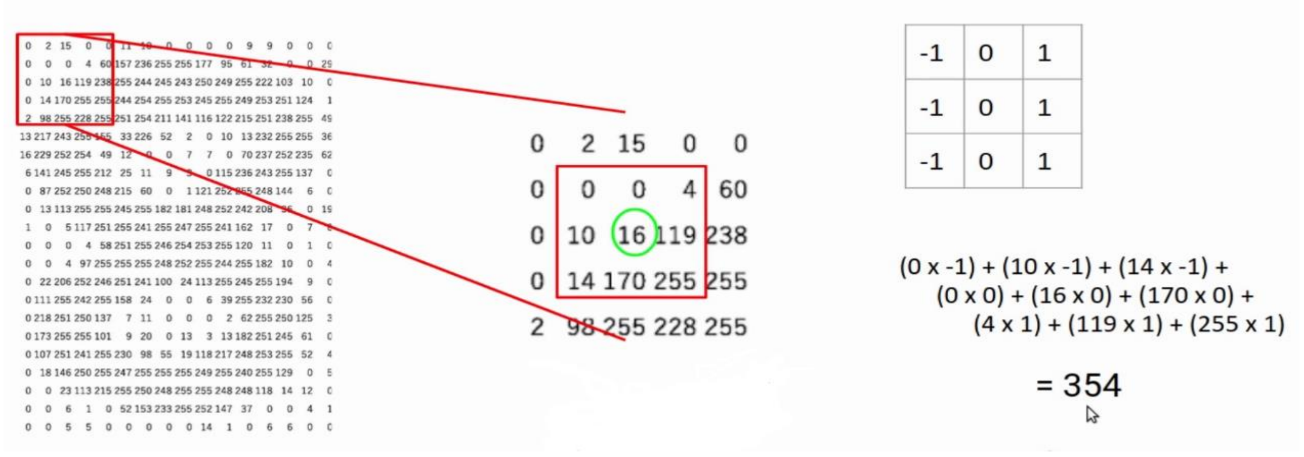
C. Data Preprocessing

Image pre-processing plays an important role to enhance the effectiveness of Convolutional Neural networks. Image preprocessing consists of operations that are applied on most primitive levels. Unless an operation like entropy is applied, these actions are not meant to alter the information of an image. Preprocessing is used to refine picture data by removing undesired distortions or increasing certain visual qualities that are relevant for subsequent processing and analysis. Here, we have used two types of image preprocessing techniques:

1) Edge Detection

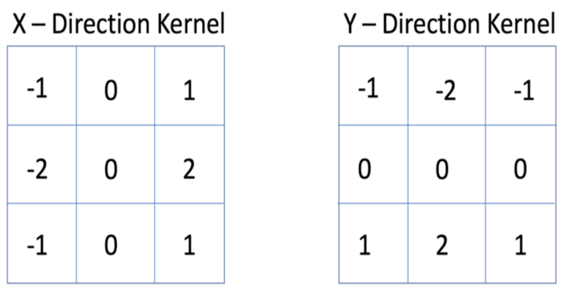
Edge detection (Fig - 4) is a technique used in image processing to determine the boundaries of an object in a given image. To sum up, the edges of an image are the parts of the image that represent the boundary or shape of the object in the image. Additionally, the pixel values near the edge show a significant difference or a sudden change in pixel values. To identify this sudden change and detect edges, the kernel(filter) is used, which performs element-wise multiplication with the image matrix. Here in the example, the portion of the image is selected as the size of the kernel then the element-wise multiplication is performed. Apart from this, the threshold value is decided. If the value of multiplication is greater than the threshold then the pixel(here 16) is considered on the edge, otherwise not.

There are various kinds of filters used for the task of Edge Detection.



0 2 15 0 0
0 0 0 4 60
0 10 16 119 238
0 14 170 255 255
2 98 255 228 255
-1 0 1
-1 0 1
-1 0 1
(0 x -1) + (10 x -1) + (14 x -1) +
(0 x 0) + (16 x 0) + (170 x 0) +
(4 x 1) + (119 x 1) + (255 x 1)
= 354

Fig - 3. Edge Detection Example



X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

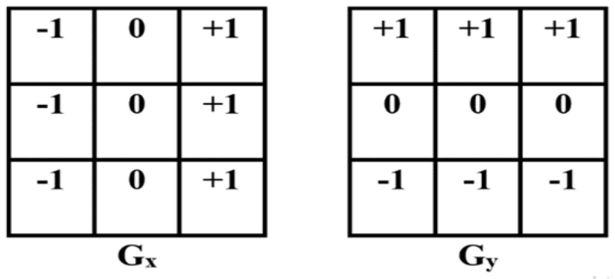
Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Fig - 5. Sobel Kernels

a) Sobel

Edge detection using Sobel is a gradient-based algorithm that computes first-order derivations in an image. The Sobel operator generates either the vector's normal or the equivalent gradient vector at each pixel of an image. It computes the vertical and horizontal derivative approximations using two 3 x 3 kernels or masks convolved with the input image. The Sobel algorithm is very simple and time-efficient and also works well while detecting smooth edges. One drawback of using Sobel is that it does not work well with diagonal direction edges. Here, examples of horizontal and vertical gradients on one image are given. Also, the summation of these two is mentioned with 'sobel'.



-1	0	+1
-1	0	+1
-1	0	+1

G_x

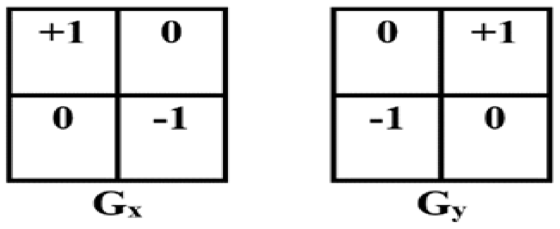
+1	+1	+1
0	0	0
-1	-1	-1

G_y

Fig - 6. Prewitt Kernels

b) Prewitt

Prewitt operator (Fig - 6) is nearly identical to the Sobel operator. It can also detect an image's vertical and horizontal edges. It is one of the most effective methods for determining the orientation and magnitude of an image. It makes use of kernels or masks. Here, the example of Prewitt is the final residue of horizontal and vertical gradients on the image. It is one of the best algorithms for horizontal and vertical edge detection but like Sobel, it doesn't handle diagonal edges.



+1	0
0	-1

G_x

0	+1
-1	0

G_y

Fig - 8. Roberts Kernels

c) Roberts

Through discrete differentiation, Roberts (Fig - 7) gradient-based operator computes the sum of squares the differences between diagonally adjacent pixels in an image. The gradient approximation is then performed. It employs the following pair of 2 x 2 kernels or masks. Roberts edge detection algorithm is very effective and also works well with diagonal edges. The example of Robert's edge detection algorithm is mentioned above.

d) Canny

Canny detects edges using a gaussian-based operator. This operator is not affected by background noise. It extracts image features without affecting or changing them. Canny edge detector employs a sophisticated algorithm derived from previous work on the Laplacian of Gaussian operator. It detects edges based on the following three criteria:

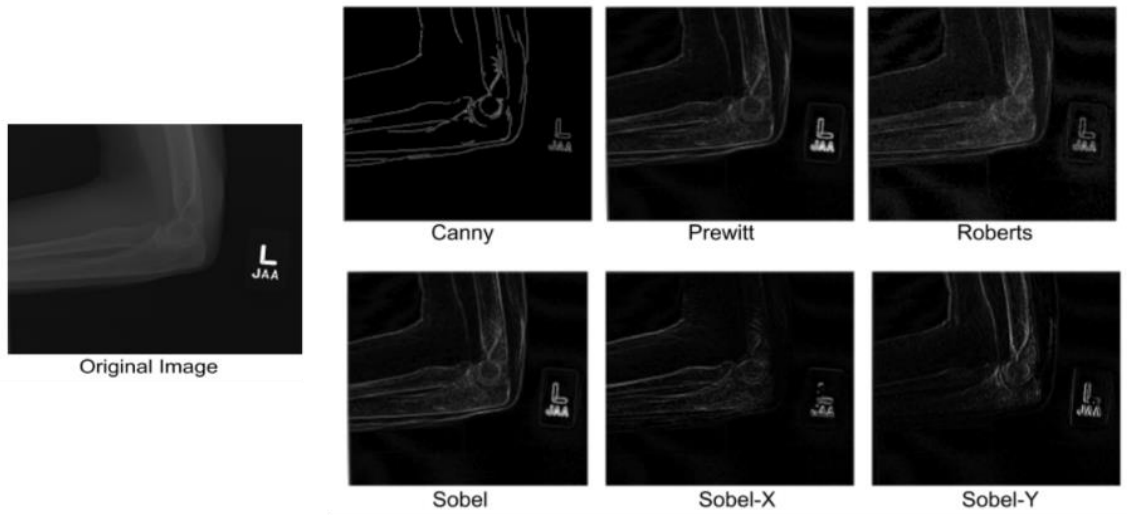


Fig - 7. Edge Detectors applied on real-time X-Rays

- Low Error rate
- Edge points must be accurately localized
- There should be just one single edge response.

The algorithm has very good localization and extracts information without altering the features but it is very complex compared to other algorithms. The example of canny edge detection is mentioned above (Fig - 8).

2) Image Sharpening

Image sharpening [14] is the other kind of image pre-processing technique used to sharpen and enhance the definition of edges in the image. Here we have used Laplacian filters for image sharpening. It is a second-order derivative filter. It detects the image as well as the horizontal and vertical directions as a whole. It is not necessary to use it separately to detect edges as well as horizontal and vertical directions. The sum of this filter's values is 0.

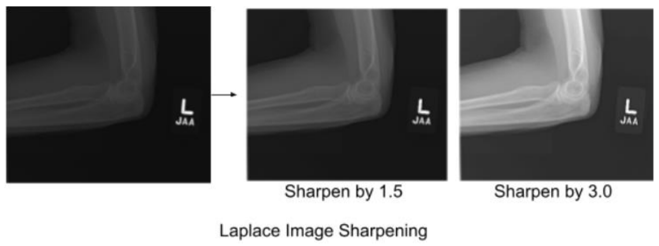


Fig - 9. Image Sharpening using Laplace

(Fig - 9) mentiones two variants of the original image sharpened by factors of 1.5 and 3.0 respectively.

D. Training

We have used Transfer Learning to train the Efficient Net B0 model which used the pre-trained weights of imagenet dataset. Out of all the efficient net models we used the B0 model as it had the least parameters, which saves the training time as well as uses less memory. We have used the TensorFlow library to train the models. Firstly, we removed the output layer and added our custom output layer containing two classes - normal and abnormal. We froze all the layers of the efficient-net B0 model except the last 20 layers so that they can learn from the new dataset. All the Batch Normalization [15] layers were frozen, even if they were in the last 20 layers. After tuning the training parameters, we found out that re-training the last 20 layers gives us the optimal accuracy. We used the Adam optimizer [16] with the learning rate of 0.0001 and categorical cross-entropy as the loss function. Out of the training dataset, we used 20% of the training data as the validation set. We firstly trained a Bone detector model which identifies the correct bone type in the image. After that, we trained different crack detector models for each type of bone.

Initially, the input image is passed through a model which detects the type of bone in the image. According to the output of the Bone Detector model, the image is sent to its respective crack detector model. We have individually trained a crack detector model for every type of bone. These crack detectors can detect abnormality in a specific kind of bone. The performance of these crack detectors is discussed in the following section.

V. EVALUATION

In this section, we will discuss the performance of our trained models. We will compare the models on different metrics like precision, F1 score, precision, and recall. We will also compare the performance of our model with the Stanford ML team's model. Apart from that, we will also compute the memory requirements and the training time for all our models which can give us an estimate of the computational requirements.

A. Model Accuracy

[Table 2] shows the performance of our models on the training set. As we can see that we have achieved an average accuracy of 75% with the Elbow, Forearm, and wrist having accuracies above 80% and finger and Shoulder having accuracies in the mid-seventies. Only Hand and Humerus had accuracies below 70%.

Bone	TP	FN	FP	TN	Accuracy	Precision	Sensitivity	F1 Score
Elbow	52	14	13	74	0.8235	0.8000	0.7879	0.7939
Finger	45	38	2	68	0.7386	0.9574	0.5422	0.6923
Forearm	47	17	7	62	0.8195	0.8704	0.7344	0.7966
Hand	22	44	4	83	0.6863	0.8462	0.3333	0.4783
Humerus	90	7	45	11	0.6601	0.6667	0.9278	0.7759
Shoulder	80	15	21	37	0.7647	0.7921	0.8421	0.8163
Wrist	68	29	1	55	0.8039	0.9855	0.7010	0.8193
Average					0.7574	0.8455	0.6955	0.7389

The F1 score also shows a similar trend with accuracy, having an average value of 0.7389. Elbow, Forearm, Shoulder, Humerus, and Wrist have their F1 score around the 0.8 mark. Finger had an F1 score of 0.6923 while Hand performed poorly on the F1 metric with a score less than 0.5.

On the other hand, the average precision value is 0.8455 with Humerus and shoulder having a precision below 0.8. All other types of bones performed very well on the precision metric. The sensitivity score for most of the bone types was below par. Only Humerus, Shoulder, and Elbow had their sensitivity scores above the 0.75 mark.

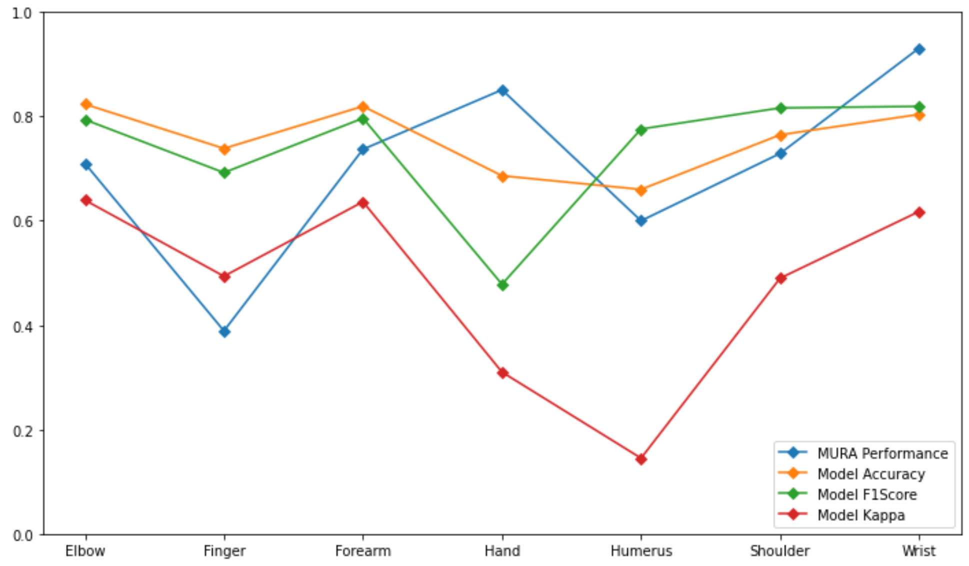


Fig - 10. Model Comparison

B. Model Comparision

(Fig - 10) illustrates the comparison between the performance of our model and the performance of the model of Stanford's ML team. We have plotted the accuracy and the F1 score of our model against the performance of the Stanford Group. As seen in the image, our model outperformed the Stanford Team’s performance on Finger. On other types of bones F1 score showed a similar trend but was below the performance of the Stanford’s ML team. Both accuracy and F1 score performed very well on Elbow, Finger, Forearm, Humerus, and Shoulder.

C. Training Time on GPU

Bone	**Training Time (in Sec)**
Elbow	2328.725
Finger	3376.15
Forearm	245.3
Hand	3726.8
Humerus	613.725
Shoulder	5780
Wrist	7030
Total Time	**23100.7**

TABLE 2. Model Training Time

[Table 3] table shows the training time of every neural network model that we have trained. The training time is directly proportional to the number of images of a particular type of bone. Bones with greater numbers of images had a larger training time compared to bones with small amounts of training images. The total time for training is 23100.7 seconds which is equivalent to 6 hours and 25 minutes. We have trained all our models on Nvidia GTX GEFORCE GPU which has resulted in faster training of our model. Without the help of the GPU, the total training time would be well above 24 hours.

D. Memory Usage

The memory usage states the total amount of memory that would be required to store all our neural networks. This metric is important because memory is a key issue in resource-constrained devices like Mobile phones and Raspberry Pi. If anyone wants to extend this project on a mobile application or pocket computer like Raspberry Pi, then they need to know the exact amount of memory that the models will use.
A single model has a size of 25.9Mb. As we have trained 7 models corresponding to each type of bone and another one for the classification of the bones. So, the total memory needed for the storage of the models will be 8 times 25.9 which is 207.2 Mb.

VI. CONCLUSION AND FUTURE WORK

In conclusion, we can say that our trained model did a pretty good job in detecting cracks in various types of bones. With the help of more training data and medical professionals, we can deploy these models in the real world the beta phase to assist the doctors in determining cracks in the X-ray images.

As discussed in the above section, our model didn’t perform well on Hand and Wrist. To improve this, we can try to use alternate models other than the Efficient Net such as DenseNet, MobileNet, and NasNet.

As data is the most important aspect in training a neural network, we can collect more data from collaborating with local hospitals and collecting their X-ray images. This will provide much more data which would eventually assist the models. We can also apply advanced Image Segmentation techniques such as Faster RCNN for narrowing the area of interest. But for this, we need the help of a medical professional who can label down the images which have visible cracks which will assist the training of Faster RCNN.

Finally, we can add images of more types of bones where fractures frequently occur in the human body such as different bones of legs and skull.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, pp. I-I, DOI: 10.1109/CVPR.2001.990517.

[2] Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger, “Densely Connected Convolutional Networks,” arXiv:1608.06993 [cs.CV]

[3] Andrew G. Howard and Menglong Zhu and Bo Chen and Dmitry Kalenichenko and Weijun Wang and Tobias Weyand and Marco Andreetto and Hartwig Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv:1704.04861 [cs.CV]

[4] Barret Zoph and Vijay Vasudevan and Jonathon Shlens and Quoc V. Le, “Learning Transferable Architectures for Scalable Image Recognition,” arXiv:1707.07012 [cs.CV]

[5] Mingxing Tan and Quoc V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.,” arXiv:1905.11946 [cs.LG]

[6] Chuanqi Tan and Fuchun Sun and Tao Kong and Wenchang Zhang and Chao Yang and Chunfang Liu, “A Survey on Deep Transfer Learning,” arXiv:1808.01974 [cs.LG]

[7] Pranav Rajpurkar and Jeremy Irvin and Aarti Bagul and Daisy Ding and Tony Duan and Hershel Mehta and Brandon Yang and Kaylie Zhu and Dillon Laird and Robyn L. Ball and Curtis Langlotz and Katie Shpanskaya and Matthew P. Lungren and Andrew Y. Ng, “MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs,” arXiv:1712.06957 [physics.med-ph]

[8] Yangling Ma, Yixin Luo, “Bone fracture detection through the two-stage system of Crack-Sensitive Convolutional Neural Network,” Informatics in Medicine Unlocked, Volume 22, 2021, 100452, ISSN 2352-9148, https://doi.org/10.1016/j.imu.2020.100452.

[9] Shipra Saxena. 2021. Edge Detection | Extracting The Edges From An Image. Analytics Vidhya Article, https://www.analyticsvidhya.com/blog/2021/03/edge-detection-extracting-the-edges-from -an-image/

[10] N. Kanopoulos, N. Vasanthavada and R. L. Baker, "Design of an image edge detection filter using the Sobel operator," in IEEE Journal of Solid-State Circuits, vol. 23, no. 2, pp. 358-367, April 1988, DOI: 10.1109/4.996.

[11] Prewitt, J.M.S. (1970). "Object Enhancement and Extraction". Picture processing and Psychopictorics. Academic Press.

[12] LS. Davis, "A survey of edge detection techniques", Computer Graphics and Image Processing, vol 4, no. 3, pp 248-260, 1975

[13] J. Canny, "A Computational Approach to Edge Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[14] J.G.M. Schavemaker, M.J.T. Reinders, J.J. Gerbrands, E. Backer,Image sharpening by morphological filtering,Pattern Recognition,Volume 33, Issue 6,2000,Pages 997-1012,ISSN 0031-3203,https://doi.org/10.1016/S0031-3203(99)00160-0.

[15] Sergey Ioffe and Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” arXiv:1502.03167 [cs.LG]

[16] Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization,” arXiv:1412.6980 [cs.LG]