Yolo v3 Based Rib Fracture Detection Method

Machine Learning Project Report

Huiping Lin, Jiaqi Xi, Haoxiang Yang, Naiqian Zheng

Peking University, Turing Class

Abstract

Medical imaging is a technique widely used for medical diagnostics. This in some cases leads to a real bottleneck when there is a lack of medical practitioners and the images have to be manually processed. In such a situation there is need to reduce the amount of manual work by partly automating the analysis. The purpose of this study is to design an optimal method and evaluate the performance of a deep learning algorithm on the detection of rib fractures in chest radiographs.

In this article, we propose a multistage machine learning method solving fracture detection task. This method is based on Yolo-v3 [10], an improved version of a very popular object detection method Yolo. Several algorithms have been attempted to solve this task, and after comparing their experiment results and discussing their superiority and inferiority, we finally adopt Yolo-v3 as our final version. Our code and experiment results are open sourced on Github [1]. Our weights can be downloaded from Peking University Cloud [3] or OneDrive Cloud [2]. We recommend you download our weights from these two websites.

1 Introduction

Chest radiograph interpretation is critical for the detection of thoracic diseases, including tuberculosis, lung cancer and rib fractures. This time-consuming task typically requires expert radiologists to read the images, leading to fatigue-based diagnostic error and lack of diagnostic

expertise in areas of the world where radiologists are not available. Recently, deep learning approaches have been able to achieve expert-level performance in medical image interpretation tasks, powered by large network architectures and fueled by the emergence of large labeled datasets.

Currently, deep learning techniques are considered to be state of the art for classification of images. There is interest in applying deep learning in radiology because of the recent success, and with promising results. In medical area, machine learning methods is now widely applied to solve classification tasks or give diagnosis on radiography images. For example, Paras Lakhani, MD etc. evaluate the efficacy of deep convolutional neural networks (DCNNs) for detecting tuberculosis (TB) on chest radiographs [6]. It turned out that the best-performing classifier had an AUC of 0.99, which was an ensemble of the AlexNet and GoogLeNet DCNNs. P Rajpurkar etc. developed CheXNeXt, a convolutional neural network to concurrently detect the presence of 14 different pathologies, including pneumonia, pleural effusion, pulmonary masses, and nodules in frontal-view chest radiographs [8]. These all show that convolutional neural networks achieves cheerful performances on chest diseases detection.

However, the existing works mainly focus on classification tasks. Given a radiography image, their networks only declare whether the image indicates a problem, or what kind of problem it indicates. More meticulous tasks like fracture detection is still in lack of methods. Existing fracture detection methods mainly focus on wrist fracture detection or ankle fracture detection [5, 7], and CNN is also considered to be a useful tool in these tasks. On condition that chest radiography images is much more complicate than wrist or ankle radiography images, because people mainly have 12 pair ribs, with a straight and sturdy vertebral in the middle, rib fracture detection task is much harder than other fracture detection tasks. Therefore, research and work focusing on this task is highly lacked.

We attempted diverse machine learning model to solve this task, and finally propose a Yolo v3 based multistage method, which achieves high rationality and superiority. Our method takes Yolo v3 as basic model, and use several data augmentation to make the training more efficiently.

2 Machine Learning Model

Our model is improved based on Yolo v3. In this section, we will first introduce the construction of Yolo v3 in detail, then give some useful data augmentation methods, and finally present our model.

2.1 Yolo v3

Yolo v3 is based on Yolo, which has the similar network structure [9]. Yolo's network architecture is inspired by the GoogLeNet model for image classification. Its network has 24 convolutional layers followed by 2 fully connected layers, and the convolutional layers are pretrained on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection. Instead of the inception modules used by GoogLeNet, Yolo simply uses 1×1 reduction layers followed by 3×3 convolutional layers. The final output of our network is the $7 \times 7 \times 30$ tensor of predictions. The full network is shown in Figure1.

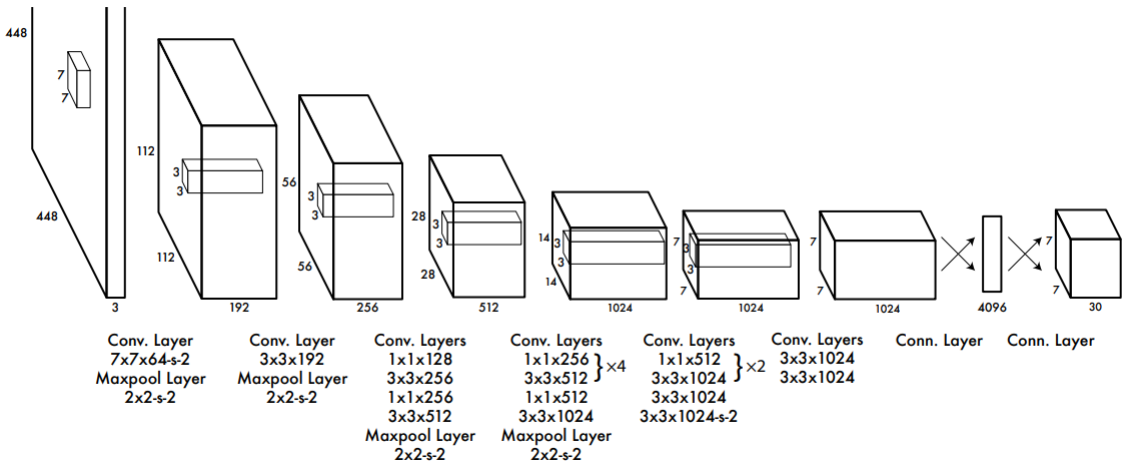


Figure 1: Yolo's Network Design

Compared with the original Yolo model, Yolo v3 uses a new network for performing feature extraction. Yolo v3's new network is a hybrid approach between the network used in Yolo v2, Darknet-19, and that newfangled residual network stuff. Yolo v3 uses successive 3×3 and 1×1 convolutional layers but now has some shortcut connections as well and is significantly larger. It has 53 convolutional layers and is called Darknet-53, which can be seen in figure 2.

It turns out that Darknet-53 performs on par with state-of-the-art classifiers but with fewer floating point operations and more speed. Darknet-53 is better than ResNet-101 and 1.5× faster. Darknet-53 has similar performance to ResNet-152 and is 2× faster. Darknet-53 also achieves the highest measured floating point operations per second. This means the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster. That's mostly because ResNets have just way too many layers and aren't very efficient.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2: Darknet-53 Network Design

2.2 Data Augmentation

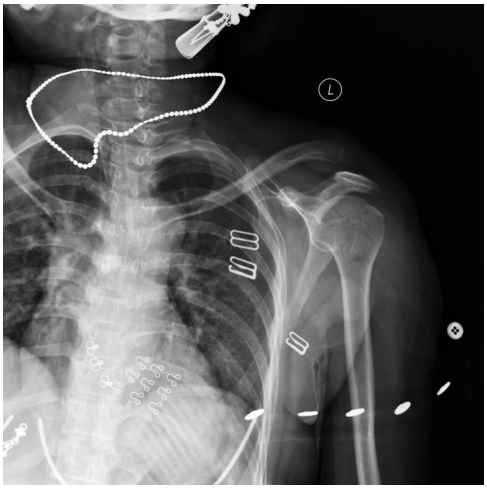
We adopt several data augmentation methods to pretreat the datasets, in order to make the images easier to be analysed and computed in the network.

Before introducing the data augmentation methods, we should first consider the difficulties we are facing when dealing with radiography images.

The chest radiography is a projection radiograph of the chest used to diagnose conditions affecting the chest, its contents, and nearby structures. It employs ionizing radiation in the form of X-rays to generate images of the chest. Its images are more like gray-scale images, though stored as .jpg or .png forms. Additionally, each chest radiography image contains a wide range of

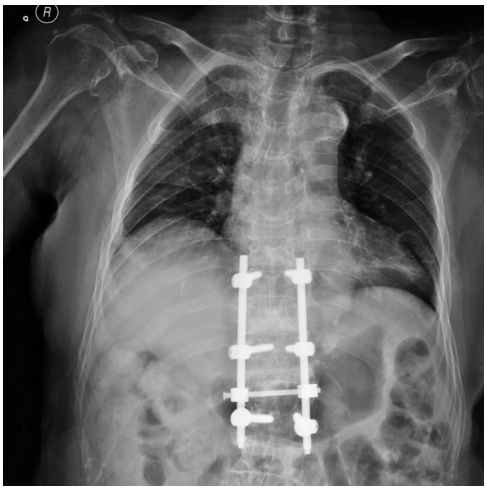
bones, but ribs are usually slender and the fractures are sometimes very slight that, which make the detection process much harder. Besides, in the chest radiography image, there may also exist necklaces, underwear shoulder straps, steel plates, or cardiac pacemakers as distracts. These may distract the "attention" of neural networks, and lead to misunderstanding. Therefore, we should also find out some methods that eliminate distraction, making the neural network model focusing on only the area of ribs. These examples can be seen in figure 3. Consequently, data augmentation methods is in need.

(a) Cardiac pacemakers.



(b) Necklaces and straps.

(c) Steel plates in arm.



(d) Steel plates in vertebral.

Figure 3: Images with distracts.

We now introduce these data augmentation methods, and then give our training methods.

- Brightness

  In some images, the bones are too dark and some areas are almost invisible with the black background. So when dealing with the dark images, enhancing the brightness is a good way to make the details in the image more distinguishable.

- Contrast

  In most images, the distracts are usually more bright than bones. So if we enhance the contract and ignore the most bright areas, or decrease the weight on the bright areas, we can reduce the influence of distraction areas. Additionally, these can make the edges of ribs more apparent.

  Brightness and Contrast adjustment are achieved in the same function, which operates linear transformation on image matrix.

- Sharpness

  Sharpness enhancement is always applied when the image is too ambiguous. This can also make the edges of ribs more apparent. Sharpness adjustment is achieved by a function which operates convolution on image matrix.

- Edge Extracting

  We initially expected to use edge extracting method and make the edge of bones more evident. We attempted different mathematical approaches and operators, but most lead to the same result that the fracture areas were totally ignored. This is because that fractures are usually slight, and their edges are more ambiguous than normal bones.

  Figure 4 shows the effect of different data augmentation methods. In figure 4 we can also see the disadvantages of edge extracting.

Therefore, in our final practice, we only adopt brightness, contrast and sharpness adjustment as image pretreatment.

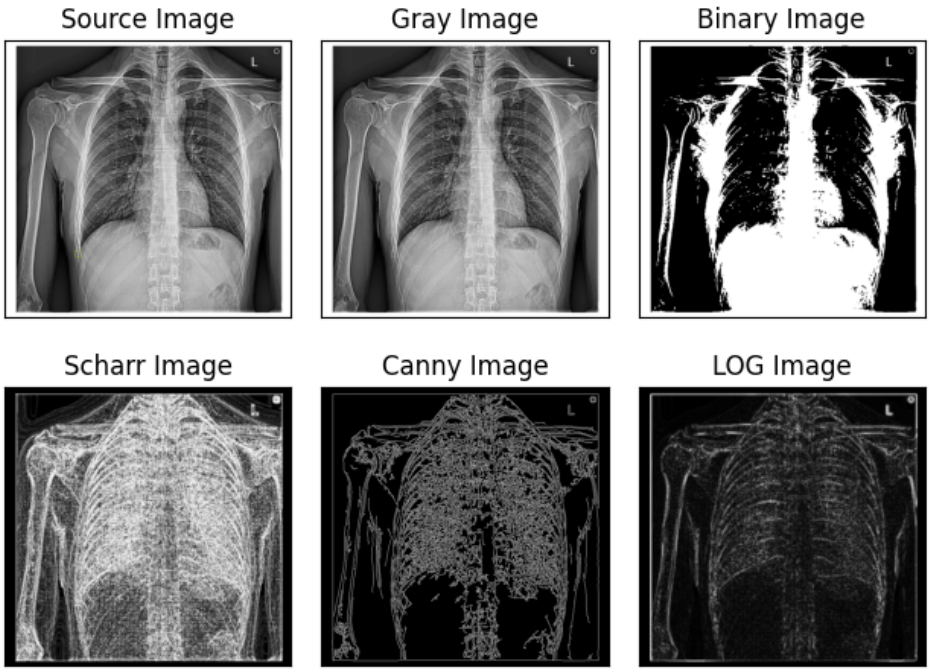


Figure 4: Different edge extracting methods.

2.3 Our Model

After introducing the basic neural network structure and data augmentation methods, we now give our training method: a 5-step multistage training model:

a) First, we use Darknet-53 as pretraining mode. In this step, we use the original Yolo v3 network (Darknet-53) to train the training dataset, and keep the training results.

b) Second, on the basis of training process of a), we use Adam [4] as the method for stochastic optimization, with learning rate=0.001, epochs=30, batch size=2 and accumulate gradient=16, and keep the training results. (Adam is a widely-used algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.)

c) Third, on the basis of training process of b), we use data augmentation methods mentioned above, and use Adam as the method for stochastic optimization, with learning rate=0.001, epochs=40, batch size=2 and accumulate gradient=16, and keep the training results.

d) Forth, on the basis of training process of c), we use data augmentation methods mentioned above, crop every image into small pieces as new images, and use Adam as the method for stochastic optimization, with learning rate=0.001, epochs=40, batch size=2 and accumulate gradient=16, and keep the training results.

e) Fifth, on the basis of training process of d), we use data augmentation methods mentioned above, **do not** crop images, and use Adam as the method for stochastic optimization, with learning rate=0.0001, epochs=40, batch size=2 and accumulate gradient=16, and keep the training results. Here we decrease the learning rate to 1/10, in order to make the training process more finetuning.

2.4 Rationality and Superiority

As we've introduced our training model, we now explain why we use a multistage model, and illustrate the rationality and superiority of our model.

This task is to give the positions of fractures using bounding boxes in every image. Considering that our training dataset is limited, and the number of fractures in each image is also small, so we should take higher usage of limited information. As we've mentioned before, data augmentation benefits our training process, because it makes the image more clear and the details more visible. Additionally, Cropping is also a good way to use our limited images. When cropping a big-size image into small pieces, the network can focus on more detailed information on each piece, and the fracture area may become much more clear and evident in the small pieces.

Therefore, at the beginning of training, we prefer to use the original dataset and network. As the training process moves on, we gradually adjust the training data by cropping or data augmenting. Considering that cropping has high randomness, and in order to avoid over fitting, we should make the training go back to its original track and aim at our original intention at the last stage. Therefore we cancel the cropping method, and only keep data augmentation method. In addition, we decrease the learning rate, making our training more finetuning.

This multistage training model considers much more factors than a single stage training model. In this model, more image features are taken into consideration, making the whole training process much more robust.

3 Experiment

In this section, we first introduce the basic setup of our experiment and the implementation we use. Next, we use detailed figures and tables to show our experimental results.

3.1 Experiment Implementation

- Server: Ubuntu Server 16.04 LTS, Kernel=4.4.0-142-generic
- GPU: Nvidia RTX 2070 Super 8GB, CUDA10.1
- CPU: AMD 3700X (8 cores, 16 threads)
- Memory: 32G

It is worth mentioning that our experiment implementation is not provided by a lab in our university. It is one of our team member's personal computer. In order to make it work, we downloaded the .iso file of Ubuntu Server 16.04 from ubuntu websites and installed it on a clean disk by ourselves, and installed the whole training environment (including GPU driver, CUDA and remote cooperation softwares) by ourselves.

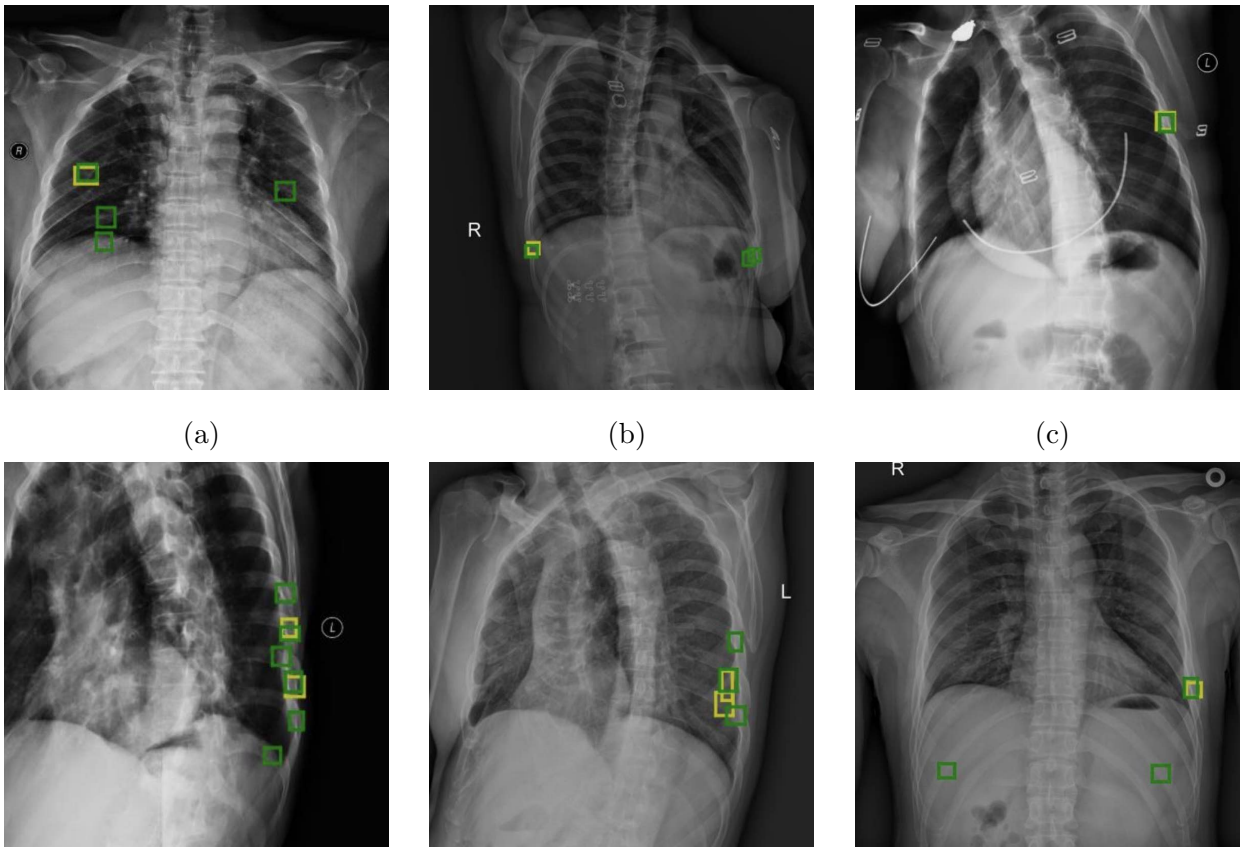
3.2 Experiment Results

We set these arguments in our file test.py: batch_size=2, conf_thres=0.5, img_size=1024, nms_thres=0.1. Here, batch_size refers to the number of training examples utilized in one iteration. conf_thres refers to confidence threshold. If and only if our confidence towards the fracutre area is greater than conf_thres, we declare it to be an area. img_size refers to the resized image size. Here we set it to 1024*1024 due to our implementation limitation. nms_thres refers to non-maximum suppression threshold, which means that if two bounding boxes overlap more than nms_thres, then we combine them into a single bounding box.

Under this circumstance, our AP50 achieves 11.8%, and total AP(50-95) achieves 3%. However, our AP75 has only nearly 0. This is because that our our resize adjust the image to 1024*1024, which makes the image more ambiguous, and the some image content is ignored. This is lim-

ited by our hardware implementation and computing ability. The development in experiment implementation can lead to better results.

The following figures5 shows some of our output images, using test data as input. Here the yellow bounding boxes are the ground truth, and the green bounding boxes are our detection results. The result shows that our model may mistake some normal positions to be fracture positions, but it can always detect true fracture positions.



(a) (b) (c)

(d) (e) (f)

Figure 5: Experiment results.

4 Conclusion

In this study, to solve the rib fracture detection task on chest radiography images, we propose a multistage machine learning method solving fracture detection task. This method is with high rationality and superiority compared with other single stage machine learning method.

References

[1] Experimental and implemention codes. https://github.com/NaturezzZ/FractureDetection-PyTorchYOLOv3.

[2] Our weights on onedrive cloud. https://pkueducn-my.sharepoint.com/:u:/g/personal/zhengnaiqian_pku_edu_cn/EQPrYihpdNJOncFh7itykQYBw8knjsXMiyDT6t_NO5_DXg?e=hMaa8d.

[3] Our weights on peking university cloud. https://disk.pku.edu.cn:443/link/4E0333B716CB2D7124ECFFA3624BEF3E.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Gene Kitamura, Chul Y Chung, and Barry E Moore. Ankle fracture detection utilizing a convolutional neural network ensemble implemented with a small sample, de novo training, and multiview incorporation. *Journal of digital imaging*, 32(4):672–677, 2019.

[6] Paras Lakhani and Baskaran Sundaram. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2):574–582, 2017.

[7] Jakub Olczak, Niklas Fahlberg, Atsuto Maki, Ali Sharif Razavian, Anthony Jilert, André Stark, Olof Sköldenberg, and Max Gordon. Artificial intelligence for analyzing orthopedic trauma radiographs: deep learning algorithms—are they on par with humans for diagnosing fractures? *Acta orthopaedica*, 88(6):581–586, 2017.

[8] Pranav Rajpurkar, Jeremy Irvin, Robyn L Ball, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis P Langlotz, et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists. *PLoS medicine*, 15(11):e1002686, 2018.

[9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[10] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.