Classification of binary fracture using CNN.

CHITTAJALLU, S.M., MANDALANENI, N.L.D., PARASA, D. and BANO, S.

2019

Classification of Binary Fracture Using CNN

Sai Manvitha Chittajallu
Dept of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Vaddeswaram, India
manvitha71@gmail.com

Navya Lakshmi Deepthi Mandalaneni
Dept of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Vaddeswaram, India
mnldeepthi@gmail.com

Dhanush Parasa
Dept of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Vaddeswaram, India
cody20734@gmail.com

Shahana Bano
Assoc Professor
Koneru Lakshmaiah Education Foundation
Vaddeswaram, India
shahanabano@icloud.com

Abstract **One of the major problems faced by any living organism since infancy are Musculoskeletal Injuries. To keep it quite simple musculoskeletal injuries are a range of disorders involving muscles, bones, tendons, blood vessels, nerves and other soft tissues. However one of the most common forms of musculoskeletal injuries are Fractures. Fractures are one of the most prevalent sores that are faced by any living organism. They are also easily overlooked by the best of physicians. Even with the help of an X-Ray they are one of the hardest symptoms to diagnose. We believe that we can provide a solution to this problem by implementing Convolutional Neural Networks (CNN) Image Processing Algorithms into the field of Medicine.**

We have designed a model using three layers of architecture which has been properly trained to identify the X-Ray images that have Fractures. To accomplish this we used large datasets that consist of 200 images of human hands, ribs, legs, and neck. These large datasets are clearly segregated to identify those images which contain fractures from those images which are perfectly fine. The Results gave us accurate predictions using some graphical representations as well as epochs of the various patients

IndexTerms: CNN, Convolutional, Pooling, Relu, Flattened, FullyConnceted, binary entropy, sequential

I. INTRODUCTION

We have been able to design a model to identify whether the given CT scan has a fracture or not by using Convolutional Neural Network (CNN) Image processing algorithm. We were able to successfully accomplish this by training our model with several image datasets of CT scans as we have discussed previously. All these Datasets are closely studied by our model and stored in the forms of multiple libraries within itself.

Some of these libraries behave as the OS of our model and they grant access to the various file systems within our model. Once access is granted the model may now begin reading the various image datasets from training as well as the testing directories. To make sure that our model performs well across all CT scans and isn't biased towards a single scan we use the Random function. Random shuffles all of the image datasets sot that the model is able to properly learn each and every CT scan irrespective of part.

We have prepared and processed our model using Python as its backbone. For our model to communicate with us regarding the output of each CT scan we used two unique libraries from Python. We imported Matplotlib as well as its numerical extension Numpy. In the world of numerical computing these 2 libraries hold the utmost importance in any python program. Python was not initially designed to perform numerical computing programs at all. However due to its short and easy to understand syntax it wasn't long before it caught the attention of the scientific and engineering community. Not before long these

2 libraries were created to help scientists as well as mathematicians to perform complex numerical algorithms. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. This is a great advantage to us as we will be able to see the results in a visualized manner which helps us identify how accurately our model has been able to function.

We were able to design an efficient algorithm implementing Computer Neural Networks in order to train our model. The algorithm consists of a step by step process which helps our model understand how the CT scan of perfectly normal human would look. Once our model has achieved this ability all those Scans that differ from that of a normal Human Being will be flagged as a fracture. Our algorithm consists of various layer which are mainly the Convolution layer, Pooling layer, Flattening layer, and Full Connection layer. Let's take a microscopic view of each layer step by step. The Convolution layer is the most important layer in our model. Its major responsibility is to properly understand the image by transforming it into an input image which is in the Dimensional Array which is further reduced with the help of the feature detector in order to form the feature map. To give you a small understanding let's take a look at an algorithm that has been designed and modeled to identify a Smiley face. Let's look at how the model transforms the image of the sad face with the help of the Convolution layer.

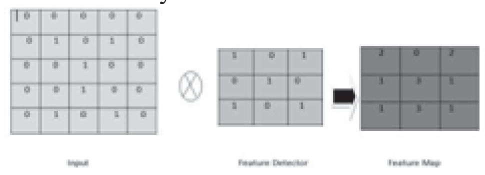


Fig. 1.Convolutional Layer

If you take a close look at the input image you will understand how it was formed. Observe how the model mapped the smiley face with the forms of 1's, the remaining area which was left untouched was mapped with 0's. This input image is further reduced to the feature map.

The Pooling Layer which is also commonly known as the downsampling layer. It is responsible for reducing the dimensionality of our obtained feature map so that assumptions can be made from combining the sub-regions. This is accomplished when the model takes the maximum values from the sub regions to form a smaller image. Let us take an example of the following input image which is stored in a 4x4 matrix after *Convolution*

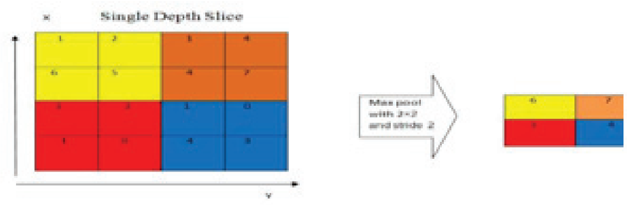


Fig. 2.Max Pooling Layer

We can observe how the pooling layer divided the larger 4x4 array into 4 sub-regions from which their maximum values were taken in order to establish the downsized 2x2 array

Once the Convolution and Pooling layer have established a proper downsized image the Flattening Layer comes into the picture. This is one of the easiest layers to understand as there isn't much to do at all. As we can deduce from the name this column so that it can be easily inserted into our artificial neural network later on in the future. Observe how the following image stored in an 3x3 matrix is flattened into a single column.

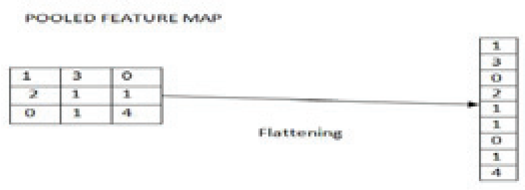
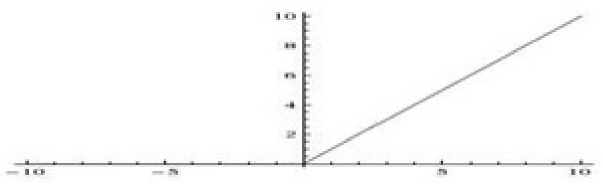


Fig. 3.Flattened Layer

The last layer that is used in any model implementing Computer Neural Networking is the Fully Connection layer. It is responsible for feeding the flattened image into the Artificial Neural Network. From here on out the Artificial Neural Network plays a vital role in our model as it takes the flattened data and it combines all of its features into a much wider variety of attributes that make it much easier for the convolutional network to classify the given images.

Each layer consists of a special unit which plays a vital role in Computer Neural Networks which is known as the Rectified Linear Unit (Relu). It is also widely known as the Activation function in deep learning models. The Relu function returns 0 if it receives any negative input, for any other positive value it return x. This can be depicted as follows ….

$$f(x) = \max(0,x) \quad (1)$$



Now that we have gone through each and every layer in a brief manner let's take a look at how we will combine all of these layers in order to form our required model.

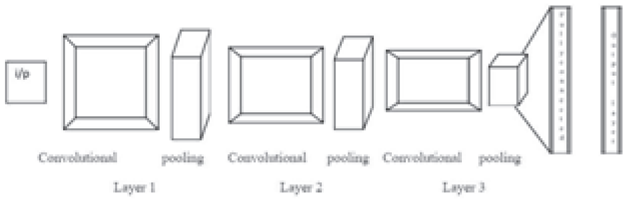


Fig. 4.Convolutional Neural Network

II. PROCEDURE

Now that we have gotten over the basics of any CNN model lets take a look into how we have constructed our model implementing everything we have read about so far. There are 3 basic steps in the implementation of our model

- Importing Data-Sets into our Model
- Training our Model
- Processing Data Acquired From Model

Let's take a closer look at how our model works step by step.

A. Importing Data-Sets:

In order to make sure that the model we have designed functions properly we will need to first import 2 major key components. We must import all the required packages that we will implement later on as well as import our data sets. This process can be further downgraded into the following stage

1. Importing Packages and Data-Sets

The first thing that we must do is import all of our required packages as well as our data-sets. Importing all of our packages isn't completed at all. It's the same process as that of any other model out in the world. Where it gets interesting is the process we must follow to import our data-sets. We have used the software Tensorflow to create our model. This software is linked with google and it uses google drive as its cloud storage. In order to import our data sets into our model we must first store all our data sets in Google Drive. From here on out once all our Data-Sets have been successfully stored we will copy its sharable link and come back to our software Tensorflow so that we can import the Data-Sets. We will provide the link to our folder within the software. Once the software tries to access our cloud drive google will send us an authentication link which we must input in our software so that it may be provided complete access to the folder consisting of our Data-Sets in our cloud drive.

2. *Splitting of Data-Sets:*

Once our model achieves complete access to or Data-Sets it will begin a process which splits our Data. This process is completely user dependant where the user may select the portion of the Data-Sets which must be used to Train our Model while the rest is used to test our Model. To give u a clear understanding if the user was to input this splitting process as 4:2 ration for a Data Set containing 100 entries. In such a scenario the model will use 80 entries to Train the model while the remaining 20 are used to Test it. In our model we have decided to give it a ratio of 9:1 for any given Data-Set.

3. *Converting Data to Appropriate Form:*

Before the data can processed by our model there is one final step. Our Model has been created in such a form to access the numerical data and have it further analysed. Our Model will find our Entries as garbage if they are not properly converted to a format that our model can recognize. The first step to accomplish this is by gray-scaling all of our images. Grayscaling is the process of converting images that are in RGB format to a B&W format. From here these images as transformed into Matrix Format. This Matrix can now be understood by our model as a matrix is nothing more then numerical data stored in a 2-D array.

B. *Training Our Model:*

Once the first part of our procedure is completed and we have successfully imported as well as converted our Data-Set we can now begin to train our model. As we are well acquainted with the various layer and their functions in any CNN model let's take a look at other functions within our Model.

1. *Passing Data Through Our Model:*

To pass the whole Data-Set at once through our model will be merely impossible. Our Model will not be given the adequate time to learn from each entry. From this we stumble upon a problem where we cant pass the entire data set and to pass each entry one after another would be simply a time taking process. In order to overcome this our model will divide our Data-Set into several intrarions to decrease the workload. The number of entries in each iteration is referred to as the batch size. Assume there are 50 entries and we were to say that the maximum batch size is 10. In such case our model will need to perform 5 iterations before the Whole Data-Set is passed through it.

2. *Binary Entropy Function:*

We will be using the Binary Entropy Function within our model in order to decide whether the given entry depicts that of a patient with or without a fracture. Our model will analyze all those patients with fractures as '1' and all those without a fracture as '0'. So basically …..

1 => CT Scan Contains Fracture
0 = > Perfectly Normal CT Scan

Binary Entropy function is able to analyze statistically how the scans which contain fractures are mapped within our model. On analysing multiple entries with fractures it will be able to predict how a scan contacting a fracture it will be mapped numerically. The output of this function is either high which is depicted by '1' or low which is depicted by '0'. The binary entropy Function can be numerically represented as follows….

$$-1N\sum i=1N[yi*\log(y\hat{}i)+(1-yi)\log(1-y\hat{}i)] \tag{2}$$

C. *Processing Data Aquired From our Model:*

The final step to our procedure is making sure that all the data that our model has analyzed is properly converted to a user understandable format. As we have looked into how the our model analyses the data-sets in the form of matrices and then it determines whether the scan has a fracture or not using our Binary Entropy Function. All of these functions are numerical and are in mathematical terms. It would be very hard for an average human being to understand such complex algorithms so it is up to us to make sure that all of these are properly transformed into a user understandable format. We will first analyze our data-set to get a clear understanding of how many entries were identified as fractures to those that were perfectly fine. We then plot all of our data with the help of Matplotlib as well as the numpy libraries that are in python so that the user can get an all around picture of his entries.

These are the 3 basic steps that we will have to follow to make sure that our model is properly working. We will follow these 3 main steps each and every time that we require the analysis of a Data-set. As we provide our model with more and more entries it will begin to become more and more accurate as it will learn and analyze each and every Data-set and use it for future entries.

RESULTS

The outputs on the tested data are

```
I am 96.46% sure this is a fracture
```

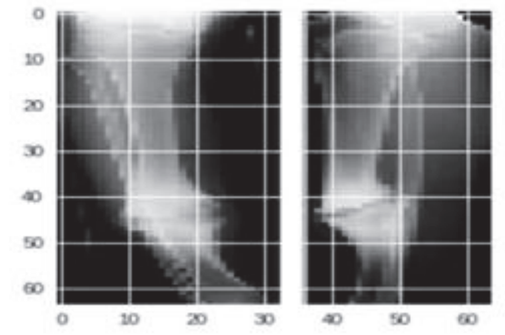
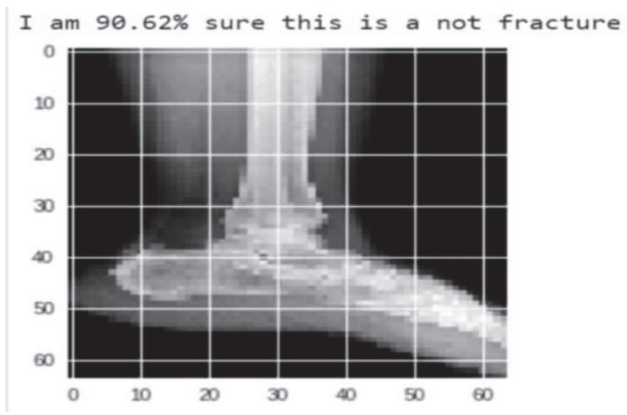
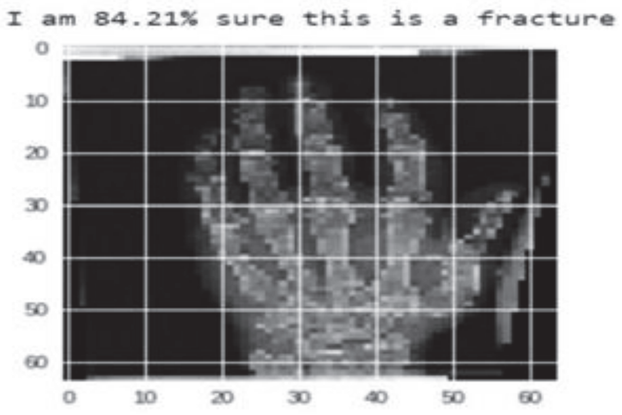
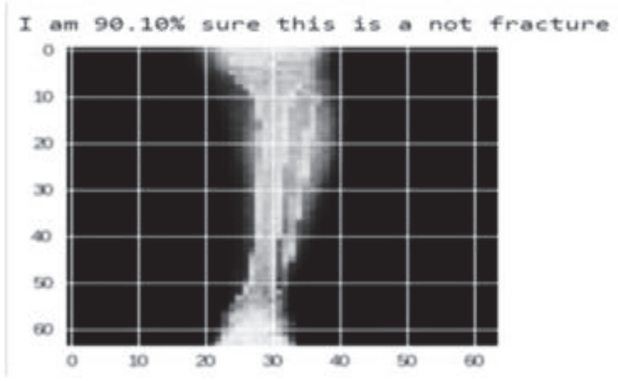
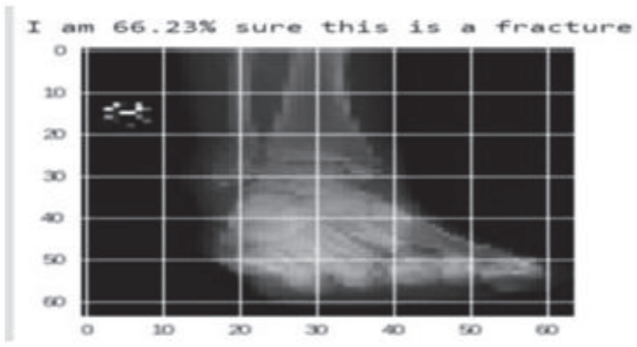


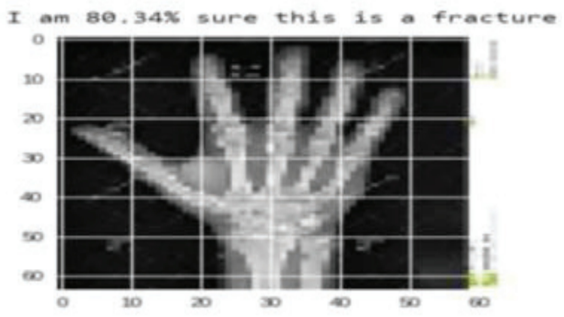
Fig. 5.Test Image(1)

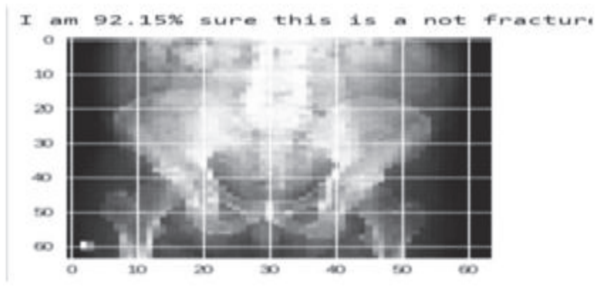

```
I am 90.62% sure this is a not fracture
```
Fig.6.Test Image(2)

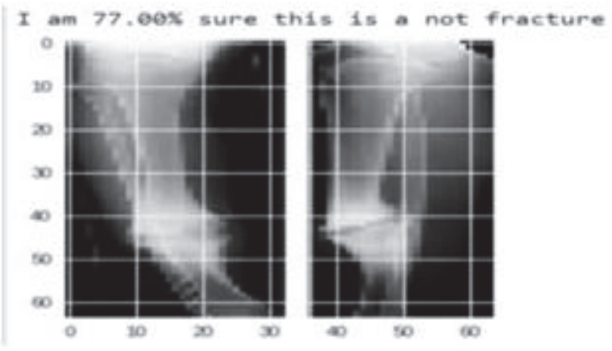

```
I am 84.21% sure this is a fracture
```
Fig.7.Test Image(3)

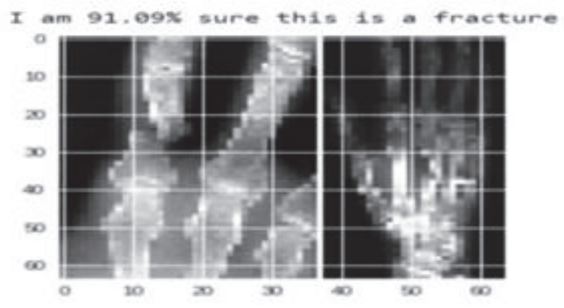

```
I am 90.10% sure this is a not fracture
```
Fig.8.Test Image(4)


```
I am 66.23% sure this is a fracture
```
Fig.9.Test Image(5)


```
I am 80.34% sure this is a fracture
```
Fig.10.Test Image(6)


```
I am 92.15% sure this is a not fractur
```
Fig.11.Test Image(7)


```
I am 77.00% sure this is a not fracture
```
Fig.12.Test Image(8)


```
I am 91.09% sure this is a fracture
```
Fig.13.Test Image(9)

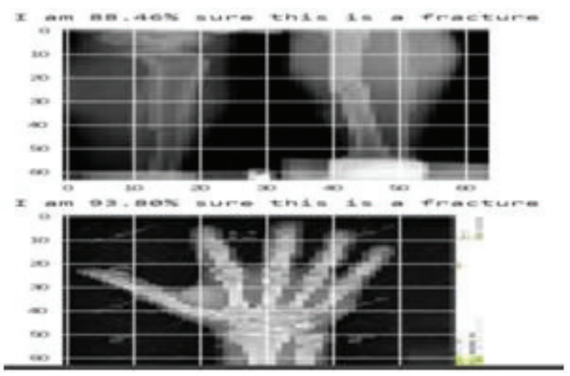

```
I am 88.46% sure this is a fracture
I am 93.80% sure this is a fracture
```
Fig.14.Test Image(10,11)

TABLE I. ACCURACY TABLE OF MODEL

SNO	Parts	Accuracy(avg)
1	Hands	80.45%
2	Legs	84.75%
3	Ribs	80.65%
4	Other	86.75%

REFERENCES

[1] Wint Wah Myint, Khin Sandar Tun, and Hla Myo Tun, " Analysis on Leg Bone fracture detection and classification using X-Ray images" Machine Learning Research. Vol. 3, No. 3, 2018 pp. 49-59. doi:10.11648/j.mlr.20180303.11

[2] B.Gajjar, S.Patel, A. Vaghela," Fracture Detection in X-ray images of long bone", vol. 5, Indus University, 2017.

[3] Convolutional Neural Networks for Automated Fracture Detection and Localization on Wrist Radiographs Yee Liang Thian, MBBS, FRCR , Yiting Li, BEng, Pooja Jagmohan, MBBS, FRCR , David Sia, MBBS, FRCR , Vincent Ern Yao Chan, MB, BCh, BAO ,Robby T. Tan, PhD, 2019https://doi.org/10.1148/ryai.2019180001

[4] Plant Leaf Disease Detection using Deep Learning and Convolutional Neural Network Anandhakrishnan MG Joel Hanson1 , Annette Joy2 ,Jerin Francis Volume 7, 2017

[5] Shin, H.R , Gao, M, Lu, L, Xu, Z, Nogues, I., …Summers,.R. M. (2016). *Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. IEEE Transactions on Medical Imaging, 35(5), 1285 1298.*doi:10.1109/tmi.2016.2528162

[6] Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., & Liang, J. (2016). Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Transactions on Medical Imaging, 35(5), 1299 *1312.*doi:10.1109/tmi.2016.2535302

[7] Thian, Y. L., Li, Y., Jagmohan, P., Sia, D., Chan, V. E. Y., & Tan, R. T. (2019). Convolutional Neural Networks for Automated Fracture Detection and Localization on Wrist Radiographs. Radiology: Artificial Intelligence, 1(1), e180001. doi:10.1148/ryai.20191800

[8] V L N Sai Komal Pendela Sai Krishna Yadlapalli Sri Kanth Yenumula Shahana Bano Convolution Neural Network for Tumor Detection in Medical Images ISSN NO:2236-6124

[9] Sai Krishna Yadlapalli P V L N Sai Komal Pendela Sri Kanth Yenumula Shahana Bano Segmentation Of Brain Tumour From Medical Images Using Watershed Algorithm ISSN NO:2236-6124