Bone Fracture Analysis in X-Ray Images Using Convolutional Neural Networks and Polygon Detection

Théo Desoil
000583396
Université Libre de Bruxelles (ULB)
Belgium
theo.desoil@ulb.be

***Abstract*—Bone fracture detection is a critical task in medical imaging, requiring precise analysis of radiographs to support diagnostic decisions. In this work, we propose and evaluate multiple deep learning pipelines for the automated classification and polygon-based localization of bone fractures in X-ray images.**

We explore three strategies: (1) separate models for classification and detection, (2) a unified multitask CNN for joint prediction, and (3) pre-trained YOLOv8 models adapted to our problem. Custom convolutional architectures were designed with careful attention to spatial resolution, loss balancing, and training stability.

Experiments were conducted on a public dataset with YOLO-style polygon annotations. Our best classification-only model achieved 85.54% test accuracy, while the multitask model reached 59.04% accuracy and an average point distance of 0.14. Detection-only models demonstrated consistent performance across runs, achieving an average point distance of 0.127. YOLOv8n and YOLOv8s yielded surprisingly lower results, likely due to annotation noise, data scarcity, and class imbalance.

Despite these challenges, our custom models proved effective and efficient, with inference times compatible with real-time applications. This study highlights the importance of architecture adaptation, training consistency, and dataset quality in developing robust medical AI systems.

***Index Terms*—Bone fracture detection, medical imaging, convolutional neural networks, multitask learning, polygon localization, YOLOv8, data imbalance, label noise.**

I. Introduction

Bone fractures are a common medical condition affecting millions of individuals worldwide every year. Accurate and timely diagnosis is crucial for effective treatment and recovery. Traditionally, the identification and classification of fractures in radiographic images rely heavily on the expertise of radiologists, which can be time-consuming and prone to inter-observer variability.

In recent years, deep learning techniques have demonstrated promising results in the field of medical imaging, particularly for automating image classification and object detection tasks. In this project, we explore several deep learning approaches to automate the classification and localization of bone fractures in X-ray images.

We designed custom convolutional neural networks (CNNs) and evaluated different strategies:

- Separate models for fracture classification and polygon-based localization.
- A multitask model jointly predicting fracture type and localization coordinates.
- State-of-the-art object detection models based on YOLOv8 for comparison.

All models were trained and tested on a publicly available dataset annotated with YOLO-format polygons. Our objective is to develop and compare deep learning frameworks capable of assisting radiologists by improving the speed and accuracy of fracture diagnosis, while analyzing the trade-offs between specialized and generic detection architectures.

II. Methodology

A. Dataset

The dataset used in this project is the "Bone Fracture Detection" dataset, available on Kaggle [1]. It consists of upper limb X-ray images annotated with both fracture class labels and polygon coordinates delineating the fracture regions. The dataset is split into training, validation, and test sets, each containing images and corresponding YOLO-format label files.

Each annotation file includes a class index (ranging from 0 to 6) and a set of normalized coordinates describing a polygon around the fractured area. This dual annotation structure allows for both image-level classification and spatial localization tasks.

B. Preprocessing and Data Analysis

All input images were resized to 224×224 pixels before being passed to the neural networks. Data preprocessing strategies differed depending on the task:

- For classification-only models, several data augmentation techniques were applied during training, including random horizontal flipping, rotation, translation, perspective distortion, and color jitter.
- For detection and multitask models, only resizing and normalization were performed. No data augmentation was applied to preserve the consistency between the images and their associated polygon labels.

Normalization was applied to all input images by scaling pixel values to the range $[-1, 1]$, using the following standardization:

- `mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]`.

An initial analysis of the dataset distribution was also conducted to verify the balance of fracture classes across the splits.

C. Model Architectures

Three types of deep learning models were designed and evaluated in this study.

1) General CNN Principle: Figure 1 illustrates the general structure of a Convolutional Neural Network (CNN). An input image is processed through a sequence of convolutional and pooling layers to extract spatial features, followed by fully connected layers for classification.

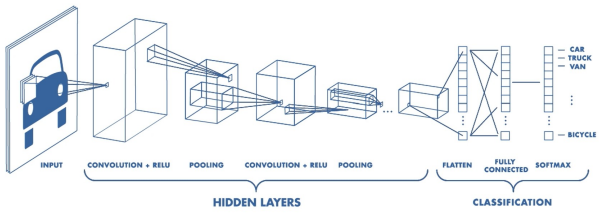

Fig. 1. General CNN architecture: alternating convolution and pooling layers extract spatial features, followed by fully connected layers for classification. Adapted from [2].

2) Classification-Only Model: The classification model consists of three convolutional layers with ReLU activations and max-pooling after each convolution, followed by two fully connected layers. A dropout layer with a rate of 0.5 was inserted before the final output layer to improve generalization. The model predicts one of the seven fracture classes.

3) Detection-Only Model: The detection model predicts eight floating-point values corresponding to four (x, y) points that delineate the fracture area. It is composed of four convolutional layers with max-pooling, followed by two fully connected layers. The model is trained using the Smooth L1 Loss (Huber Loss) to regress the polygon coordinates.

4) Joint Classification and Detection Model: To address both classification and localization tasks simultaneously, a multitask CNN named **BoneFractureMultiTaskDeepNet** was developed. The backbone consists of 10 convolutional layers, with max-pooling applied after every two layers to preserve spatial dimensions and avoid the collapse of feature maps. The shared convolutional encoder is followed by two separate branches:

- A classification head outputting the fracture class.
- A detection head outputting the polygon coordinates.

Dropout with a rate of 0.6 was applied before the heads to improve regularization.

The total loss $\mathcal{L}_{\text{total}}$ minimized during training is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{bbox}} \tag{1}$$

where $\mathcal{L}_{\text{class}}$ is the cross-entropy loss for classification, and $\mathcal{L}_{\text{bbox}}$ is the Smooth L1 loss for polygon regression.

D. Training Strategy

All models were trained using the Adam optimizer with an initial learning rate of 0.001 for classification and detection-only models, and 0.0003 for the multitask model. Early stopping was applied based on validation loss, with a patience of 20 epochs for classification and detection tasks, and 25 epochs for the multitask training.

For detection tasks and multitask, a `ReduceLROnPlateau` scheduler was used to dynamically adjust the learning rate based on validation loss stagnation.

Training was performed on an NVIDIA GPU, with a batch size of 32 images and standard TensorBoard monitoring.

E. YOLOv8 Training

We additionally explored the performance of off-the-shelf object detection models using the YOLOv8 architecture. Two variants were tested:

- **YOLOv8n (nano)**: trained for 100 epochs with a 640×640 input size.
- **YOLOv8s (small)**: trained for 150 epochs with an increased 800×800 input size.

Both models were fine-tuned on the fracture dataset using the Adam optimizer and early stopping based on validation performance. The YOLO models were initialized with pre-trained weights from the COCO dataset.

III. Results

A. Classification-Only

1) Overview of Training Strategy: Multiple models were trained using a continuous training pipeline implemented in `train_loop.py`. Each training session ran with early stopping (20 epochs patience), and both logs (for TensorBoard) and model weights were saved. The goal was to explore the variability of model performances and select the best model using the validation accuracy as a metric.

2) Performance Comparison of All Training Sessions: To analyze the global behavior across runs, we plotted all training and validation curves. Figure 2 shows that while some models converged early, others suffered from overfitting or stagnation.

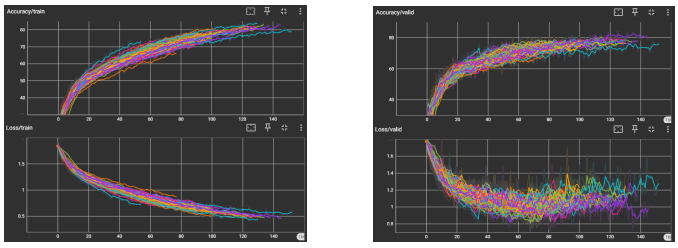

Fig. 2. Training and validation curves across all training sessions.

The model achieving the highest validation accuracy was selected and evaluated on the test set. This model reached a test accuracy of **85.54%**.

3) Best Model Evaluation: The training dynamics of the best model are shown in Figure 3. The curves exhibit stable convergence and consistent validation performance.

We also computed the confusion matrix to analyze class-specific predictions (Figure 4). The results show reliable

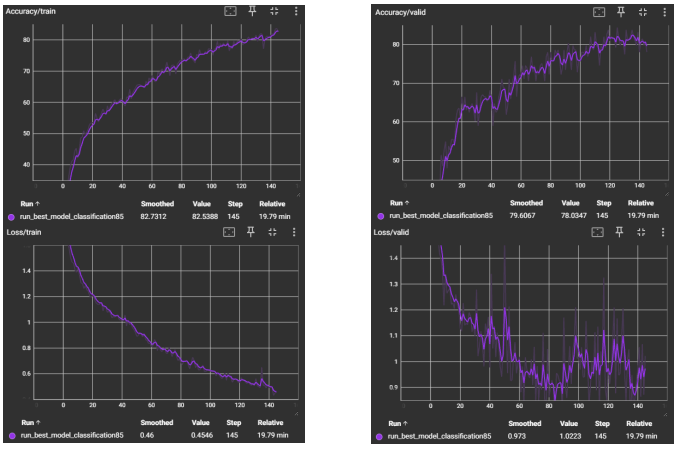

Fig. 3. Training and validation curves of the selected best model.

classification for most classes, although a few confusion errors remain between wrist and fingers categories.

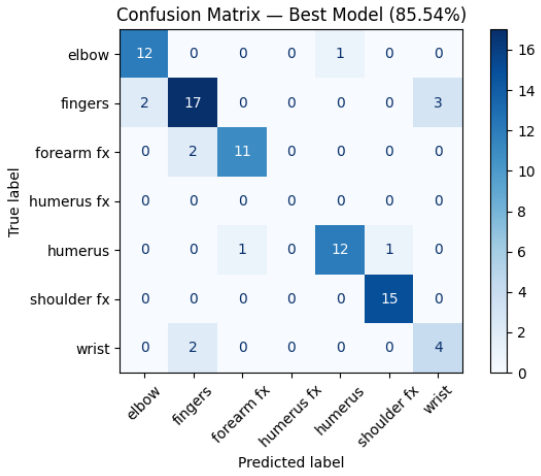

Fig. 4. Confusion matrix of the best model (test accuracy: 85.54%).

4) Early Stopping Justification: To justify the use of early stopping, we trained one model without any stopping mechanism for 1000 epochs. As shown in Figure 5, the validation performance plateaued early and fluctuated despite continuous training, indicating no further learning. This highlights the importance of early stopping to avoid overfitting and computational waste.

B. Detection-Only Results

The detection-only task focused on predicting the polygon coordinates surrounding fractures in X-ray images. A systematic experimental protocol was applied to optimize the model architecture, loss functions, training strategies, and preprocessing techniques.

1) Architecture Selection: We initially compared architectures with 3, 4, and 5 convolutional layers using the MSE loss function. Based on preliminary experiments, the architecture

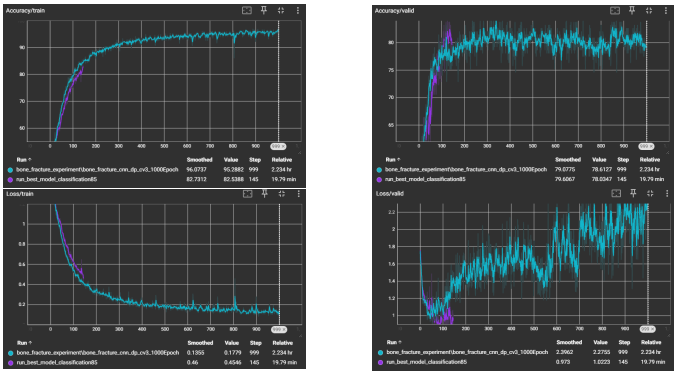

Fig. 5. Long training session without early stopping (1000 epochs).

with 4 convolutional layers was selected, as it provided a good balance between complexity and performance.

2) Criterion Selection: Next, we evaluated different loss functions:

- Mean Squared Error (MSE)
- Smooth L1 Loss (Huber Loss)
- L1 Loss

As shown in Figure 6, Smooth L1 Loss provided significantly better convergence compared to MSE and L1 Loss, with lower validation loss and more stable training. Thus, Smooth L1 Loss was selected for the following experiments.

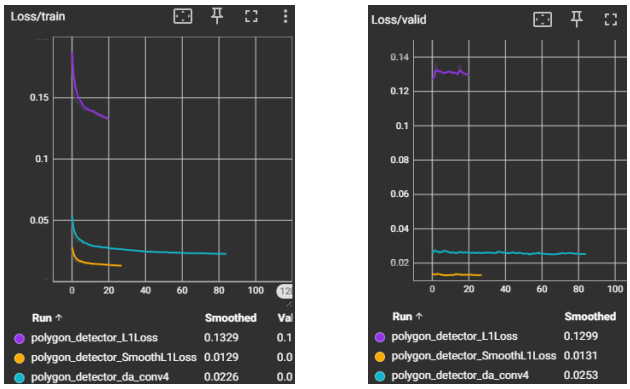

Fig. 6. Training and validation loss comparison for different loss functions.

3) Scheduler Selection: Several learning rate schedulers were tested:

- StepLR
- CosineAnnealingLR
- ReduceLROnPlateau

Figure 7 shows that all schedulers led to comparable final validation losses. However, ReduceLROnPlateau was selected because it adapts dynamically to the model's performance: it reduces the learning rate only when the validation loss stagnates, making it more responsive to the true training dynamics. Compared to fixed schedulers like StepLR or CosineAnnealingLR, ReduceLROnPlateau avoids unnecessary learning rate drops, promoting smoother convergence. Nonetheless, it can react more slowly if not properly tuned. In our experiments, it

provided slightly better overall stability and early convergence, and was therefore retained for the final model.

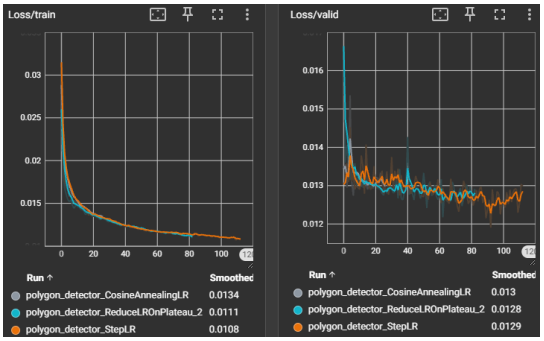

Fig. 7. Training and validation loss comparison for different loss functions learning rate schedulers.

4) Preprocessing and Transformation Tests: Different image resizing and preprocessing strategies were explored:

- Image resizing to 640×640 and 800×800 pixels
- Standard normalization (mean=0.5, std=0.5)
- Color jittering
- Full data augmentation

As shown in Figure 8, training without any normalization or augmentation at a resized image size of 800×800 yielded the lowest validation loss. Applying color jittering or full data augmentation systematically degraded model performance, confirming that heavy input distortions negatively affect precise coordinate regression.

Although removing normalization gave slightly better validation results, we decided to retain normalization for the final model. Normalization ensures consistency across all experiments and is a widely accepted practice in deep learning workflows. As emphasized by Lundervold and Lundervold in their comprehensive review of deep learning in medical imaging, proper normalization of input data is crucial for stabilizing training dynamics and improving the generalization ability of neural networks across varying datasets [3].

5) Final Model Selection via Loop Training: Based on the best identified settings (4 convolutions, Smooth L1 Loss, ReduceLROnPlateau, normalization), multiple models were trained in a loop. Validation curves are summarized in Figure 9.

From this batch, the best-performing model was selected:

- Model: `_2025-04-27_23-26-46.pth`
- **Mean Squared Error (MSE):** 0.025986
- **Average Point Distance:** 0.127 (normalized units)

The final detection-only model predicts fracture locations with high precision and stability. Figure 10 illustrates a example where the predicted polygon (in green) closely matches the ground-truth polygon (in red).

C. Combined Predictions Using Separate Models

After selecting the best classification-only and detection-only models from previous experiments, we combined them

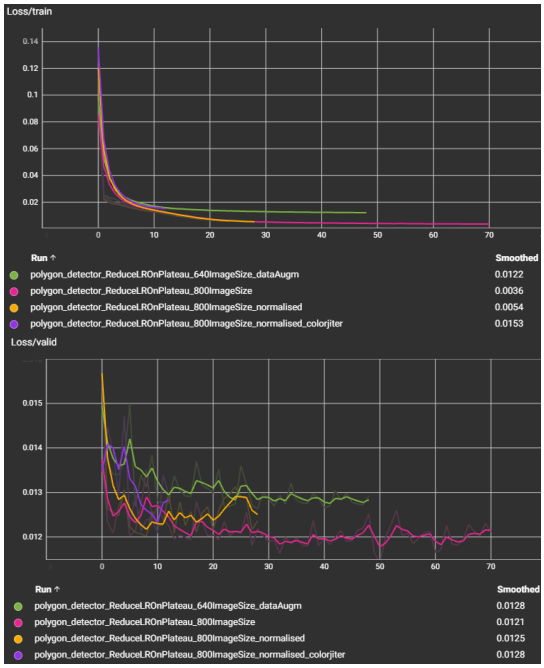

Fig. 8. Training and validation loss comparison for different preprocessing strategies.

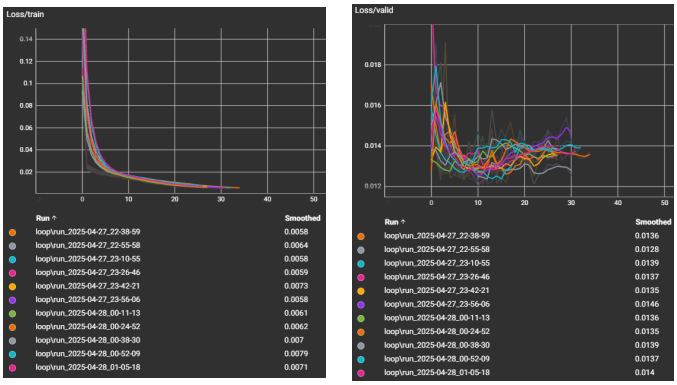

Fig. 9. Training and validation loss curves during loop training across multiple models.

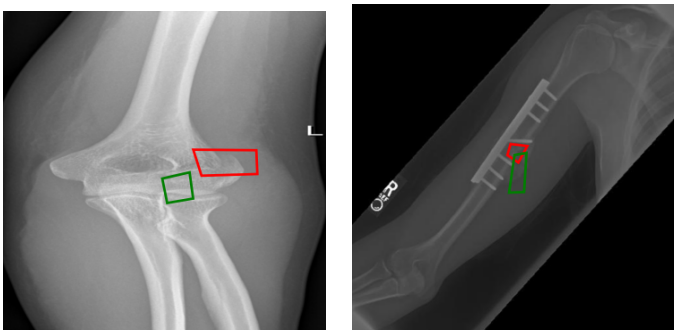

Fig. 10. Comparison of ground-truth (red polygon) and predicted (green polygon) polygons on test images using the best detection-only model.

to simultaneously predict the fracture type and its precise localization. For each test image, the classifier predicts the fracture class, while the detector predicts the polygon coordinates delineating the fracture area.

Figure 11 shows an example of the combined prediction, comparing the ground-truth label and localization with the predicted results.

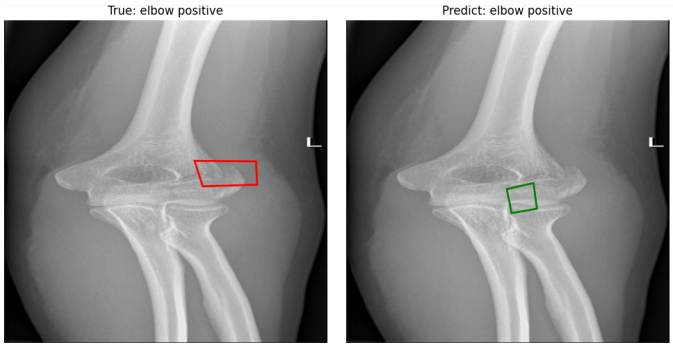

Fig. 11. Dual prediction example: ground-truth label and polygon (left) vs predicted class and polygon (right) using separate specialized models.

This combination approach allowed us to leverage the strengths of each specialized model to achieve complete fracture analysis, covering both classification and localization tasks.

D. Joint Classification and Detection Results

1) Experimental Strategy: In the multitask setting, the objective was to simultaneously predict the fracture class and the polygon localization coordinates. A systematic exploration of model depth, optimization strategies, and regularization was conducted.

a) Architecture Search.: We initially experimented with 4, 5, 6, and 7 convolutional layers, each followed by max-pooling. The results showed that increasing the depth to 5, 6, or 7 layers progressively improved the classification accuracy but had only moderate impact on localization. An attempt to use 8 convolutional layers failed due to excessive spatial downsampling from repeated pooling operations, resulting in 1×1 feature maps and loss of spatial information.

b) Architecture Search.: We initially experimented with 4, 5, 6, and 7 convolutional layers, each followed by max-pooling. The results showed that increasing the depth to 5, 6, or 7 layers progressively improved the classification accuracy but had only moderate impact on localization. An attempt to use 8 convolutional layers failed due to excessive spatial downsampling from repeated pooling operations, resulting in 1×1 feature maps and loss of spatial information.

To overcome this limitation, we designed a deeper architecture inspired by the VGG configurations explored by Simonyan and Zisserman [4]. These configurations rely on stacks of small 3×3 convolutional filters with periodic max-pooling, allowing deeper models while preserving spatial resolution. Following this principle, we developed a network with 10 convolutional layers and applied max-pooling only every two layers. This strategy preserved a 7×7 spatial resolution at the encoder's output, enabling the model to retain meaningful spatial features for localization.

c) Normalization and Learning Rate.: Early experiments without normalization showed that the model struggled to converge properly. Thus, images were normalized between $[-1, 1]$, improving stability. In parallel, the initial learning rate was reduced from 0.001 to 0.0003 to better match the deeper network.

d) Loss Computation.: It is important to note that the total loss used for optimization was the sum of:

- The cross-entropy loss for classification
- The Smooth L1 loss for polygon coordinate regression

This has important implications: improvements in one task (e.g., classification) could be masked by stagnation in the other, potentially triggering early stopping based on the combined loss plateau.

e) Optimization Strategy.: After stabilization with normalization and learning rate tuning, we introduced a learning rate scheduler (ReduceLROnPlateau) with a patience of 10 epochs and a reduction factor of 0.8. We also experimented with higher dropout rates (0.6 and 0.7) to combat overfitting.

f) Absence of Data Augmentation.: No data augmentation (e.g., flips, rotations) was applied in the multitask setting. This decision was made to maintain strict consistency between the image and polygon annotations, but may partially explain why classification accuracy remained lower than in the classification-only setting where augmentations were used.

2) Model Selection and Results: Several multitask models were trained and evaluated. Training and validation accuracy and loss curves are shown in Figures 12 and 13.

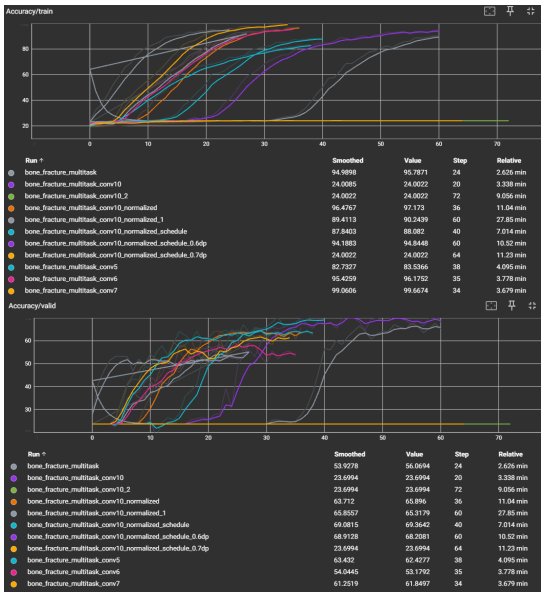

Fig. 12. Training and validation accuracy for different multitask models.

The best-performing model (BoneFractureMultiTaskDeepNet with 10 convolutional layers, normalization, scheduler,

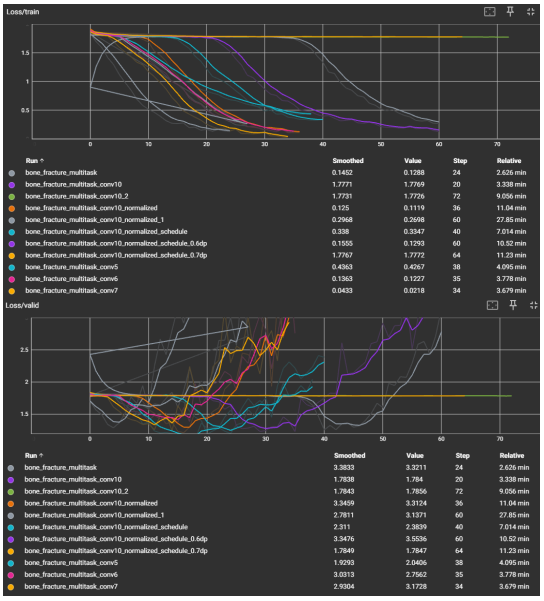

Fig. 13. Training and validation loss for different multitask models.

and a dropout rate of 0.6) achieved the following results on the **test set**:

- **Classification Accuracy:** 59.04%
- **Mean Squared Error (MSE):** 0.030441
- **Average Point Distance (normalized):** 0.140001

A qualitative prediction example is shown in Figure 14, demonstrating the simultaneous prediction of fracture type and localization polygon.

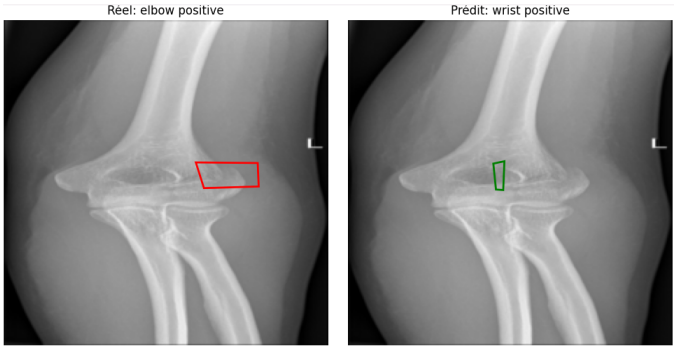

Fig. 14. Joint prediction example: ground-truth label and polygon (left) vs predicted class and polygon (right) using the multitask model.

This multitask model enabled joint prediction of fracture class and localization within a single architecture. However, training such a deep multitask network proved more complex and sensitive than training two specialized models separately. The model required more careful tuning of hyperparameters, longer training times, and exhibited greater sensitivity to the learning rate, dropout, and scheduler settings. While inference with a single model is theoretically faster at deployment, achieving good performance required significantly more effort compared to training separate classification and detection models.

E. YOLOv8 Training and Results

1) YOLOv8n: The lightweight YOLOv8n model ("nano" version) was trained for 100 epochs with a 640×640 input size. On the test set, YOLOv8n achieved:

- **mAP0.5:** 19.9%
- **mAP0.5:0.95:** 6.81%

Detection performance is insufficient for clinical relevance. Performance is particularly poor on subtle fracture types such as "wrist positive" and "fingers positive".

2) YOLOv8s: Given the limited performance of YOLOv8n, we transitioned to YOLOv8s ("small" version), a larger model with approximately 11 million parameters (vs 3 million for the model n). The training settings were adapted:

- Increased image size: 800×800 pixels
- Number of epochs: 150

After training, YOLOv8s achieved the following metrics on the test set:

- **mAP0.5:** 13.5%
- **mAP0.5:0.95:** 5.31%

Contrary to expectations, YOLOv8s did not significantly outperform YOLOv8n despite its increased complexity and input size. Detection performance remained limited, particularly for fine-grained fracture detection.

TABLE I
COMPARISON OF YOLOV8N AND YOLOV8S ON THE TEST SET

Model	mAP0.5	mAP0.5:0.95	Inference Time
YOLOv8n	19.9%	6.81%	0.9 ms
YOLOv8s	13.5%	5.31%	8.8 ms

IV. DISCUSSION

The experimental results show that our custom CNN model achieved a test accuracy of 85.54%, which is competitive given the limited size and variability of the dataset. This indicates that the model successfully learns discriminative features for bone fracture classification, even across multiple anatomical regions.

However, a deeper analysis of the dataset revealed several limitations that likely impacted model performance.

A. Class Distribution Imbalance

We discovered that class 3 ("humerus fracture") was entirely absent from the training, validation, and test sets. This explains its absence in the confusion matrix and indicates a structural imbalance issue in the dataset generation or split procedure.

In addition to this missing class, we found that a significant portion of the dataset was entirely unused in training due to missing or empty annotation files. Out of the full dataset:

- **Train set:** 3631 images total, only 1804 labeled (1827 images discarded)
- **Validation set:** 348 images total, only 173 labeled
- **Test set:** 169 images total, only 83 labeled

TABLE II
CLASS DISTRIBUTION ACROSS DATASET SPLITS (CLASS 3 MISSING).

Class	Train (#)	Validation (#)	Test (#)	Total (#)
0	306 (16.96%)	28 (16.18%)	13 (15.66%)	347
1	433 (24.00%)	41 (23.70%)	22 (26.51%)	496
2	283 (15.69%)	37 (21.39%)	13 (15.66%)	333
3	0 (0.00%)	0 (0.00%)	0 (0.00%)	0
4	299 (16.57%)	31 (17.92%)	14 (16.87%)	344
5	315 (17.46%)	19 (10.98%)	15 (18.07%)	349
6	168 (9.31%)	17 (9.83%)	6 (7.23%)	191
Total	**1804**	**173**	**83**	**2060**

This means that approximately **50% of the available data was ignored** during training, validation, and testing due to the absence of annotations.

Combined with the missing class and class imbalance, this significantly reduced the effective size and diversity of training samples, making the learning task more difficult and introducing biases into the model. These limitations must be taken into account when interpreting performance metrics and generalization capability.

B. Labeling Errors

Beyond class imbalance, we also identified significant labeling inconsistencies in the dataset. Some images contained incorrect class annotations, making it difficult for the model to learn reliable decision boundaries.

Figure 15 illustrates such a case: the ground-truth label is ”fingers positive”, while the fracture clearly belongs to the ”wrist positive” category.

In addition to the misclassified label, the polygon (shown in red) fails to capture the actual location of the fracture. It is incorrectly placed far from the wrist region, where a metallic surgical pin is clearly visible. This further confirms that certain annotations are not only semantically incorrect but also spatially misaligned.

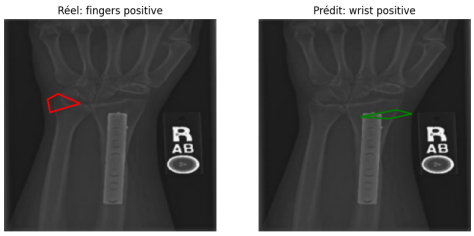

Fig. 15. Example of a labeling error: wrong class and inaccurate polygon localization.

Such noisy annotations introduce confusion during training and penalize both classification and detection performance. They are especially detrimental to larger models like YOLOv8s or multitask networks, which are more sensitive to label inconsistencies due to their higher capacity and reliance on clean supervision.

C. Training Observations

The confusion matrix revealed that some classes, such as ”humerus fracture” (expected class 3), were not predicted at all due to missing training data. Meanwhile, classes like ”fingers positive” and ”wrist positive” were sometimes confused—possibly due to visual proximity or labeling inconsistencies. The model still managed to generalize relatively well across other classes.

Early stopping also proved effective: training beyond the convergence point did not yield better performance, as demonstrated by our 1000-epoch test. This validated the need for regularization and training efficiency strategies.

Despite these issues, the model achieves promising results and inference speed suitable for real-time applications. However, further dataset cleaning and rebalancing would likely improve classification accuracy and robustness.

D. Why YOLOv8s Underperformed Despite Its Capacity

Surprisingly, YOLOv8s underperformed compared to the smaller YOLOv8n, despite having significantly more parameters and theoretical expressive power. This unexpected result can be attributed to several factors related to both the dataset and the training configuration:

- **Limited dataset size.** The full dataset consists of only **686 X-ray images**, which is relatively small for training deep convolutional architectures. Larger models like YOLOv8s typically require more data to generalize effectively. On limited datasets, they tend to *overfit*—capturing noise or spurious patterns that do not transfer well to new examples.
- **Labeling issues and annotation noise.** Multiple sources of annotation error were identified during our exploration:
  - The *“Humerus fracture”* class (ID 3) is entirely absent from all three dataset splits (train/valid/test).
  - Some labels are incorrect, such as a “fingers” fracture label applied to what is clearly a wrist image (see Figure 15).
  - A subset of images contains empty or invalid label files.

  These noisy labels confuse deeper networks more than shallow ones, as they have more parameters and capacity to learn undesired correlations.

In summary, YOLOv8s likely failed to outperform YOLOv8n due to a mismatch between its model complexity and the dataset’s size and quality. This highlights a key challenge in medical imaging: *bigger models are not always better*, especially when training data is limited or noisy.

E. Impact of Data Augmentation on Model Variability

Another key insight emerged from our training loops: the use of data augmentation introduces a significant source of randomness into the learning process.

In the classification task, where several augmentation techniques were applied (random flips, rotations, translations, etc.), we observed noticeable variability in performance across training runs. As shown in Figure 2, models differed in:

- Final accuracy scores.
- Convergence speed and trajectory.

- Stability of validation performance.

In contrast, during the detection-only experiments conducted without any data augmentation the training loop produced highly consistent models, both in terms of validation loss and coordinate accuracy. This consistency suggests that, in the absence of randomness from data augmentation, the training process is more deterministic and repeatable.

While data augmentation is known to improve generalization, especially with small datasets, it also introduces nondeterminism that can affect reproducibility and evaluation. This observation reinforces the need to run multiple experiments and report averaged metrics when working with randomized training pipelines.

V. Conclusion

In this study, we explored the application of deep learning for the automated classification and localization of bone fractures on X-ray images. We proposed and evaluated three main approaches: a classification-only model, a detection-only model, and a multitask architecture combining both objectives. Additionally, we benchmarked our solutions against state-of-the-art object detection models from the YOLOv8 family.

Our custom classification model achieved a strong test accuracy of 85.54%, while the detection model reached a polygon localization error below 0.13 on average. The multitask architecture though harder to train and slightly less accurate in classification successfully performed both tasks jointly, demonstrating the feasibility of end-to-end fracture analysis using a unified model.

We also uncovered critical limitations in the dataset, including class imbalance, missing annotations, and labeling errors, all of which impaired model performance. Interestingly, YOLOv8s, despite its increased complexity, underperformed compared to its smaller counterpart YOLOv8n highlighting the importance of matching model capacity to data quality and volume.

Finally, we observed that data augmentation, while beneficial for generalization, introduced significant variability in training dynamics and final accuracy, suggesting a need for controlled or synchronized augmentation strategies especially in multitask setups.

Future work will aim to:

- Improve annotation consistency and correct labeling errors.
- Rebalance class distributions and exploit currently unused images.
- Investigate more advanced architectures.
- Incorporate synchronized geometric augmentations to enhance detection training.

Despite the challenges, our results demonstrate that well-designed deep learning pipelines can provide accurate and real-time assistance for fracture analysis, paving the way for clinical decision support tools in radiology.

Acknowledgments

We thank the Roboflow and Kaggle communities for providing the original X-ray dataset.

All code used in this study, including model architectures, training scripts, evaluation tools, and result visualizations, is publicly available at https://github.com/Teytey2002/Bones_Dectetion_NN.

References

[1] P. K. Darabi, "Bone Fracture Detection (Computer Vision Project)," https://www.kaggle.com/datasets/pkdarabi/bone-fracture-detection-computer-vision-project, 2023, accessed: 2025-04-19.

[2] MathWorks, "Introduction to Deep Learning: What Are Convolutional Neural Networks?" https://nl.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html, 2016, accessed: 2025-04-19.

[3] A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on MRI," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2020, accessed: 2025-04-19. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0895611120300197

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556