End-To-End Pediatric Bone Fracture Localization with Data Augmentation

Jingwen Wu
Stanford University
jingwenw@stanford.edu

1. Abstract

In pediatric patients, bone fractures can account for 25% of all childhood injuries, and approximately 1 in 3 children experience at least one bone fracture before age 17 [7]. Given the frequency of bone trauma in youth and the limited availability of trained radiologists for image interpretation, effective treatment is dependent on accurate and prompt diagnosis [15]. Thus, object detection via deep learning CNNs can serve as effective supplementary information for diagnosing bone fractures. There are two significant challenges of performing object detection on medical imaging datasets: 1) class imbalance, where the rarity of certain medical conditions leads to limited availability of images [8], and 2) the prevalence of noisy bounding box annotations in medical images that are difficult to interpret [32]. In this paper, we finetune state-of-the-art RT-DETR [35] and YOLOv9 [28] end-to-end object detection models on the GRAZPEDWRI-DX pediatric bone fracture X-ray dataset [18] and obtain increases in model performance compared to baseline results after utilizing data augmentation techniques to mitigate class imbalance. We additionally analyze the robustness of these models to erroneous/noisy labels by further perturbing the bounding box annotations of training images, and find that model performance is generally robust to label noise for this dataset.

2. Introduction

Since bone fractures and other forms of bone trauma are common in young patients, an accurate and timely diagnosis is crucial to ensuring successful treatment. As Lindsey et al. demonstrate, the supplementary information provided by automated diagnosis tools can improve the accuracy of fracture detection and diagnosis by emergency medical clinicians [15]. Often, these medical professionals need to interpret X-rays and make a speedy diagnosis in emergency situations where radiologists may not be available, but they are not trained in orthopedics or radiology; thus, deep neural network models could provide valuable information to prevent misdiagnosis [15].

In this project, we finetune the state-of-the-art RT-DETR [35] and YOLOv9 [28] object detection models and utilize data augmentation techniques to perform object localization on a dataset of pediatric bone fracture X-rays, in order to generate bounding box labels for 9 classes (bone anomaly, bone lesion, foreign body, fracture, metal, periosteal reaction, pronator sign, soft tissue, text) corresponding to locations of interest. We apply and compare both one-shot and two-shot object detectors to analyze the tradeoff between accuracy and efficiency.

We perform bone fracture localization on pediatric bone X-ray images by predicting the bounding boxes for different instances, including fractures, lesions, metal (e.g. implants from previous orthopedic surgeries), periosteal reactions (new bone formation), or other abnormalities in the bone structure, using the GRAZPEDWRI-DX dataset [18] as described in the Datasets section. Specifically, we finetune CNN object detection models (RT-DETR [35], YOLOv9 [28]) on the input of each train X-ray image, which is annotated with bounding box coordinates for each object of interest and the corresponding class label for each set of coordinates; the output consists of all predicted bounding box coordinate(s) and class(es). Each image can have multiple instances of multiple classes. We evaluate model performance primarily using mAP (Mean Average Precision) at IoU thresholds of 50 and 50-95, and secondarily using precision and recall.

2.1. Related Work

2.1.1 One-Stage Detection

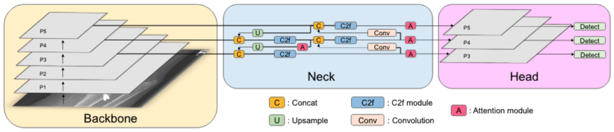


Figure 1: YOLOv8-AM model architecture with additional attention modules. Source: Chien et al. [6]

The YOLO models are a series of one-stage object detection systems that simultaneously predict object classes and bounding boxes through one regression task [20]. The current SoTA model on the GRAZPEDWRI-DX dataset is Chien et al.'s YOLOv8 with Attention Mechanisms (see Fig. 1), which experiments with adding four attention modules (i.e., Convolutional Block Attention [30], Global Attention [16], Efficient Channel Attention [29], and Shuffle Attention [34]) to the original YOLOv8 model [6, 21]. These variations on attention mechanisms can help the model focus on the relevant parts of the image across different dimensions, and thus predict bounding boxes more accurately [6]. Out of these four models, YOLOv8+ResCBAM performs best, achieving an overall mAP-50 score of 0.658 on GRAZPEDWRI-DX using an image size of 1024 [6].

This is followed closely by Chien et al.'s adapted YOLOv9-E model with additional brightness/contrast data augmentation, which achieves a mAP-50 score of 0.6562 on GRAZPEDWRI-DX with an input image size of 1024 [4], although YOLOv9-E outperforms YOLOv8+ResCBAM on the lower-resolution input image size of 640 (mAP-50 score of 0.6546 vs 0.6295). The performance of these two models is comparable, but YOLOv9-E requires more compute resources due to its increased complexity [4].

Dibo et al. also proposed the DeepLOC model (Deep Learning-based Bone Pathology Localization and Classification) which is based on YOLOv7 and achieves a mAP-50 score of 0.654 on GRAZPEDWRI-DX [9]. The YOLOv7 architecture is mainly characterized by model reparameterization and dynamic label assignment [27]. DeepLOC modifies YOLOv7 by adding Shifted Window Transformer blocks [17] and Global Attention blocks [16] in order to lessen the computational complexity of the original Transformer module and calculate attention across the channel dimensions of the input image [9].

2.1.2 Two-Stage Detection

Other general bone fracture detection systems have focused on transfer learning through finetuning two-stage detectors, with a main focus on Faster-RCNN [22]. The Faster-RCNN model is a two-module network (region proposal, detector) that improves upon R-CNN [11] and Fast-RCNN [10] through a Region Proposal Network (RPN) [22]. The RPN uses a CNN to first identify potential regions where any instances of objects of interest may be located in the input image, and then only retains the regions that it is most confident about [22].

One benefit of the Faster-RCNN model is that the RPN backbone can be substituted with many known CNN networks. The original Faster-RCNN authors [22] experimented with using backbones such as ZF [33], VGG-16 [23] and ResNet101 [12]. Tabarestani et al. further applied this experimentation to fracture detection by finetuning Faster-RCNN with InceptionV2 [13], ResNet101 [12], and Inception-ResNet V2 [24] backbones on the MURA dataset [19] of upper-body bone X-rays to achieve an AP@0.5 of 0.634 [25]. Abbas et al. also successfully finetuned the last layers of Faster-RCNN to detect lower-body bone fractures [1]. However, Faster-RCNN can be more inefficient to train due to the large number of parameters in most backbone models (e.g., approximately 138 million parameters for VGG [23]). Furthermore, through our own experimentation with finetuning Faster-RCNN on GRAZPEDWRI-DX, we were not able to achieve reasonable mAP scores, suggesting that two-stage detectors may be more challenging to finetune than one-stage detectors for our dataset, especially when compute resources are limited.

2.1.3 Noisy Medical Image Annotations

As Xue et al. notes, the quality of object detection is significantly biased by the quality of bounding box annotations, and such noisy annotations are common for medical imaging datasets where the inherent ambiguity of images can lead to errors in expert labelling [32]. Thus, robust medical image object detection is a crucial area of study, especially in scenarios where there is class imbalance or a lack of easily-accessible expert-annotated medical data [32].

Research in this area has primarily centered on representing model uncertainty about labels for training images, modifying loss functions to account for noisy labels, or filtering out noisy samples [32]. For instance, Xue et al. proposes a skin lesion classification network to detect images with noisy labels through online uncertainty sample mining [31]. Since noisy labels are a general issue in deep learning that can lead to networks overfitting erroneous labels, Tanaka et al. also proposes a joint optimization framework that continuously updates and potentially corrects noisy labels, which has proven to be effective [26].

3. Technical Approach

3.1. YOLOv9 Baseline

The baseline is obtained by finetuning a pretrained YOLOv9 model on GRAZPEDWRI-DX. Compared to previous YOLO models, YOLOv9 is unique as it introduces the Programmable Gradient Information (PGI) framework, which attempts to solve potential information loss issues as denoted by the Information Bottleneck Principle [4]. Although issues with deep network training are typically attributed to vanishing/exploding gradients, the Information Bottleneck Principle describes another issue with training deep neural networks where the various many layers of transformations applied to the input data, especially for larger models, can cause certain information to be lost from the original image [4].

Although Chien et al. provide the mAP scores obtained by finetuning YOLOv9 on this dataset [4], we reproduce their method ourselves as a baseline for a more accurate comparison based on compute constraints, by finetuning a YOLOv9-C model using the pretrained weights on our train split of the dataset. We use an existing off-the-shelf YOLOv9 implementation provided by the YOLOv9 authors [3, 4], and the starting code provided by Chien et al. and Wang et al. [5, 28], for training and evaluation. Ultralytics is also used to compute evaluation metrics (e.g. mAP, precision, recall) and visualize results [14].

3.2. RT-DETR

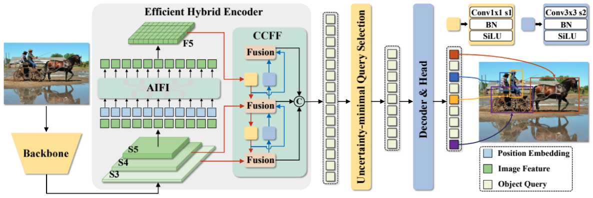


Figure 2: The RT-DETR architecture, which improves upon DETR by using an efficient hybrid encoder. Source: Zhao et al. [35]

An issue with the chosen YOLO baseline is that the accuracy and efficiency of YOLO models is somewhat limited by the Non-Maximum Suppression process, which is required to filter out incorrect or duplicate object detection labels [35]. Thus, since Detection Transformer (DETR) models do not require this NMS step, they are a viable and potentially more efficient alternative to consider [35].

The RT-DETR model is a realtime end-to-end detector which does not need to propose regions or filter out the various bounding box candidates, since it matches the bounding box predictions with the objects in the image directly [35]. As shown in Fig. 2, the model speeds up the original DETR model by using a hybrid encoder, and uses uncertainty-minimal query selection to determine the ”object queries” that the decoder will match to each bounding box prediction [35]. Uncertainty-minimal query selection is motivated by the difficulty in concurrently representing features for both the classification and localization of an object, and is empirically shown to produce more meaningful features [35]. Given $\hat{\mathcal{X}}$ as the encoder feature representation, $\mathcal{P}, \mathcal{C}$ as predicted localization and classification, and $\hat{\boldsymbol{\mathcal{Y}}} = \{\hat{c}, \hat{\boldsymbol{b}}\}, \boldsymbol{\mathcal{Y}} = \{\boldsymbol{c}, \boldsymbol{b}\}$ as the predicted and ground-truth categories/bounding boxes, this method adds the minimization of the difference between prediction and classification distributions (Eq. 1) as an objective in the loss function (Eq. 2):

$$U(\hat{\boldsymbol{\mathcal{X}}}) = ||\mathcal{P}(\hat{\boldsymbol{\mathcal{X}}}) - \mathcal{C}(\hat{\boldsymbol{\mathcal{X}}})|| \qquad (1)$$

$$\mathcal{L}(\hat{\boldsymbol{\mathcal{X}}}, \hat{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\mathcal{Y}}) = \mathcal{L}_{box}(\hat{\boldsymbol{b}}, \boldsymbol{b}) + \mathcal{L}_{cls}(\mathcal{U}(\hat{\boldsymbol{\mathcal{X}}}), \hat{\boldsymbol{c}}, \boldsymbol{c}) \qquad (2)$$

To the best of our knowledge, RT-DETR has not yet been applied to bone fracture detection; thus, we will finetune RT-DETR on our dataset using the existing off-the-shelf model implementation as explained by the authors [35] and provided by the Ultralytics code library for training and evaluation [14]. We also use Ultralytics for computation of evaluation metrics (mAP, precision, recall, etc.) and additional visualization of results.[14].

3.3. Data Augmentation

One challenge with utilizing medical imaging datasets is the natural issue of class imbalance: certain medical abnormalities, or the lack thereof, may be rarer than others, and high-quality expert labeled data is difficult to acquire in large quantities [8]. For object detection, the quality and accuracy of labels can be especially imperative to the model performance, as learning incorrect labels is counterproductive. Thus, data augmentation is one proposed solution - techniques such as random cropping, rotations, and brightness/contrast adjustments can effectively expand dataset size and mitigate class imbalance [8].

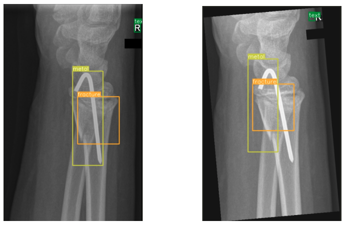
Thus, we finetune YOLOv9 and RT-DETR on an augmented GRAZPEDWRI-DX dataset by implementing data augmentation techniques from scratch. Our proposed augmentation method is as follows: each training image is randomly selected for augmentation with a chosen probability p_a. Then, the image undergoes the following successive transformations, where each probability is a tunable hyperparameter:

1) random crop with probability p_c
2) rotation by 5 degrees with probability p_r
3) brightness adjustment:
image $\leftarrow \alpha \cdot$ image $+ \beta$, with $\alpha = 1.2, \beta = 20$, with probability p_b (make image brighter)
image $\leftarrow \alpha \cdot$ image $- \beta$, with $\alpha = 0.9, \beta = 20$, with probability $1 - p_b$ (make image darker)

For an image, the random crop procedure is:
1) Obtain $(x_{bl}, x_{bh}, y_{bl}, y_{bh})$: the minimum x-value, maximum x-value, minimum y-value, and maximum y-value seen out of all bounding box labels for the image.
2) Obtain the newly cropped image coordinates (x_l, x_h, y_l, y_h) from a uniform distribution:
$\{x_l, y_l\} \sim Uniform(0, \min(\{x_{bl}, y_{bl}\}, 0.2))$
$\{x_h, y_h\} \sim Uniform(\max(\{x_{bh}, y_{bh}\}, 0.85), 1)$
0.2 is chosen as the upper bound for lower cropping, and 0.85 as the upper bound for upper cropping, meaning that the minimum area preserved for a cropped image is $0.65 \cdot 0.65 = 42.25\%$.
3) Update the bounding boxes:
$(x, y, w, h) \leftarrow (x - x_l, y - y_l, w, h)$

4) Crop the image using (x_l, x_h, y_l, y_h).

Finally, we finetune each model on the original training set combined with the augmented image set.



(a) before augmentation

(b) after augmentation

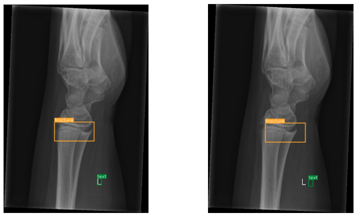
Figure 3: Sample image before and after crop, brightness, and rotation augmentation with noisy bbox annotations.

Another aspect of interest in this project is how robust the model is to noisy annotations. Although research in this area primarily focuses on modifications based on model uncertainty, loss functions that account for noisy labels, or filtering out noisy samples [32], we wish to analyze the degree to which this noise negatively affects model performance, and whether this effect is significant enough for concern. In the above data augmentation scheme, the random rotation of 5 degrees can cause bounding box labels to be inaccurate. Figure 3 displays such an example where the bounding boxes become noisy as a result of augmentation. This noise is intentionally added to analyze whether the model can still maintain reasonable performance despite the presence of noisy or erroneous annotations. We do not perform any augmentation or noising on the validation or test sets. Thus, if we select $p_r > 0$, then the training data is augmented with a number of noisy annotations approximately proportional to p_r.

3.4. Additional Noising

To further evaluate model robustness, we implement another method to add "noise" to the training labels through perturbation. In our proposed method, each training image is randomly selected for "noising" with probability p_n. Then, the chosen image is "noised" as follows:

1) For each bounding box annotation (x, y, w, h), compute perturbation:
$w_{perturb} \sim Uniform(0, w_{perturb\_max})$
$h_{perturb} \sim Uniform(0, h_{perturb\_max})$
2) Adjust out-of-bounds annotations:
If $x + w_{perturb} \geq 1 : w_{perturb} \sim Uniform(0, 1 - x)$
If $y + h_{perturb} \geq 1 : h_{perturb} \sim Uniform(0, 1 - y)$
3) Set new x and y:
$x, y \leftarrow x + w_{perturb}, y + h_{perturb}$



(a) before noising

(b) after noising

Figure 4: Sample image before and after noising with bbox annotations.

Figure 4 displays the effect of adding perturbations on the bounding box labels. The ground-truth labels have been shifted enough to be considered erroneous.

4. Dataset

Split	**anom.**	**lesion**	**foreign**	**fracture**	**metal**	**perios.**	**pronat.**	**tissue**	**text**
train	184	26	8	12612	567	2409	408	316	16588
train+aug v1 YOLO	202	28	11	13834	622	2623	448	343	18238
train+aug v1 RT-DETR	202	30	8	13814	616	2627	455	347	18235
val	53	8	0	3740	168	697	104	89	4754
test	39	11	0	1738	83	347	55	59	2380

Table 1: Instances of each class for each data split.

We use the GRAZPEDWRI-DX dataset, which consists of 20,327 grayscale pediatric wrist X-rays annotated with bounding boxes for 9 classes: bone anomaly, bone lesion, foreign body, fracture, metal, periosteal reaction, pronator sign, soft tissue, and text [18]. The annotations are validated by pediatric radiologist experts with board certifications [18]. We use the YOLO bounding boxes format: (x, y, w, h), where x, y are the center coordinates of the bounding box in the horizontal and vertical directions respectively, w is the box width, and h is the box height. All bounding box values are in decimal form.

We divide the dataset into train, validation, and test sets, using a random 70/20/10 split, as done in previous works by Chien et al. [4, 6]. There are 14204 train examples, 4094 validation examples, and 2029 test examples in the original split. For training and evaluation, we use an image size of 640 pixels. No additional data preprocessing was necessary.

Table 1 displays the number of instances of each class in each given data split: we note that each image can have multiple instances of multiple classes. Given the severe class imbalance, data augmentation was performed through the methods discussed in Section 3.3. Thus, we also provide the class breakdown of augmentation runs on the training set for two models to further illustrate the augmentation method. Because an image can have multiple bounding box annotations, and fractures/text are by far the two most prevalent classes and occur in almost every image, we choose to augment each class with an equal probability instead of only augmenting images with underrepresented classes, and find the former is empirically more effective.

5. Experiments, Results, Discussion

5.1. Experimental Details

5.1.1 Training

Hyperparameters We train all models on AWS using a NVIDIA A10G GPU. All YOLOv9 models are trained for 20 epochs and RT-DETR models are trained for 10 epochs. All models are trained and evaluated using a batch size of 16. For the YOLOv9 model, we use the same starting hyperparameters provided by Chien et al., given the proven success of these chosen values from Chien et al.'s YOLOv9 experimentations on this dataset: SGD optimizer with momentum=0.937, starting lr=0.01, weight decay=5e-4 [4]. For RT-DETR, Ultralytics [14] auto-determined the best optimizer as AdamW, with the chosen hyperparameters of starting lr=0.000769, momentum=0.9.

Augmentation V1 For augmentation as described in section 3.3, we experiment with different choices and ultimately find the most success with $p_a = 0.1, p_r = 0.1, p_b = 0.9, p_c = 0.2$. We run data augmentation separately for each model. For YOLOv9, we train on 14204 original examples and 1399 augmented images, for a total of 15603 images. For RT-DETR, we train on 14204 original examples and 1388 augmented images, for 15592 images total.

Augmentation V2 We choose $p_a = 0.1, p_r = 0, p_b = 1, p_c = 0$ to observe the effects of brightness augmentation only without adding noise to bounding box labels. For YOLOv9, we add 1497 augmented images, for 15701 images total. For RT-DETR, we add 1406 images and train on 15610 images total.

Noise We choose $p_n = 0.2$ and produce a noised training set still with 14204 images, where each image has probability p_n of having noisy labels. We set $w_{perturb\_max}, h_{perturb\_max}$ both to 0.1.

5.1.2 Evaluation

The primary evaluation metric is Mean Average Precision (mAP), where mAP $= \frac{1}{N}\sum_i^N AP_i$ for N classes and AP is average precision. mAP 50 indicates the mAP value at the IoU threshold of 0.5, whereas mAP 50-95 indicates the mAP at the IoU thresholds of 0.5 to 0.95, and IoU $= \frac{|X \cap X_t|}{|X \cup X_t|}$ for predicted bounding box X and ground-truth box X_t.

Precision and recall are used as secondary evaluation metrics, where Precision $= \frac{\text{TP}}{\text{TP+FP}}$ and Recall $= \frac{\text{TP}}{\text{TP+FN}}$.

We additionally use the Ultralytics library [14] and the starter code of YOLOv9 [28, 5] for further evaluation of our models (confusion matrices, PR curves, batch labels/predictions, feature visualizations).

5.2. Results and Discussion

Model	mAP 50	mAP 50-95	Precision	Recall
YOLOv9* (BL)	0.569	0.373	**0.723**	0.516
RT-DETR	0.571	0.379	0.702	0.537
YOLOv9+AugV1	0.612	0.400	0.590	0.592
RT-DETR+AugV1	**0.624**	**0.413**	0.665	**0.616**
YOLOv9+AugV2	0.604	0.403	0.683	0.576
RT-DETR+AugV2	0.61	0.396	0.642	0.61
YOLOv9+Noise	0.589	0.384	0.631	0.571
RT-DETR+Noise	0.607	0.403	0.694	0.583

Table 2: Model performance evaluated on the test set.

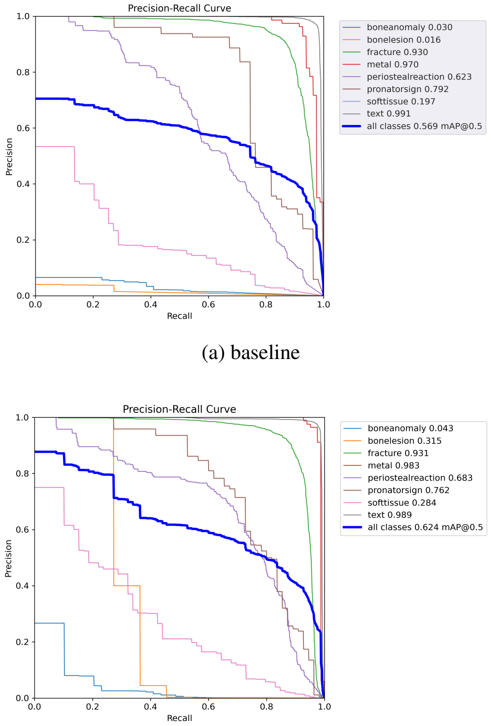
Table 2 displays the results of all models evaluated on the test set using the model weights that achieved the highest mAP scores on the validation set.

5.2.1 YOLOv9 Baseline and RT-DETR

We were unable to replicate the mAP@50 of 0.6562 obtained by Chien et al. using YOLOv9-E [4], potentially due to GPU compute limitations allowing us to only train for 20 epochs, compared to 100 epochs in the original paper. Additionally, RT-DETR slightly outperforms our baseline on mAP and recall despite being finetuned for 10 epochs compared to 20 for YOLOv9. We observe that during training, RT-DETR converges faster than YOLOv9, and takes an average of 7 minutes to train on each epoch, compared to approximately 10 minutes per epoch for YOLOv9. As described in section 3.2, the ability of RT-DETR to potentially converge faster than YOLOv9 could be due to the efficiency speed-ups brought by the hybrid encoder and because RT-DETR does not run the NMS algorithm [35].

5.2.2 Models with Data Augmentation

Figure 5 compares the PR curves of our YOLOv9 baseline with RT-DETR+AugV1. We observe the baseline has trouble predicting bounding boxes especially for bone anomaly (mAP@50=0.03), bone lesion (mAP@50=0.016), and soft tissue (mAP@50=0.197), which are also among the classes with the least number of instances (see Table 1). Thus, it is very likely that the baseline performance suffers from class imbalance in the dataset. In contrast, RT-DETR+AugV1 achieves a mAP@50=0.043 on bone anomaly, mAP@50=0.315 on bone lesion, and mAP@50=0.284 on soft tissue, which are significant improvements arising from simple data augmentation, despite the inclusion of erroneous or noisily labelled training data due to our random rotation augmentation method. Furthermore, the AugV1 YOLOv9 and RT-DETR slightly outper-



(a) baseline

(b) RT-DETR+AugV1

Figure 5: PR curves of baseline vs RT-DETR+AugV1.

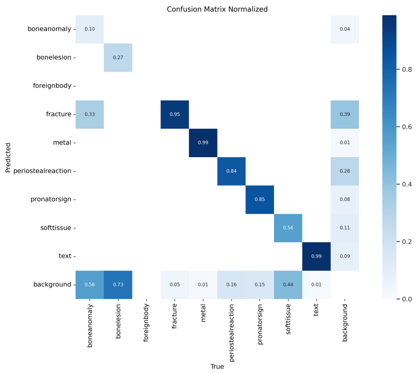
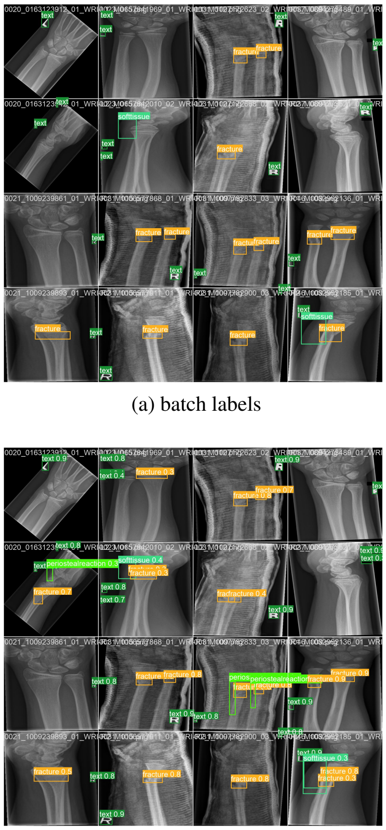


Figure 6: RT-DETR+AugV1 confusion matrix on test set.

form the AugV2 models, suggesting that the addition of randomly cropped and rotated images improves model performance compared to brightness adjustments alone. We hypothesize that these erroneous labels help regularize the model and prevents train set overfitting to some degree, since a select subset of the original train images are augmented with incorrect or noisy bounding box annotations. Furthermore, only adjusting image brightness might not be enough to offset overfitting, as RT-DETR and YOLOv9 can likely easily learn an affine transformation of pixel values.

From Figure 6, we observe that although classes such as fracture and metal are mostly classified correctly, RT-DETR+AugV1 confuses the bone anomaly class with the fracture class most often, and the soft tissue class is often undetected. Figure 7 displays a side-by-side comparison of ground-truth (left) and predicted bounding boxes (right) by the RT-DETR+AugV1 model on a sample batch of images in the validation set. We observe that RT-DETR+AugV1 is generally able to label the locations of objects accurately, but occasionally incorrectly adds extra bounding box labels, especially for the periosteal reaction class. It is mostly confident in labelling text annotations, likely due to text labels being surrounded by dark pixels in most radiograph images and being easy to distinguish due to the distinctive and standardized shapes of letters. However, it often errs by predicting multiple bounding boxes of the same class in overlapping regions, such as seen in the image in row 2, column 3. This could be due to the model's lack of confidence in its prediction, and/or its erroneous determination that there are multiple instances of fractures in this region.

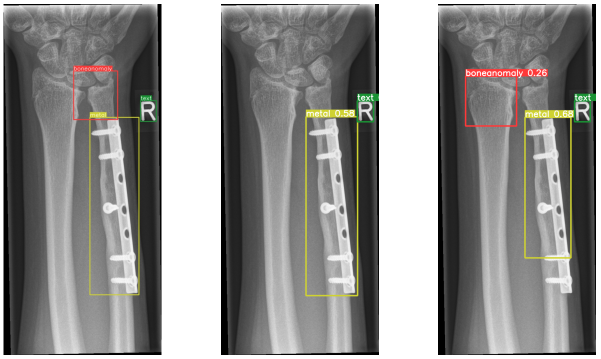


(a) batch labels

(b) batch predictions

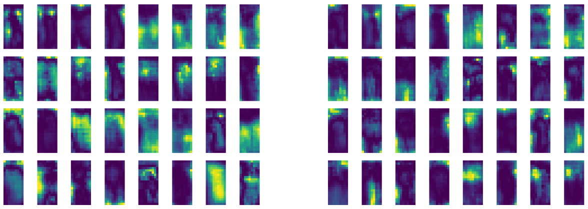
Figure 7: RT-DETR+AugV1 test batch labels vs predictions.

Figure 8 displays the bounding box ground truth and predictions by the YOLOv9 baseline and YOLOv9+AugV1 models on a particularly challenging example image. The



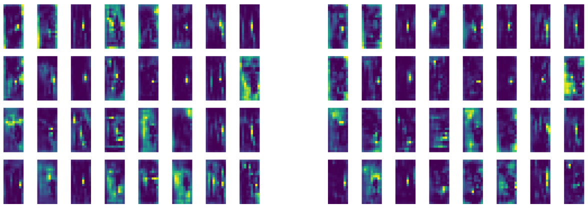
(a) ground-truth (b) baseline (c) YOLO+AugV1

Figure 8: Image with bbox annotations for ground truth, baseline YOLOv9, and YOLOv9+AugV1.



(a) baseline (b) YOLOv9+AugV1

Figure 9: Feature visualizations after the first SPPELAN block in the YOLOv9 head (stage 10) for Fig. 8.

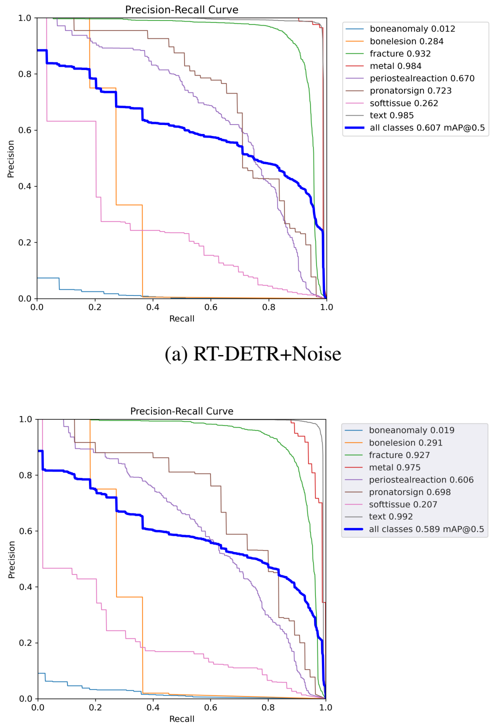


(a) baseline (b) YOLOv9+AugV1

Figure 10: Feature visualizations after the last ELAN block before the detection head (stage 37) for Fig. 8.

baseline completely fails to detect the bone anomaly instance, while YOLO+AugV1 incorrectly localizes the anomaly in the left bone instead of the right bone and also fails to capture the entirety of the metal in the image. We additionally use the YOLOv9 starter code for detection [5, 28, 4] to produce feature visualizations at various stages of the YOLOv9 models. Figure 9 shows the feature visualizations of the two YOLOv9 models after the first SPPELAN block in the head, and Figure 10 visualizes the features in the stage immediately before the detection head. We observe that in earlier stages and layers of the YOLOv9 models, the features are more generalized and span larger areas of the image, while later stages tend to extract more fine-grained and localized features, as indicated by the more pixellated bright regions of images in Figure 10. We also note that in Figure 9, the YOLOv9+AugV1 model tends to extract features in the upper left region of the image, potentially explaining the erroneous localization of the boneanomaly instance; contrastingly, the features extracted in Figure 10 tend to be similar for both models.

5.2.3 Models with Data Noise



(a) RT-DETR+Noise

(b) YOLOv9+Noise

Figure 11: PR curves of RT-DETR+Noise and YOLOv9+Noise.

Figure 11 displays the PR curves for the two models with added data noise. The two noise models underperform the baseline on bone anomaly, but significantly outperform the baseline on bone lesion, suggesting that specific classes benefit more from added bounding box noise.

Although YOLOv9+Noise does underperform most other models, it notably outperforms the YOLOv9 baseline on mAP@50, mAP@50-95, and precision; similarly, the RT-DETR+Noise models outperform RT-DETR on mAP@50, mAP@50-95, and recall. Both underperform the non-noised models in precision, signifying that adding noisy labels actually increases the model's ability to predict all instances in an image, while sacrificing the accuracy of the predicted bounding boxes themselves. Again, one hypothesis is that adding noisy/erroneous labels has a regularization effect and reduces the degree of overfitting on the train data, and in turn, the IoU of predicted and ground-truth bounding box labels. However, since YOLOv9+AugV1 and RT-DETR+AugV1 generally outperform the noise models, this suggests that noisy labels combined with augmentation methods may be preferable to noisy label perturbations alone. Overall, the YOLOv9 and RT-DETR models appear to be fairly robust to erroneous or noisy labels.

6. Conclusion/Future Work

We finetune the RT-DETR [35] and YOLOv9 [28] object detection models on the GRAZPEDWRI-DX dataset [18] of pediatric bone fractures with annotated bounding box and class labels. We additionally implement data augmentation and noising techniques to offset class imbalance and analyze model robustness to erroneous labels. Our best model, RT-DETR+AugV1, achieves a mAP@50 of 0.624 and mAP@50-95 of 0.413 on the test set. We find that despite the inclusion of a small number of erroneous/noisy samples due to random rotation, RT-DETR+AugV1 achieves the highest mAP@50, mAP@50-95, and recall scores on the test set, although it achieves lower precision scores than other models. This is potentially attributed to the "regularization" effect produced by the addition of noisy labels, which alleviates overfitting on the train set. Additionally, when further noising a small number of labels through perturbation, we still observe comparable model performance, suggesting that the models are generally robust to mislabelled data.

Further work would center around the following areas:

1) Determining the upper bound for the threshold of p_n, the probability that an image's bounding box coordinates will be noisily perturbed, without observing significant decreases in model performance.

2) Training and evaluating models on at least 10 different random seeds to produce results that are robust to randomness, and training for more epochs. Due to compute constraints, we were unable to do so for this project.

3) Exploring the effects of other forms of data augmentation, such as random flip, translations, and altering image resolution, on model performance.

4) Analyzing model performance when performing both noise perturbations and data augmentation on the training data.

7. Contributions and Acknowledgements

As I am a one-person group, I finetuned the YOLOv9 and RT-DETR models on the GRAZPEDWRI-DX dataset, implemented the data augmentation and noise methods for this project, ran all experiments on AWS, and individually wrote this report.

This project is solely completed for CS231N with valued mentorship from Tiange Xiang - a huge thank you for all of your assistance throughout the project.

1. The project builds on the starter code provided by Chien et al. in https://github.com/RuiyangJu/YOLOv9-Fracture-Detection [5] by adding 1) data augmentation, 2) additional noising, and 3) code to utilize visualization methods provided by the YOLOv9 repository and Ultralytics [14]. Chien et al.'s repository utilizes code from Wang et al.'s YOLOv9 implementation at https://github.com/WongKinYiu/yolov9 [28, 3].
2. We use the Ultralytics library provided at https://github.com/ultralytics/ultralytics [14] to write code for RT-DETR model finetuning and evaluation.
3. The above repositories utilize the PyTorch framework for model training and evaluation from https://github.com/pytorch/pytorch [2].

References

[1] W. Abbas, S. M. Adnan, M. A. Javid, F. Majeed, T. Ahsan, S. S. Hassan, et al. Lower leg bone fracture detection and classification using faster rcnn for x-rays images. In *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pages 1–6. IEEE, 2020. 2

[2] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. 8

[3] H.-S. Chang, C.-Y. Wang, R. R. Wang, G. Chou, and H.-Y. M. Liao. YOLOR-based multi-task learning. *arXiv preprint arXiv:2309.16921*, 2023. 3, 8

[4] C.-T. Chien, R.-Y. Ju, K.-Y. Chou, and J.-S. Chiang. Yolov9 for fracture detection in pediatric wrist trauma x-ray images, 2024. 2, 3, 4, 5, 7

[5] C.-T. Chien, R.-Y. Ju, K.-Y. Chou, and J.-S. Chiang. Yolov9-fracture-detection. https://github.com/RuiyangJu/YOLOv9-Fracture-Detection, 2024. 3, 5, 7, 8

[6] C.-T. Chien, R.-Y. Ju, K.-Y. Chou, E. Xieerke, and J.-S. Chiang. Yolov8-am: Yolov8 with attention mechanisms for pediatric wrist fracture detection, 2024. 1, 2, 4

[7] C. Cooper, E. M. Dennison, H. G. Leufkens, N. Bishop, and T. P. van Staa. Epidemiology of childhood fractures in

britain: a study using the general practice research database. *Journal of bone and mineral research*, 19(12):1976–1981, 2004. 1

[8] M. Cossio. Augmenting medical imaging: A comprehensive catalogue of 65 techniques for enhanced data analysis, 2023. 1, 3

[9] R. Dibo, A. Galichin, P. Astashev, D. V. Dylov, and O. Y. Rogov. Deeploc: Deep learning-based bone pathology localization and classification in wrist x-ray images, 2023. 2

[10] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 2

[11] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 2

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 2

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 2

[14] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO, Jan. 2023. 3, 5, 8

[15] R. Lindsey, A. Daluiski, S. Chopra, A. Lachapelle, M. Mozer, S. Sicular, D. Hanel, M. Gardner, A. Gupta, R. Hotchkiss, et al. Deep neural network improves fracture detection by clinicians. *Proceedings of the National Academy of Sciences*, 115(45):11591–11596, 2018. 1

[16] Y. Liu, Z. Shao, and N. Hoffmann. Global attention mechanism: Retain information to enhance channel-spatial interactions. *CoRR*, abs/2112.05561, 2021. 2

[17] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021. 2

[18] E. Nagy, M. Janisch, F. Hržić, E. Sorantin, and S. Tschauner. A pediatric wrist trauma x-ray dataset (grazpedwri-dx) for machine learning. *Scientific data*, 9(1):222, 2022. 1, 4, 8

[19] P. Rajpurkar, J. Irvin, A. Bagul, D. Ding, T. Duan, H. Mehta, B. Yang, K. Zhu, D. Laird, R. L. Ball, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng. Mura: Large dataset for abnormality detection in musculoskeletal radiographs, 2018. 2

[20] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 2

[21] D. Reis, J. Kupec, J. Hong, and A. Daoudi. Real-time flying object detection with yolov8, 2024. 2

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[24] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 2

[25] S. S. Tabarestani, A. Aghagolzadeh, and M. Ezoji. Bone fracture detection and localization on mura database using faster-rcnn. In *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–6. IEEE, 2021. 2

[26] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5552–5560, 2018. 2

[27] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. 2

[28] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024. 1, 3, 5, 7, 8

[29] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. *CoRR*, abs/1910.03151, 2019. 2

[30] S. Woo, J. Park, J. Lee, and I. S. Kweon. CBAM: convolutional block attention module. *CoRR*, abs/1807.06521, 2018. 2

[31] C. Xue, Q. Dou, X. Shi, H. Chen, and P.-A. Heng. Robust learning at noisy labeled medical images: Applied to skin lesion classification. In *2019 IEEE 16th International symposium on biomedical imaging (ISBI 2019)*, pages 1280–1283. IEEE, 2019. 2

[32] C. Xue, L. Yu, P. Chen, Q. Dou, and P.-A. Heng. Robust medical image classification from noisy labeled data with global and local representation guided co-training. *IEEE transactions on medical imaging*, 41(6):1371–1382, 2022. 1, 2, 4

[33] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 2

[34] Q. Zhang and Y. Yang. Sa-net: Shuffle attention for deep convolutional neural networks. *CoRR*, abs/2102.00240, 2021. 2

[35] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen. Detrs beat yolos on real-time object detection, 2024. 1, 3, 5, 8