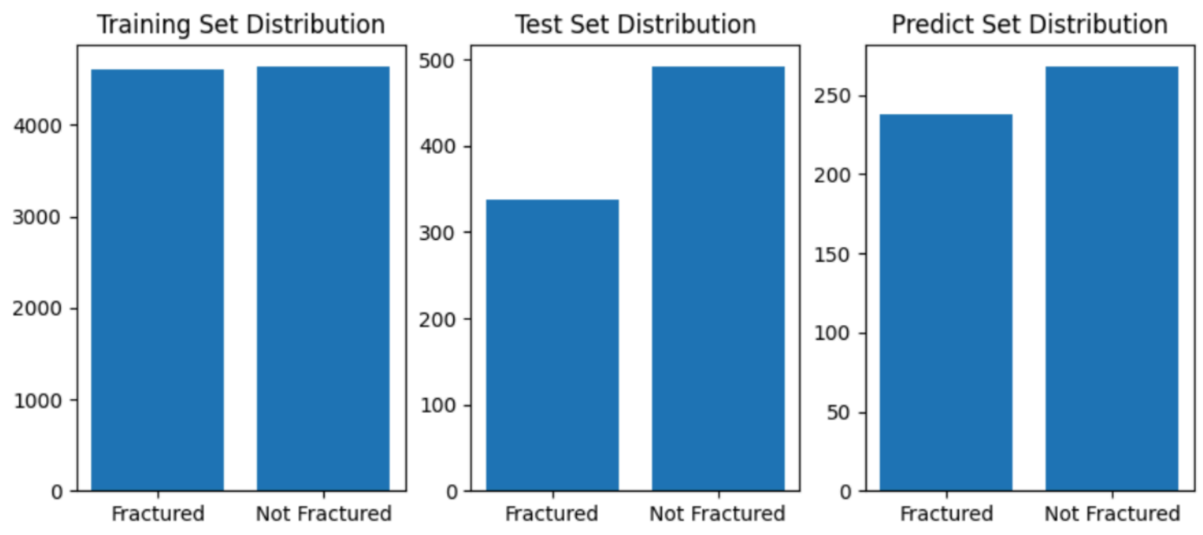
Exploratory Data Analysis

First, to make it easy to understand we renamed the test directory to predict and renamed the val directory to test.

Let's look at the 3 directories distribution between classes :



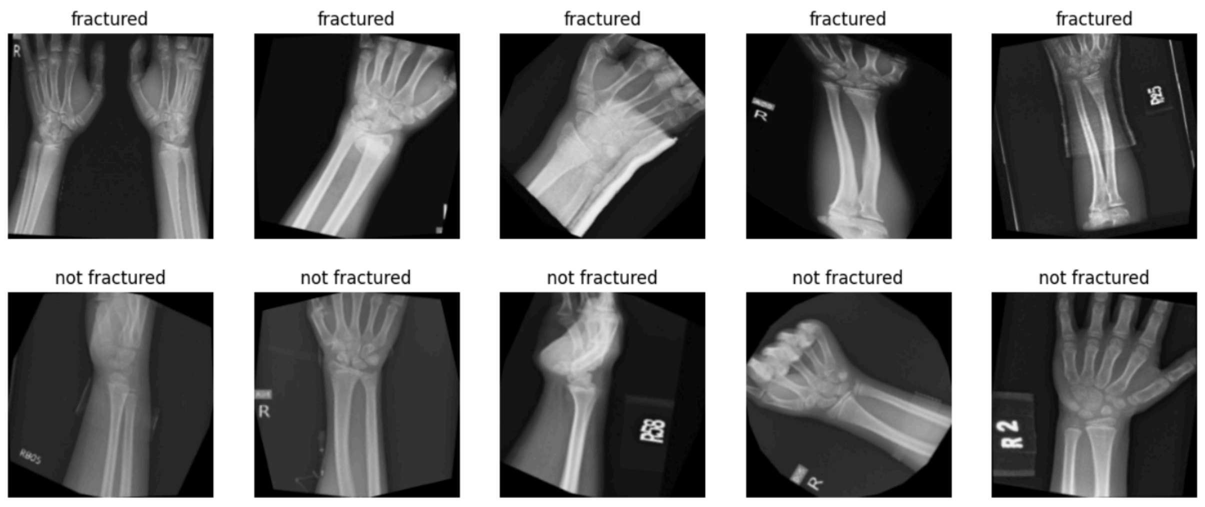
We observe a slight imbalance in the test and predict sets but nothing of great magnitude.

Now lets see if all images have the same dimensions :

Results from code : Sample image sizes from training set: [(617, 1024), (2328, 2928), (761, 1024), (408, 1024), (1517, 2021)]

We can see that images have different sizes, we will need to keep that in mind and resize all images to the same size in the preprocessing step for our CNN to work.

Here is a sample of images from our train directory :



We will be performing data augmentation to the train set to increase our model's performance.

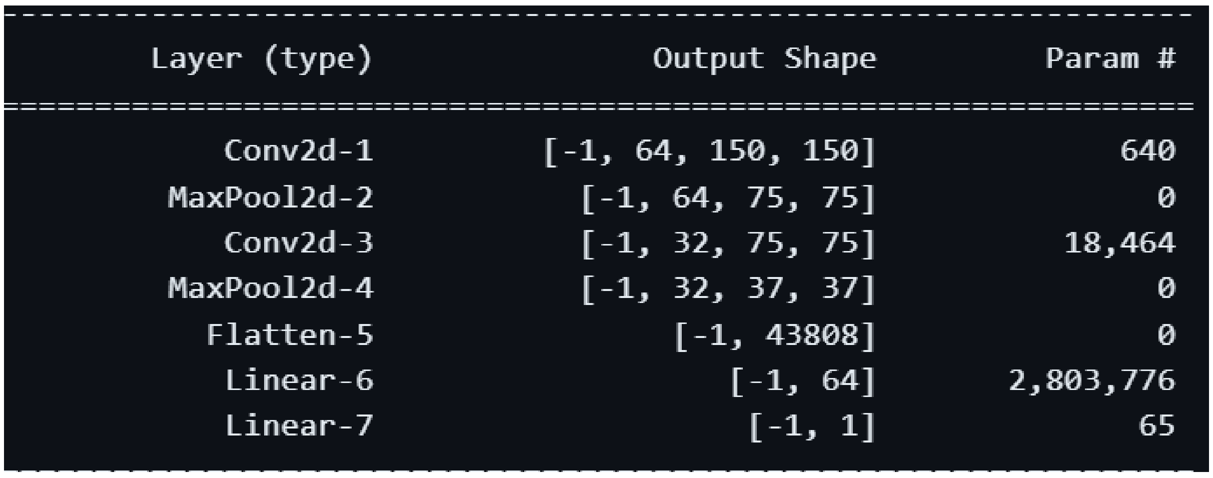
Here is an example of the same image augmented multiple times :

Data preprocessing

Here are the 5 steps we chose to take
1. Loading images
2. Grayscale conversion
3. Resizing to 150x150 with padding to keep the aspect ratio
4. Normalizing pixels from [-1 , 1]
5. Data augmentation on the training set only

Model development process

We chose convolutional neural networks since they perform well on images.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 64, 150, 150]             640
         MaxPool2d-2          [-1, 64, 75, 75]               0
            Conv2d-3          [-1, 32, 75, 75]          18,464
         MaxPool2d-4          [-1, 32, 37, 37]               0
           Flatten-5               [-1, 43808]               0
            Linear-6                  [-1, 64]       2,803,776
            Linear-7                   [-1, 1]              65
----------------------------------------------------------------
```

We trained the model with the Adam optimiser and the binary cross entropy loss function.

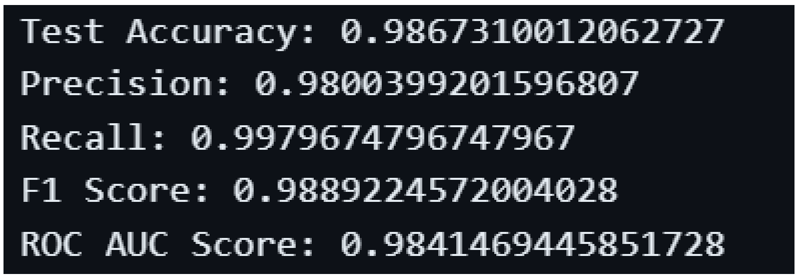
Training was performed using the pytorch library and a RTX 4060 gpu.

First test was done with 15 epochs. We later tested with an image resizing of 224x224 and got slightly worse results so we stuck with 150x150.

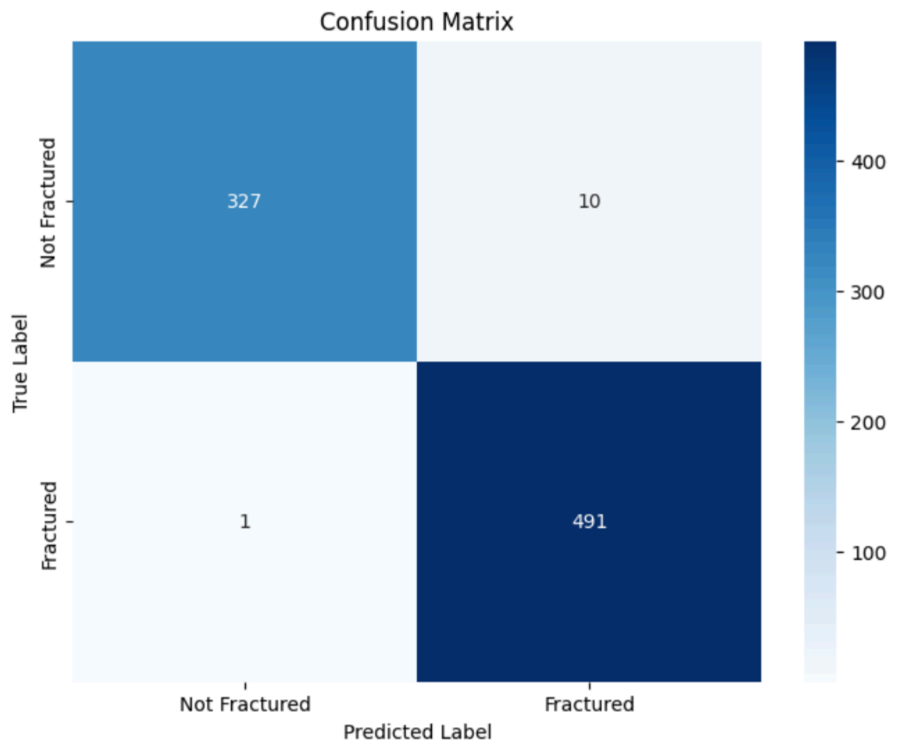
We noticed by increasing epochs and looking at validation loss per epoch that the model was reaching values as low as 0.05 validation loss.

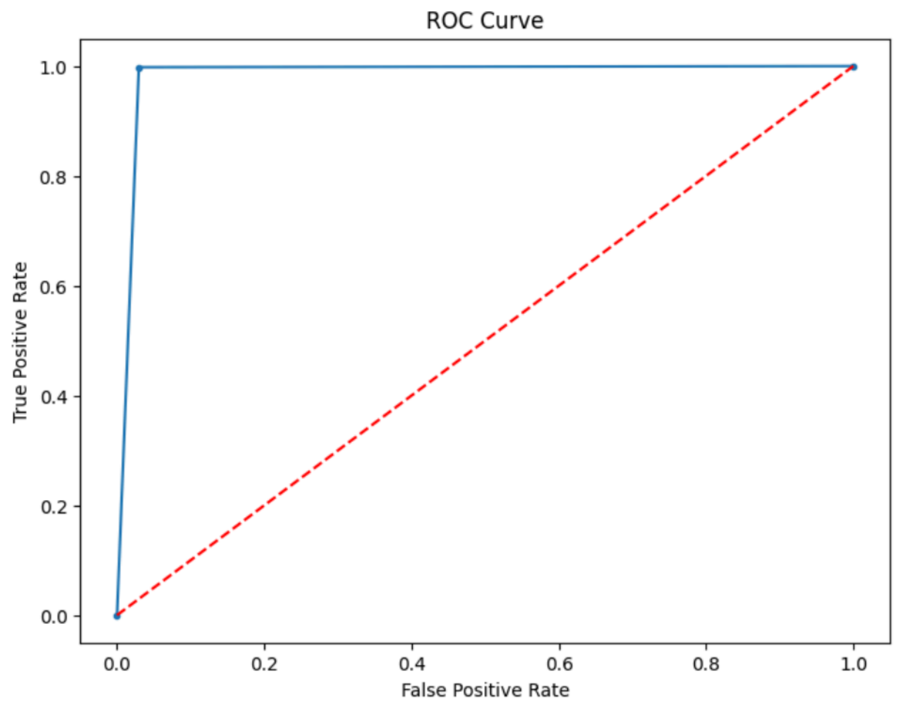
To optimize the model we decided to include a threshold validation loss number in our training loop.
That way, when the validation loss would drop below 0.05, the training loop would stop and we would get a high performing model. This was reached after 40 epochs.

Evaluation metrics

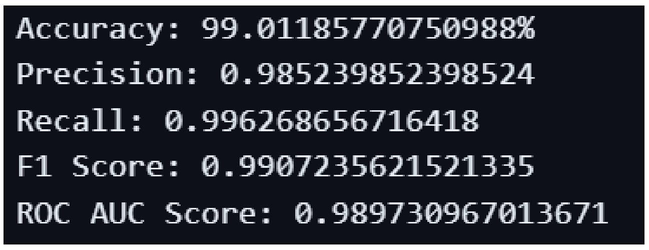


```
Test Accuracy: 0.9867310012062727
Precision: 0.9800399201596807
Recall: 0.9979674796747967
F1 Score: 0.9889224572004028
ROC AUC Score: 0.9841469445851728
```

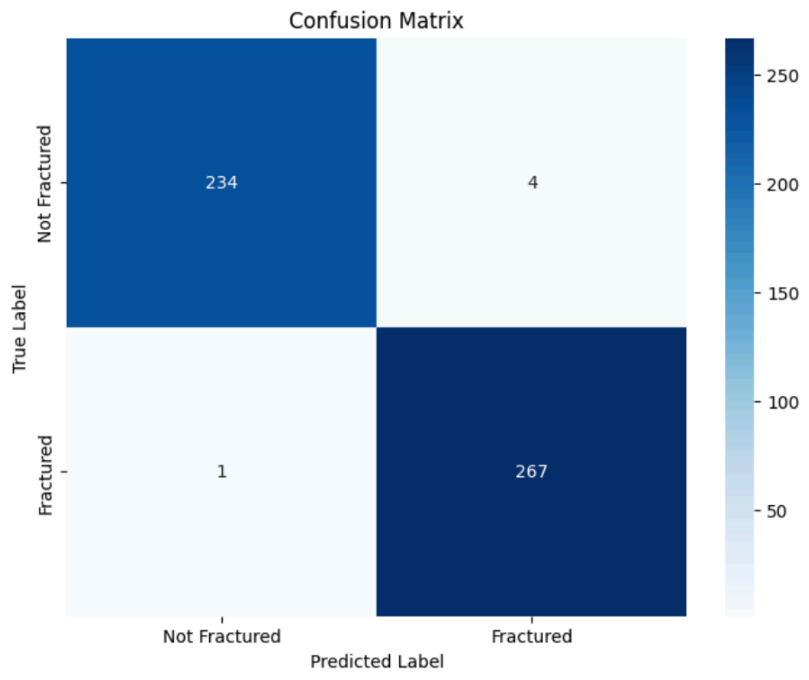




Prediction metrics (since the predicting dataset is tagged)



```
Accuracy: 99.01185770750988%
Precision: 0.985239852398524
Recall: 0.996268656716418
F1 Score: 0.9907235621521335
ROC AUC Score: 0.989730967013671
```



Conclusions about the model's performance.

Performance :

Quite high at the moment. Now, we could decide to include a threshold in our predictions in order to maximize recall and be almost sure to include all the true broken bones predicted as broken bones. (only 1 image was misclassified as not being a broken bone)

That way, we would never miss a patient with a broken bone but the doctor will spend more time with cases that do not have a broken bone but are classified as if they did.

Limitations :

Here the model is trained with X Rays only, we tested it and if you predict with a normal pipe and a broken pipe it will tell you the broken pipe is a broken bone.

Meaning that for our predicting accuracy to be in the same ballpark, we need to predict only with pictures we know are X Rays of human parts.

The dataset must represent the real-world scenarios well. If the training data is not diverse enough, the model might not generalize well to unseen data.

Also, the lower the image quality, the worse the X Ray has been done and if the fracture is very small or unnoticeable the less precise the model will be at predicting and we might not catch the broken bone.

Selling point :

If a doctor's ability to identify a broken bone from an Xray is lower than 99%, then our model can help doctors identify patients with broken bones.

The model can also be used as a way for doctors to save time and only focus on individuals classified as having a broken bone.
Obviously, doctors need to listen to the patient's symptoms and pains in order to include all patients potentially at risk.

Our model predicts in less than a second if an Xray is a broken bone or not. Allowing doctors to spend less time looking at the Xray's.