Kaggle Competition: Spinal Fracture Detection

Jacob Schneider-Martin

Overview

1. Clinical Background
2. Competition Background
3. Clinical Dataset
4. Model Development
5. Conclusion

Clinical Background

Spinal Fractures

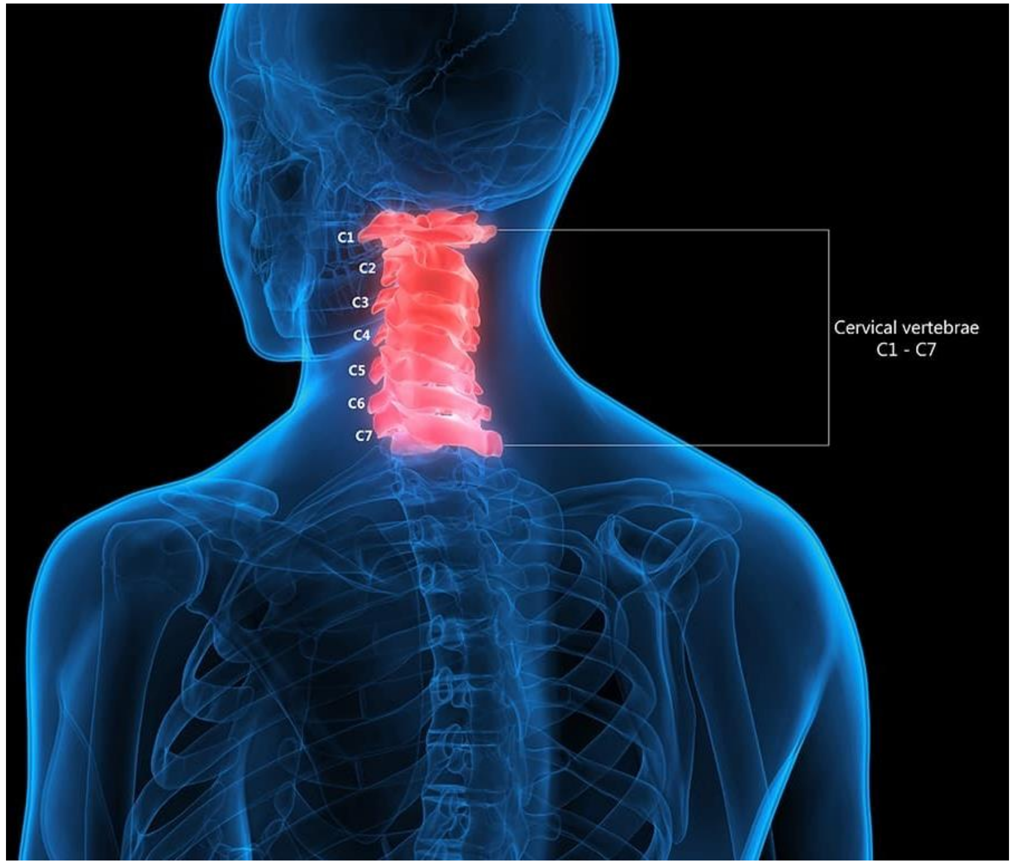
Clinical Background

- 1.5 million annual spinal fractures in U.S. result in 17k spinal cord injuries
- Most common site of vertebrae fracture is cervical spine (neck)
- Increased rate of spinal fractures in older populations

Spinal Fractures

Diagnosis

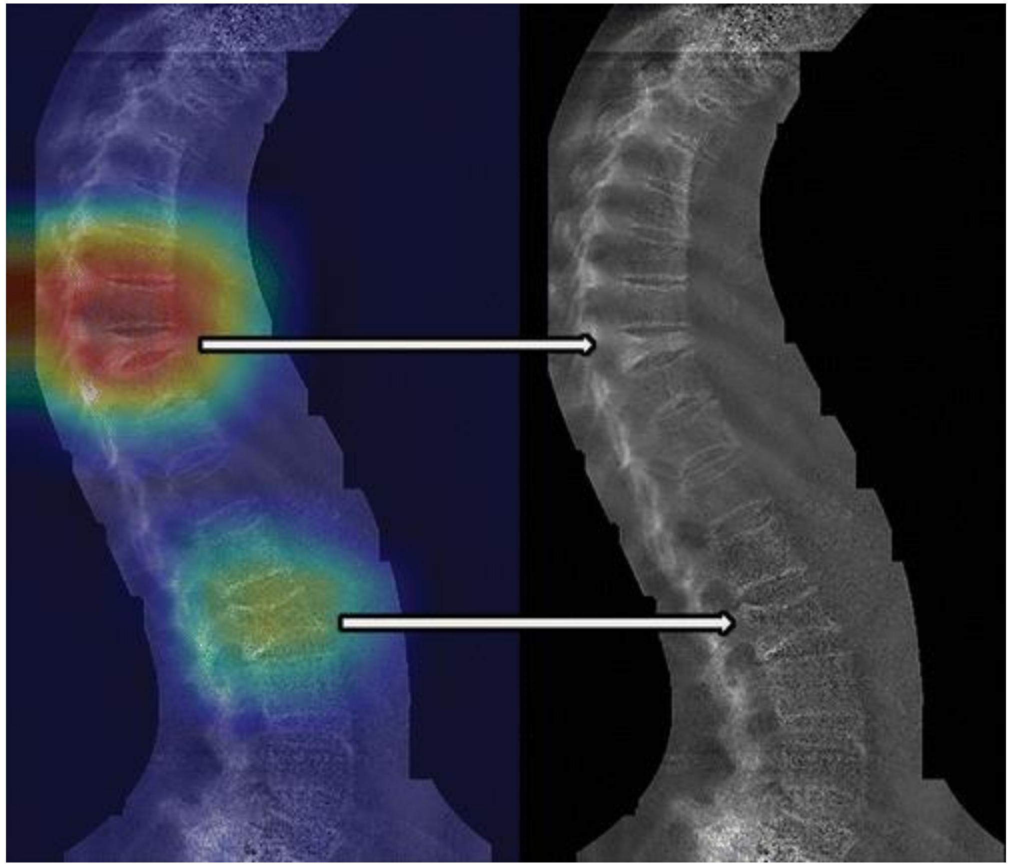
- Diagnosis of adult spinal fractures is performed with computed tomography (CT) scan capture
- In elderly patients, spinal fractures can be **difficult to detect on imaging**
  - Due to superimposed degenerative disease and osteoporosis
- **Critical** to quickly detect and determine location of vertebral fractures
  - Prevent neurologic deterioration & trauma



Competition Background

Kaggle Competition

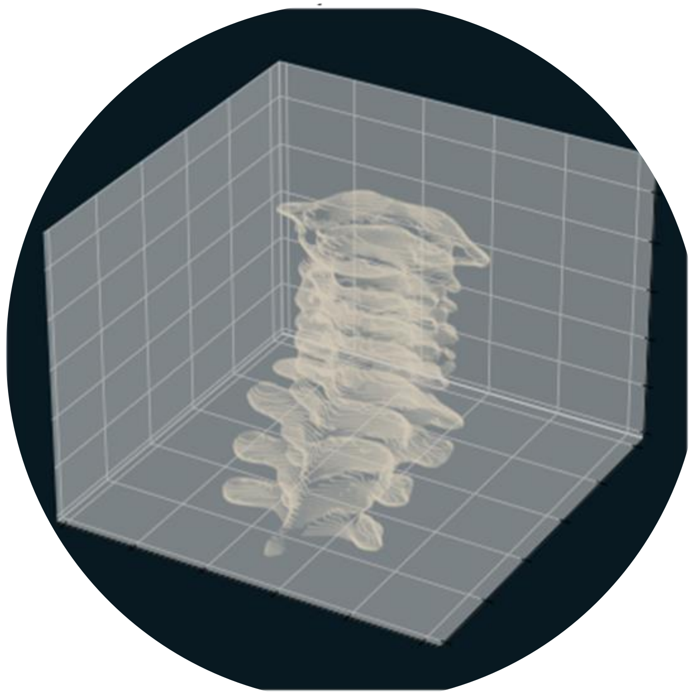
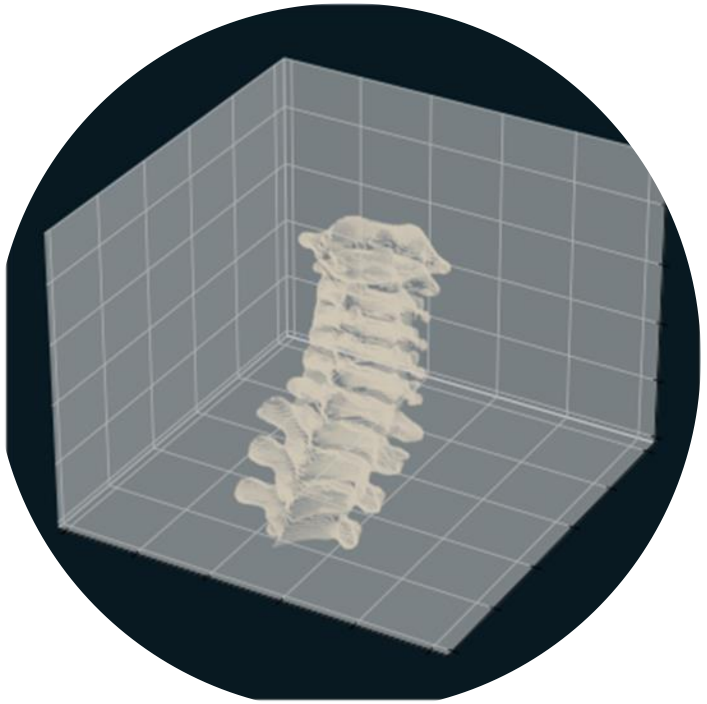
- AI models can detect and localize spinal fractures, which may improve clinical outcomes
- Deep learning classification models require well-labeled image data for development, but **datasets are not widely available** for spinal fracture CT images
- Kaggle competition dataset was released for the development of **deep learning models for spinal fracture detection and localization**
- Kaggle competition consisted of 880+ teams and $30k in prize money



Clinical Dataset

Clinical Dataset

Background

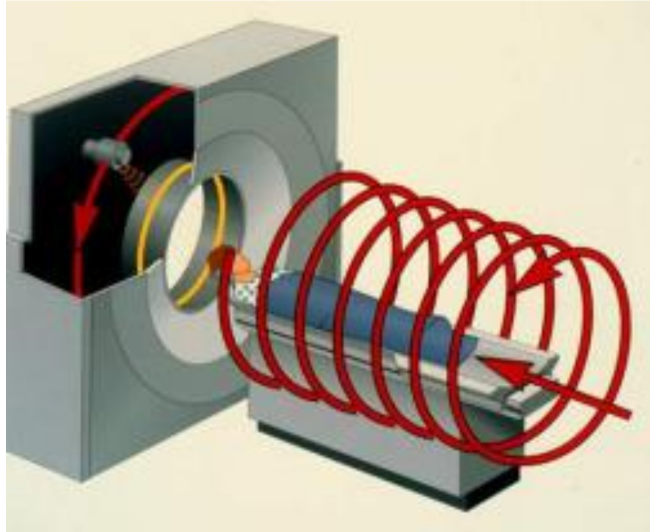



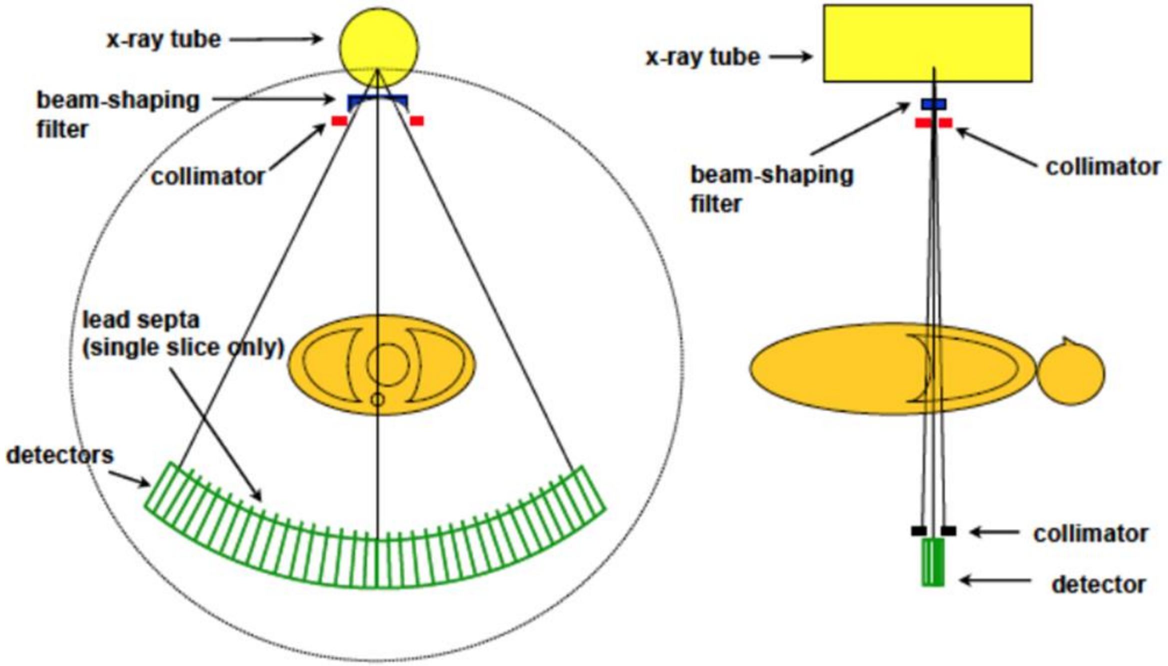
- Dataset organized by Radiological Society of North America, American Society of Neuroradiology and American Society of Spine Radiology
  - 12 research institutions from 9 countries contributed to dataset
- Dataset focused on fractures in cervical vertebrae C1-C7
  - Images annotated by medical experts
- Dataset
  - Public dataset contains n=2,019 CT scans
  - Hidden test contains n=1,500 CT scans

CT Scan

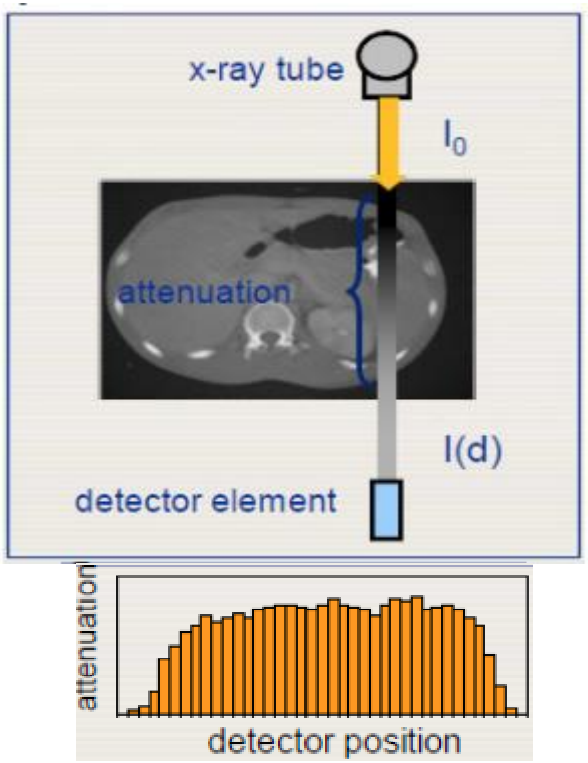
Image Capture

- Rotating x-ray tube and detectors measure x-ray attenuations from different projection angles
- Multiple attenuations are processed using tomographic reconstruction to create cross-sectional slices of patients, which can be viewed as a 3D surface rendering
- CT scanners have many hardware components that affect image capture and quality, such as the x-ray tube, beam filter, detector array and data acquisition system





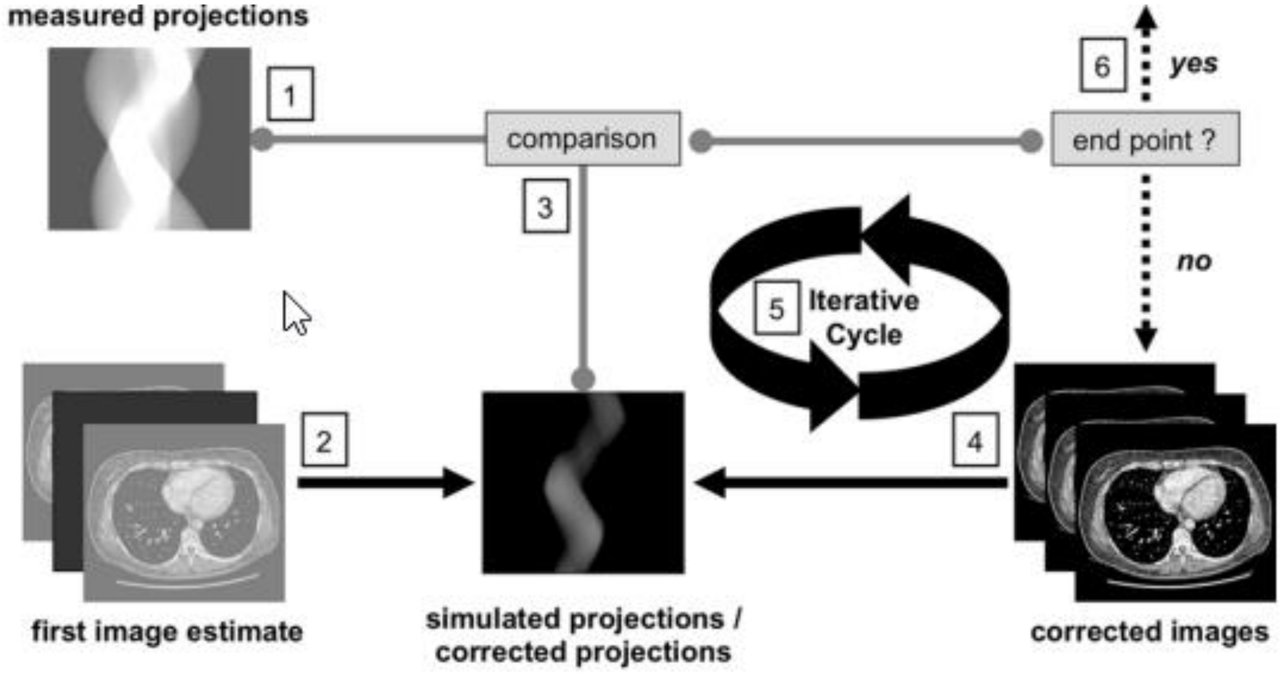
Helical CT scan capture



CT Scan

Image Reconstruction

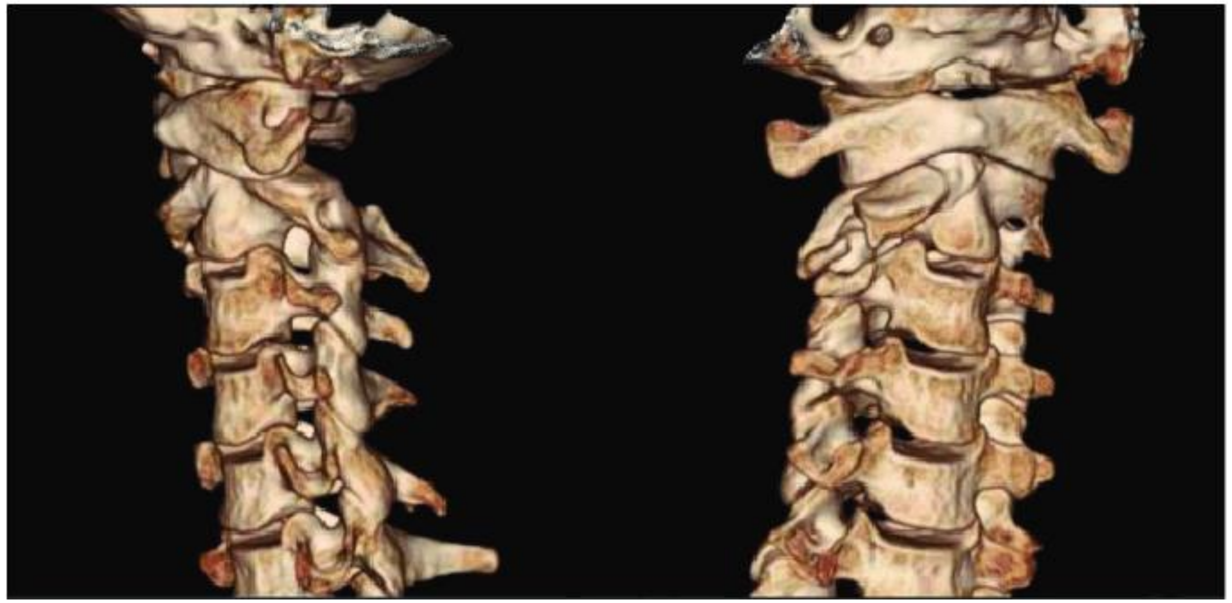
- Scan projection radiograph is a low exposure image that is first performed to help plan scan: determine scan range, field of view and gantry angle
- Reconstruction completed using filtered back projection (FBP) or iterative reconstruction
  - While reconstruction can be completed via open-source packages (i.e.: SIRT algorithm in ASTRA Python toolbox), reconstruction is typically completed shortly after scan using vendor-specific algorithm
- CT scans in Kaggle dataset were reconstructed by competition organizers
  - FBP reconstruction used with “bone kernel”
  - Bone kernel: higher resolution, but more noise



Iterative reconstruction

CT Scan

Image Quality



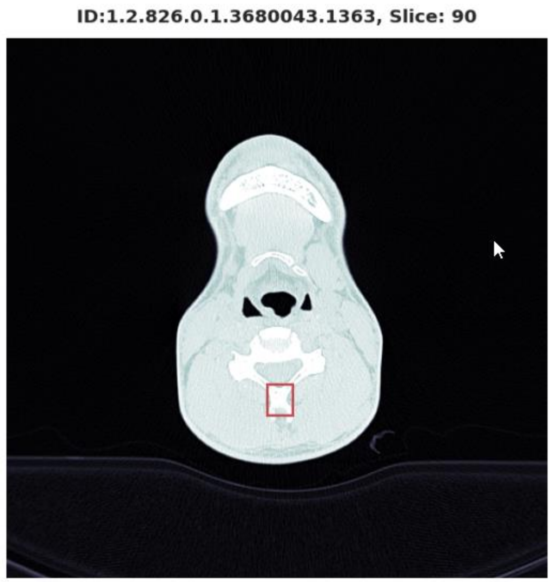
- Image quality dependent on contrast, resolution and noise
- Primary acquisition parameters:
  - Tube voltage, tube current, rotation time
- Secondary acquisition parameters:
  - Detector configuration and beam collimation
  - Pitch
  - Type of reconstruction
  - Reconstruction filter
  - Slice thickness
- Due to different scanners and acquisition parameters, CT scans in Kaggle dataset vary in image quality

Clinical Dataset

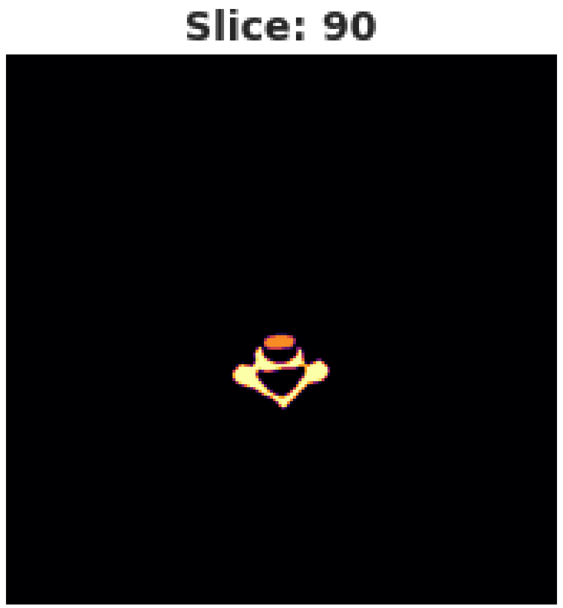
Background

- N=2019 patients have **scan-level** fracture information for vertebrae C1-C7
  - Large dataset: **700k+ image slices**
- CT scans: 0.5 to 1mm slice thickness, axial orientation
- Image slice size: 512 x 512 pixels
- DICOM format

StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7
1.2.826.0.1.3680043.25891	1	0	0	0	0	1	1	0
1.2.826.0.1.3680043.17325	1	0	0	0	1	1	0	0
1.2.826.0.1.3680043.30177	1	0	0	1	0	0	1	1
1.2.826.0.1.3680043.23052	0	0	0	0	0	0	0	0
1.2.826.0.1.3680043.27299	1	0	0	0	0	0	0	1



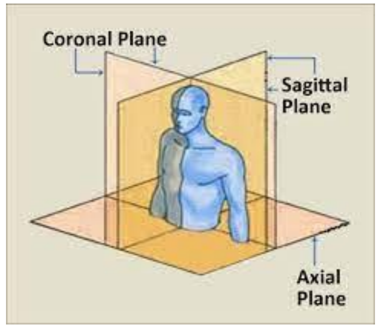
- N=235 scans with fractures have **slice-level** bounding boxes that show fracture location
- Bounding boxes are in .csv format in axial orientation

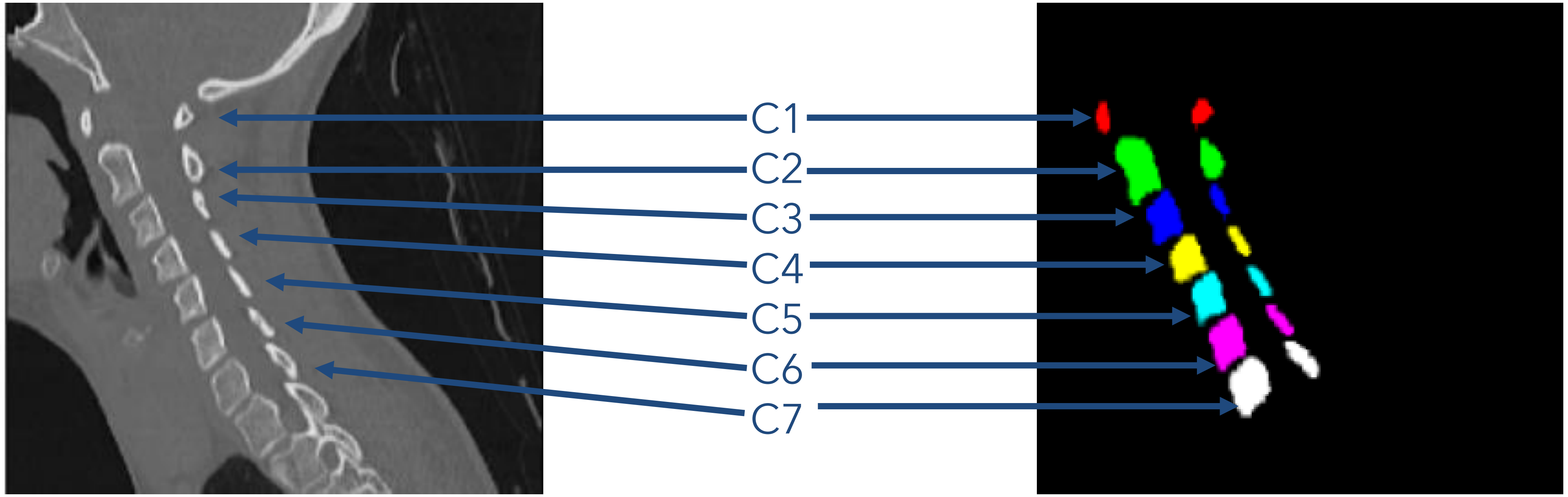


- N=87 scans have **slice-level** segmentations for vertebrae
- Semantic segmentation: label each pixel with its class
- Segmentations are in NIFTI format in sagittal orientation

Clinical Dataset

Segmentations in Sagittal Orientation

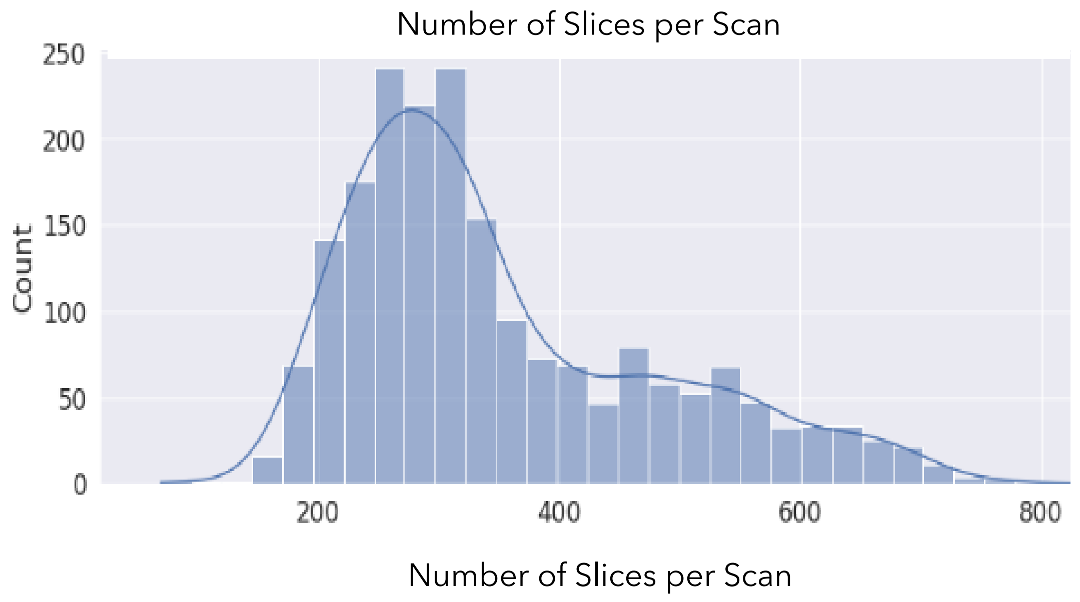


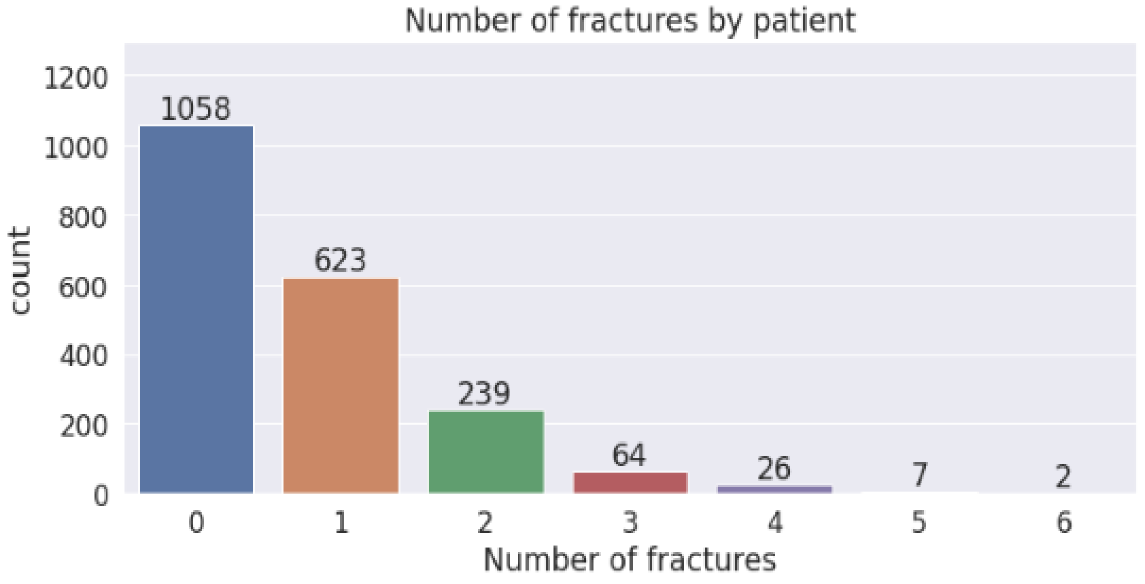


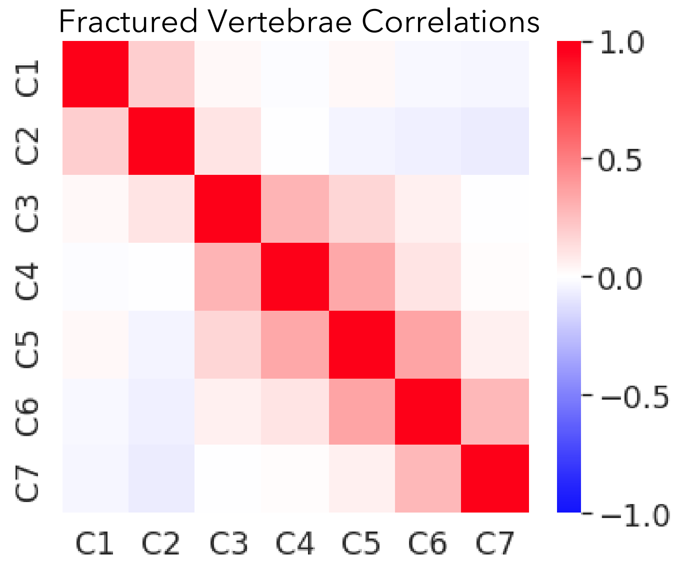
Clinical Dataset

Exploratory Analysis

- N=7 different cervical vertebrae shown in images, as well as other anatomical structures, such as thoracic spine, neck and skull
- Similar number of CT scans for fractured and non-fractured patients
- Similar number of fractures across different vertebrae
- CT scans most frequently have ~300 image slices, although certain patients have 800+ slices
- Most fractured patients have a fracture in only a single vertebra
- Scans with multiple fractures indicate that adjacent vertebrae are more likely to both be fractured



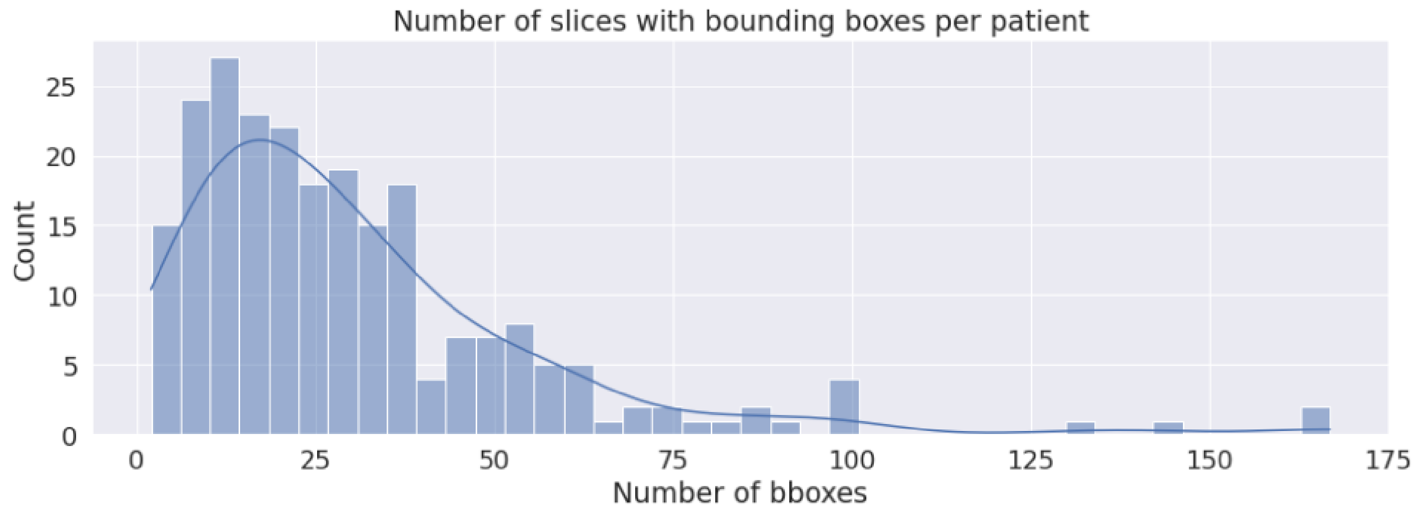


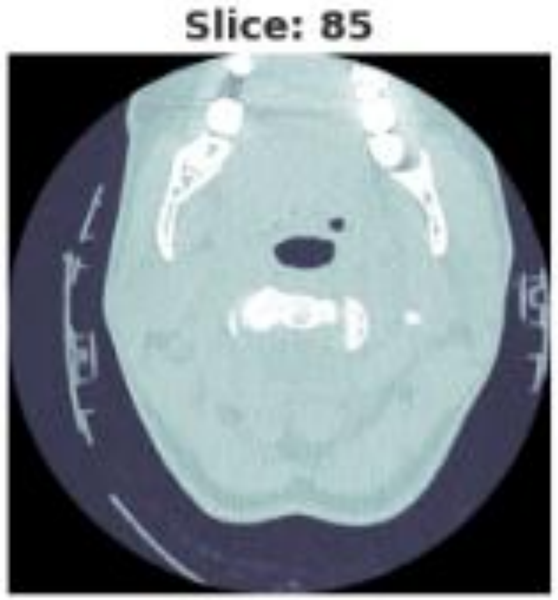


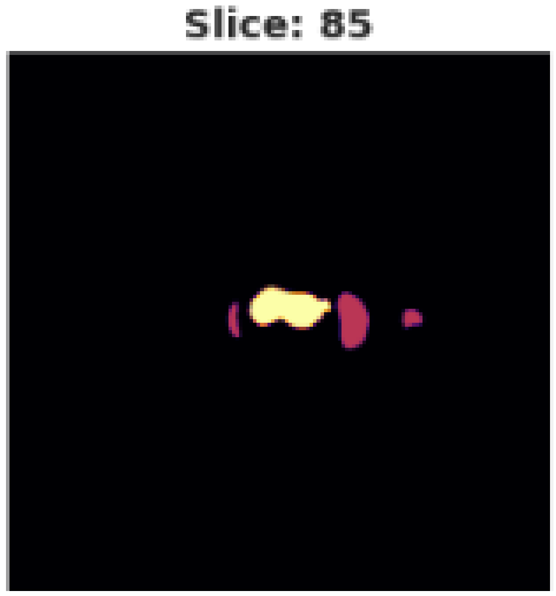
Clinical Dataset

Technical Challenges

- Differences in patient position and size, as well as image quality and field of view, across scans
- Class imbalance
  - Based on fractured patients with bounding boxes, fractures show up in only ~25 slices out of ~300 total slices per scan
- Lack of annotated data: ground truth data gives scan-level fracture information, but **slice-level ground truth data is limited**
  - Slice-level segmentations available for 4% of patients
  - Slice-level fracture bounding boxes available for 24% of patients with fractures
- Segmentation data: Certain classes consist of small numbers of pixels







Model Development

Model Development Steps

1. Development Approach
2. Data Loading
3. Training
4. Pseudo-Labeling
5. Semi-Supervised Learning

Model Development Approach

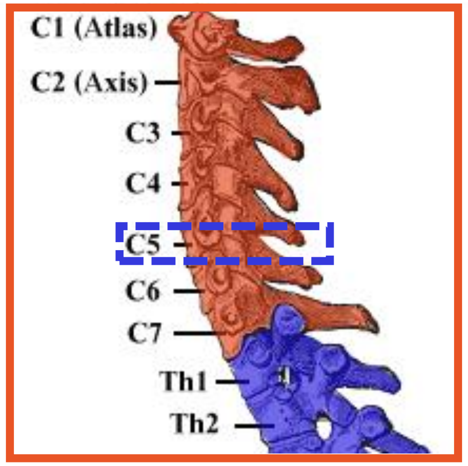
- **Problem: Scan-level** fracture annotations are available, but **slice-level** fracture annotations are not readily available
  - Training scan-level fracture detection model would most likely not be successful due to minimal annotated data
- **Solution:** Leverage bounding box data (n=235 patients) to train slice-level fracture detection model
  - For patients with bounding box annotations, make an educated **assumption** that all **slices without bounding boxes do not contain fractured vertebrae**
  - Use bounding box information to provide slice-level (axial) localization to model

Scan-level annotation

StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7
1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0

Slice-level annotation

StudyInstanceUID	Slice	C1	C2	C3	C4	C5	C6	C7
1.2.826.0.1.3680043.1363	90	0	0	0	0	1	0	0



Orange indicates scan-level annotations available for entire dataset and corresponding ROI for model. Blue indicates slice-level annotations and corresponding ROI for model. Bounding box information was leveraged to provide axial localization, rather than in sagittal and coronal directions.

Model Development Steps

1. Development Approach
2. Data Loading
3. Training
4. Pseudo-Labeling
5. Semi-Supervised Learning

Data Loading Pipeline

Data storage

- Save images in online directory
- Change orientation of segmentations from sagittal to axial
- Create CSV with image filenames and ground truth metadata targets
- Data split: 80% training, 10% validation, 10% test
- Validation and test set patient scans not included in training

Dataset class

- Iterate through CSV and load individual images and ground truth targets
- Apply transformation & normalization to image

Sampling

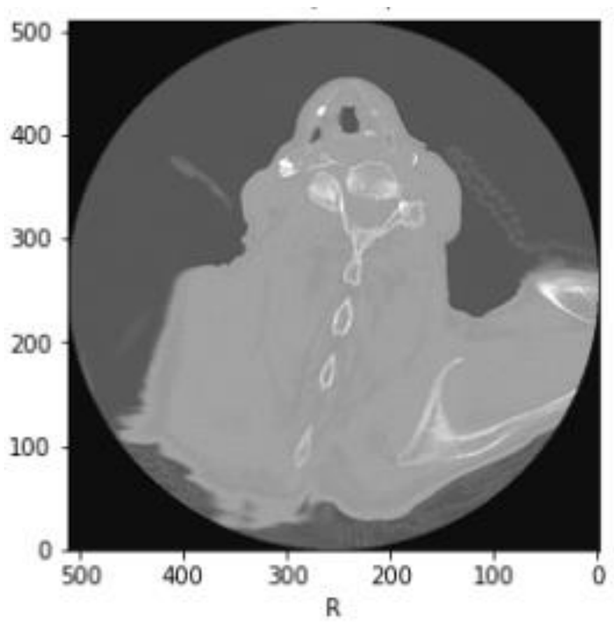
- Determine order of data that is loaded into model
- Training data randomly shuffled each epoch
- Validation data loaded in identical order each epoch using manual seed
- Custom Sampler allows for oversampling

Dataloaders

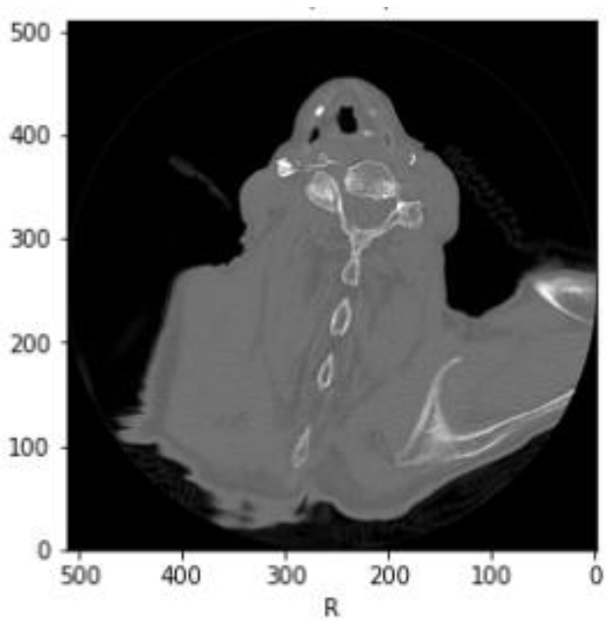
- Parallelize data loading process to increase speed and memory
- Individual dataloaders for training, validation and test
- Different transformations for training, validation and test

Data Pre-Processing

- Goal: allow network to see inputs with uniform intensities across dataset
- Intensity pre-processing
  - Transformation: truncate outlier Hounsfield intensity values that are not within expected range (i.e.: artifacts) using Clamp transform
  - Normalization: transform Hounsfield scale to intensity values in range [0, 1]



Original Image Slice



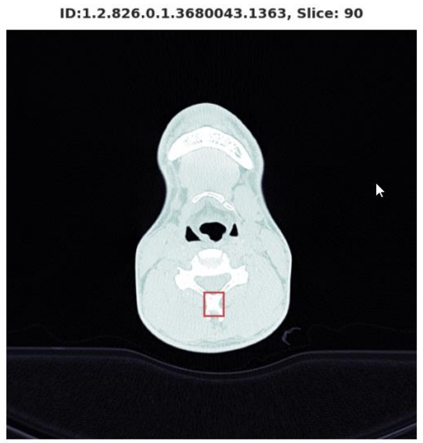
Transformed & normalized image slice

Data Loading

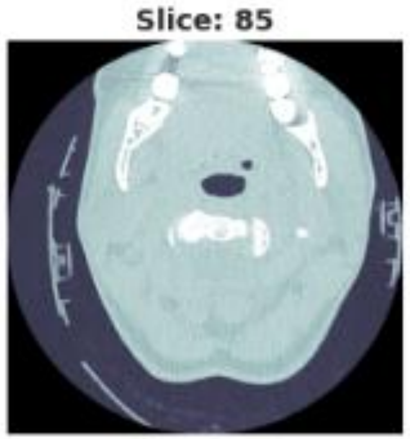
- Extract metadata (which vertebrae are fractured in each slice) from bounding box data to use as ground truth targets for fracture detection model
- Train fracture detection model using n=235 fractured patients with bounding boxes and n=235 non-fractured patients
  - Use all image slices (110k+ images) from this patient cohort to train model

Input Images

C5 Fracture



No Fracture



Targets

StudyInstanceUID	Slice	C1	C2	C3	C4	C5	C6	C7
1.2.826.0.1.3680043.1363	90	0	0	0	0	1	0	0

StudyInstanceUID	Slice	C1	C2	C3	C4	C5	C6	C7
1.2.826.0.1.3680043.10051	85	0	0	0	0	0	0	0

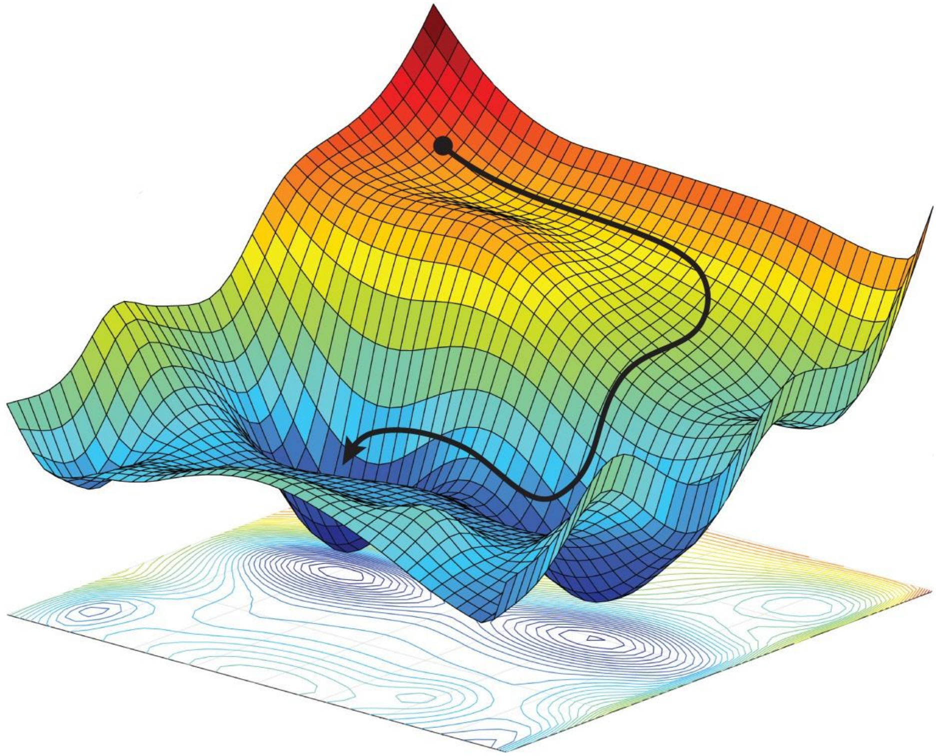
Model Development Steps

1. Development Approach
2. Data Loading
3. Training
4. Pseudo-Labeling
5. Semi-Supervised Learning

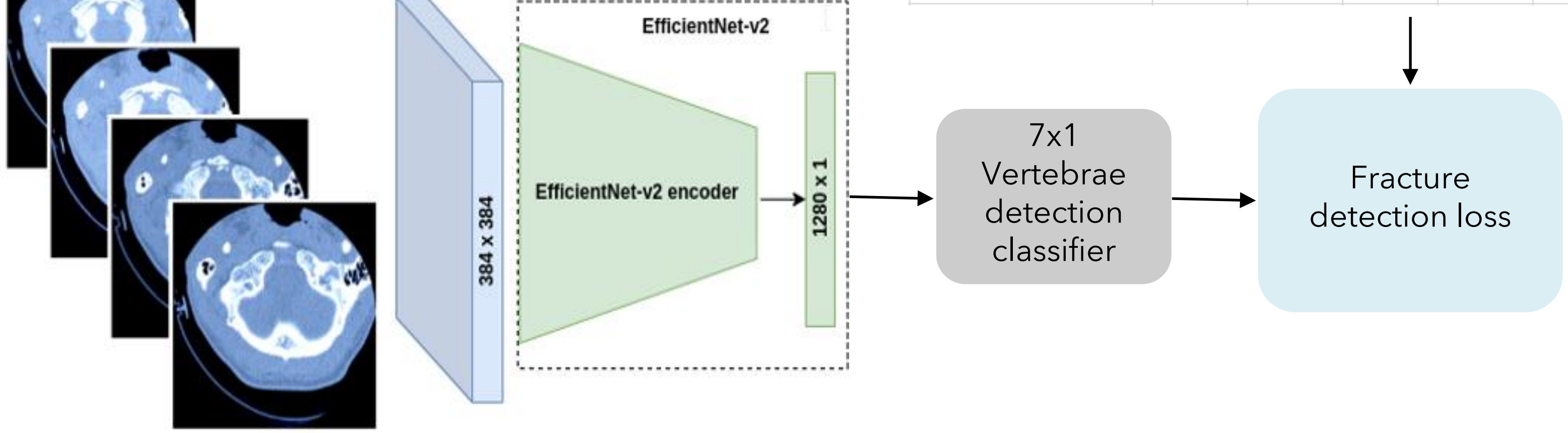
2D Model Training

Fracture Detection

- **2D Baseline:** Use pre-trained EfficientNetV2 (21.5M parameters) as feature extractor to create fracture detection model
  - EfficientNetV2 has improved parameter efficiency and accuracy
  - Finetune final layer using challenge dataset



StudyInstanceUID	Slice	C1	C2	C3	C4	C5	C6	C7
1.2.826.0.1.3680043.1363	90	0	0	0	0	1	0	0



2D Model Training

Fracture Detection

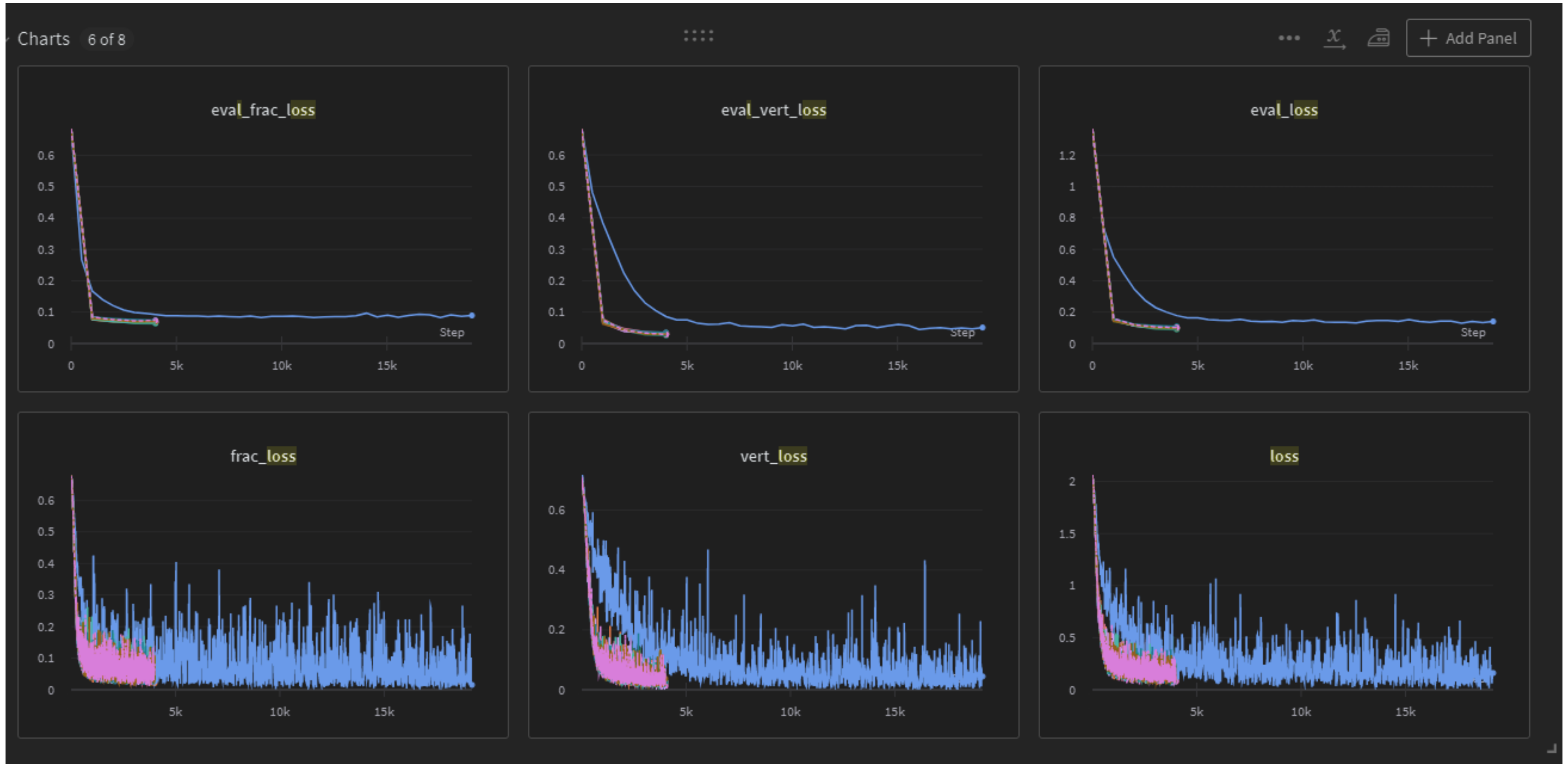
- Training conducted with Nvidia P100 GPU in Kaggle environment
- Computational constraints: 16GB memory
  - Increased batch size to maximum possible number of images
- Loss and metrics recorded with Weights & Biases

PyTorch

scikit-image
image processing in python

W&B

TorchIO



2D Model Training

- Augmentations: dataset is not very large, so very heavy augmentations were not applied
  - N=6 augmentations were applied, most with 30% probability: random resized crop, vertical and horizontal flips, rotations, RGB shift, CLAHE (sharpens contrast to identify small fractures)
  - Metric: Per competition rules, weighted binary cross entropy (BCE) used to evaluate submissions
  - Metric favors a more conservative model: sensitivity more important than specificity

$$L_{ij} = -w_j * [y_{ij} * log(p_{ij}) + (1 - y_{ij}) * log(1 - p_{ij})]$$

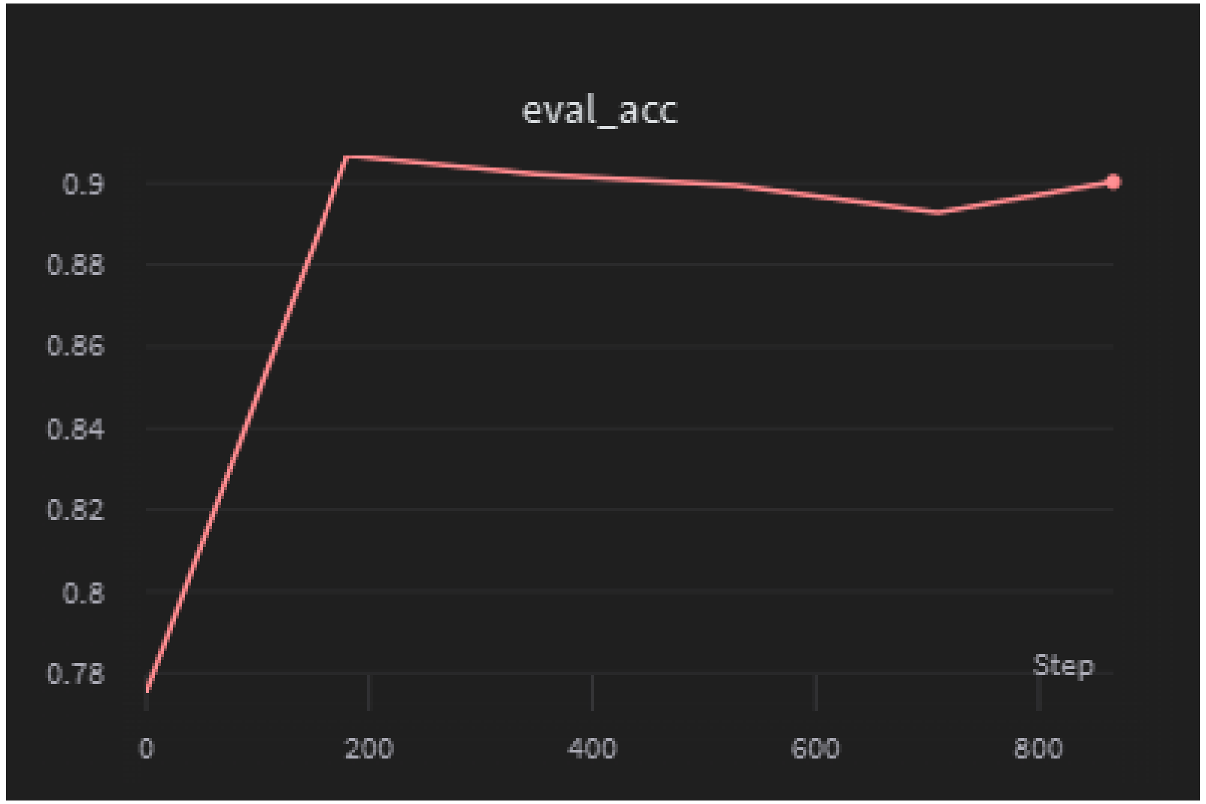
$$w_j = \begin{cases} 1, & \text{if vertebrae negative} \\ 2, & \text{if vertebrae positive} \\ 7, & \text{if patient negative} \\ 14, & \text{if patient positive} \end{cases}$$

- Loss: Weighted binary cross-entropy, identical to metric
- Additional metric :
  - Fracture accuracy: correctly classified vertebrae/total vertebrae

2D Model Training & Evaluation

Fracture Detection

Results: 90% accuracy for fracture detection

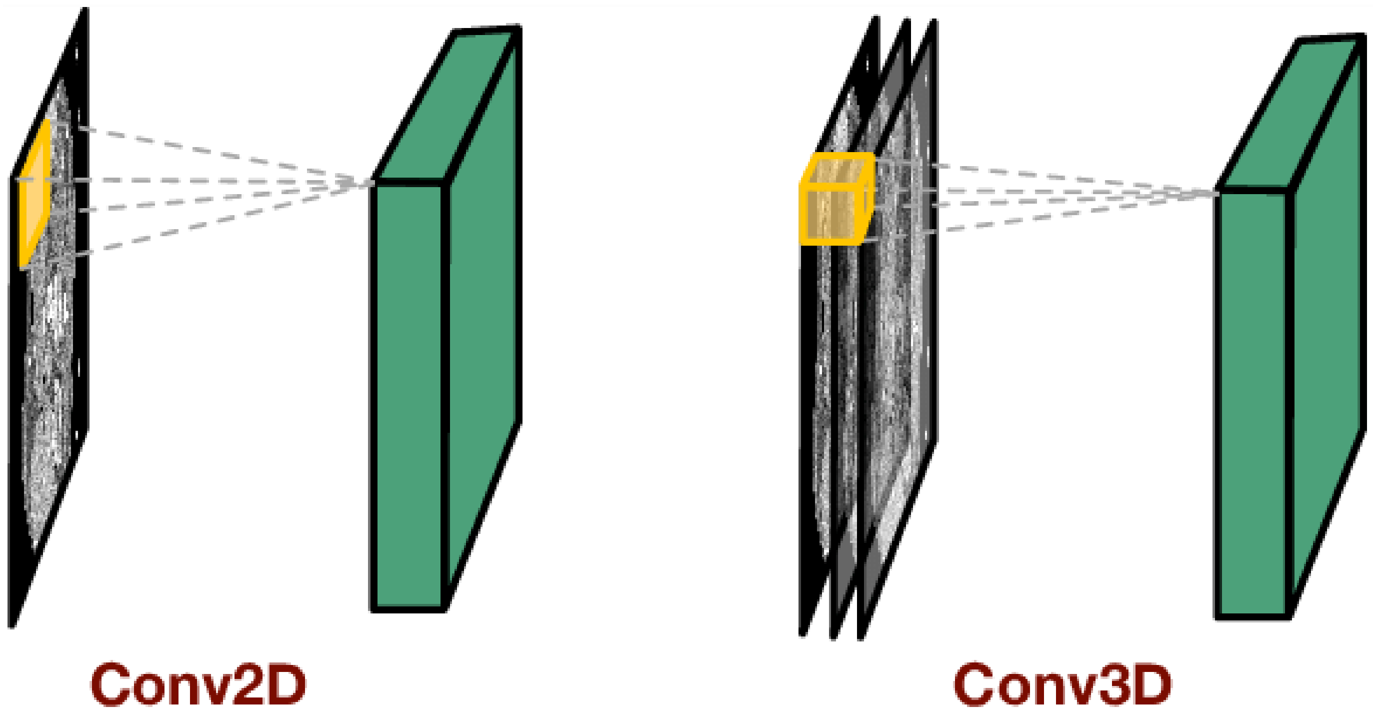


Hyperparameter optimization

- Model: EfficientNetV2
- Optimizer: Adam
- Learning rate: 1E-3
- Loss: weighted BCE
- Regularization: 20% dropout

2D vs 3D Modeling

- With baseline created using **2D model**, next step was to attempt to improve results using **3D model**
- 3D model architectures include significantly more contextual information from adjacent slices, which can improve results
- Input is 3D volumetric patches rather than 2D slices

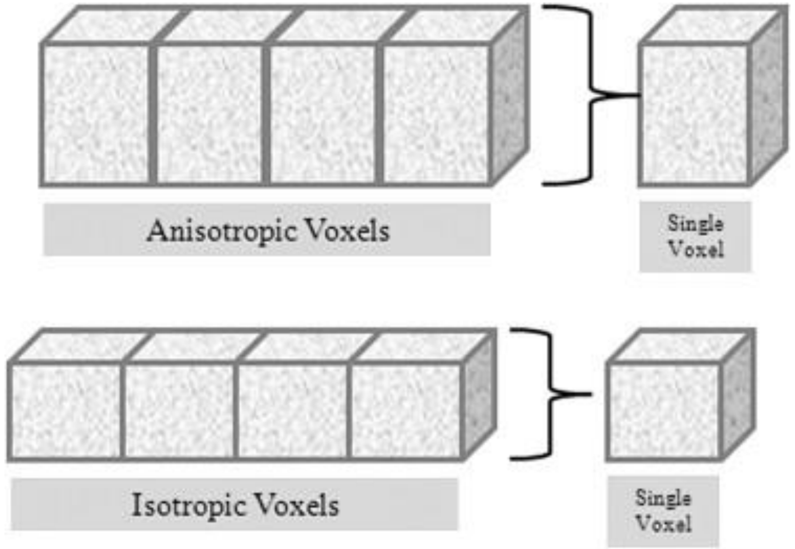


- Use pre-trained 3D ResNet-18 model (Med3D Net, 33M parameters) as feature extractor to create fracture detection model
  - Med3D model was pre-trained on medical images, rather than ImageNet, which can decrease convergence time for new medical imaging tasks
  - Finetune final layer using challenge dataset

Med3D: Transfer Learning for 3D Medical Image Analysis

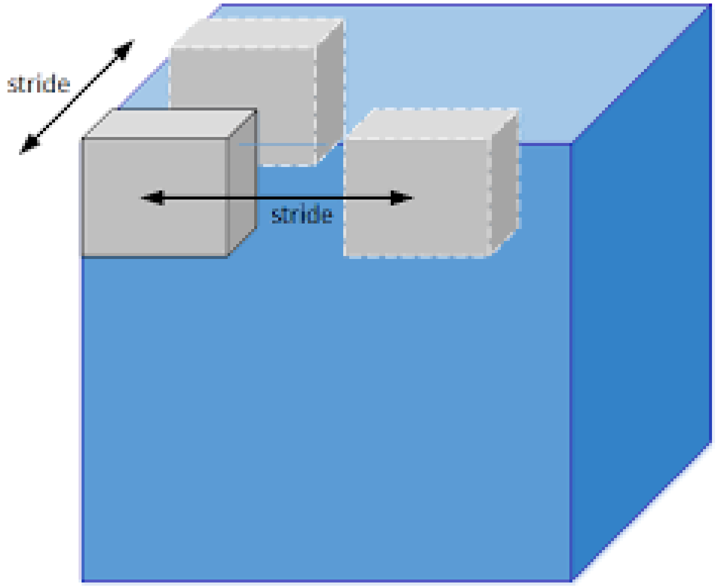
3D Data Pre-Processing

- Intensity pre-processing was previously completed in 2D modeling
- Spatial pre-processing allows network to see inputs with uniform sizes across dataset
  - Resampling: standardize physical resolution by downsampling to 1mm isotropic voxel sizing, which was lowest resolution in dataset
  - Voxel size hyperparameter: tradeoff between more anatomical data in receptive field versus improved resolution
  - Downsampling allows for faster computations



3D Data Loading

- 2D data loading pipeline is maintained, with additional features added
  - Due to GPU memory limitations, patch-based pipeline was implemented
- Images are divided into patches of 56x56x56 pixels
  - This spans length of approximately two cervical vertebrae so that contextual information is included
- Patch LabelSampler: randomly extract patches from volume based on class probabilities
  - Probabilities
    - No fracture: 0.7
    - Fracture: 0.3
  - This effectively implements oversampling of fracture data
  - Sample n=8 patches per volume per epoch
- TorchIO queue to improve data parallelism
- PyTorch DataLoader is based on TorchIO queue



3D patches sampled from 3D volume

3D Model Training & Evaluation

Fracture Detection

Training

- Augmentations N=6 augmentations were applied, most with 30% probability
  - Random patch scaling, vertical and horizontal flips, rotations, RGB shift, CLAHE
- Hyperparameter optimization
  - Model: 3D ResNet-18
  - Optimizer: Adam
  - Learning rate: 1E-3
  - Loss: weighted BCE
  - Regularization: 20% dropout

Results

- 88% accuracy for fracture detection, which was lower than 90% accuracy for 2D EfficientNetV2 model
  - This was surprising, as 3D models have more contextual information, which is almost always helpful
  - Post-competition, other competitors also reported slightly worse results for 3D models on this dataset
- Slice-based strategy using 2D EfficientNetV2 was selected as approach

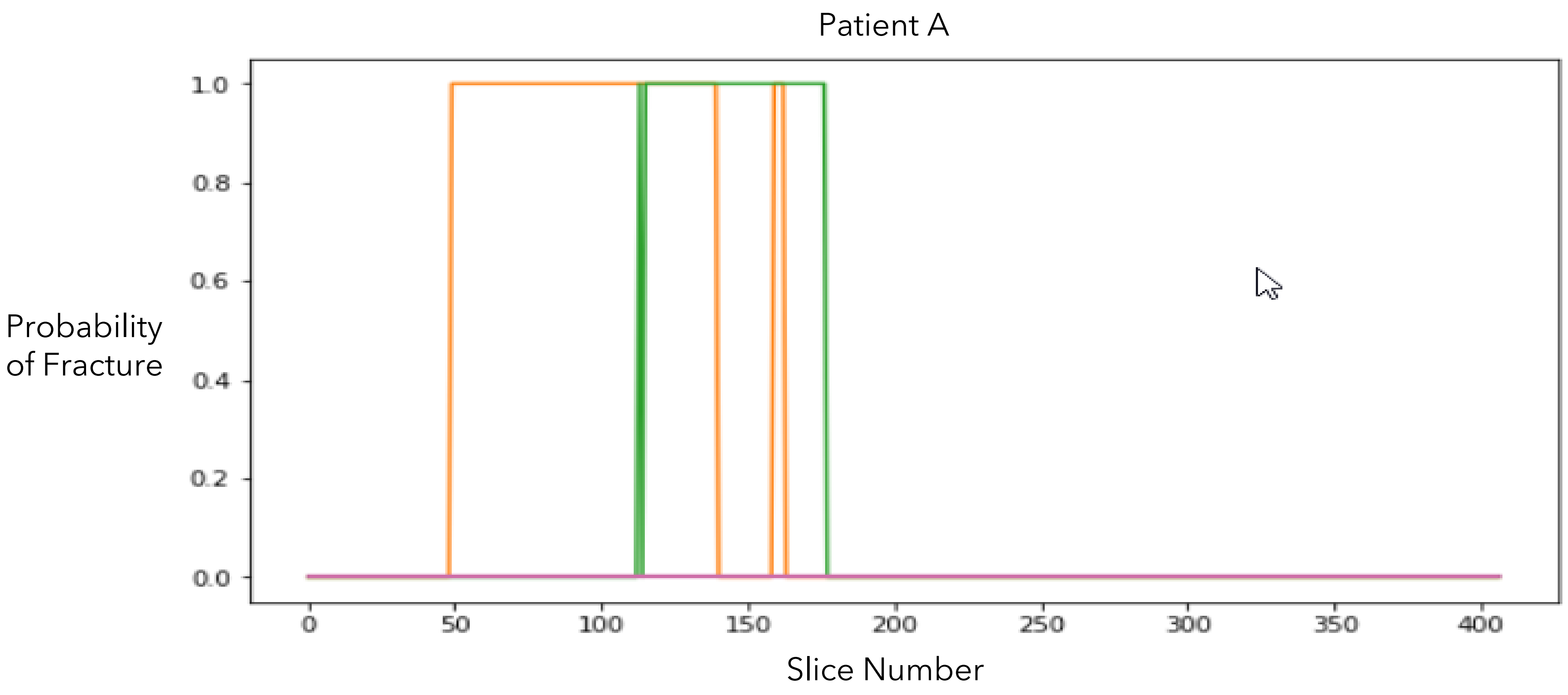
Model Development Steps

1. Development Approach
2. Data Loading
3. Training
4. Pseudo-Labeling
5. Semi-Supervised Learning

Model Inference

Fracture Detection

- Fracture detection model inference performed on rest of fractured dataset (n=734 patients without bounding boxes) to generate slice-level predictions of vertebrae fracture
  - Not necessary to perform slice-level inference on non-fractured patients, as no slices contain fractures
- Generated pseudo-labels for all image slices in dataset that predict probability of fractured vertebrae
  - Probabilities were binarized in post-processing
  - Colors below correspond to binarized probability of fracture for 7 different vertebrae in all image slices for a particular patient



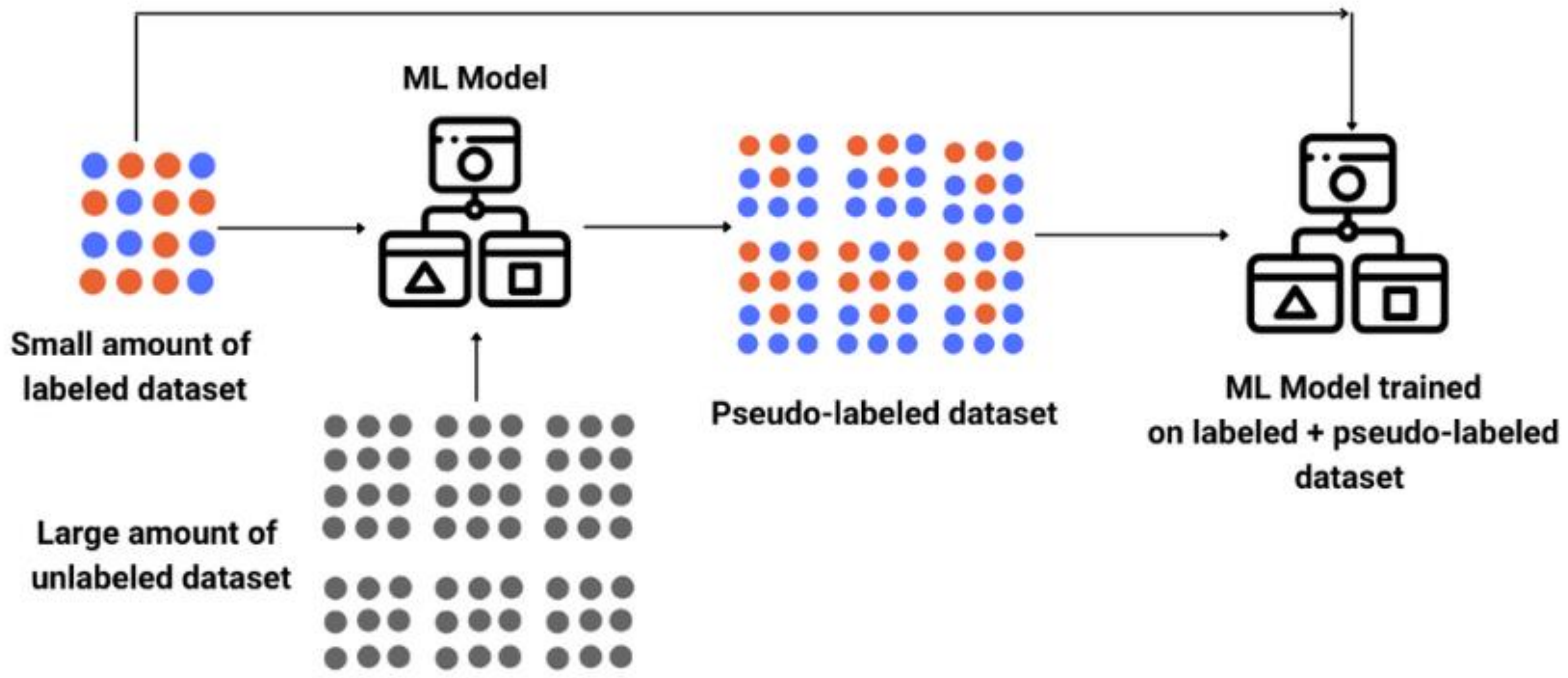
Model Development Steps

1. Development Approach
2. Data Loading
3. Training
4. Pseudo-Labeling
5. Semi-Supervised Learning

Model Re-trained with Entire Dataset

Semi-supervised Learning

- With labels and pseudo-labels generated for all slices in dataset, EfficientNetV2 network was **re-trained**
  - **Semi-supervised** learning: combination of supervised and unsupervised learning



- Network re-trained using n=2019 patients with all 700k+ image slices and slice-level labels/pseudo-labels
- Data split: 90% training, 10% validation, hidden public test set from Kaggle (n=420 patients)

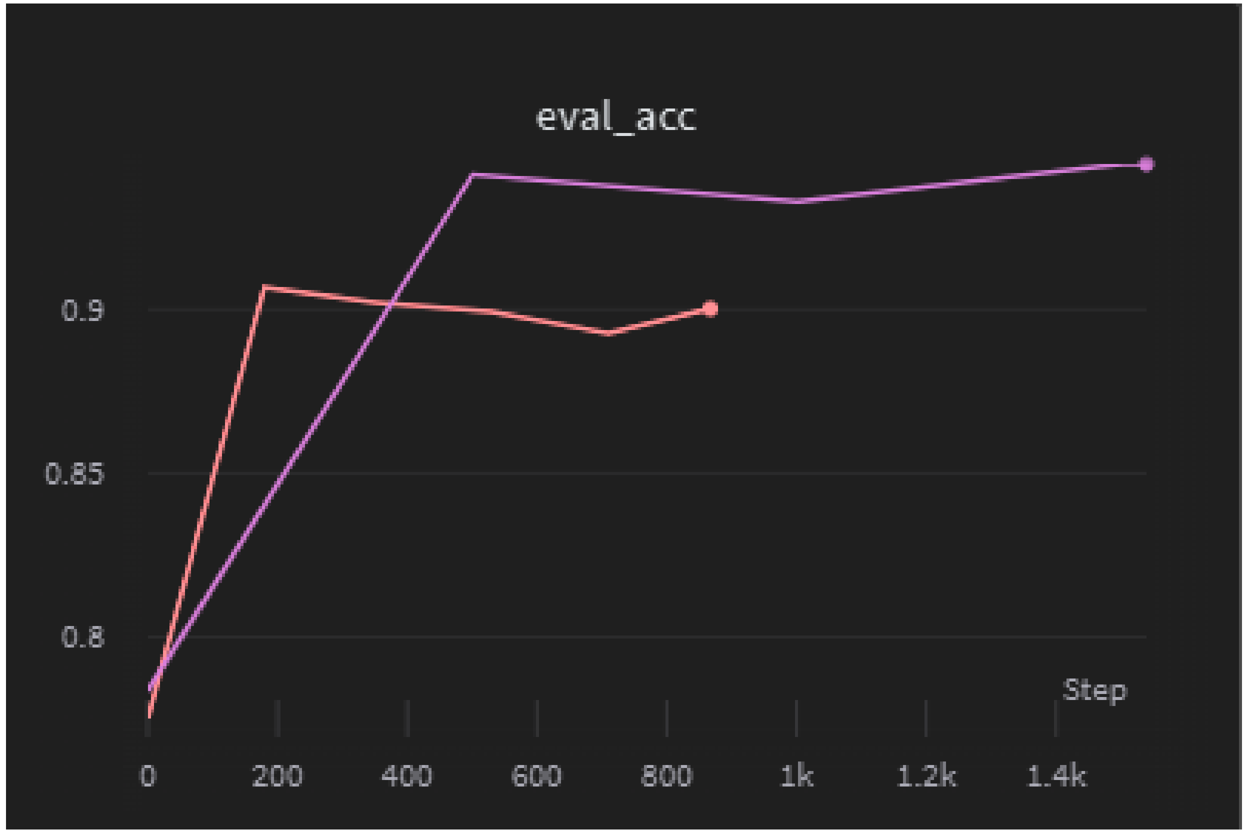
Model Re-trained & Evaluation

Fracture Detection

- Augmentations: dataset is large, so heavier augmentations were applied
  - N=6 augmentations were applied, most with 40% probability: random resized crop, vertical and horizontal flips, rotations, RGB shift, CLAHE
- Loss & metric: Weighted binary cross-entropy
- Additional metric: fracture accuracy
- Hyperparameters from earlier training worked well
  - Model: EfficientNetV2
  - Optimizer: Adam
  - Learning rate: 1E-3
  - Loss: weighted BCE
  - Regularization: 20% dropout
- **Final score:** 0.45 public and 0.50 private
  - Top 7% submission

Results: 94% evaluation accuracy for fracture detection

+4% increase with semi-supervised learning



Model Development Steps

1. Development Approach
2. Data Loading
3. Training
4. Pseudo-Labeling
5. Model Tuning

Conclusion

Summary

- Results were quite predictive, with 94% evaluation accuracy and 0.50 weighted BCE for hidden test set
- Potential Improvements
  - Calculate sensitivity and specificity, as well as accuracy, to aid in error analysis
  - Model more robust at predicting non-fractures than at predicting fractures
    - Improved use of oversampling, or implementing focal loss, may help mitigate accuracy differences between classes, which would be key for a clinical product
  - Ensemble modeling
- Constraints:
  - Time: started three month-long competition with one month left
  - Limited computational power prevented pursuing additional hyperparameter tuning
  - Kaggle weekly GPU quota led to highest scoring model submission coming in slightly after competition deadline

Discussion & Conclusion

- During competition, Kaggle discussion topics and notebooks were helpful for initial data exploration and model development
- Post-competition, notebooks of highest scoring submissions were helpful for lessons learned
- Highest scoring notebooks used various approaches:
  - Semantic segmentation: train model based on limited segmentation data and leverage localization information
    - 2D: with 2D masks generated, train object detector to determine bounding boxes for all vertebrae
    - 3D: train 3D U-Net to create masks & bounding boxes for all vertebrae
  - Utilize contextual information
    - Similar to my experience, most 3D CNN classification models were not very successful
    - 2.5D approach was helpful: each 2D input slice to model also had information from adjacent slices
  - Sequential models (RNN, LSTM, transformer) were used to learn features of entire vertebrae
  - Heavy augmentations: Mixup, random brightness were particularly helpful
- Results suggest that deep learning fracture detection models have the potential to be used as powerful tools to aid clinicians in rapidly diagnosing the location of cervical spinal fractures
- In the future, models may be helpful in prioritizing positive CT scans for reviews in high volume hospitals and in underserviced areas