

01.116 AI for Healthcare

AI Tool for Occult Hip Fracture Detection

Project Report



Rahul Parthasarathy 1003718

Varsha Venkatesh 1003646

Huang Zhibo 1003558

Table of Contents

Abstract	3
1 Introduction	3
2 Related Work	4
3 Methodology	4
3.1 Classification	4
3.1.1 Method 1 - Using a Single Classifier to Classify 9 Grids	4
3.1.2 Method 2 - Using a Single Classifier to Classify Single Grids	6
3.1.3 Method 3 - Using a Single Classifier to Classify the Whole Image	6
3.2 Test Dataset Distribution	6
3.3 Experiments	7
3.4 Results	7
3.4.1 Method 1 Results	8
3.4.2 Method 2 Results	9
3.4.3 Method 3 Results	10
4 Object Detection	11
4.1 Motivation	11
4.2 Dataset Pre-processing	11
4.3 Model Selection	13
4.4 Experiments	15
4.5 Results	15
5 Conclusion	17
5.1 Classification	17
5.2 Object Detection	18
6 Code Availability	18
7 Citations	18

Abstract

Hip fractures are a leading cause of death and disability among older adults. Hip fractures are also the most commonly missed diagnosis on pelvic radiographs, and delayed diagnosis leads to higher cost and worse outcomes for the patient. Numerous attempts have shown promise or helping diagnose and classify diseases using image data. However, there exists many limitations including multiple types of fractures, patient and hospital process variables and presence of implants. We initially developed a classification model that proved to us that the model is indeed learning, shown through salient mapping. We developed an object detection deep learning model using YoloV5, trained with 158 bounding-box-annotated pelvic X-ray images. The model yields a precision of 0.75 and Mean Average Precision (mAP) of 0.77. This model demonstrates potential for the clinician's workflow to be complemented by deep-learning methods to provide more efficient and accurate patient diagnosis. Further research, exploration, training and testing is needed to achieve results and outcomes that will allow the model to be implemented in reality.

1 Introduction

Hip fractures are a leading cause of death and disability among older adults. Hip fractures are diagnosed on the basis of these images and are the most frequently occurring fracture type in elderly people; however, the misdiagnosis rate ranges from 4 to 9%¹. Hip fractures are the most commonly missed diagnosis on pelvic radiographs, and delayed diagnosis leads to higher cost and worse outcomes. The current process of identifying potential fractures is by inspecting the X-Ray, gauging a rough idea of the situation and sending the patient for further scans. We aim to remove this middle step of inspecting the X-Ray by automating the process using Computer Vision and Deep Learning.

The goal of our model is to predict whether the X-Ray looks completely normal or whether there is a potential fracture, in which case further scans should be done. For this particular case, sensitivity is a more important metric as compared to specificity. One of the main constraints of this project is the size of the fracture/injury; the fractures are generally very small and not easily detectable by non-clinicians

To detect a fracture, we approached to solve the problem in two approaches - Image classification and object detection. Classification aims to classify the given hip X-ray images into fracture and no-fracture. Object detection involves detecting if the image has a fracture and locate the fracture.

2 Related Work

Previous studies have shown models that have achieved an accuracy of up to 96.1% for detecting fractures in pelvic X-rays and radiographs². One of these studies include deep learning models that can detect in radiographs patient and hospital variables, and analyse their contribution to the inner workings of a fracture detection model³. An Inception v-3 deep learning model used to featurize radiographs into an embedded 2048-dimensional representation. Another study addresses the limitation of different forms of fractures which leads to inaccurate results due to the conditions at the time of the imaging⁴. To overcome this limitation, they utilised an encoder-decoder structures neural network that incorporates radiology reports along with the training images. These reports reflect the radiologists experience and expertise, and complement the image by capturing possible missed information to improve the model's decision making. The last study in our literature review developed a universal deep learning algorithm, PelviXNet, which detects all trauma related findings in a pelvic radiograph⁵. They used a weakly supervised approach with point-annotated images and achieved performance comparable to radiologists.

3 Methodology

As of 23rd April, we were given a total of 1352 images out of it, 117 fracture images, the images are separated into 9 batches. For each batch of images, the clinician provided us with CSV files that indicate if the image is a fracture or normal, and if it is a fractured image, a coordinate for the bounding box will be given. For all the fracture image given, the fracture will be displacement fracture⁶. We approached to solve the problem in two approaches - Image classification and object detection.

3.1 Classification

We approach the problem through classification as it is a baseline of the problem. Classification aims to classify a given fracture into a normal fracture image.

3.1.1 Method 1 - Using a Single Classifier to Classify 9 Grids

A fracture in an X-ray can be very small (refer to Fig 2), as compared to the entire image, which comes in very high definition ranging from 2 million to 12 million pixels. To overcome this problem, we decided to split the image into 9 parts, each part was resized to 224 x 224 pixels before feeding into a model. To determine if each of the parts has a fracture, we check if the image has an overlap of 20% with the bounding box. We use a single deep learning model to train on all images that have been sliced out of the X-ray. This method has 12078 normal images and 318 fracture images for training.

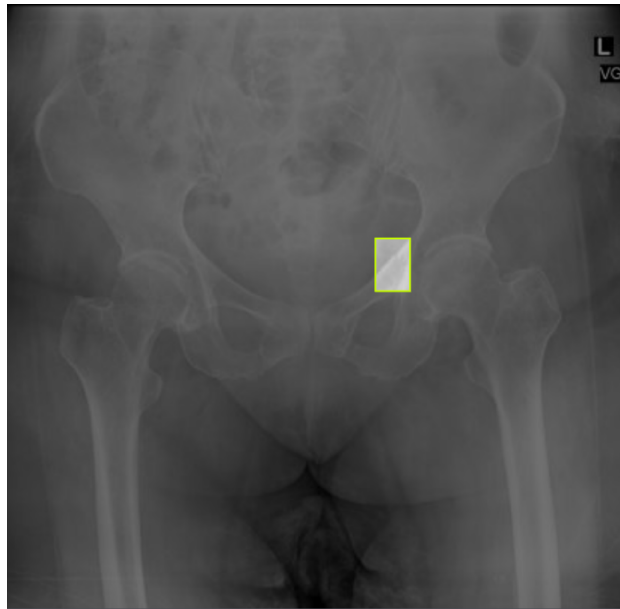


Figure 1 (Fracture X_ray with very small bounding box) from the clinician

This method has increased the original datasets by nine times, but it also dilutes the fracture image future. This is because the fracture occurs at a very small part of the image after slicing the image into 9 pieces, usually only one or two out of the 9 slices will have the fracture in the images.

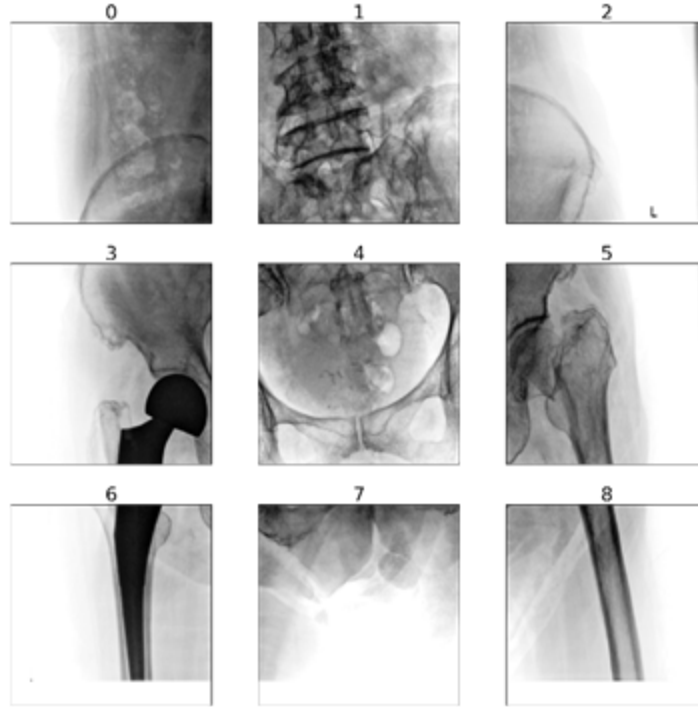


Figure 2 (A typical X-ray Images that slice into 9 images, each image comes with a grid index on top of it) with a red bounding box where the fracture located

3.1.2 Method 2 - Using a Single Classifier to Classify Single Grids

Similar to Method 1 (Section 3.1.1), we sliced an X-ray image into 9 images and each sliced image was given a grid index (refer to Fig 2). We used a grid index to determine the original location of the new image in the original X-ray. From Table 1, we can observe the number of fractured images is very small compare to normal images. This leads to a very imbalanced dataset for training, but at the same time, we can observe that most of the fracture occurs at the grid index 3, 4, 5 and 7. As most of the X-ray take taken with the same postures, this lead each grid index often is capture very specific part of the body and grid 3, 4, 5, 7 are the grids capturing the body parts that are more prone to a fracture.

Thus, in this method, we aim to train one deep neural network on the data from one particular grid. We have picked the grid with index 3,4,5 and 7 to train four separate models. The motivation for this approach is to train a model that will be specialized in recognizing fracture for a certain part of the body. Each image after slicing will be reshaped to 244 x 224 pixels before feeding into a model.

Table 1(The amount of Fracture in each grid for Training)

Grid Index	Fracture	Normal
0	11	1331

1	3	1339
2	8	1334
3	93	1249
4	88	1254
5	61	1281
6	8	1334
7	42	1330
8	4	1338
Total :	318	12078

3.1.3 Method 3 - Using a Single Classifier to Classify the Whole Image

Lastly, we approach the problem with a most intuitive approach by training a neural network that tries to classify a complete X-ray image into a fracture or normal images. This is the approach that is done by most studies. Each image is resized to 640 x 640 pixel before feeding into a model. This approach has 1245 normal and 97 fracture images for training.

3.2 Test Dataset Distribution

Since all the three methods use different training datasets, the test dataset is different as well. Method 1 has 89 fracture and 748 normal images for testing, while Method 3 has 72 fractured images and 21 normal images for testing. We decided to have an overlap between training and testing dataset as there are very little fractured images, and fracture can come in a lot of variation. The goal is let the model expose to a wide range of fracture in both testing and training as detecting a fracture is more important than detecting a normal images.

Method 2 requires us to train four separate classifiers on four sets of images, refer to Table 2 for the distribution of test dataset that are used in Method 2.

Table 2 Test Data set distribution for Method 2

Grid Index	Fracture	Normal
3	31	62

4	9	84
5	30	63
7	6	87
Total :	76	296

3.3 Experiments

For all the method we proposed, we have used two model architectural. The first architectural is GoogLeNet with an additional one fully connected layers with 150 nodes, with experiment, we realize that the additional layers help the model to plot a better salient map. The second architectural we named it as CustomeNet is a model with 4 convolution layers, the model carryout max pooling, batch normalization and a ReLU activation after each convolution. Each convolution layers has a 3v3 kernel with stride 1 with no padding. After the convolution, the output will be flattened and pass to 2 layers of fully connected layer, with 1000 and 150 nodes, Relu used as activation function. Both architectural will have a log softmax layer at the end of the output.

ADAM is used as the optimizer with an initial learning rate of 0.001, the loss function used is negative log loss with various weight. Due to the data imbalance and very few images in fracture images we have various our weight for the loss function where the model will be penalized differently a fractured image is misclassified. Refer to the formulate below for the how weighted loss are computed. L_n is referring the original loss for the particular x_n and y_n

$$l(x, y) = \sum_{n=1}^N \frac{1}{\sum_{n=1}^N W_{ym}} l_n$$

We have also performed data augmentation through random rotation. For method 2 and 3 we have performed data argumentation to balance the dataset. The amount of augmentation performed using the following equation

$$Augmentation\ Factor, N = floor\left(\frac{Number\ of\ normal\ samples}{Number\ of\ fracture\ samples}\right)$$

For every fracture, we will perform N augmentation so that the model will receive a relatively equal amount of the fracture and normal image.

To improve the interpretability of the model, we plot the salient map for the test data sets every 100 steps for method 2 and 3. The model is saved for every epoch if it managed to achieve a better test accuracy.

3.4 Results

For all 3 methods of classification, we carry out experiment on both GoogLeNet and CustomNet to train with different weights. The first weight is used to penalize the Normal image, and the second weight is used to penalized Fracture image. We carry out data augmentation on the training data, but not the test data.

3.4.1 Method 1 Results

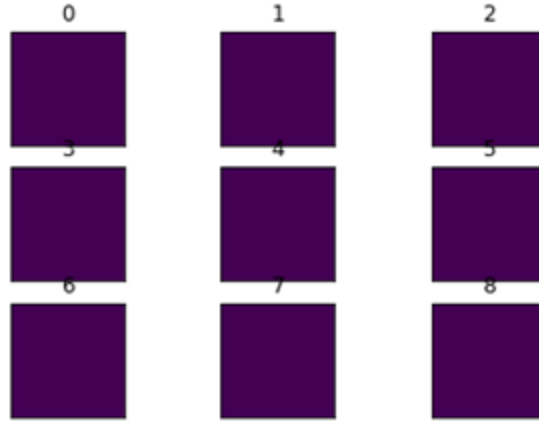
From table 2 we can see that model did not manage to come out with a good prediction that can classify Normal and Fracture images. This can be because the model is either classifying all the images as a fracture or normal. We can observe that CustomNet tend to classify all the images as fracture with less penalty on the fractured image compare to the GoogLeNet

Table 3(Performance of Method 1)

	GoogLeNet		CustomNet	
Weights	True Positive Rate	Accuracy	True Positive Rate	Accuracy
[0.5,0.5]	0	0.974	0	0.974
[0.5, 10]	0	0.974	1.0	0.025
[0.5, 10.5]	1.0	0.025	1.0	0.025

We compare the Salient mapping for both GoogLeNet and CustomNet at different weights, but most of the salient map appears to be all 0, except for GoogLeNet with the weights [0.5,10.5].

Figure 3 (Typical Salient Map for both GoogLeNet and CustomNet)



Refer to the Figure 4 and 5, we can see that the Salient mapping is able to capture the shape of the X-ray and exclude the implants form the X-ray where a fracture will not have occurred. This shows that the model is able learn the possible features that are not causing a fracture, but not necessarily a feature that can cause a fracture.

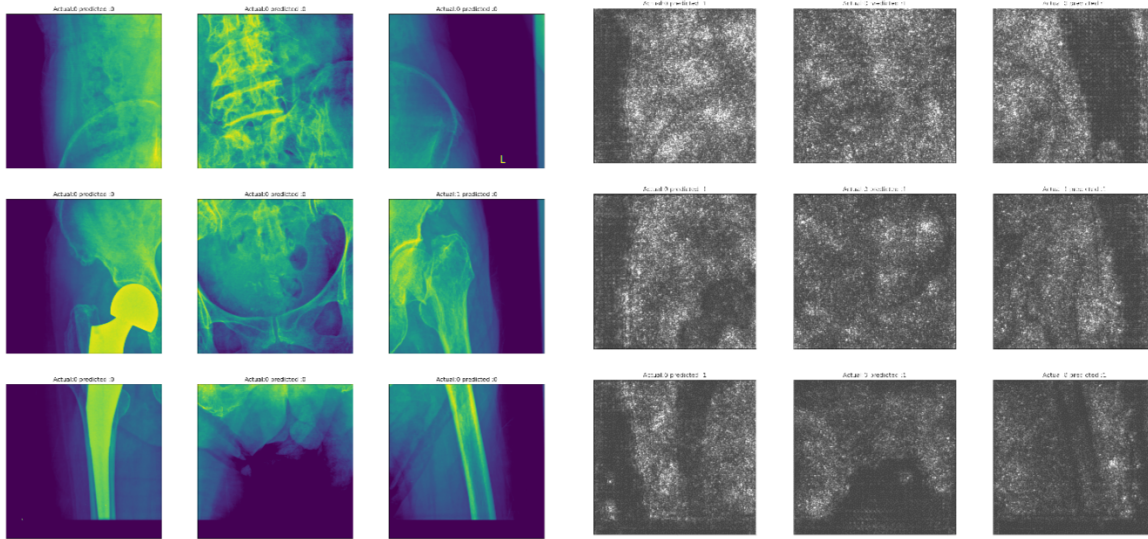


Figure 4(Original X-ray)

Figure 5(Salient Mapping from GoogLeNet when True positive rate =1)

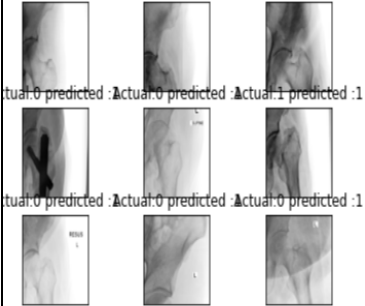
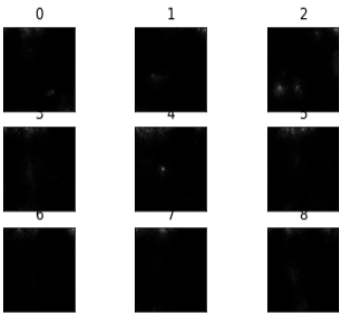
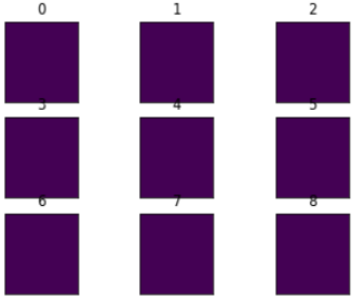
3.4.2 Method 2 Results

Method 2 for classification has produced very similar results. Refer to table 3, we have trained one model for the grid index with given specific weight for the loss function. This can be because the imbalance data and variation of fracture. The salient mapping, however, tends to behave better for GoogLeNet.

Table 4 Model Performance for GoogLeNet and CustomNet for Method 2

	Grid	GoogLeNet		CustomNet	
Weight		True Positive Rate	Accuracy	True Positive Rate	Accuracy
[0,5, 0.5]	3	0	0.667	1	0.333
	4	0	0.903	1	0.0967
	5	0	0.677	1	0.322
	7	0	0.931	1	0.0690
[0,5, 1.5]	3	1	0.333	1	0.333
	4	1	0.0967	1	0.0967
	5	1	0.322	1	0.322
	7	1	0.0690	1	0.0690
[0,5, 2]	3	1	0.333	1	0.333
	4	1	0.0967	1	0.0967
	5	1	0.322	1	0.322
	7	1	0.0690	1	0.0690

However, we also observed that GoogLeNet tend to produce a better salient map compare to the CustomNet (Due to the school GPU cluster is down on 2nd of May we were not able to generate the better image with better definition) We can see that although both models perform poorly in this method, but google map is able to learn better compare to the CustomNet as CustomNet produce almost a empty salient map

		
Original X-ray	Salient Map for GoogLeNet at 1000 steps	Salient Map for CustomNet at 1000 steps

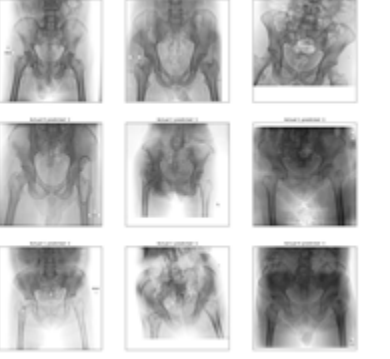
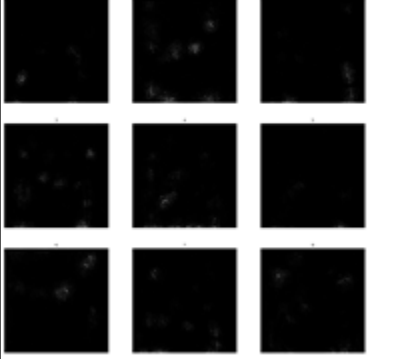

3.4.3 Method 3 Results

In this method, we have observed similar performance. From Method 1 and 2 where the model is either classifying all the images as fracture. This can mean that all 3 methods we are using are not able to learn anything useful to classify the fracture from normal X-ray. We plot the salient at every 100 steps of the training, from the salient map we can see that compared to method 2 the model very quickly loss direction on what to learn.

Table 5 The performance for method 3

Weight	GoogLeNet		CustomNet	
	True Positive Rate	Accuracy	True Positive Rate	Accuracy
[1,1]		0.774		0.774
[1,1.5]		0.774		0.774

Table 6 The comparison for Salient map for GoogleNet and Custom Net for Weight =[0.5,1.5]

		
The original X-ray.	Salient Map at 100 Steps for GoogLeNet	Salient Map at 1000 Steps for GoogLeNet

4 Object Detection

We chose YoloV5 as it is known to perform well with small sized datasets, small target region and trains very fast as well. YoloV5 architecture has 3 different backbones of varying sizes (small, medium and large). We experimented with the small and medium models. It was redundant to try out the large model as the dataset is small (which will lead to overfitting) and that the computation requirements were too high. We tested out varying batch sizes based on computation availability.

We decided to divide each image into smaller tiles. This would remove the need to resize the image and loss feature information. Training in this resulted in much better performance in terms of accuracy, precision, recall and IOU scores.

4.1 Motivation

Since our dataset is very small, using image classification alone will not be as effective due to the lack of semantic details. The images are from a very broad distribution and hence it's hard to understand what the exact feature. This is evident from the classification experiment we conducted, which shows that the salient mapping is not really very effective in finding out the defective regions. Hence object detection will help us due to localizing the defective region, which would tell the model where the fracture is leading to better classification performance. So ultimately our aim was to use the object detection model to classify, rather than detect boxes. Hence, we measured results in terms of precision recall as compared to IoU scores.

4.2 Dataset Pre-processing

We had 158 training images and 43 testing images with fractures in them with 50 training and 30 test images of non fractures images for all our experiments. Our dataset is very small and most of our dataset is non-fractured images. Hence it is imperative that we pre-process our data to extract maximum performance to classify fractures. NOTE: The number of images is different from that of classification due to tiling and rebalancing of imbalanced classes.

The different strategies that we tried out:

- 1) Remove dataset imbalance by over sampling the fractured data images
- 2) Data augmentation such as random translations and random rotations by 10 percent alongside contrast and saturation tweaking
- 3) Dividing the large 3000x3000 (approx.) images into smaller images of different tile sizes (we tried 750, 850 and 650 and otherwise the fracture was bigger than the tile
- 4) We felt that there was a lot of empty space (non-fracture region in fracture box) in each of the annotations and as such, hence we additionally added random downscaling by a small factor (10 percent) to the ground truth bounding boxes.

Figure 6 - Examples of Dataset

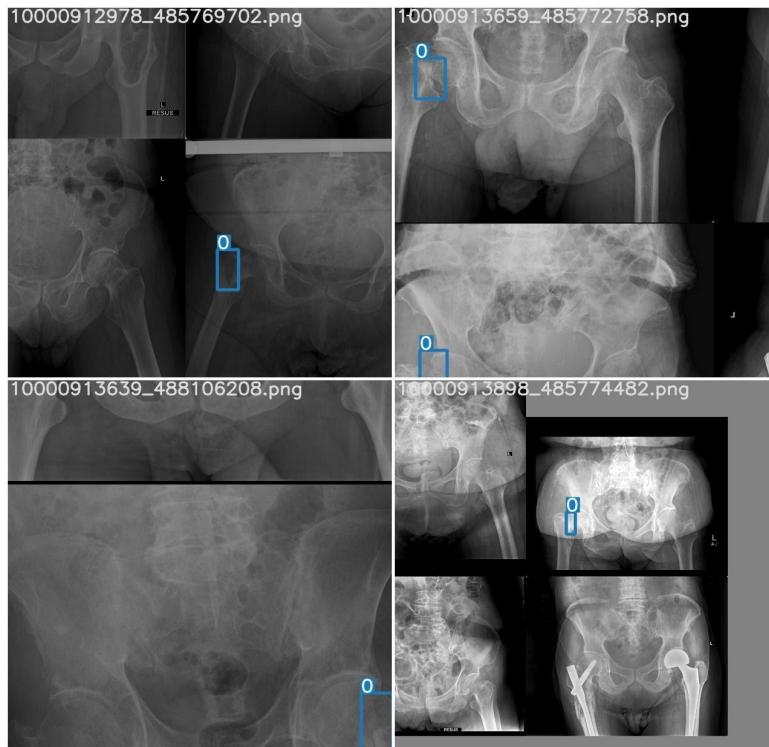
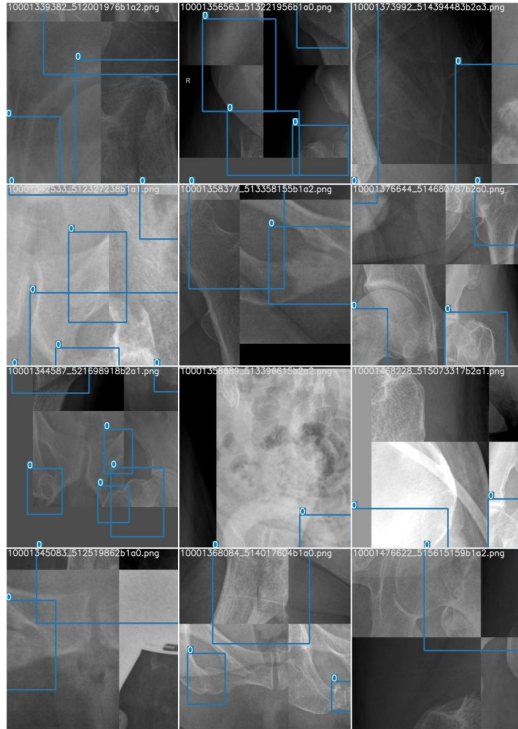


Figure 7 – Examples of generated bounding boxes



4.3 Model Selection

There are several models out there for object detection but we need to prioritize multiple factors such as:

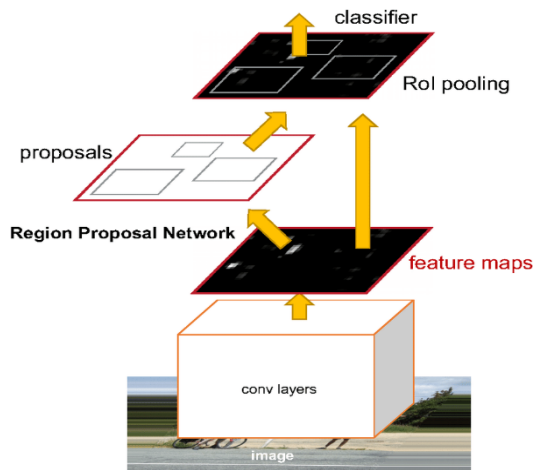
- 1) Works well with less data (not very dense)
- 2) Good at detecting smaller objects
- 3) Great classification head
- 4) Fast training (for rapid prototyping)

Hence, we shortlisted 4 models (if a model has different variants, we are considering the smallest and shallowest variant, e.g.: YoloV5s, instead of YoloV5m)

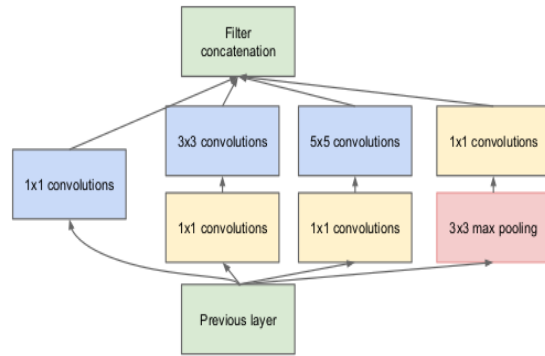
- 1) Faster RCNN: Very good accuracy scores, taught in course
- 2) InceptionNet: Backed by Google, very good at detecting multi scale fractures
- 3) YoloV5: Fast Training speed, good at small object detection, Better at classification
- 4) CenterNet: Fast inference time

Rcnn, Inception block, YoloV5, CenterNet

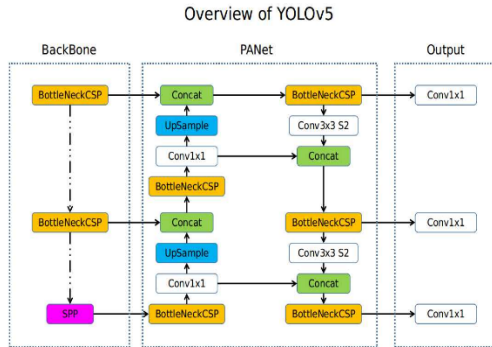
Figure 8 – Model Architectures: a)Faster RCNN b)Inception Block c)YoloV5 d)CenterNet



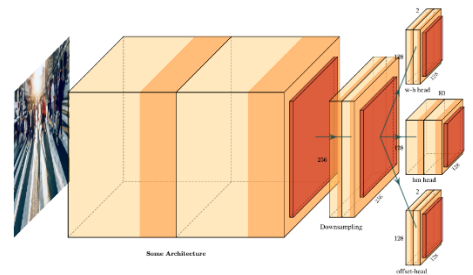
a



b



c



d

Figure 9 – Comparison between YoloV5 and Faster RCNN

	YOLO v5	Faster RCNN
Inference Speed	✓	
Detection of small or far away objects	✓	
Little to no overlapping boxes	✓	
Missed Objects	✗	✗
Detection of Crowded objects	✓	✓

We selected YoloV5 because although FasterRCNN is slightly better with mAP scores (both produce good mAP), yoloV5 trains faster, less computationally intensive, better for small objects and overall, better for classification. Inception is too dense in comparison and does not do well with smaller objects as

per our research. CenterNet has too many false positives as it does not use Non-Maximum Suppression (NMS) while LeNet was difficult for us to implement (though the results are around the same theoretically)

4.4 Experiments

We trained every experiment until 10 epochs after convergence and only took the best epoch. We use SGD optimizer with an initial learning rate of 0.0032 and a final learning rate of 0.12 with a momentum value of 0.843.

- 1) The first experiment was conducted with just the uncropped images
- 2) The second experiment consisted of using tiled data of size 750 to run the experiments alongside translations and rotations
- 3) The third experiment consisted of using tiled data of size 750 to run the experiments with variable ground truth bounding box size (downscale by a small factor)
- 4) The fourth experiment consisted of using tiled data of size 850 with variable ground truth bounding box size.

All experiments had basic augmentations such as translations and rotations, hue saturation and contrast change added at random.

We tried every experiment 3 times and took the average of the results. We did try training with Faster RCNN as well but we obtained no valuable results. This was either due to hyperparameter mismatch or just robustness of model. We also tried more experiments that yielded no such result.

4.5 Results

Table 3 – Results at Confidence Score 0.1

Experiment	Precision	Recall	mAP
Uncropped images	0	0	0
Tiled 750	0.261	0.148	0.187
Tiled 750 variable gt	0.344	0.186	0.342
Tiled 850 variable gt	0.75	0.14	0.778
Tiled 950 variable gt	0.75	0.03	0.77

During our class project presentation, the Tiled 750 model was our best model, even with such a low confidence score. We took Prof. Ngai-Man's suggestion that our bounding box ground truth are too generous and hence it does not fully localize the object. We fixed this by adding variable ground truth where we resize the ground truth bounding box of some inputs by a factor. This has impacted our model significantly.

Clearly the Tiled 850 with variable ground truth has the best results. It has a precision of 0.75 while the recall suffers with 0.14 but the mAP score is 0.778 which is very impressive. This shows that our model is very sensitive which is exactly what the requirements state.

Figure 10 – Training graph and evaluation graphs for the Tile 850 variable GT model

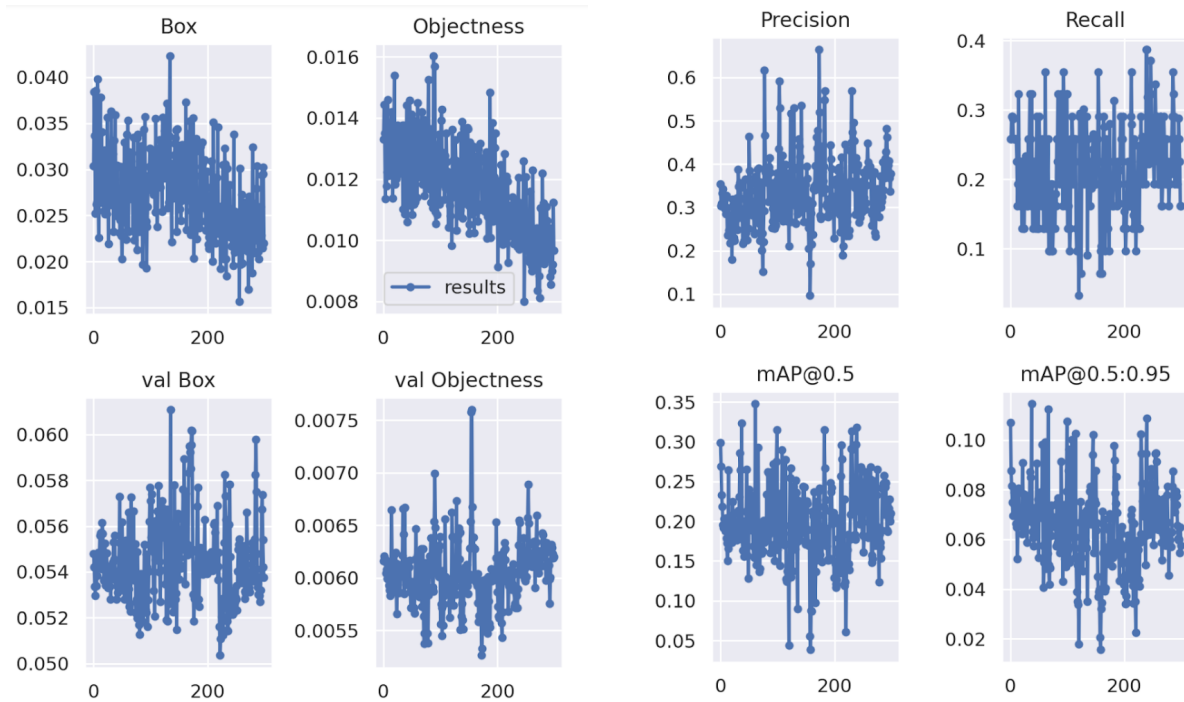


Figure 11 – Prediction Example

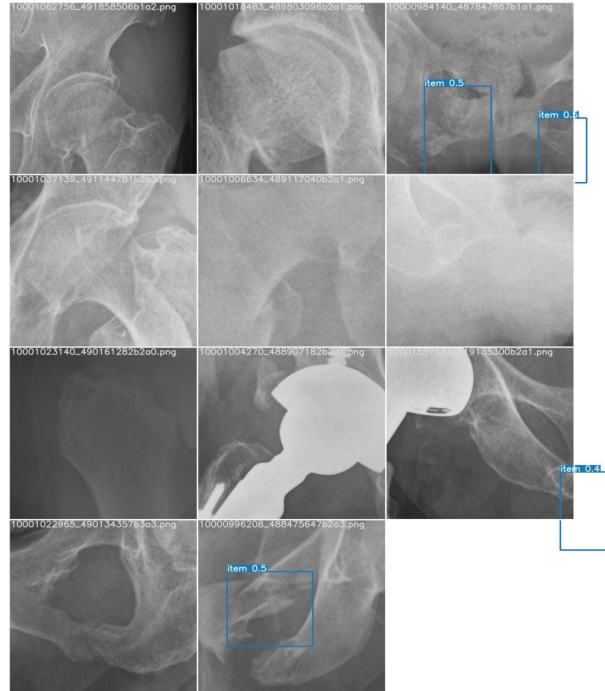
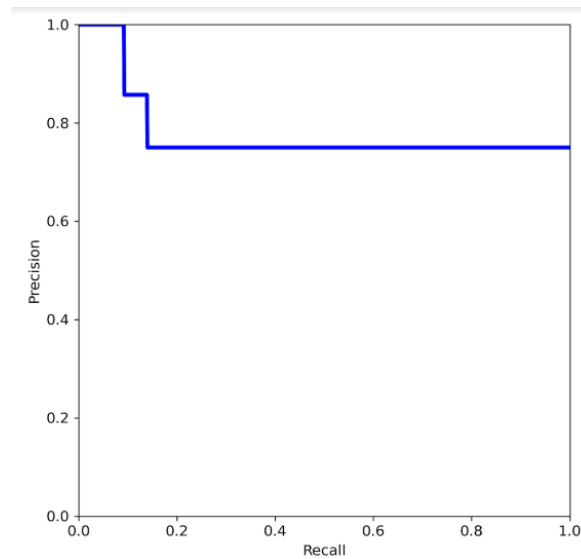


Figure 12 – Precision Vs Recall



To explain the poor recall, this is because our confidence score is very low, leading to the output of several bounding boxes but we want our precision to be high as in a medical use case, a sensitive model is more important than a highly specific model. We can train longer and tweak the threshold to increase recall if needed as well. The reason for drop in recall when tile size increased to 950 was because there is just so much more spatial information, leading to more bounding boxes being outputted at the low threshold.

5 Conclusion

5.1 Classification

From the results of the classification, we have observed that it is very difficult to train a model that can effectively classify the fracture and normal x-ray. This can be because the very limited dataset we have, the fracture can come in many forms and one can be very different from another, such as the type of fracture and part of the hip that experiences a fracture. Among the datasets we have, we have a total 107 fracture X-ray, compare the studies that are done by other research teams mentioned in the related works, most of the team have at least 2000 over images of fracture images. Thus, having enough representation of a fracture is very important for the training of the model.

From all the experiments we have conducted, we can see that method 2 with GoogLeNet seems to be the most potential solution, as it is able to plot more reasonable salient maps compared to method 1 and 3 of the classification. Thus method 2 with GoogLeNet is capable of picking features during training and more likely to perform better when there are more datasets.

5.2 Object Detection

In conclusion, our best model is when we use 198 training images and 73 test images (distribution described earlier) with a tile size of 850x850 and variable ground truth size by downscaling at 10% randomly (though the randomness is the same for all experiments so that the same images are downsized as to ensure fairness). We achieved a precision of 0.75 and a recall of 0.14 with a mAP score of 0.77. This is a very good result with the limited data we have available as the clinician requested for a highly sensitive model. A precision of 0.77 is considered sensitive. The recall is low because of the low confidence of the model. It is also proof that the model is indeed learning the correct features and with more training data, we are confident we can bring up the precision while also bringing up the recall.

6 Code Availability

Classification: https://github.com/storyinvisible/AI_Healthcare_Hip_Fracture

Object Detection: <https://github.com/Hsengiv2000/AIHC-Project>

7 Citations

1. Pinto, A. et al. Traumatic fractures in adults: missed diagnosis on plain radiographs in the Emergency Department. *Acta Bio-Med.* **89**, 111 (2018).
2. Cheng, CT., Wang, Y., Chen, HW. et al. A scalable physician-level deep learning algorithm detects universal trauma on pelvic radiographs. *Nat Commun* 12, 1066 (2021).
<https://doi.org/10.1038/s41467-021-21311-3>
3. Badgeley, M.A., Zech, J.R., Oakden-Rayner, L. et al. Deep learning predicts hip fracture using confounding patient and healthcare variables. *npj Digit. Med.* **2**, 31 (2019).
<https://doi.org/10.1038/s41746-019-0105-1>
4. Lee, C., Jang, J., Lee, S. et al. Classification of femur fracture in pelvic X-ray images using meta-learned deep neural network. *Sci Rep* **10**, 13694 (2020).
<https://doi.org/10.1038/s41598-020-70660-4>
5. Cheng, CT., Wang, Y., Chen, HW. et al. A scalable physician-level deep learning algorithm detects universal trauma on pelvic radiographs. *Nat Commun* 12, 1066 (2021).
<https://doi.org/10.1038/s41467-021-21311-3>
6. Dr Graham Lloyd-Jones BA MBBS MRCP FRCR - Consultant Radiologist -. "Introduction to Trauma X-ray Fracture Displacement." *Radiology Masterclass*. Web. 02 May 2021.
7. Ren, Shaoqing et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017): 1137–1149. Crossref. Web.
8. Duan, Kaiwen et al. "CenterNet: Keypoint Triplets for Object Detection." 2019 IEEE/CVF International Conference on Computer Vision (ICCV) (2019): n. pag. Crossref. Web.
9. Redmon, Joseph et al. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): n. pag. Crossref. Web.
10. Szegedy, Christian et al. "Going Deeper with Convolutions." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): n. pag. Crossref. Web.

