Fracture Detection in Pediatric Wrist Trauma X-ray Images Using YOLOv8 Algorithm

Rui-Yang Ju[1] **and Weiming Cai**[2,*]

[1]Tamkang University, Department of Electrical and Computer Engineering, New Taipei City, 251301, Taiwan
[2]Jingjiang People's Hospital, Department of Hand and Foot Surgery, Jingjiang City, 214500, China
[*]Corresponding author: 1318746637@qq.com

ABSTRACT

Hospital emergency departments frequently receive lots of bone fracture cases, with pediatric wrist trauma fracture accounting for the majority of them. Before pediatric surgeons perform surgery, they need to ask patients how the fracture occurred and analyze the fracture situation by interpreting X-ray images. The interpretation of X-ray images often requires a combination of techniques from radiologists and surgeons, which requires time-consuming specialized training. With the rise of deep learning in the field of computer vision, network models applying for fracture detection has become an important research topic. In this paper, we train YOLOv8 (the latest version of You Only Look Once) model on the GRAZPEDWRI-DX dataset, and use data augmentation to improve the model performance. The experimental results show that our model have reached the state-of-the-art (SOTA) real-time model performance. Specifically, compared to YOLOv8s models, the mean average precision (mAP 50) of our models improve from 0.604 and 0.625 to 0.612 and 0.631 at the input image size of 640 and 1024, respectively. To enable surgeons to use our model for fracture detection on pediatric wrist trauma X-ray images, we have designed the application "Fracture Detection Using YOLOv8 App" to assist surgeons in diagnosing fractures, reducing the probability of error analysis, and providing more useful information for surgery. Our implementation code is released at https://github.com/RuiyangJu/Bone_Fracture_Detection_YOLOv8.

Introduction

In hospital emergency rooms, radiologists are often asked to examine patients with fractures in various parts of the body, such as the wrist and arm. Fractures can generally be classified as open or closed, with open fractures occurring when the bone pierces the skin, and closed fractures occurring when the skin remains intact despite the broken bone. Before performing surgery, the surgeon must inquire about the medical history of the patients and conduct a thorough examination to diagnose fracture. In recent medical imaging, three types of devices, including X-ray, Magnetic Resonance Imaging (MRI), and Computed Tomography (CT), are commonly used to diagnose fracture[1]. And X-ray is the most widely used device due to its cost-effectiveness.

Fractures of the distal and ulna account for the majority of wrist trauma in pediatric patients[2,3]. In prestigious hospitals of developed areas, there are many experienced radiologists who are capable of correctly analyzing X-ray images; while in some small hospitals of underdeveloped regions, there are only young and inexperienced surgeons who may be unable to correctly interpret X-ray images. Therefore, a shortage of radiologists would seriously jeopardize timely patient care[4,5]. Specifically, some hospitals in Africa have even limited access to specialist reports[6], which badly affects the probaility of the sucess of surgery. According to the survey[7,8], the percentage of X-ray images misinterpreted have reached 26%.

With the advancement of deep learning, neural network models have been introduced in medical image processing[9–12]. In recent years, researchers have started to apply object detection models to fracture detection[13–16], which is a popular research topic in computer vision (CV).

Deep learning methods in the field of object detection are divided into two-stage and one-stage algorithms. Two-stage algorithm models such as R-CNN[17] and its improved models[18–24] generate location and class probabilities in two stages. Whereas one-stage algorithm models directly produce the location and class probabilities of objects, resulting in the improvement of the model inference speed. In addition to the classical one-stage algorithm models, such as SSD[25], RetinaNet[26], CornerNet[27], CenterNet[28], and CentripetalNet[29], You Only Look Once (YOLO) series algorithm models[30–32] are preferred for real-time applications due to the good balance between the model accuracy and inference speed.

In this paper, we first design the model using YOLOv8 algorithm[33] and train it on the GRAZPEDWRI-DX[34] dataset to detect wrist fracture in children. After evaluating the performance of YOLOv8 model, we utilize data augmentation to improve it. The experimental results demonstrate that the model we trained using data augmentation has reached the state-of-the-art (SOTA) real-time model performance.

The contributions of this paper are summarized as follows:

- We introduce YOLOv8 algorithm for fracture detection and use data augmentation to improve the performance. The experimental results demonstrate that our model achieves the SOTA real-time model performance on the GRAZPEDWRI-DX dataset.

- This work develops an application to detect wrist fracture in children, which aims to help pediatric surgeons interpret X-ray images without the assistance of the radiologist, and reduce the probability of X-ray image analysis errors.

This paper is structured as follows: Section **Related Work** describes the deep learning methods for detecting fracture, and describes the application of YOLOv5 model in medical image processing. Section **Proposed Method** introduces the entire process of training and the architecture of our model. Section **Experiments** compares the performance of YOLOv5 and YOLOv8 models in detecting fracture, and presents the improved performance of our model trained using data augmentation. Section **Application** describes our proposed application to assist pediatric surgeons in analyzing X-ray images. Finally, Section **Conclusions and Future Work** discusses the conclusions and future work of this paper.

Related Work

In recent years, neural networks have been widely utilized in fracture image data for fracture detection. Guan *et al.*[35] achieved an average precision of 82.1% on 3,842 thigh fracture X-ray images using the Dilated Convolutional Feature Pyramid Network (DCFPN). Wang *et al.*[36] employed a novel R-CNN[17] network ParalleNet as the backbone network for fracture detection on 3,842 thigh fracture X-ray images. In addition to thigh fracture detection, about arm fracture detection, Guan *et al.*[37] used R-CNN for detection on Musculoskeletal-Radiograph (MURA) dataset[38] and obtained an average precision of 62.04%. Ma and Luo[39] used Faster R-CNN[19] for fracture detection on a part of 1,052 bone images of the dataset and the proposed CrackNet model for fracture classification on the whole dataset. Wu *et al.*[40] used ResNeXt101[41] and FPN[42] on the Feature Ambiguity Mitigate Operator (FAMO) dataset[40] for fracture detection on various body parts. Qi *et al.*[43] utilized Fast R-CNN[18] with ResNet50[44] as the backbone network to detect nine different types of fractures on 2,333 fracture X-ray images. Sha *et al.*[45,46] used YOLOv2[47] and Faster R-CNN[19] models for fracture detection on 5,134 CT images of spine fractures, respectively. Xue *et al.*[48] utilized the Faster R-CNN model for hand fracture detection on 3,067 hand trauma X-ray images, achieving an average precision of 70.0%. Although most of the works using R-CNN series models have shown excellent detection accuracy, the inference speed is not satisfactory.

YOLO series models[30–32] offer a balance of performance in terms of the model accuracy and inference speed, which is suitable for mobile devices in real-time X-ray images detection. YOLOv5 model[49], which was proposed by Ultralytics in 2021, has been deployed on mobile phones as the "iDetection" application. On this basis, Yuan *et al.*[50] employed external attention and 3D feature fusion techniques in YOLOv5 model to detect skull fractures in CT images. Warin *et al.*[51] used YOLOv5 model to detect maxillofacial fractures in 3,407 maxillofacial bone CT images, and classified the fracture conditions into frontal, midfacial, mandibular fractures and no fracture. Rib fractures are a precursor injury to physical abuse in children, and chest X-ray (CXR) images are preferred for effective diagnosis of rib fracture conditions because of their convenience and low radiation dose. Tsai *et al.*[52] used data augmentation with YOLOv5 model to detect rib fractures in CXR images. And Burkow *et al.*[53] applied YOLOv5 model to detect rib fractures in 704 pediatric CXR images, the model obtained the F2 score value of 0.58. To identify and detect mandibular fractures in panoramic radiographs, Warin *et al.*[54] used convolutional neural networks (CNNs) and YOLOv5 model to implement it. Fatima *et al.*[55] used YOLOv5 model to localize vertebrae, which is important for detecting spinal deformities and fractures, and obtained an average precision of 0.94 at an IoU (Intersection over Union) threshold of 0.5. Moreover, Mushtaq *et al.*[56] applied YOLOv5 model to localize the lumbar spine and obtained an average precision value of 0.975. Nevertheless, relatively few researches have been reported on pediatric wrist fracture detection using YOLOv5 model. While YOLOv8 was proposed by Ultralytics in 2023, we use this algorithm to train the model for the first time in pediatric wrist fracture detection.

Proposed Method

In this section, we introduce the process of the model training, validation and testing on the dataset, the architecture of our model, and the data augmentation technique employed during training. Figure 1 illustrates the flowchart depicting the model training process and performance evaluation. We randomly divide the 20,327 X-ray images of the GRAZPEDWRI-DX dataset into the training, validation, and test set, where the training set is expanded to 28,408 X-ray images by data augmentation from the original 14,204 X-ray images.

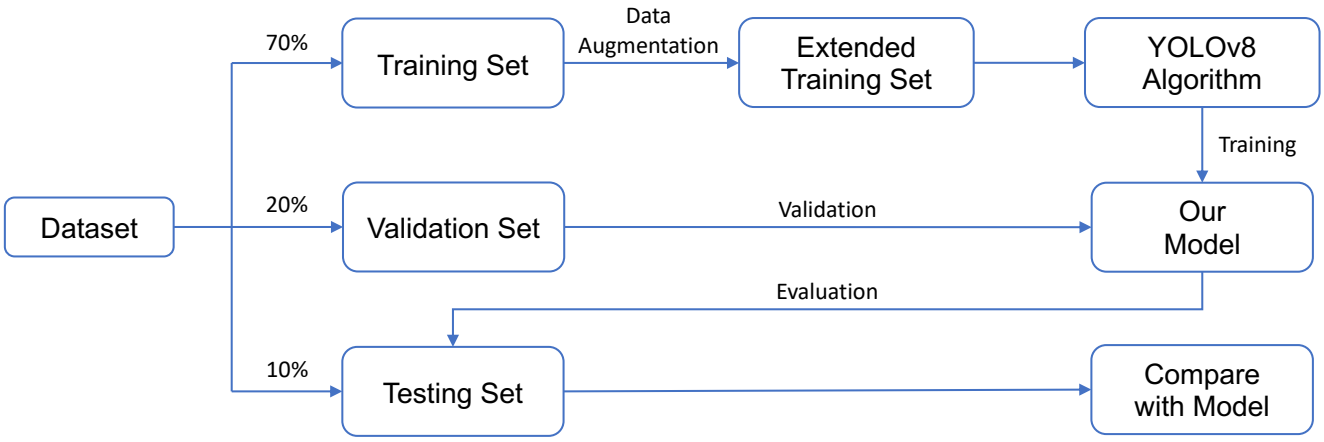

Figure 1. Flowchart of the model training, validation and testing on the dataset. The extended training set is used to double the number of X-ray images by data augmentation.

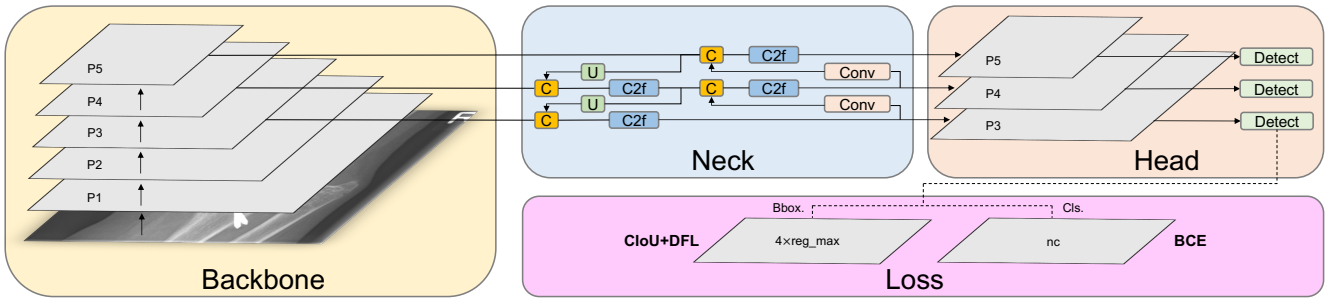

Figure 2. The architecture of YOLOv8 algorithm model. The overall architecture is divided into four parts, including Backbone, Neck, Head, and Loss.

Data Augmentation

During the model training process, data augmentation is employed in this work to extend the dataset. Specifically, we adjust the contrast and brightness of the original X-ray image to enhance the visibility of bone-anomaly. This is achieved using the addWeighted function available in OpenCV (Open Source Computer Vision Library). The equation is presented below:

$$Output = Input_1 \times \alpha + Input_2 \times \beta + \gamma, \tag{1}$$

where α represents the weight assigned to the first input image, which corresponds to the image contrast, and is set to 1.2; β denotes the weight assigned to the second input image, and since mixup is not utilized, β is set to 0; and γ represents the scalar value added to each sum, which corresponds to the image brightness, and is set to 30. The image after adjusting the contrast and brightness is shown in Figure 3.

Model Architecture

Our model architecture consists of Backbone, Neck, and Head, as shown in Figure 4. In the following subsections, we introduce the design concepts of each part of the model architecture, and the modules of different parts.

Backbone

The backbone of our model uses Cross Stage Partial (CSP)[57] architecture to split the feature map into two parts. The first part uses convolution operations, and the second part is concatenated with the output of the previous part. The CSP architecture improves the learning ability of the CNNs and reduces the computational cost of the model.

YOLOv8[33] introduces C2f module by combining the C3 module and the concept of ELAN from YOLOv7[30], which allows the model to obtain richer gradient flow information. The C3 module consists of 3 ConvModule and n DarknetBottleNeck, and the C2f module consists of 2 ConvModule and n DarknetBottleNeck connected through Split and Concat, as illustrated in Figure 4, where the ConvModule consists of *Conv-BN-SiLU*, and n is the number of the bottleneck. Unlike YOLOv5[49], we use the C2f module instead of the C3 module.

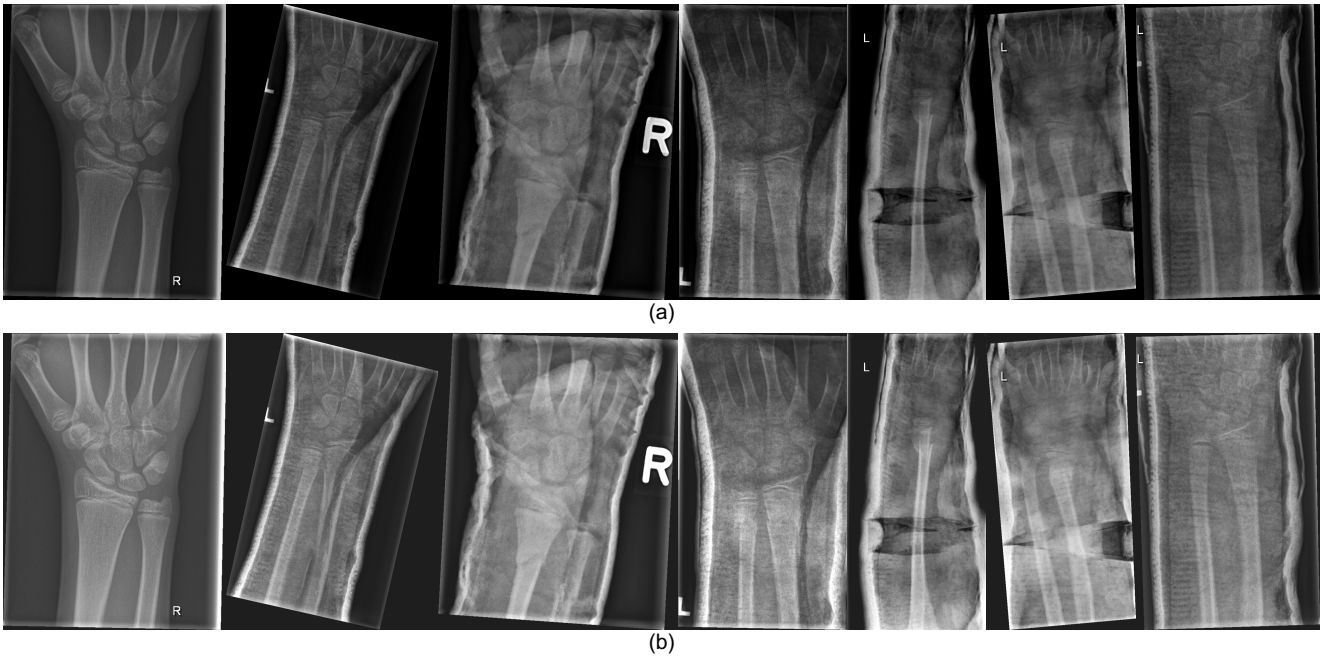

Figure 3. Examples of pediatric wrist trauma X-ray images using data augmentation. (a) the original images, (b) the adjusted images.

Furthermore, we reduce the number of blocks in each stage compared to YOLOv5 to further reduce the computational cost. Specifically, our model reduces the number of blocks to 3,6,6,3 in Stage 1 to Stage 4, respectively. Additionally, we adopt the Spatial Pyramid Pooling - Fast (SPPF) module in Stage 4, which is an improvement from Spatial Pyramid Pooling (SPP)[58] to improve the inference speed of the model. These modifications lead to our model with a better learning ability and shorter inference time.

Neck

Generally, deeper networks obtain more feature information, resulting in better dense prediction. However, excessively deep networks reduce the location information of the object, and too many convolution operations will lead to information loss for small objects. Therefore, it is necessary to use Feature Pyramid Network (FPN)[42] and Path Aggregation Network (PAN)[59] architectures for multi-scale feature fusion. As illustrated in Figure 4, the Neck part of our model architecture uses multi-scale feature fusion to combine features from different layers of the network. The upper layers acquire more information due to the additional network layers, whereas the lower layers preserve location information due to fewer convolution layers.

Inspired by YOLOv5, where FPN upsamples from top to bottom to increase the amount of feature information in the bottom feature map; and PAN downsamples from bottom to top to obtain more the top feature map information. These two feature outputs are merged to ensure precise predictions for images of various sizes. We adopt FP-PAN (Feature Pyramid-Path Aggregation Network) in our model, and delete convolution operations in upsampling to reduce the computational cost.

Head

Different from YOLOv5 model utilizing a coupled head, we use a decoupled head[31], where the classification and detection heads are separated. Figure 4 illustrates that our model deletes the objectness branch and only retains the classification and regression branches. Anchor-Base employes a large number of anchors in the image to determine the four offsets of the regression object from the anchors. It adjusts the precise object location using the corresponding anchors and offsets. In contrast, we adopt Anchor-Free[60], which identifies the center of the object and estimates the distance between the center and the bounding box.

Loss

For positive and negative sample assignment, the Task Aligned Assigner of Task-aligned One-stage Object Detection (TOOD)[61] is used in our model training to select positive samples based on the weighted scores of classification and regression, as shown in Equation 2 below:

$$t = s^{\alpha} \times u^{\beta}, \tag{2}$$

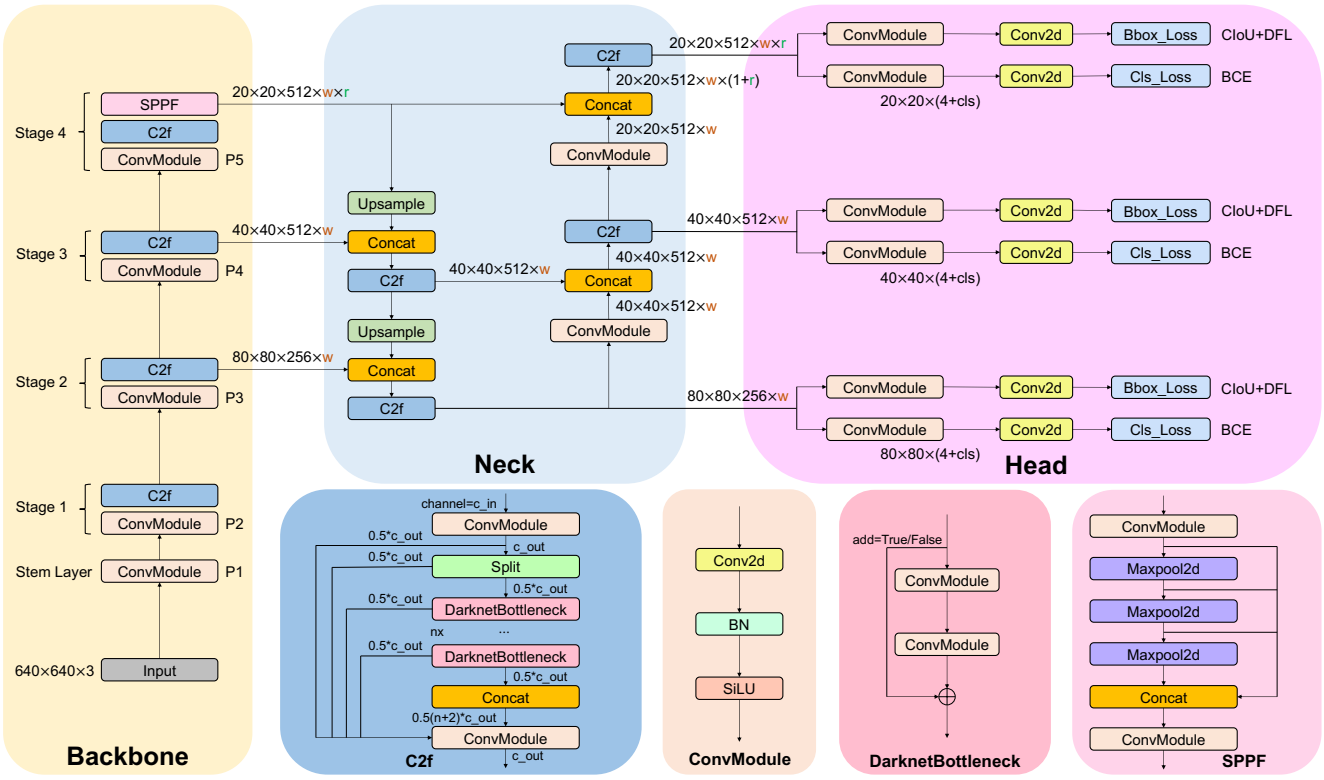

Figure 4. Detailed illustration of our model architecture. The Backbone, Neck, and Head are the three parts of our model, and C2f, ConvModule, DarknetBottleneck, and SPPF are modules.

where s is the predicted score corresponding to the labeled class, and u is the IoU of the prediction and the ground truth bounding box.

In addition, our model has classification and regression branches, where the classification branch uses Binary Cross-Entropy (BCE) Loss, and the equation is shown below:

$$Loss_n = -w\left[y_n \log x_n + (1-y_n)\log(1-x_n)\right], \tag{3}$$

where w is the weight, y_n is the labeled value, and x_n is the predicted value of the model.

The regression branch uses Distribute Focal Loss (DFL)[62] and Complete IoU (CIoU) Loss[63], where DFL is used to expand the probability of the value around the object y. Its equation is shown as follows:

$$DFL(\mathscr{S}_n, \mathscr{S}_{n+1}) = -((y_{n+1}-y)\log(\mathscr{S}_n) + (y-y_n)\log(\mathscr{S}_{n+1})), \tag{4}$$

where the equations of $\mathscr{S}_n$ and $\mathscr{S}_{n+1}$ are shown below:

$$\mathscr{S}_n = \frac{y_{n+1}-y}{y_{n+1}-y_n}, \ \mathscr{S}_{n+1} = \frac{y-y_n}{y_{n+1}-y_n}. \tag{5}$$

CIoU Loss adds an influence factor to Distance IoU (DIoU) Loss[64] by considering the aspect ratio of the prediction and the ground truth bounding box. The equation is shown below:

$$CIoU_{Loss} = 1 - IoU + \frac{Distance_2^2}{Distance_C^2} + \frac{v^2}{(1-IoU)+v}, \tag{6}$$

where v is the parameter that measures the consistency of the aspect ratio, defined as follows:

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w^p}{h^p})^2, \tag{7}$$

where w is the weight of the bounding box, and h is the height of the bounding box.

Experiments

Dataset

Medical University of Graz provides a public dataset named GRAZPEDWRI-DX[34], which consists of 20,327 X-ray images of wrist trauma in children. These images were collected from 6,091 patients between 2008 and 2018 by multiple pediatric radiologists at the Department of Pediatric Surgery of the University Hospital Graz. The images are annotated in 9 different classes by placing bounding boxes on them.

To perform the experiments shown in Table 4 and Table 5, we divide the GRAZPEDWRI-DX dataset randomly into three sets: training set, validation set, and test set. The sizes of these sets are approximately 70%, 20%, and 10% of the original dataset, respectively. Specifically, our training set consists of 14,204 images (69.88%), our validation set consists of 4,094 images (20.14%), and our test set consists of 2,029 images (9.98%). The code for splitting the dataset can be found on our GitHub; however, it should be noted that each split is random and therefore not reproducible.

Table 1. Model performance comparison of YOLOv8s and YOLOv8m trained on the GRAZPEDWRI-DX dataset using SGD and Adam optimizers. For training with the SGD optimizer, the initial learning rate is 1×10^{-2}; for training with the Adam optimizer, the initial learning rate is 1×10^{-3}.

Model	Size	Optimizer	Best Epoch	mAP^{val} 50	mAP^{val} 50-95	Speed GPU 3080Ti (ms)
YOLOv8s	640	SGD	56	0.611	0.389	4.4
YOLOv8s	640	Adam	57	0.604	0.383	4.3
YOLOv8s	1024	SGD	36	0.623	0.395	5.4
YOLOv8s	1024	Adam	47	0.625	0.399	4.9
YOLOv8m	640	SGD	52	0.621	0.396	4.9
YOLOv8m	640	Adam	62	0.621	0.403	5.5
YOLOv8m	1024	SGD	35	0.624	0.402	9.9
YOLOv8m	1024	Adam	70	0.626	0.401	10.0

Table 2. Detailed results for each class evaluated on YOLOv8s model when the input image size is 1024, with evaluation metrics including boxes, instances, precision, recall, and mAP.

Class	Size	Images	Boxes	Instances	Precision	Recall	mAP^{val} 50	mAP^{val} 50-95
all	1024	4094	47435	9613	0.674	0.605	0.623	0.395
boneanomaly	1024	4094	276	53	0.505	0.094	0.110	0.035
bonelesion	1024	4094	45	8	0.629	0.250	0.416	0.212
fracture	1024	4094	18090	3740	0.885	0.903	0.947	0.572
metal	1024	4094	818	168	0.878	0.899	0.920	0.768
periostealreaction	1024	4094	3453	697	0.645	0.684	0.689	0.357
pronatorsign	1024	4094	567	104	0.561	0.713	0.611	0.338
softtissue	1024	4094	464	89	0.324	0.315	0.251	0.125
text	1024	4094	23722	4754	0.961	0.984	0.991	0.750

Table 3. Detailed results for each class evaluated on our model when the input image size is 1024, with evaluation metrics including boxes, instances, precision, recall, and mAP.

Class	Size	Images	Boxes	Instances	Precision	Recall	mAP^{val} 50	mAP^{val} 50-95
all	1024	4094	47435	9613	0.694	0.592	0.631	0.402
boneanomaly	1024	4094	276	53	0.510	0.151	0.169	0.076
bonelesion	1024	4094	45	8	0.658	0.243	0.414	0.213
fracture	1024	4094	18090	3740	0.899	0.896	0.947	0.569
metal	1024	4094	818	168	0.898	0.890	0.924	0.780
periostealreaction	1024	4094	3453	697	0.721	0.654	0.700	0.359
pronatorsign	1024	4094	567	104	0.534	0.683	0.611	0.342
softtissue	1024	4094	464	89	0.367	0.236	0.241	0.120
text	1024	4094	23722	4754	0.961	0.981	0.991	0.754

Evaluation Metric

Intersection over Union (IoU)

Intersection over Union (IoU) is a classical metric for evaluating the performance of the model for object detection. It calculates the ratio of the overlap and union between the generated candidate bounding box and the ground truth bounding box, which measures the intersection of these two bounding boxes. The IoU is represented by the following equation:

$$IoU = \frac{area(C) \cap area(G)}{area(C) \cup area(G)}, \tag{8}$$

where C represents the generated candidate bounding box, and G represents the ground truth bounding box containing the object. The performance of the model improves as the IoU value increases, with higher IoU values indicating less difference between the generated candidate and ground truth bounding boxes.

Precision-Recall Curve

Precision-Recall Curve (P-R Curve)[65] is a curve with recall as the x-axis and precision as the y-axis. Each point represents a different threshold value, and all points are connected as a curve. The recall (R) and precision (P) are calculated according to the following equations:

$$Recall = \frac{T_P}{T_P + F_N}, \; Precision = \frac{T_P}{T_P + F_P}, \tag{9}$$

where True Positive (T_P) denotes the prediction result as a positive class and is judged to be true; False Positive (F_P) denotes the prediction result as a positive class but is judged to be false, and False Negative (F_N) denotes the prediction result as a negative class but is judged to be false.

F1-score

The F-score is a commonly used metric to evaluate the model accuracy, providing a balanced measure of performance by incorporating both precision and recall. The F-score equation is as follows:

$$F\text{-}score = \frac{\left(1 + \beta^2\right) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{10}$$

When β = 1, the F1-score is determined by the harmonic mean of precision and recall, and its equation is as follows:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2T_P}{2T_P + F_P + F_N} \tag{11}$$

Table 4. Quantitative comparison (mean average precision) with state-of-the-art models on the GRAZPEDWRI-DX dataset when an input image size is 640. Speed CPU ONNX is the total time of validation per image on the CPU Intel Core i5, and Speed GPU 3080Ti is the total time of validation per image on the GPU NVIDIA GeForce RTX 3080Ti. The total time includes the preprocessing, inference, and post-processing time.

Model	Size	mAP^{val} 50	mAP^{val} 50-95	Speed CPU ONNX	Speed GPU 3080Ti	Params	FLOPs
YOLOv5n	640	0.589	0.339	\	2.8ms	1.77M	4.2B
YOLOv8n	640	0.601	0.374	67.4ms	2.9ms	3.01M	8.1B
Ours	640	0.605	0.379	111.3ms	3.4ms	3.01M	8.2B
YOLOv5s	640	0.601	0.357	\	3.3ms	7.03M	15.8B
YOLOv8s	640	0.604	0.383	191.5ms	4.3ms	11.13M	28.5B
Ours	640	0.612	0.392	285.1ms	4.9ms	11.13M	28.7B
YOLOv5m	640	0.613	0.371	\	4.0ms	20.89M	48.0B
YOLOv8m	640	0.621	0.403	536.4ms	5.5ms	25.84M	78.7B
Ours	640	0.629	0.404	685.9ms	5.1ms	25.84M	78.7B
YOLOv5l	640	0.620	0.379	\	5.6ms	46.15M	107.8B
YOLOv8l	640	0.624	0.403	1006.3ms	7.4ms	43.61M	164.9B
Ours	640	0.637	0.406	1370.8ms	7.2ms	43.61M	164.9B

Table 5. Quantitative comparison (mean average precision) with state-of-the-art models on the GRAZPEDWRI-DX dataset when an input image size is 1024. Speed CPU ONNX is the total time of validation per image on the CPU Intel Core i5, and Speed GPU 3080Ti is the total time of validation per image on the GPU NVIDIA GeForce RTX 3080Ti. The total time includes the preprocessing, inference, and post-processing time.

Model	Size	mAP^{val} 50	mAP^{val} 50-95	Speed CPU ONNX	Speed GPU 3080Ti	Params	FLOPs
YOLOv5n	1024	0.600	0.347	\	3.2ms	1.77M	4.2B
YOLOv8n	1024	0.605	0.387	212.1ms	3.3ms	3.01M	8.1B
Ours	1024	0.608	0.391	260.4ms	4.4ms	3.01M	8.1B
YOLOv5s	1024	0.622	0.371	\	4.4ms	7.03M	15.8B
YOLOv8s	1024	0.625	0.399	519.5ms	4.9ms	11.13M	28.5B
Ours	1024	0.631	0.402	717.1ms	6.2ms	11.13M	28.5B
YOLOv5m	1024	0.624	0.380	\	7.1ms	20.89M	48.0B
YOLOv8m	1024	0.626	0.401	1521.5ms	10.0ms	25.84M	78.7B
Ours	1024	0.635	0.411	1724.4ms	9.4ms	25.85M	78.7B
YOLOv5l	1024	0.626	0.378	\	11.3ms	46.15M	107.8B
YOLOv8l	1024	0.636	0.404	2671.1ms	15.1ms	43.61M	164.9B
Ours	1024	0.638	0.415	3864.5ms	13.6ms	43.61M	164.9B

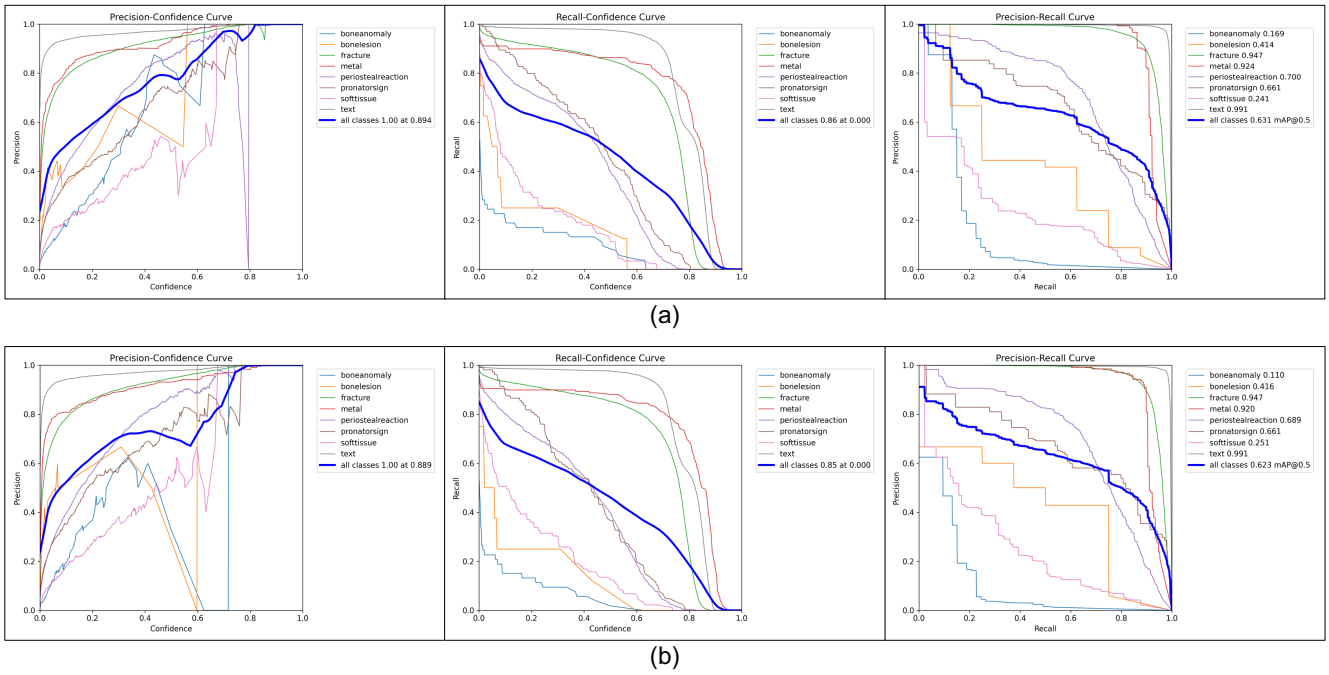

Figure 5. Detailed illustration of the validation at the input image size of 1024, (a) is our model, and (b) is YOLOv8s model.

Experiment Setup

During the model training process, we utilize pre-trained YOLOv8 model from the MS COCO (Microsoft Common Objects in Context) val2017 dataset[66]. The research reports provided by Ultralytics[33,49] suggests that YOLOv5 training requires 300 epochs, while training YOLOv8 requires 500 epochs. Since we use pre-trained model, we initially set the total number of epochs to 200 with a patience of 50, which indicate that the training would end early if no observable improvement is noticed after waiting for 50 epochs. In the experiment comparing the effect of the optimizer on the model performance, we notice that the best epcoh of all the models is within 100, as shown in Table 1, mostly concentrated between 50 and 70 epochs. Therefore, to save computing resources, we adjust the number of epochs for our model training to 100.

As the suggestion[33] of Glenn, for model training hyperparameters, the Adam optimizer[67] is more suitable for small custom datasets, while the SGD optimizer[68] perform better on larger datasets. To prove the above conclusion, we train YOLOv8 algorithm models using the Adam and SGD optimizers, respectively, and compare the effects on the model performance. The comparison results are shown in Table 1.

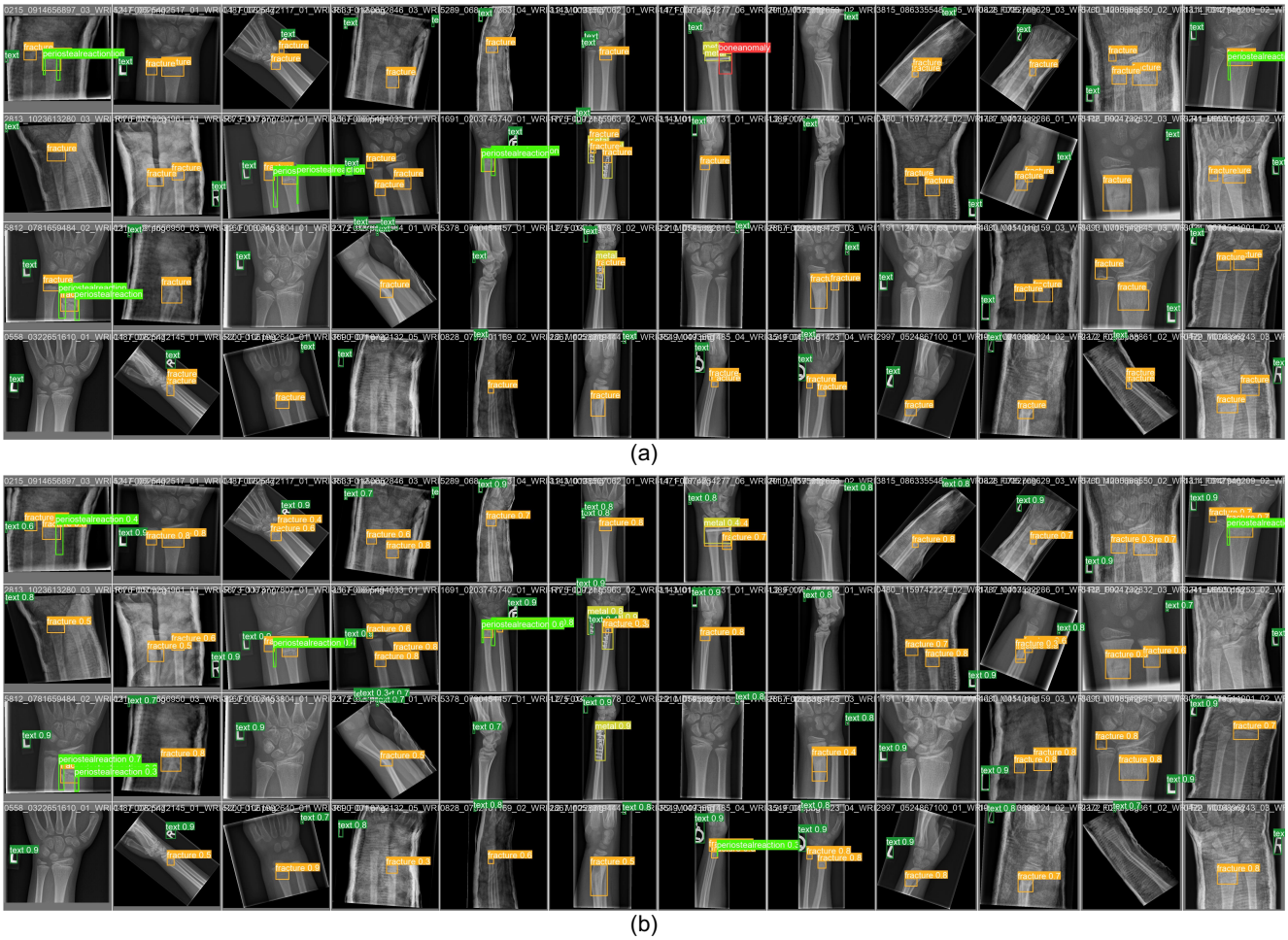

Figure 6. The prediction examples of our model on the pediatric wrist trauma X-ray images. (a) the manually labeled images, (b) the predicted images.

For the experiments, we choose the SGD optimizer[68] with an initial learning rate of 1×10^{-2}, a weight decay of 5×10^{-4}, and a momentum of 0.937 during our model training. We set the input image size to 640 and 1024 for training on a single GPU GeForce RTX 3080Ti 12GB with a batch size of 16. We train the model using Python 3.8 and PyTorch 1.8.2, and recommend readers to use Python 3.7 or higher and PyTorch 1.7 or higher for training. It is noteworthy that due to GPU memory limitations, we choose 3 worker threads to load data on GPU GeForce RTX 3080Ti 12GB when training our model. Therefore, using GPUs with larger memory and more computing power can effectively increase the speed of model training.

Experimental Results

Before training our model, in order to choose an optimizer that has a more positive effect on the model performance, we compare the performance of models trained with the SGD[68] optimizer and the Adam[67] optimizer. As shown in Table 1, using the SGD optimizer to train the model requires less epochs of weight updates. Specifically, for YOLOv8m model with an input image size of 1024, the model trained with the SGD optimizer achieves the best performance at the 35th epoch, while the best performance of the model trained with the Adam optimizer is at the 70th epoch. In terms of mAP and inference time, there is not much difference in the performance of the models trained with the two optimizers. Specifically, when the input image size is 640, the mAP value of YOLOv8s model trained with the SGD optimizer is 0.007 higher than that of the model trained with the Adam optimizer, while the inference time is 0.1ms slower. Therefore, according to the above experimental results and the suggestion by Glenn[33,49], for YOLOv8 model training on a training set of 14,204 X-ray images, we choose the Adam optimizer. However, after using data augmentation, the number of X-ray images in the training set extend to 28,408, so we switch to the SGD optimizer to train our model.

In order to prepare for data augmentation, we calculate the experimental data for each class respectively, as shown in Table 2. Among all classes, YOLOv8s model has good accuracy in detecting fracture, metal and text, with mAP 50 of each above 0.9.

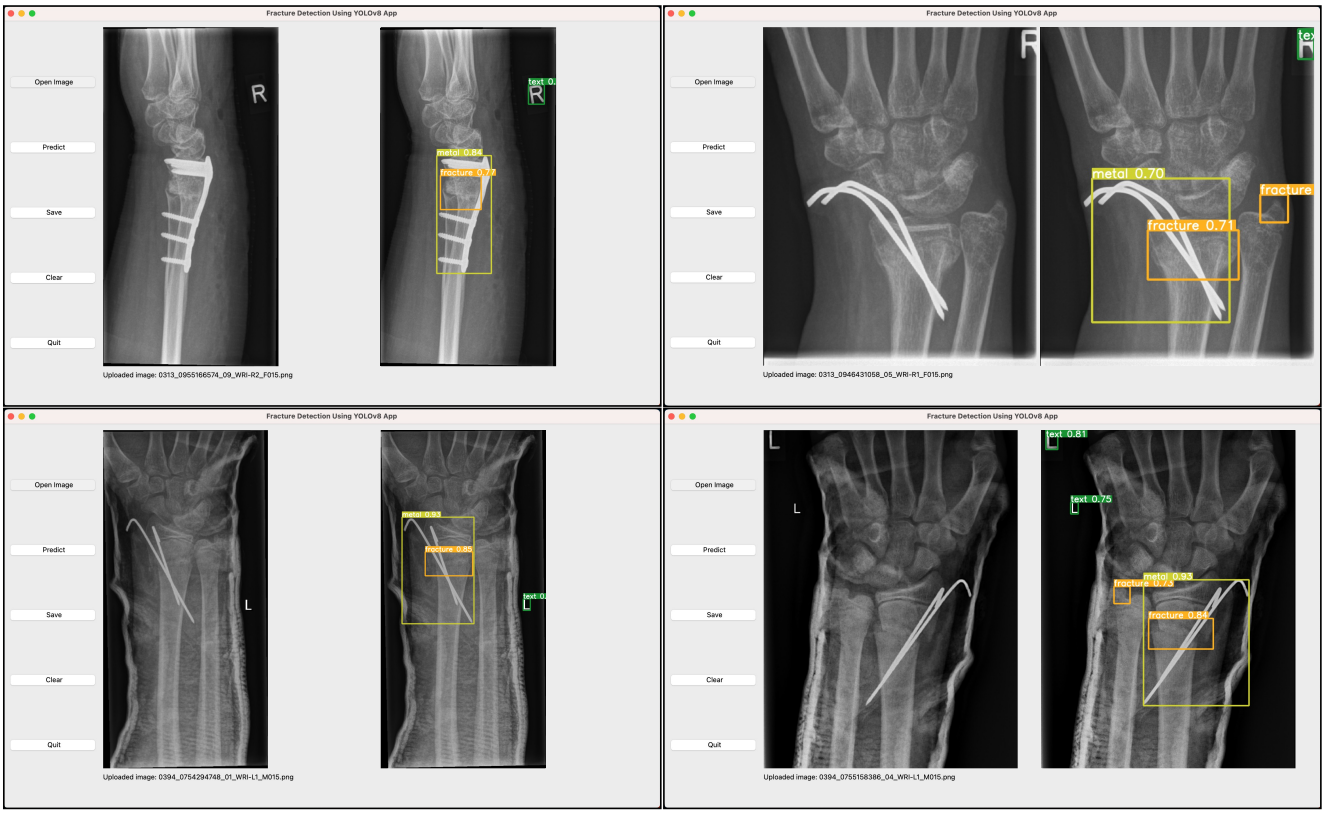

Figure 7. Example of using the GUI application "Fracture Detection with YOLOv8 Application" on macOS.

On the opposite, the detection ability of bone-anomaly is poor, with mAP 50 of 0.11. Therefore, we increase the contrast and brightness of X-ray images to make bone-anomaly easier to detect. Table 3 presents the predictions of our model for each class. Compared with YOLOv8s model, the mAP value predicted by our model for bone-anomaly increased from 0.11 to 0.169, an increase of 53.6%. Fig. 5 also shows that our model has a better performance in detecting bone-anomaly, which enables the improvement of the overall model performance. The above experimental results demonstrate that data augmentation has a positice effect on the model performance.

After using data augmentation, our models have a better mAP value than that of YOLOv8 model, and have reached the state-of-the-art (SOTA) real-time model performance, as shown in Table 4 and Table 5. Specifically, when the input image size is 640, compared with YOLOv8m model and YOLOv8l model, the mAP 50 of our model improves from 0.621 to 0.629, and from 0.623 to 0.637, respectively. Although the inference time on the CPU is increased from 536.4ms and 1006.3ms to 685.9ms and 1370.8ms, respectively, the number of parameters and FLOPs are the same, which means that our model can be deployed on the same computing power platform.

This paper aims to design a pediatric wrist fracture detection application, so we use our model for fracture detection. Fig. 6 shows the results of manual annotation by the radiologist and the results predicted using our model. These results demonstrate that our model can correctly detect fractures in most routine cases, however the presence of metal punctures badly affects the accuracy of prediction.

Application

After completing model training, we utilize a Python library that includes the Qt toolkit, PySide6, to develop a Graphical User Interface (GUI) application. Specifically, PySide6 is the Qt6-based version of the PySide GUI library from the Qt Company.

According to the model performance evaluation results in Table 4 and Table 5, we choose our model with YOLOv8s algorithm and the input image size of 1024, to perform fracture detection. Our model is exported to onnx format, and is applied to the GUI application. Fig. 7 depicts the flowchart of the GUI application operation on macOS. As can be seen from the illustration, our application is named "Fracture Detection Using YOLOv8 App". Users can open and predict the images, and save the predictions in this application. In summary, our application is designed to assist pediatric surgeons in analyzing fractures on pediatric wrist trauma X-ray images.

Conclusions and Future Work

Ultralytics proposed the latest version of YOLO series (YOLOv8) in 2023. Although there are relatively few research works on YOLOv8 model for medical image processing, we apply it to fracture detection and use data augmentation to improve the model performance. We randomly divide the dataset, consisting of 20,327 pediatric wrist trauma X-ray images from 6,091 patients, into training, test, and validation sets to train the model and evaluate the performance.

Furthermore, we develop an application named "Fracture Detection Using YOLOv8 App" to analyze pediatric wrist trauma X-ray images for fracture detection. Our application aims to assist pediatric surgeons in interpreting X-ray images, reduce the probability of misclassification, and provide a better information base for surgery. The application is currently available for macOS, and in the future, we plan to deploy different sizes of our model in the application, and extend the application to iOS and Android. This will enable inexperienced pediatric surgeons in hospitals located in underdeveloped areas to use their mobile devices to analyze pediatric wrist X-ray images.

References

1. Fractures, health, hopkins medicine. https://www.hopkinsmedicine.org/health/conditions-and-diseases/fractures (2021).
2. Hedström, E. M., Svensson, O., Bergström, U. & Michno, P. Epidemiology of fractures in children and adolescents: Increased incidence over the past decade: a population-based study from northern sweden. *Acta orthopaedica* **81**, 148–153 (2010).
3. Randsborg, P.-H. *et al.* Fractures in children: epidemiology and activity-specific fracture rates. *JBJS* **95**, e42 (2013).
4. Burki, T. K. Shortfall of consultant clinical radiologists in the uk. *The Lancet Oncol.* **19**, e518 (2018).
5. Rimmer, A. Radiologist shortage leaves patient care at risk, warns royal college. *BMJ: Br. Med. J. (Online)* **359** (2017).
6. Rosman, D. *et al.* Imaging in the land of 1000 hills: Rwanda radiology country report. *J. Glob. Radiol.* **1** (2015).
7. Mounts, J., Clingenpeel, J., McGuire, E., Byers, E. & Kireeva, Y. Most frequently missed fractures in the emergency department. *Clin. pediatrics* **50**, 183–186 (2011).
8. Erhan, E., Kara, P., Oyar, O. & Unluer, E. Overlooked extremity fractures in the emergency department. *Ulus Travma Acil Cerrahi Derg* **19**, 25–8 (2013).
9. Adams, S. J., Henderson, R. D., Yi, X. & Babyn, P. Artificial intelligence solutions for analysis of x-ray images. *Can. Assoc. Radiol. J.* **72**, 60–72 (2021).
10. Tanzi, L. *et al.* Hierarchical fracture classification of proximal femur x-ray images using a multistage deep learning approach. *Eur. journal radiology* **133**, 109373 (2020).
11. Chung, S. W. *et al.* Automated detection and classification of the proximal humerus fracture by using deep learning algorithm. *Acta orthopaedica* **89**, 468–473 (2018).
12. Choi, J. W. *et al.* Using a dual-input convolutional neural network for automated detection of pediatric supracondylar fracture on conventional radiography. *Investig. radiology* **55**, 101–110 (2020).
13. Gan, K. *et al.* Artificial intelligence detection of distal radius fractures: a comparison between the convolutional neural network and professional assessments. *Acta orthopaedica* **90**, 394–400 (2019).
14. Kim, D. & MacKinnon, T. Artificial intelligence in fracture detection: transfer learning from deep convolutional neural networks. *Clin. radiology* **73**, 439–445 (2018).
15. Lindsey, R. *et al.* Deep neural network improves fracture detection by clinicians. *Proc. Natl. Acad. Sci.* **115**, 11591–11596 (2018).
16. Blüthgen, C. *et al.* Detection and localization of distal radius fractures: Deep learning system versus radiologists. *Eur. journal radiology* **126**, 108925 (2020).
17. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587 (2014).
18. Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 1440–1448 (2015).
19. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. neural information processing systems* **28** (2015).
20. Cai, Z. & Vasconcelos, N. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6154–6162 (2018).

21. Lu, X., Li, B., Yue, Y., Li, Q. & Yan, J. Grid r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7363–7372 (2019).

22. Pang, J. *et al.* Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 821–830 (2019).

23. Zhang, H., Chang, H., Ma, B., Wang, N. & Chen, X. Dynamic r-cnn: Towards high quality object detection via dynamic training. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, 260–275 (Springer, 2020).

24. He, K., Gkioxari, G., Dollár, P. & Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969 (2017).

25. Liu, W. *et al.* Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 21–37 (Springer, 2016).

26. Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988 (2017).

27. Law, H. & Deng, J. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, 734–750 (2018).

28. Duan, K. *et al.* Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6569–6578 (2019).

29. Dong, Z. *et al.* Centripetalnet: Pursuing high-quality keypoint pairs for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10519–10528 (2020).

30. Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696* (2022).

31. Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430* (2021).

32. Wang, C.-Y., Yeh, I.-H. & Liao, H.-Y. M. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206* (2021).

33. Glenn, J. Ultralytics yolov8. https://github.com/ultralytics/ultralytics (2023).

34. Nagy, E., Janisch, M., Hržić, F., Sorantin, E. & Tschauner, S. A pediatric wrist trauma x-ray dataset (grazpedwri-dx) for machine learning. *Sci. Data* **9**, 222 (2022).

35. Guan, B., Yao, J., Zhang, G. & Wang, X. Thigh fracture detection using deep learning method based on new dilated convolutional feature pyramid network. *Pattern Recognit. Lett.* **125**, 521–526 (2019).

36. Wang, M. *et al.* Parallelnet: Multiple backbone network for detection tasks on thigh bone fracture. *Multimed. Syst.* 1–10 (2021).

37. Guan, B., Zhang, G., Yao, J., Wang, X. & Wang, M. Arm fracture detection in x-rays based on improved deep convolutional neural network. *Comput. & Electr. Eng.* **81**, 106530 (2020).

38. Rajpurkar, P. *et al.* Mura dataset: Towards radiologist-level abnormality detection in musculoskeletal radiographs. In *Medical Imaging with Deep Learning* (2018).

39. Ma, Y. & Luo, Y. Bone fracture detection through the two-stage system of crack-sensitive convolutional neural network. *Informatics Medicine Unlocked* **22**, 100452 (2021).

40. Wu, H.-Z. *et al.* The feature ambiguity mitigate operator model helps improve bone fracture detection on x-ray radiograph. *Sci. Reports* **11**, 1–10 (2021).

41. Xie, S., Girshick, R., Dollár, P., Tu, Z. & He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500 (2017).

42. Lin, T.-Y. *et al.* Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125 (2017).

43. Qi, Y. *et al.* Ground truth annotated femoral x-ray image dataset and object detection based method for fracture types classification. *IEEE Access* **8**, 189436–189444 (2020).

44. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).

45. Sha, G., Wu, J. & Yu, B. Detection of spinal fracture lesions based on improved yolov2. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 235–238 (IEEE, 2020).

46. Sha, G., Wu, J. & Yu, B. Detection of spinal fracture lesions based on improved faster-rcnn. In *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, 29–32 (IEEE, 2020).

47. Redmon, J. & Farhadi, A. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7263–7271 (2017).

48. Xue, L. *et al.* Detection and localization of hand fractures based on ga_faster r-cnn. *Alex. Eng. J.* **60**, 4555–4562 (2021).

49. Glenn, J. Ultralytics yolov5. https://github.com/ultralytics/yolov5 (2022).

50. Yuan, G., Liu, G., Wu, X. & Jiang, R. An improved yolov5 for skull fracture detection. In *Exploration of Novel Intelligent Optimization Algorithms: 12th International Symposium, ISICA 2021, Guangzhou, China, November 20–21, 2021, Revised Selected Papers*, 175–188 (Springer, 2022).

51. Warin, K. *et al.* Maxillofacial fracture detection and classification in computed tomography images using convolutional neural network-based models. *Sci. Reports* **13**, 3434 (2023).

52. Tsai, H.-C. *et al.* Automatic rib fracture detection and localization from frontal and oblique chest x-rays. In *2022 10th International Conference on Orange Technology (ICOT)*, 1–4 (IEEE, 2022).

53. Burkow, J. *et al.* Avalanche decision schemes to improve pediatric rib fracture detection. In *Medical Imaging 2022: Computer-Aided Diagnosis*, vol. 12033, 597–604 (SPIE, 2022).

54. Warin, K. *et al.* Assessment of deep convolutional neural network models for mandibular fracture detection in panoramic radiographs. *Int. J. Oral Maxillofac. Surg.* **51**, 1488–1494 (2022).

55. Fatima, J., Mohsan, M., Jameel, A., Akram, M. U. & Muzaffar Syed, A. Vertebrae localization and spine segmentation on radiographic images for feature-based curvature classification for scoliosis. *Concurr. Comput. Pract. Exp.* **34**, e7300 (2022).

56. Mushtaq, M., Akram, M. U., Alghamdi, N. S., Fatima, J. & Masood, R. F. Localization and edge-based segmentation of lumbar spine vertebrae to identify the deformities using deep learning models. *Sensors* **22**, 1547 (2022).

57. Wang, C.-Y. *et al.* Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 390–391 (2020).

58. He, K., Zhang, X., Ren, S. & Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis machine intelligence* **37**, 1904–1916 (2015).

59. Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8759–8768 (2018).

60. Tian, Z., Shen, C., Chen, H. & He, T. Fcos: A simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis Mach. Intell.* **44**, 1922–1933 (2020).

61. Feng, C., Zhong, Y., Gao, Y., Scott, M. R. & Huang, W. Tood: Task-aligned one-stage object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 3490–3499 (IEEE Computer Society, 2021).

62. Li, X. *et al.* Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Adv. Neural Inf. Process. Syst.* **33**, 21002–21012 (2020).

63. Zheng, Z. *et al.* Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybern.* **52**, 8574–8586 (2021).

64. Zheng, Z. *et al.* Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 12993–13000 (2020).

65. Boyd, K., Eng, K. H. & Page, C. D. Area under the precision-recall curve: point estimates and confidence intervals. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, 451–466 (Springer, 2013).

66. Lin, T.-Y. *et al.* Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755 (Springer, 2014).

67. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

68. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).