Computer Vision for Ankle Replacement Surgery

Benjamin M. Gitai

Princeton High School, Princeton NJ

Abstract

Roughly 1% of adults suffer from ankle pain that could benefit from surgery, but ankle replacements are rarely performed because they often fail, causing immense pain. I thus sought to develop artificial intelligence tools to predict in advance which patients are at risk for implant failure. I trained my model using Stanford's open-source LERA database of Ankle X-rays. My analyses showed that traditional edge detection methods do not work well on X-ray images. To solve this problem, I developed a computer vision object-detection algorithm to identify and segment each ankle bone. To do so, I designed an image pre-processing pipeline, including color inversion, to standardize the LERA X-ray images. I then used the CVAT annotation tool to draw detections and contours for each bone, which I used to train a detection algorithm to isolate and segment each ankle bone. These new tools outperformed existing approaches, enabling me to extract measurements of the fibula and the talar center migration angles, and the talar tilt angle. Finally, I developed a convolutional neural net to predict ankle replacement failure based on these automated measurements. These efforts highlight limitations in standard image processing methods, establish a robust approach for automated ankle X-ray measurements, and hold promise for improving patient outcomes for ankle replacement surgeries.

Introduction

Ankle replacements have become more common as better techniques have been developed (Figure 1B, 1C). Currently, about 1% of American adults suffer from ankle pain that could benefit from surgery (Figure 1A), but ankle replacements are rarely performed because they often fail. In fact, about 10% of ankle replacements actually fail within five years [1]. This poses a problem, as when implants fail, it causes the patient immense pain and requires additional surgery [2]. Many different events can cause failure, and failures are hard to predict. Therefore, it would be beneficial for surgeons to have the ability to identify which patients might benefit from surgical intervention prior to ankle failure.

To make such predictions, surgeons need information about specific measurements that will be informative about the state of the bones. Measurements of fibula and tibia bone angles are thought to be informative [2,3]. Specifically, the talar center migration angle, talar tilt angle, and distal tibial articular angle can tell the surgeon about the health of the ankle (Figure 1B, C). However, currently those measurements can only be done manually and are thus slow, such that there is no large-scale database of such measurements that could be used to train an AI prediction model.

To address this problem, I examined X-ray images that are used to assess surgical candidates. I noticed that many of the pre-op measurements were being done using hand-drawn lines and measurements; therefore, I wondered whether computer vision analysis of images would provide faster and more accurate analyses of the X-ray image pre-op measurements. I also wanted to create an algorithm to classify the specific kind of injury, as well as to predict likelihood of implant failure. To meet these goals, I broke the problem down into a series of steps that included isolating the specific bones to examine, segmenting those bones, and then analyzing the contours of those segments. This information can then inform the surgeon about the likelihood of surgical intervention success. The ultimate goal of the algorithm is to allow doctors to use this AI approach together with clinical X-ray images to minimize burdensome failures (Figure 1D).

Methodology and Results

Dataset

To begin to address the question of how to assess ankles for surgical success, I first needed to obtain a large dataset for my analyses. The Stanford Lower Extremity Radiographs (LERA) dataset [4] is a publicly available dataset that contains X-ray images of feet, knees, ankles, and hips. For my work, I only used those that contain images of feet and ankles. I also sorted the foot and ankle images into “healthy” and “unhealthy” classes based on the clinical information provided for each image in the dataset. Because I am focusing on the ankle, my dataset contained 150 usable images (Figure 2A).

Pre-processing

Before I could do any analyses of the images, I first needed to reduce noise and variance (Figure 2A, 2B). The dataset was highly variable; for example, some images were on black backgrounds, others had white backgrounds (Figure 2C). Additionally, the quality and sizes were different for each image. To combat these issues, I started by standardizing the dataset through basic pre-processing. Specifically, I wrote code that changed the images to all have white backgrounds by taking the average pixel value of an image and inverting it based on whether that value was closer to black or white, thus standardizing the background on all images. I then applied a sharpening filter from cv2 [5] to increase contrast in the image. This code can be found at [6], and the user can choose the preprocessing options, including Smoothing, Sharpening, Inverting, Resizing, Cleaning, and Contrasting. By applying these pre-processing steps, I was able to reduce variation and noise in the images, which allowed me to move on to the next steps (Figure 2C).

Bone Isolation and Labeling

The next step was to identify and assign labels to each bone in the images. With the set of pre-processed images, I created a detection algorithm. I used the program “CVAT” [7] to draw detection boxes around each bone and region I wanted to analyze, and assigned each region a specific associated label (e.g., “Tibia”) (Figure 1A). CVAT is a program which allows for the import of images, creation of detection boxes/segmentation, and then the output of those boxes/segments [7]. The boxes were then output into the YOLO.txt format, allowing us to train my model using YOLOv8 [7,8] a publicly available library from Ultralytics (see below).

Model Training

YOLOv8, or You Only Look Once, from Ultralytics is an open-source library for the creation of computer vision AI models. YOLOv8 allows you to create your own models for image detection, segmentation, classification, etc. It uses .txt files that give the location of the boxes/outlines, making it relatively small when it comes to storage. It also has an extensive pretrained model, which is useful as it allows for high quality models to be created from relatively low amounts of data. Because 150 images is on the low side for training, I decided to modify a pretrained model, in this case, the yolov8m.pt pretrained model, as there was not enough data to build a model from scratch. This is also why I choose YOLO over other computer vision libraries such as Segment Anything, which also requires more data [9]. I selected the medium-sized model, as it was the largest model that worked on the GPUs I had access to.

The labeling algorithm was made with YOLOv8 by separating the images and labels into associated folders for training, validation, and testing [8]. Using my pre-processed images, I was able to train the detection algorithm to be able to draw bounding boxes around the bones (Figure 2D). My model has a precision of 0.84562 and a recall of 0.2 (Figure 3). Given how little training data there was (only 150 images), this was a robust score, as most models require thousands of images to obtain precision scores of 0.8 or higher [10].

With my custom model, I was able to make labeling predictions for the bounding box of each bone and get the coordinates of each bounding box (Figure 2E). With those coordinates, I made a NumPy mask for the images to isolate just the bones I wanted to examine [11] (Figure 2E). I focused on ankle-associated bones, but in the future this model can be extended to apply to any other kind of bone. For my purposes, I knew that including foot bones other than the tibia, fibula, and talus would likely lead to additional and unnecessary noise, regardless of what I ended up doing with the image. By isolating only the necessary bones, I could limit the amount of additional, and potentially noisy, unnecessary data.

Segmentation

In order to calculate pre-op measurements, I needed to obtain and analyze the contours of each bone (Figure 4A). My initial idea was to use a standard edge detection method: “Canny edge detection” from OpenCV [11,12]. However, this method had serious flaws, as the pores and cavities of the bones that are visible within X-ray images cause these X-ray images to be noisy and the edge detection to fail. This made it not only extremely difficult to get a clean outline, but also meant that different parameters were needed for each different image. Therefore, I instead decided to develop a segmentation model instead of using standard edge detection. I considered using an unsupervised learning system (e.g., Segment Anything (SAM) [9]); however, since I needed the model to be as accurate to the bone as possible, I worried that some edges might get muddy using an unsupervised learning system. Therefore, I instead decided to use a supervised learning system (Figure 4B). I again used YOLO for this step, as it also has an available segmentation system that was compatible with my images. Similar to the detection algorithm, I again used CVAT [7] for labeling (Figure 4C). For pre-op measurements, I used "AP views" [13] to measure the talar center migration angle, talar tilt angle, and distal tibial articular angle (Figure 4D). 56 of the images were removed from analysis because they did not include AP views of the ankle, resulting in 94 images for the model. Although I only had 94 images to build the model, and the annotations were done by hand, the model worked relatively well (Figure 4E). Importantly, even despite some noise in overlap regions, I was able to successfully extract the mask of each bone. Separate models were created for each bone in order to analyze the bones completely separately. This was a necessary step as each bone required different measurements. Similar to the detection algorithm, I made the segmentation system modular. This means that the system can easily be modified or added to with new bones and/or measurements [6].

For image segmentation AI, the metrics typically used to evaluate a model are mean average precision scores (mAP), which describe how close the segment made by the computer is to the actual segment [10]. Mean average precision is a combination of precision and recall, making it the strongest measurement for computer vision [10]. It measures the average precision, which is the precision at each recall level, giving a single value representing the area under the precision-recall curve. To assess performance, I measured the mAP50 scores. For the tibia, the mAP50 (B), mAP50-95 (B), mAP50 (M), and mAP50-95 (M) were 0.995, 0.96581, 0.995, and 0.96815 respectively (Figure 5). For the fibula, the scores were 0.995, 0.7516, 0.995, and 0.78924, respectively (Figure 6). For the talus, the scores were 0.995, 0.62752, 0.995, and 0.61886, respectively (Figure 7). These values indicated that each of the bone segmentation models that I developed performed extremely well.

Determining Contours

Next, I needed to determine the shape of each bone in order to be able to obtain precise measurements of the relevant bones (Figure 8A). To do so, I needed to determine the contours of the bone shapes (Figure 8B). Contours were obtained using Canny edge detection of the segmented mask [5,12]. Each contour was then analyzed independently. By analyzing each bone contour separately, I was able to obtain measurements for each bone. I also processed my images to ensure that they would be analyzed properly. For example, I bordered my images to prevent contours from going all the way to the top of the image and being incorrectly analyzed as a curve rather than a shape. I also rotated images as needed [5,6].

Measuring angles

To make pre-operation measurements, I needed to determine the correct positions and angles of the bones (Figure 4D) [3]. All pre-op measurements were based on the angle between the midline of the tibia and a second line specific to each measurement (Figure 8C, 8D). One issue I encountered early on was image rotation. Some X-rays placed the foot at odd angles so that the images were not vertically straight; as a result, those maxima were altered, leading to errors in finding the bottom of the tibia (Figure 8D). To fix this problem, I first measured the angle of the midline and then if it exceeded a threshold of 10 degrees off straight, I rotated the image. I then analyzed the image after rotation [6].

For the tibia, I first measured the angle of the midline. I calculated the midline by taking the ⅙ to ½ midpoints of the tibia contour. I focused on this region as it the most straight region of the contour for all bones due to it not being near any joints. To calculate the distal tibial articular angle, I needed to find the bottom line of the tibia. The Distal Tibial Articular Angle was calculated by taking the highest local maxima in the lower half of the tibia (to rule out any local maxima at the top) that aren't directly next to each other (more than the height of the image/50 ~ 13 pixels) [3,6]. The angle between this line between maxima and the tibia midline creates the distal tibial articular angle. I did this by calculating the local maximum on the bottom of the tibia and plotting the line between this local maximum and the tibia midline. To find the distal tibial articular angle, I compared the line between maxima and the center line of the tibia [3].

The "fibula and talar center migration angle" was the most straightforward to analyze. I calculated the Talar Center Migration Angle by taking the absolute minimum point of the fibula and the minimum point of the tibia and taking the angle of the line between the two minimum points and the tibia midline [3].

The last angle I calculated was the talar tilt angle. This was calculated in a similar fashion to the distal tibial articular angle. For the talar tilt angle, I needed to calculate the top line of the talus. In order to do that, I found the two local maxima of the talus and plotted the line between them. This line was then compared to the tibia center line in order to calculate the talar tilt angle [3].

Based on my calculations, all angles measured by my automated pipeline were within 5% error of my manual measurements and were similar to the measurements reported in other studies [14][3]. All measurements also fell within the normal distribution range [3]. In the future, these measurements can be compared against manual measurements made by clinical practitioners to confirm the accuracy of my method.

Like all other parts of my system, my contour analysis is also modular. This will allow the extraction of additional features as necessary in the future. It will also provide the basis for any future analysis of other bones [6].

Classification

The final step of the process was to begin the process of using the automated measurements to generate a prediction model for classifying ankles into "Healthy" and "Unhealthy" categories based on the extracted data. To do so, I created an algorithm using TensorFlow [15]. The LERA dataset [4] only labeled images as "healthy" or "unhealthy" without further explanation or clinical information. I therefore separated images into "healthy" and "unhealthy" classes and trained the model on that classification [6]. The algorithm I created performed well at identifying obviously unhealthy bones, such as those with breaks or post-operative implants. However, I lacked sufficient training data to properly classify images with more subtle issues, suggesting that the model's training dataset was underpowered. This classification system should improve with additional training data and demonstrates promise for the future applications of my pipeline to predicting ankle surgery outcomes.

Discussion: Conclusions and Future Directions

Over time, ankle replacements have a chance of failing. Implant failures cause pain and the need for additional surgery, so automated algorithms that can help predict such failures will be helpful to surgeons and patients. The pipeline I developed was successful in automating steps to isolate bones of interest, segment those bones, and analyze the contours of those segments. My bone isolation system functions with high precision but low recall, which is usable for this functionality, allowing the user to specify which bones to analyze. This system can be expanded to any other bone or multiple bones as well. I was also able to increase the accuracy of my model by pre-processing my images through standardization of background and rotation. In addition, I was successful in segmenting each bone I wanted to analyze, and in analyzing the contours of these bone masks to be able to successfully provide necessary pre-op measurements. The system is modular, and even the way the contours are measured can easily be changed. In its current state, the system can be used to quickly and effectively give pre-op measurements for ankle replacement surgery. The biggest use of this system is likely yet to come, as I have designed it such that any bone can be analyzed with only minor tweaks and new data. With additional data and higher quality data, I will hopefully be able to expand my classification algorithm to make better predictions of surgery outcomes. The simplest use would be to categorize the type of injury (sprain, fracture, etc.) based on either the image or some combination of the image and the doctor's report.

Finally, the classification algorithm can be run again as additional data become available, ultimately leading to the ability to classify ankles into healthy vs unhealthy in an automated fashion. My system would benefit from additional clinical data that would be much more in-depth, for example, providing more information about the type of injury or surgery and other possibly relevant clinical information. With such data, I can retrain my model to improve accuracy and compare against real clinical measurements. Perhaps most interestingly, in the future one could utilize my pipeline to help generate an algorithm that can predict how likely an implant is to fail before it actually does. If I can configure the prediction algorithm to the point where I can know not only *if* the implant is going to fail but also *why*, my algorithm can prevent such failures and potentially save patients from additional pain and surgery.

Figures

Figure 1: Introduction

1A, Image of a healthy ankle. 1B, image of an ankle with arthritis in need of replacement [16]. Arrow indicates site of osteoporosis. 1C, Same ankle as 1B post op [16]. 1D, Schematic of the order of operations for the project.

Figure 2: Outline of Bone Isolation steps

2A, Cartoon of goal result. Masking to reduce noise. 2B, Schematic of steps necessary to achieve goal bone isolation. 2C, example of effect of preprocessing: left side shows image before processing, right side is after processing. 2D, an example of training data inside CVAT [7]. 2E, an example of the results from running the detection algorithm on an image with the detection boxes overlaid on the left, on the right is the same image masked to only contain the detected regions.

Figure 3: Detection Algorithm Metrics

3: Metrics of the detection model: train/box_loss, bounding box loss during training, indicates how the model is learning to predict bounding boxes. train/cls_loss, classification loss during training, how well the model classifies the objects into classes. train/dfl_loss is distribution focal loss which measures the accuracy of the location of the boxes. val/box_loss, bounding box loss during validation. val/cls_loss, classification loss during validation. val/dfl_loss, distribution focal loss during validation. metrics/precision(B) shows precision during training, measuring the proportion of correctly predicted objects out of all predicted objects. metrics/recall(b), recall, measuring the proportion of correctly detected objects out of all actual objects in the dataset. metrics/mAP50(B), mean average precision at IoU threshold 0.5 (Intersection over Union of 50%), combines precision and recall. metrics/mAP50-95(B) shows mean average precision over IoU thresholds from 0.5 to 0.95, a stricter and more comprehensive measurement of quality. All data output from YOLO [8]. All metrics that end with (B) refer to metrics of the detection box.

Figure 4: Segmentation

4A, Cartoon of goal result, which is to extract contour of each bone. 4B, schematic of planned necessary steps for image segmentation. 4C, example of training data from inside CVAT [7]. 4D, Images of ankle angle measurements [3]. 4Di, The lateral distal tibial angle (LDTA) is formed by the distal tibial articular surface and the anatomical axis of the tibia (normal values 89° ± 3°) [3]. 4Dii, The anterior distal tibial angle (ADTA) is formed by the mechanical axis of the tibia and the joint orientation line of the ankle in the sagittal plane (normal values 80° ± 3°) [3]. 4Diii, The tibial-talar angle (a) is defined by the tibial and talar articular surfaces in the ankle joint; if it measures > 10°, the joint is defined as incongruent [3]. 4E, Segmentation result, tibia input on the right, segment estimation (from model) in the center, output contour on the right.

Figure 5: Tibia Segmentation Data

5A, Metrics of the tibia segmentation model: train/box_loss, bounding box loss during training, indicates how the model is learning to predict bounding boxes. train/cls_loss, classification loss during training, how well the model classifies the objects into classes. train/dfl_loss is distribution focal loss which measures the accuracy of the location of the boxes. val/box_loss, bounding box loss during validation. val/cls_loss, classification loss during validation. val/dfl_loss, distribution focal loss during validation. metrics/precision(B) and metrics/precision(M) shows precision during training, measuring the proportion of correctly predicted objects out of all predicted objects. metrics/recall(B) and metrics/recall(M), recall, measuring the proportion of correctly detected objects out of all actual objects in the dataset. metrics/mAP50(B) and metrics/mAP50(M), mean average precision at IoU threshold 0.5 (Intersection over Union of 50%), combines precision and recall. metrics/mAP50-95(B) and metrics/mAP50-95(M) shows mean average precision over IoU thresholds from 0.5 to 0.95, a stricter and more comprehensive measurement of quality. All data output from YOLO [8]. All metrics that end with (B) refer to metrics of the detection box and all metrics that end with (M) refer to the metrics of the segmentation.

Figure 6: Talus Segmentation Data

6A, Metrics of the talus segmentation model [8]: train/box_loss, bounding box loss during training, indicates how the model is learning to predict bounding boxes. train/cls_loss, classification loss during training, how well the model classifies the objects into classes. train/dfl_loss is distribution focal loss which measures the accuracy of the location of the boxes. val/box_loss, bounding box loss during validation. val/cls_loss, classification loss during validation. val/dfl_loss, distribution focal loss during validation. metrics/precision(B) and metrics/precision(M) shows precision during training, measuring the proportion of correctly predicted objects out of all predicted objects. metrics/recall(B) and metrics/recall(M), recall, measuring the proportion of correctly detected objects out of all actual objects in the dataset. metrics/mAP50(B) and metrics/mAP50(M), mean average precision at IoU threshold 0.5 (Intersection over Union of 50%), combines precision and recall. metrics/mAP50-95(B) and metrics/mAP50-95(M) shows mean average precision over IoU thresholds from 0.5 to 0.95, a stricter and more comprehensive measurement of quality. All data output from YOLO [8]. All metrics that end with (B) refer to metrics of the detection box and all metrics that end with (M) refer to the metrics of the segmentation.

Figure 7: Fibula Segmentation Data

6A, Metrics of the fibula segmentation model [8]: train/box_loss, bounding box loss during training, indicates how the model is learning to predict bounding boxes. train/cls_loss, classification loss during training, how well the model classifies the objects into classes. train/dfl_loss is distribution focal loss which measures the accuracy of the location of the boxes. val/box_loss, bounding box loss during validation. val/cls_loss, classification loss during validation. val/dfl_loss, distribution focal loss during validation. metrics/precision(B) and metrics/precision(M) shows precision during training, measuring the proportion of correctly predicted objects out of all predicted objects. metrics/recall(B) and metrics/recall(M), recall, measuring the proportion of correctly detected objects out of all actual objects in the dataset. metrics/mAP50(B) and metrics/mAP50(M), mean average precision at IoU threshold 0.5 (Intersection over Union of 50%), combines precision and recall. metrics/mAP50-95(B) and metrics/mAP50-95(M) shows mean average precision over IoU thresholds from 0.5 to 0.95, a stricter and more comprehensive measurement of quality. All data output from YOLO [8]. All metrics that end with (B) refer to metrics of the detection box and all metrics that end with (M) refer to the metrics of the segmentation.

Figure 8: Contour Analysis

8A, Cartoon of goal result, which is to analyze each contour to get medical measurement angles of the ankle. 8B, schematic of planned necessary steps for contour analysis. 8C, example result of test image, on the left is the image given to the AI and on the right is the output, below is the output measurements. 8D, example of rotation in final result, original image given on the left, output result on the right, final values underneath [6].

References

1. Burton A, Aynardi MC, Aydogan U. Demographic Distribution of Foot and Ankle Surgeries Among Orthopaedic Surgeons and Podiatrists: A 10-Year Database Retrospective Study. Foot & ankle specialist. 2021;14. doi:10.1177/1938640020910951

2. Frank VJ, Lichte P, Gutteck N, Bouillon B, Arbab D. Comparison of the European Foot and Ankle Score (EFAS) and the Self-Reported Foot and Ankle Score (SEFAS) in patients with foot and ankle surgery. Archives of orthopaedic and trauma surgery. 2024 [cited 5 Nov 2024]. doi:10.1007/s00402-024-05585-y

3. Bonasia DE, Dettoni F, Femino JE, Phisitkul P, Germano M, Amendola A. TOTAL ANKLE REPLACEMENT: WHY, WHEN AND HOW? The Iowa Orthopaedic Journal. 2010;30: 119.

4. LERA- Lower Extremity RAdiographs. In: Center for Artificial Intelligence in Medicine & Imaging [Internet]. [cited 5 Nov 2024]. Available: https://aimi.stanford.edu/datasets/lera-lower-extremity-radiographs

5. GitHub - opencv/opencv: Open Source Computer Vision Library. In: GitHub [Internet]. [cited 5 Nov 2024]. Available: https://github.com/opencv/opencv

6. GitHub - BenGitai/Computer-Vision-for-Ankle-Replacement-Surgery: The code needed for this paper. In: GitHub [Internet]. [cited 5 Nov 2024]. Available: https://github.com/BenGitai/Computer-Vision-for-Ankle-Replacement-Surgery

7. GitHub - cvat-ai/cvat: Annotate better with CVAT, the industry-leading data engine for machine learning. Used and trusted by teams at any scale, for data of any scale. In: GitHub [Internet]. [cited 5 Nov 2024]. Available: https://github.com/cvat-ai/cvat

8. GitHub - ultralytics/ultralytics: Ultralytics YOLO11 🚀. In: GitHub [Internet]. [cited 5 Nov 2024]. Available: https://github.com/ultralytics/ultralytics

9. GitHub - facebookresearch/segment-anything: The repository provides code for running inference with the SegmentAnything Model (SAM), links for downloading the trained model checkpoints, and example notebooks that show how to use the model. In: GitHub [Internet]. [cited 5 Nov 2024]. Available: https://github.com/facebookresearch/segment-anything

10. Shah D. Mean Average Precision (mAP) Explained: Everything You Need to Know. V7; [cited 5

Nov 2024]. Available: https://www.v7labs.com/blog/mean-average-precision

11. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020;585: 357–362.

12. Canny J. A Computational Approach to Edge Detection. [cited 5 Nov 2024]. Available: http://dx.doi.org/10.1109/TPAMI.1986.4767851

13. Patel P, Russell TG. Ankle Radiographic Evaluation. StatPearls [Internet]. StatPearls Publishing; 2023.

14. Barg A, Wimmer MD, Wiewiorski M, Wirtz DC, Pagenstert GI, Valderrabano V. Total Ankle Replacement: Indications, Implant Designs, and Results. Deutsches Ärzteblatt International. 2015;112: 177.

15. TensorFlow. [cited 5 Nov 2024]. doi:10.5281/zenodo.13989084

16. Total Ankle Replacement on HSS.edu. 2022. Available: https://www.hss.edu/condition-list_ankle-replacement-arthroplasty.asp

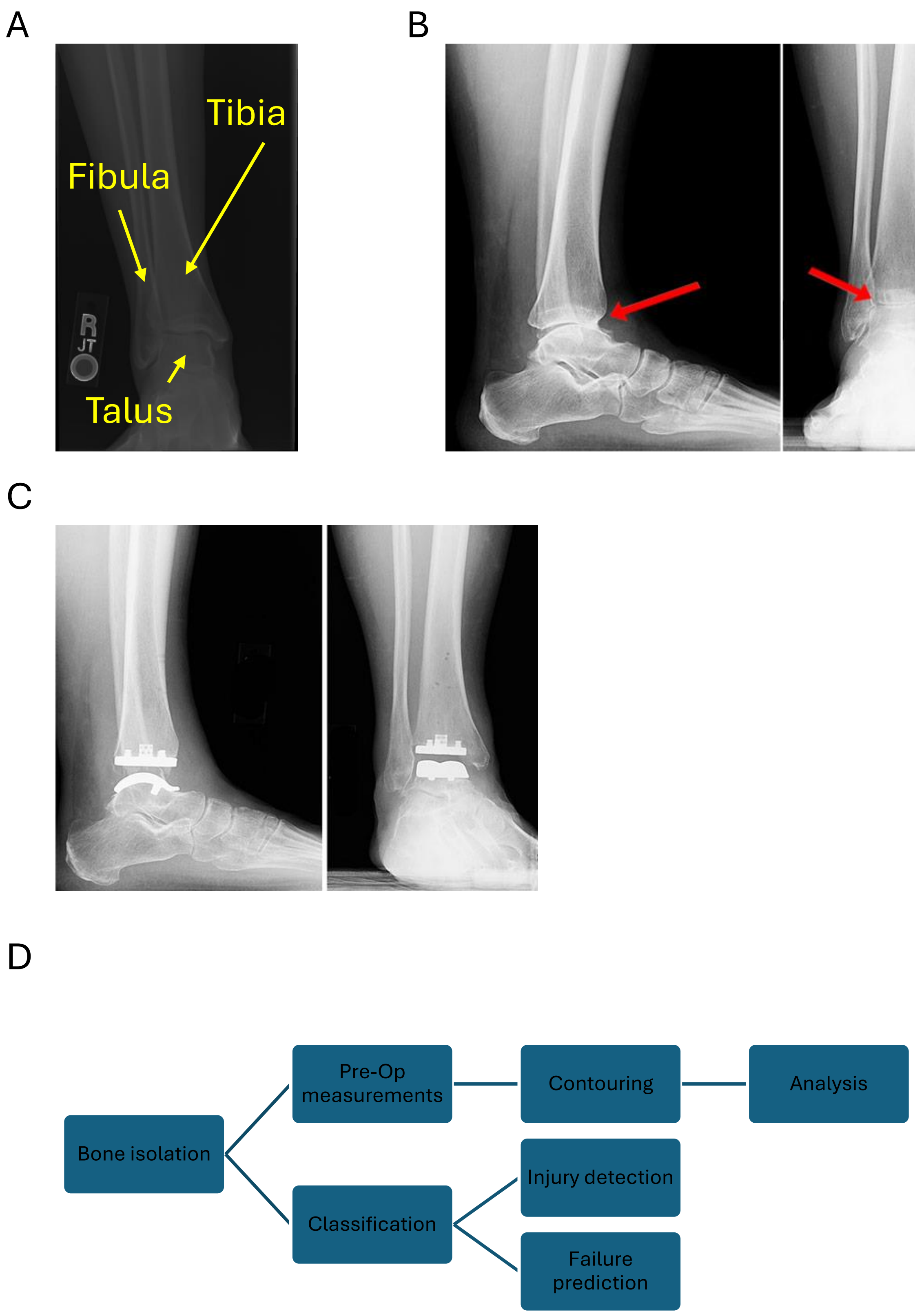

A
Tibia
Fibula
R
JT
Talus
B
C
D
Bone isolation
Pre-Op measurements
Contouring
Analysis
Classification
Injury detection
Failure prediction

Figure 1

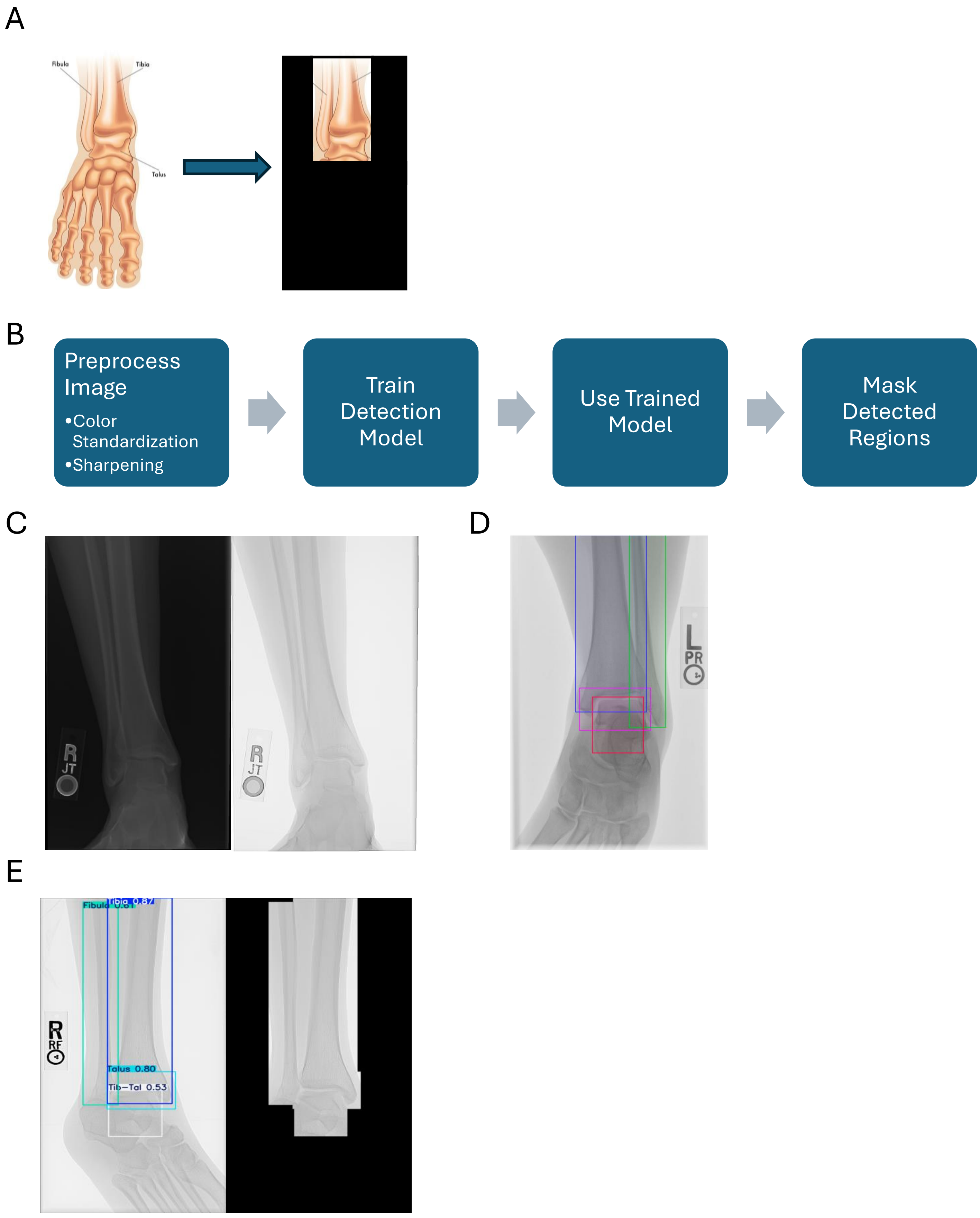
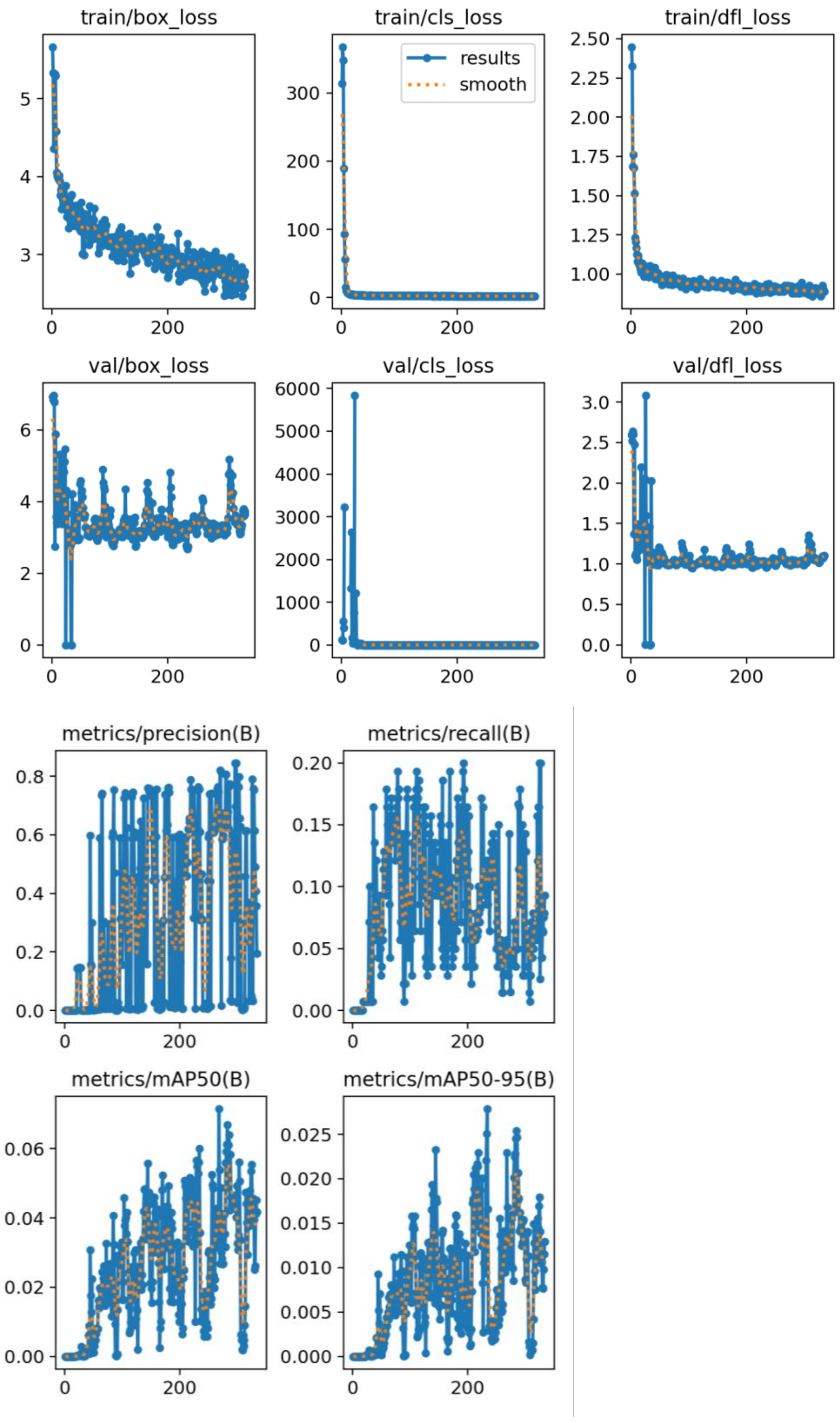

A
Fibula
Tibia
Talus
B
Preprocess Image
•Color Standardization
•Sharpening
Train Detection Model
Use Trained Model
Mask Detected Regions
C
R
JT
D
L
PR
E
Tibia 0.87
Fibula
Talus 0.80
Tib–Tal 0.53
R
RF

Figure 2



train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
val/box_loss
val/cls_loss
val/dfl_loss
metrics/precision(B)
metrics/recall(B)
metrics/mAP50(B)
metrics/mAP50-95(B)

Figure 3

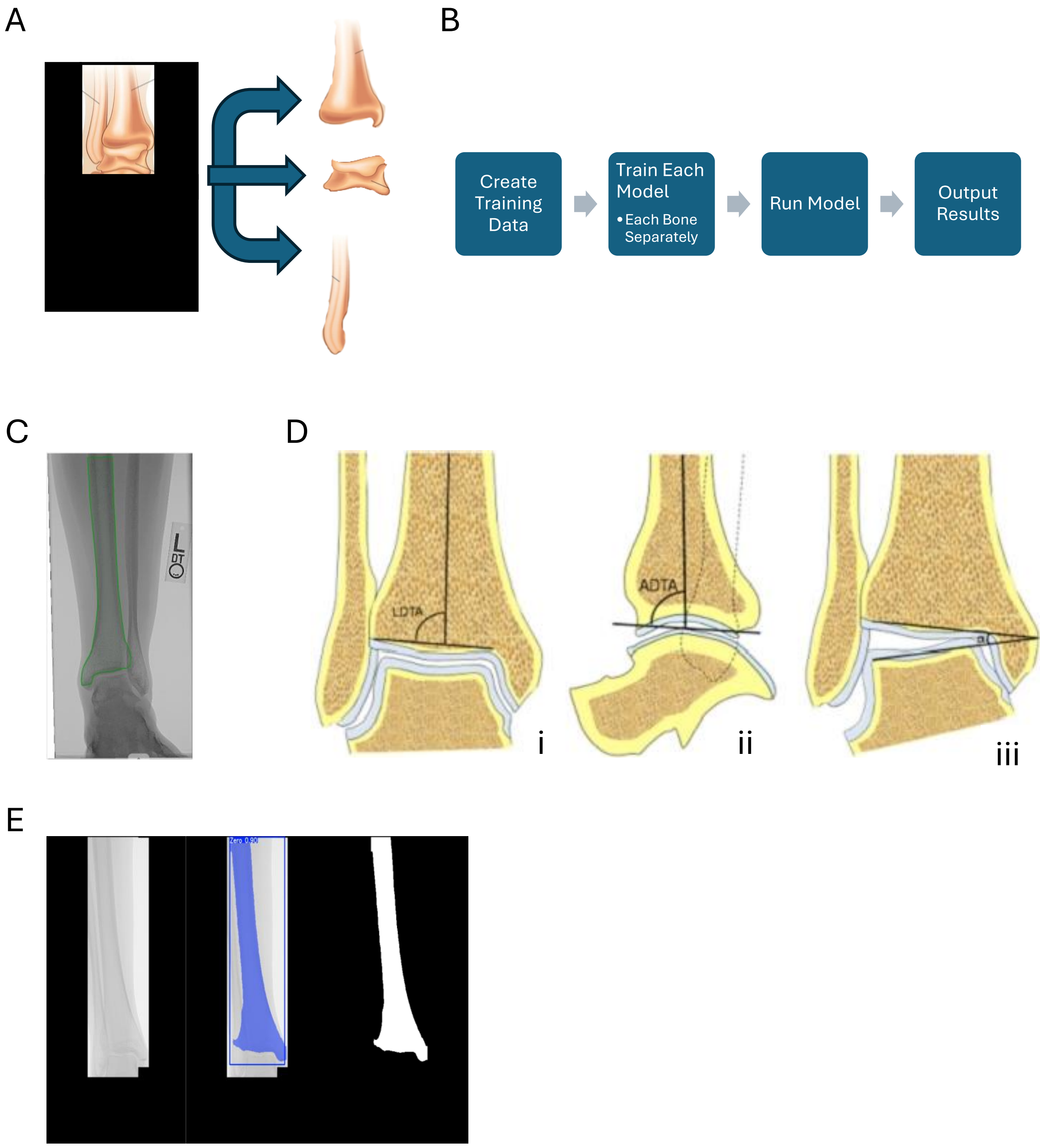

A
B
Create Training Data
Train Each Model
• Each Bone Separately
Run Model
Output Results
C
L DT
D
LDTA
ADTA
i
ii
iii
E
Zero 0.90

Figure 4

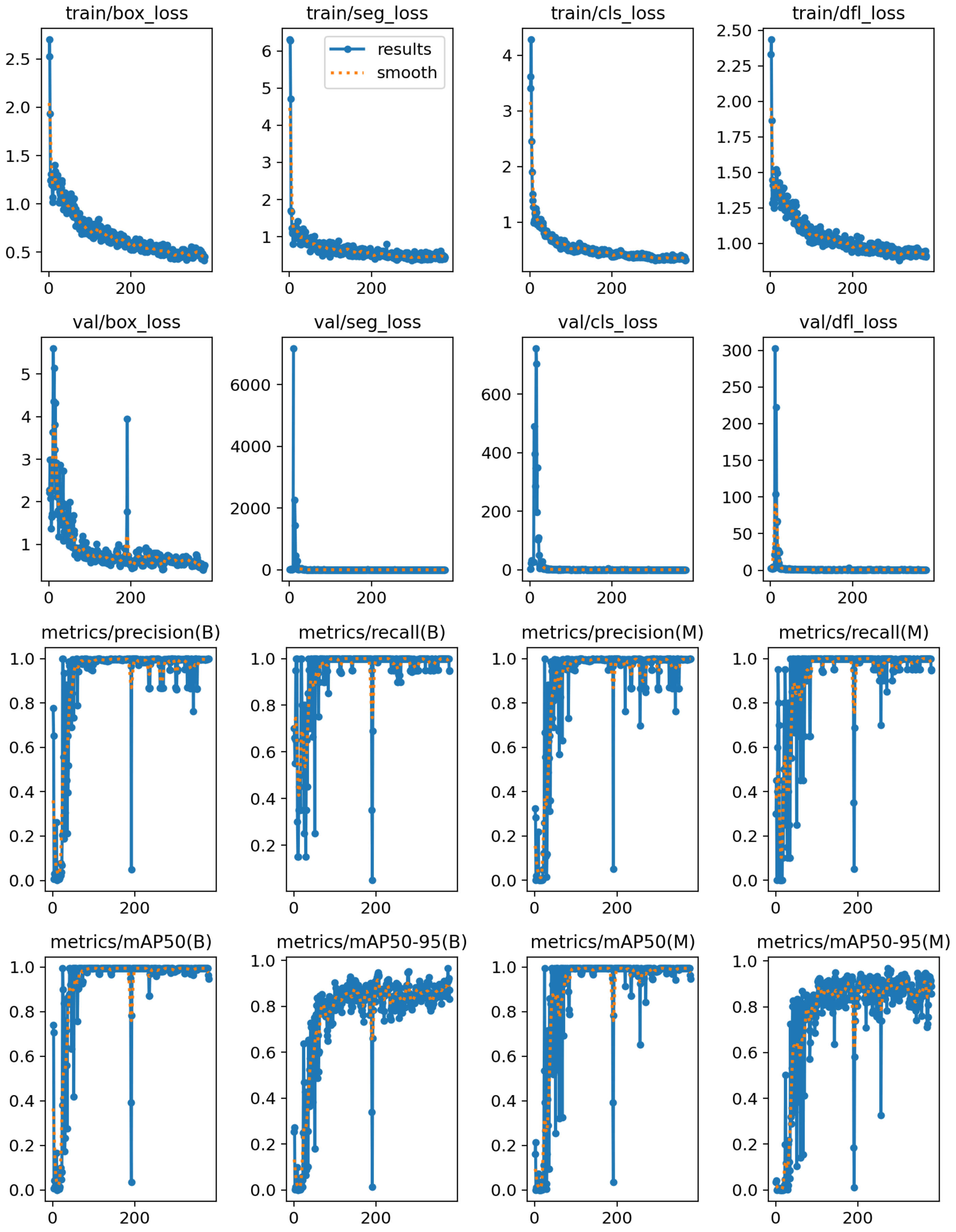
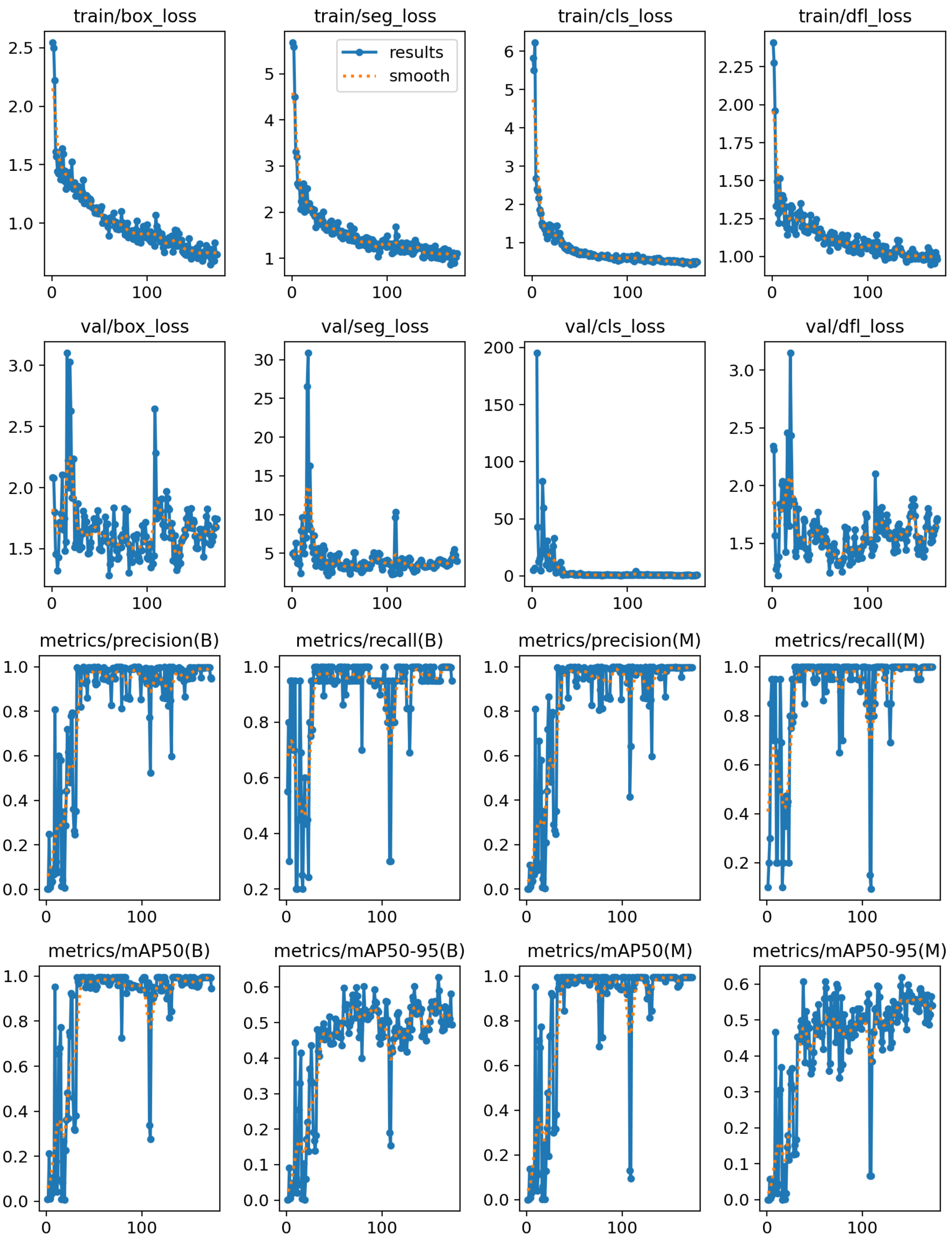
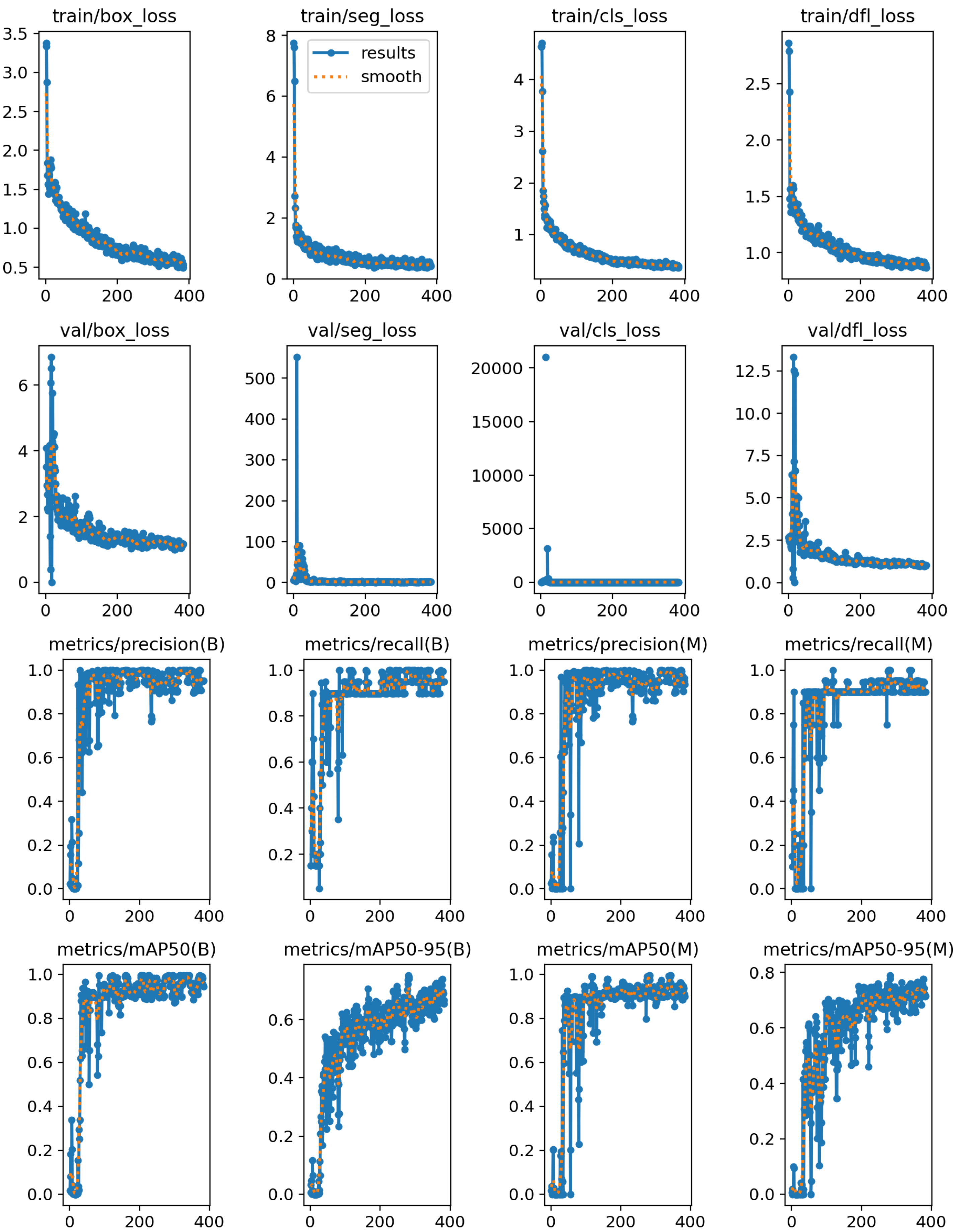

train/box_loss
train/seg_loss
results
smooth
train/cls_loss
train/dfl_loss
val/box_loss
val/seg_loss
val/cls_loss
val/dfl_loss
metrics/precision(B)
metrics/recall(B)
metrics/precision(M)
metrics/recall(M)
metrics/mAP50(B)
metrics/mAP50-95(B)
metrics/mAP50(M)
metrics/mAP50-95(M)

Figure 5

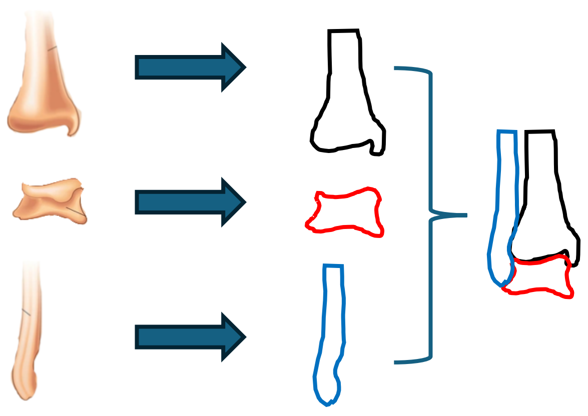

train/box_loss
train/seg_loss
results
smooth
train/cls_loss
train/dfl_loss
val/box_loss
val/seg_loss
val/cls_loss
val/dfl_loss
metrics/precision(B)
metrics/recall(B)
metrics/precision(M)
metrics/recall(M)
metrics/mAP50(B)
metrics/mAP50-95(B)
metrics/mAP50(M)
metrics/mAP50-95(M)
Figure 6



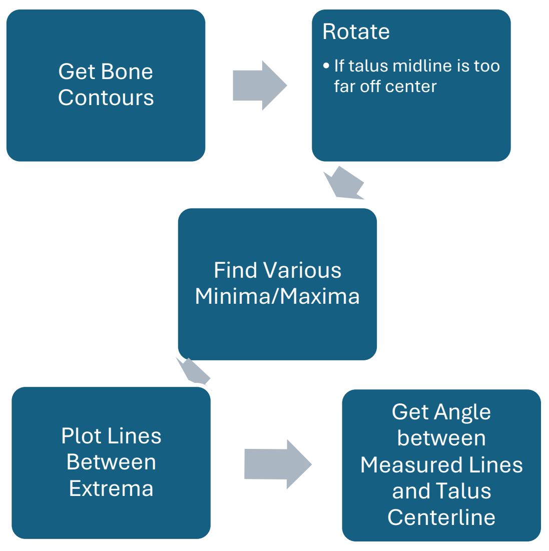
train/box_loss
train/seg_loss
train/cls_loss
train/dfl_loss
results
smooth
val/box_loss
val/seg_loss
val/cls_loss
val/dfl_loss
metrics/precision(B)
metrics/recall(B)
metrics/precision(M)
metrics/recall(M)
metrics/mAP50(B)
metrics/mAP50-95(B)
metrics/mAP50(M)
metrics/mAP50-95(M)

Figure 7

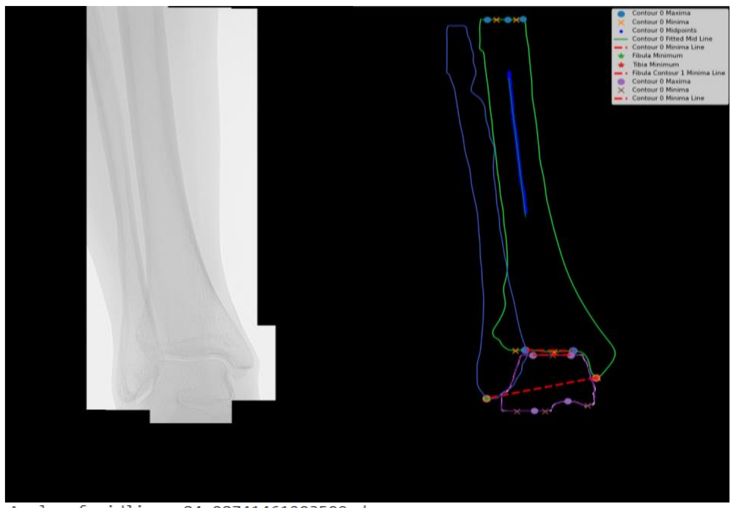
A



B

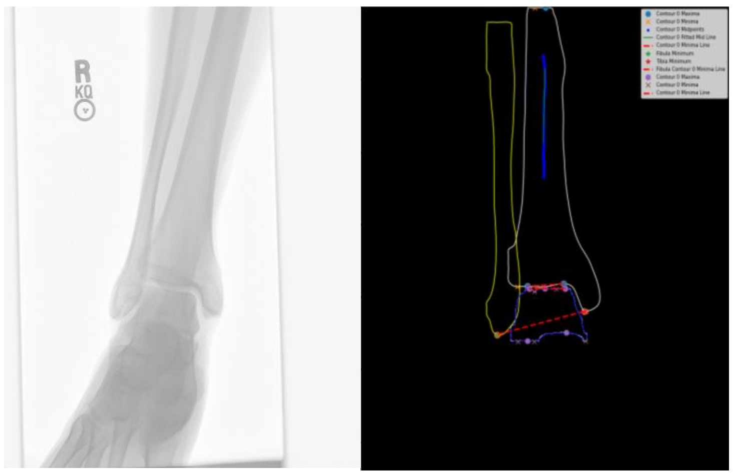

Get Bone Contours
Rotate
• If talus midline is too far off center
Find Various Minima/Maxima
Plot Lines Between Extrema
Get Angle between Measured Lines and Talus Centerline

C



```
Angle of midline: 84.08741461203509 degrees
Contour 0 Angle between midline and minima line: 82.87085694895562 degrees
Angle between midline and minimum line: -96.76779810385491
Angle between midline and talus line: -84.20345216866717
```

D



```
Tibia Midline Angle: -88.4359649058567 degrees
Distal Tibial Articular Angle: 84.59130048256375 degrees
Talar Center Migration Angle: 74.50427566405372 degrees
Talar Tilt Angle: 88.31432924233286 degrees
```

Figure 8