CERVICAL SPINE FRACTURE DETECTION VIA COMPUTED TOMOGRAPHY SCANS

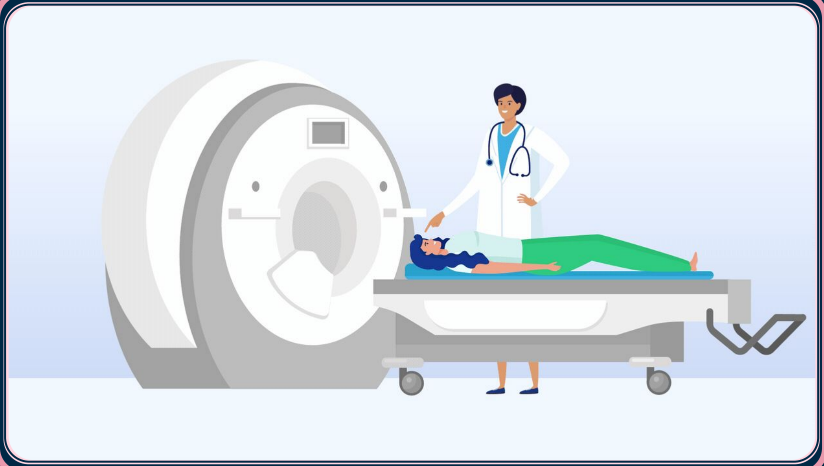
Tran Duc Tuan
Nguyen Trong Hieu
Le Quang Hung

TABLE OF CONTENTS

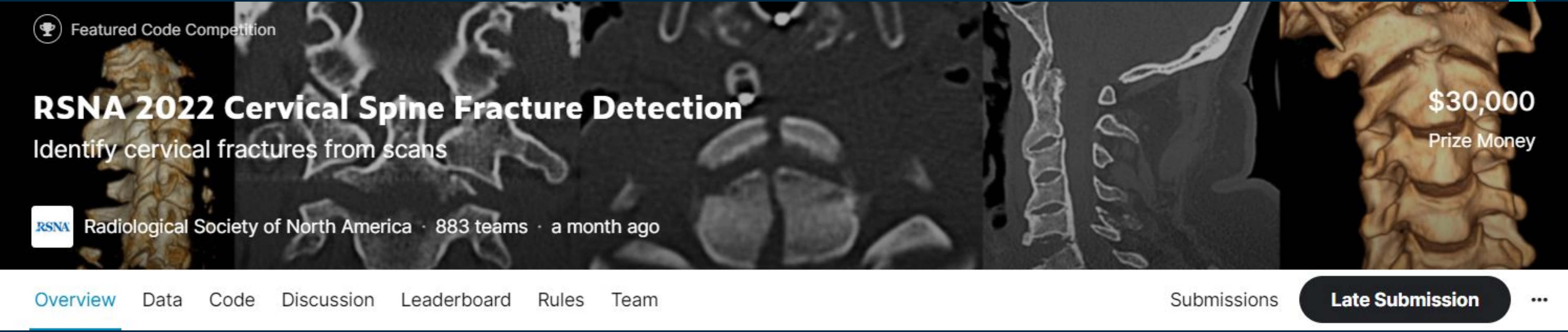
INTRODUCTION 01

PROBLEM & MOTIVATION

- There have been over 1.5 million cases suffered from spine fractures annually in the United States alone.
- The early detection and localization of spine fractures can play an essential role in preventing neurologic deterioration and paralysis after trauma.
- It often requires computed tomography (CT) to be performed instead of radiographs (x-rays), which might be more time-consuming and require specialists or experts to carefully examine patients’ spine



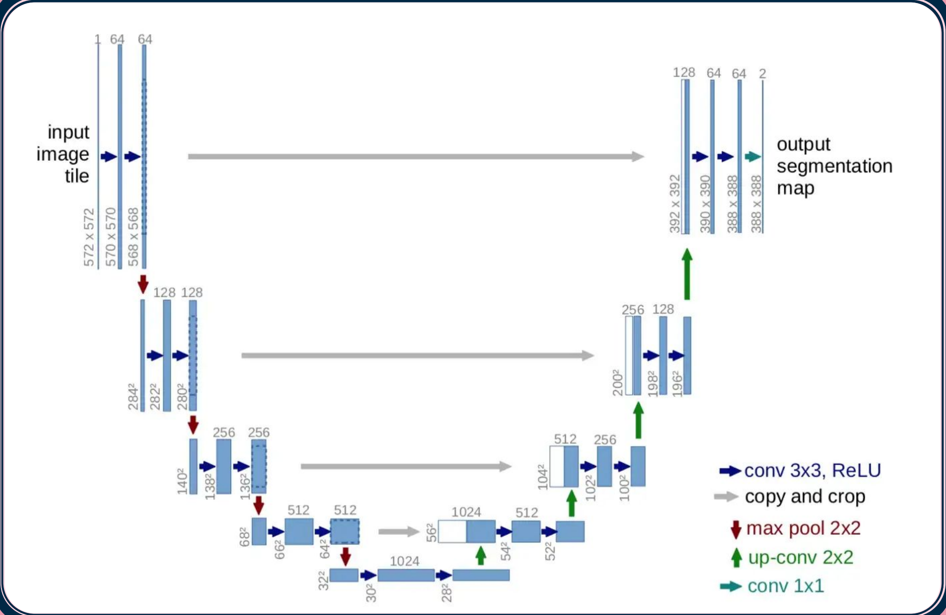
PROBLEM & MOTIVATION



- A competition on Kaggle, namely RSNA 2022 Cervical Spine Fracture Detection, was held to find the best AI-based method to support the early detection and localization of cervical spine fracture.

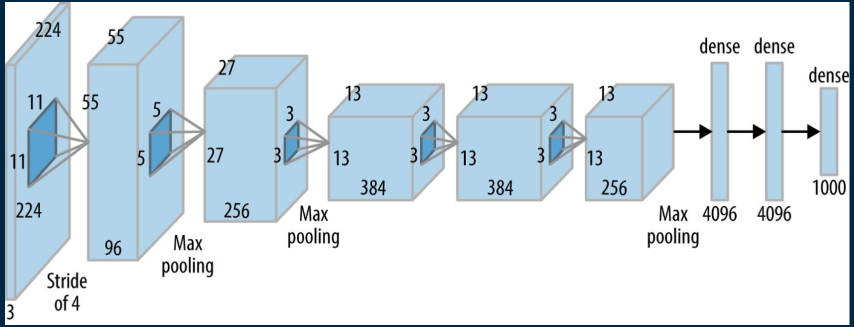
RELATED WORKS

- U-Net was first proposed as an deep learning approach for medical image segmentation, which is the task of classifying each pixel in an image.

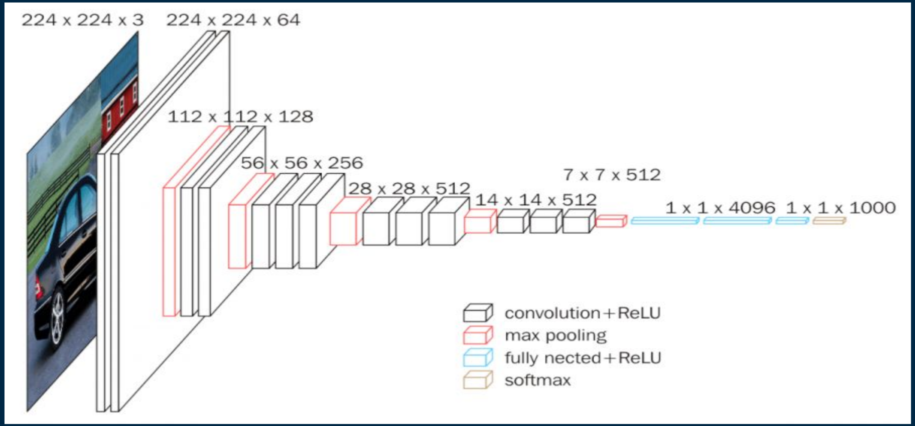


RELATED WORKS

- Convolutional Neural Networks (CNNs) serves as backbone in a variety of computer vision tasks such as image classification, detection, segmentation, etc.



Alex-Net architecture

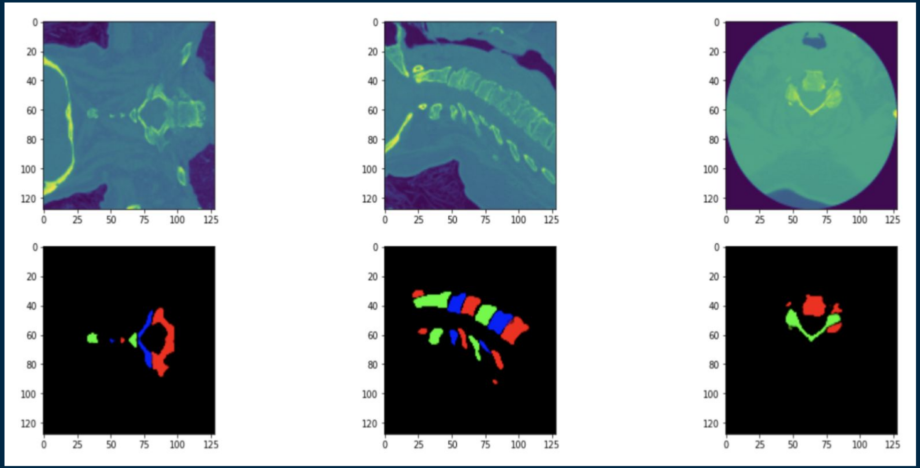


VGG16 architecture

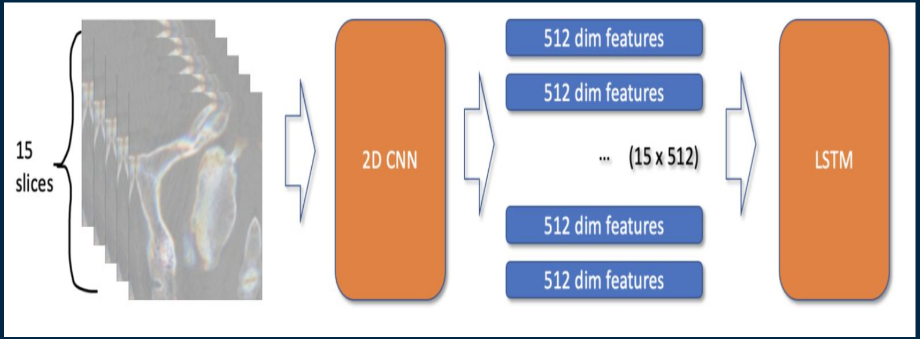
RELATED WORKS

Top 1 solution from the Kaggle Contest

Stage 1: 3D Semantic Segmentation

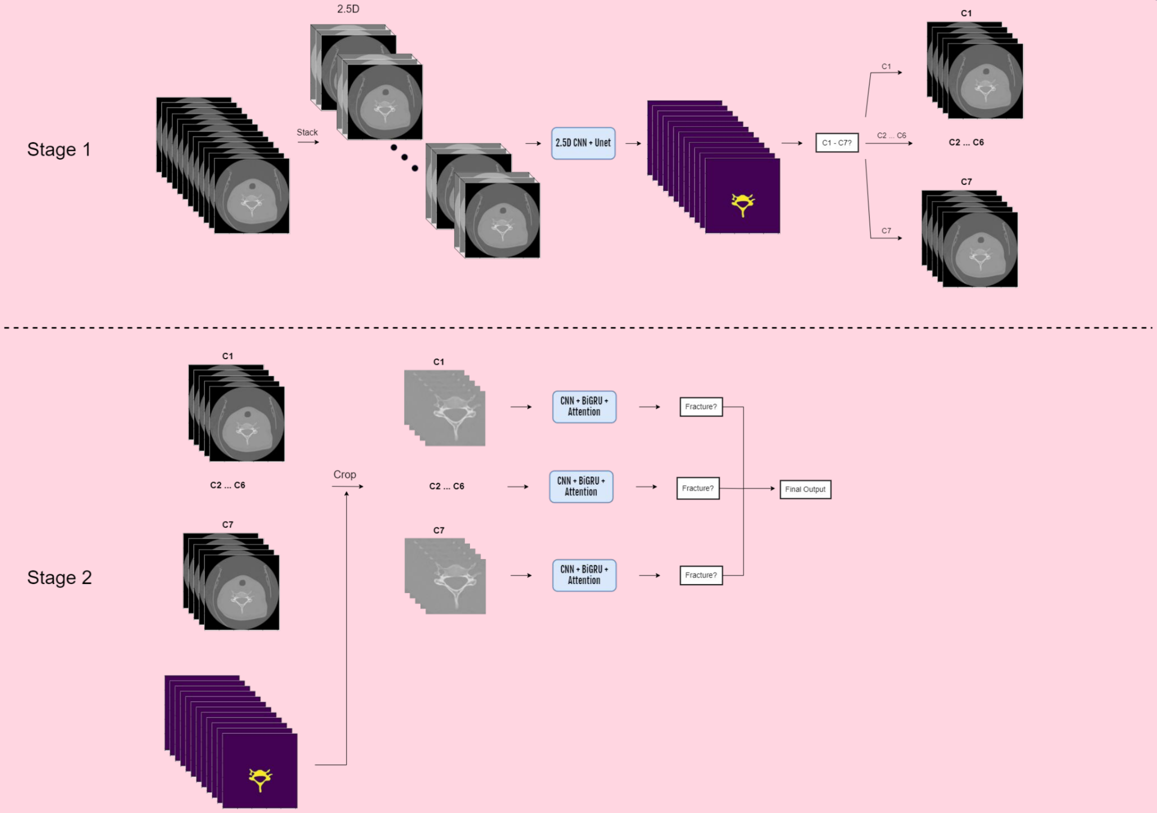


Stage 2: 5D + LSTM Classification



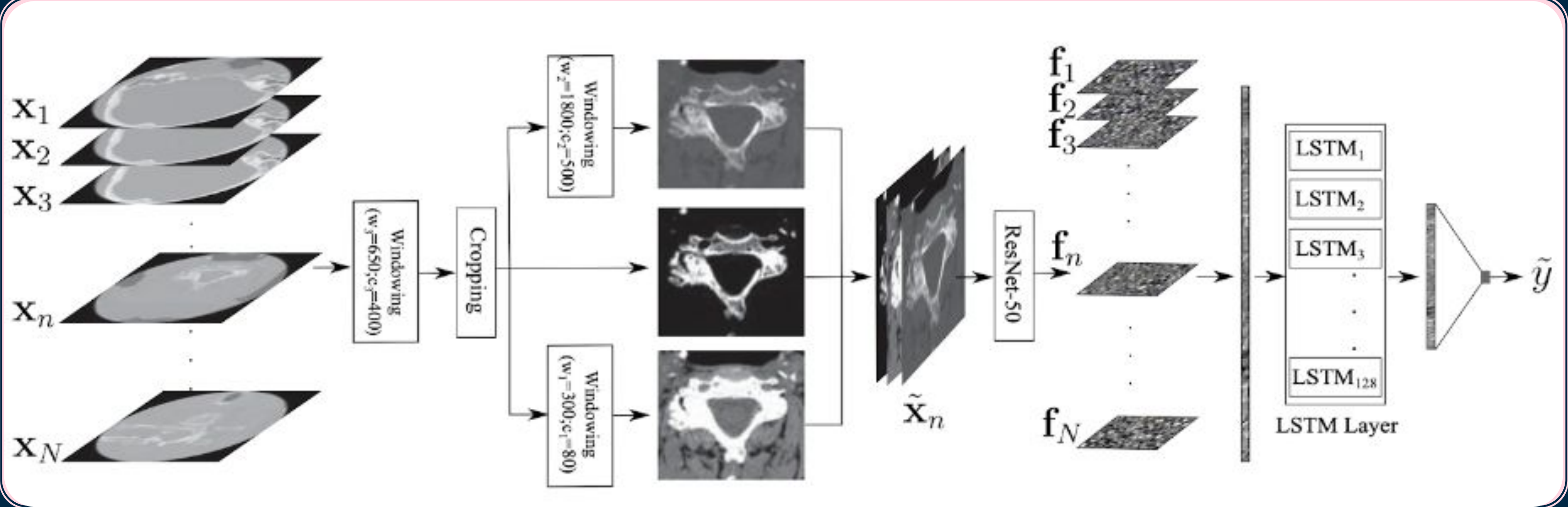
RELATED WORKS

Top 2 solution from the Kaggle Contest



RELATED WORKS

Deep convolutional neural network (DCNN) with a bidirectional long-short term memory (BiLSTM) for cervical spine fracture detection.



CONTRIBUTIONS

- Experimented two approaches to the mentioned problem, which are 3D classification and 2D classification.
- Find out how to use the data provided by the contest organizer effectively with each approach.
- Propose an architecture which is not time-consuming, resource-consuming.

DATASET

02

DATASET STRUCTURE:

```
rsna-2022-cervical-spine-fracture-detection
├── segmentations
│   ├── 1.2.826.0.1.3680043.10633.nii
│   ├── 1.2.826.0.1.3680043.10921.nii
│   └── ...
├── train_images
│   ├── 1.2.826.0.1.3680043.10001
│   │   ├── 1.dcm
│   │   ├── 2.dcm
│   │   └── ...
│   └── ...
├── test_images
│   ├── 1.2.826.0.1.3680043.22327
│   │   ├── 1.dcm
│   │   ├── 2.dcm
│   │   └── ...
│   └── ...
├── train.csv
└── train_bounding_boxes.csv
```

All segmentations are in NIFTI format

All images are in DICOM format

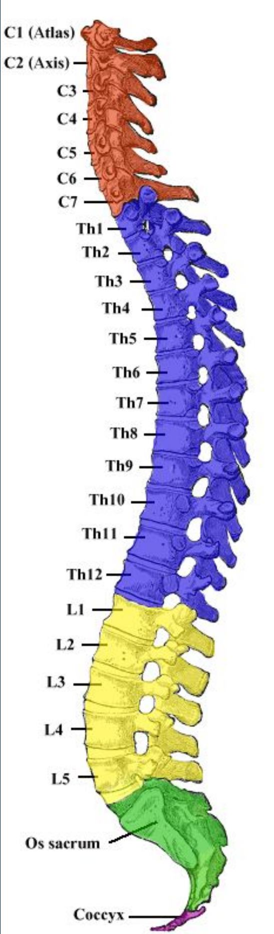
Stores fracture labels for each patient

Provides bounding boxes information on (i.e. fracture location)

DATASET DISCOVERY & EXPLANATION

- Labels for training images (in *train.csv* file):
  - ➢ patient_overall
  - ➢ C1 - C7

	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7
0	1.2.826.0.1.3680043.6200	1	1	1	0	0	0	0	0
1	1.2.826.0.1.3680043.27262	1	0	1	0	0	0	0	0
2	1.2.826.0.1.3680043.21561	1	0	1	0	0	0	0	0
3	1.2.826.0.1.3680043.12351	0	0	0	0	0	0	0	0
4	1.2.826.0.1.3680043.1363	1	0	0	0	0	1	0	0

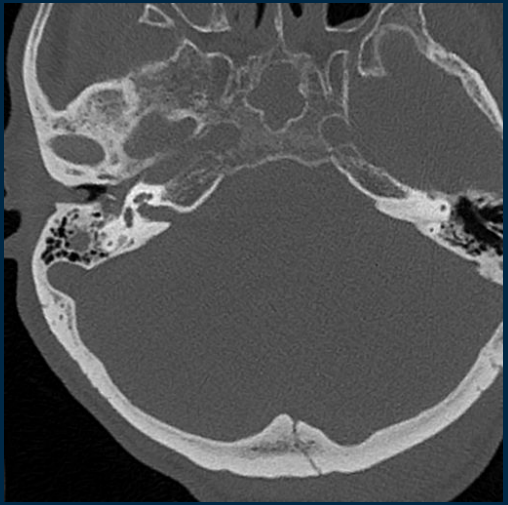


Dataset	Size	#vertebrae	#masks	#training studies	#testing studies
RSNA	512 — 768	7	87	2019	1080

Dataset overview

DATASET DISCOVERY & EXPLANATION

- DICOM files are loaded via *pydicom* library in Python.



A sample image loaded from a DICOM file

DATASET DISCOVERY & EXPLANATION

- Problem: Impossible to directly check whether a slice image is corresponding to which bone. => **Segmentation** is come to action.
- 87 segmentation masks, and those masks are in NIFTI format.

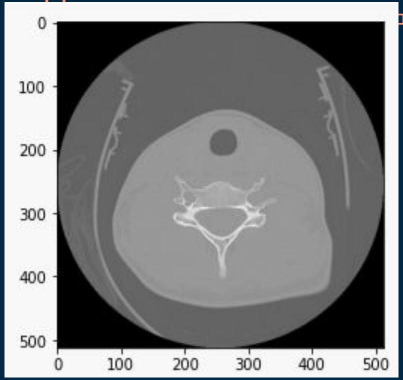
  => Loaded via *nibabel* - a Python library
- Segmentation masks loaded are in 3D format.

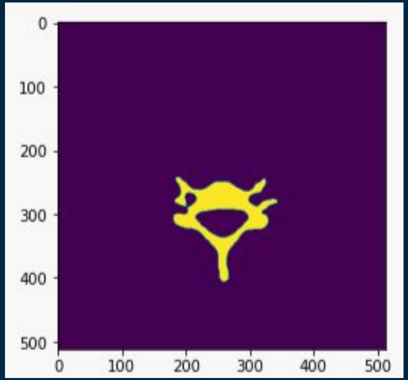
DATASET DISCOVERY & EXPLANATION

- In segmentation masks, unique values would indicate which bone for each slice.
- Unique values for this example: [0, 6]

  0: Background

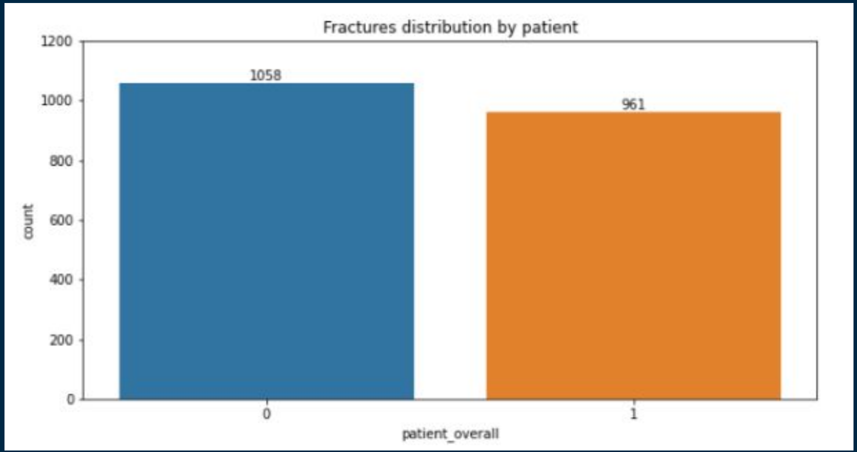
  6: Bone C6



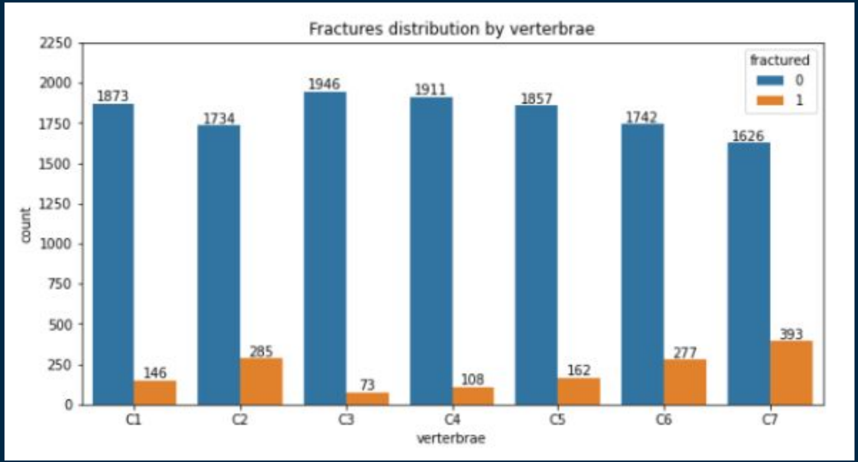


Up: Sample image
Down: Corresponding segmentation

EXPLORATORY DATA ANALYSIS (EDA)

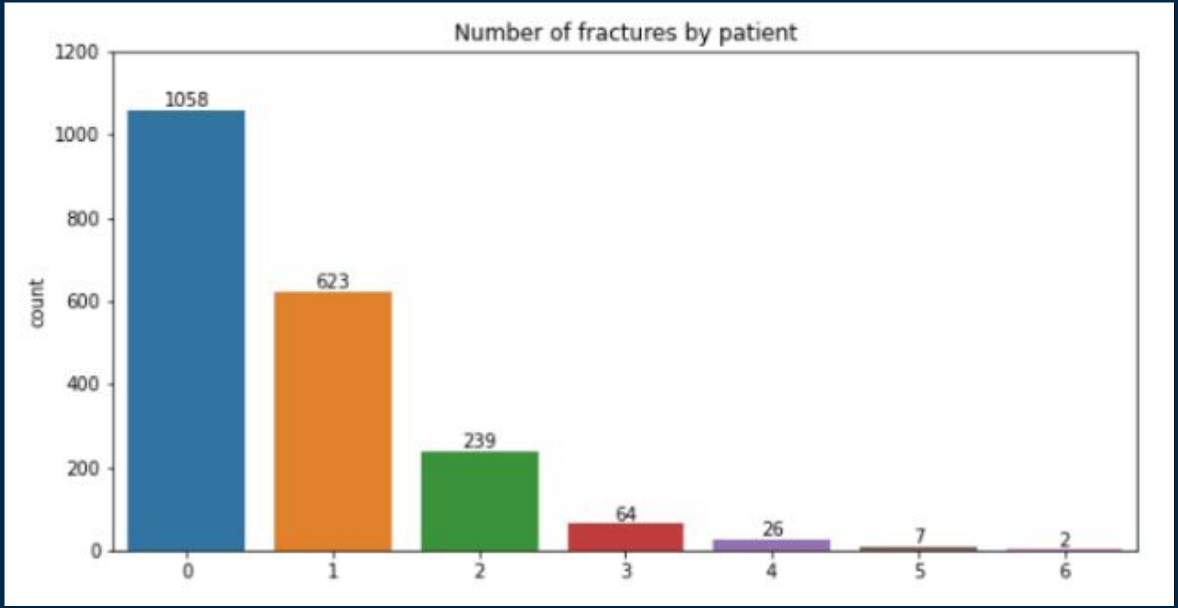


Data distribution (overall)



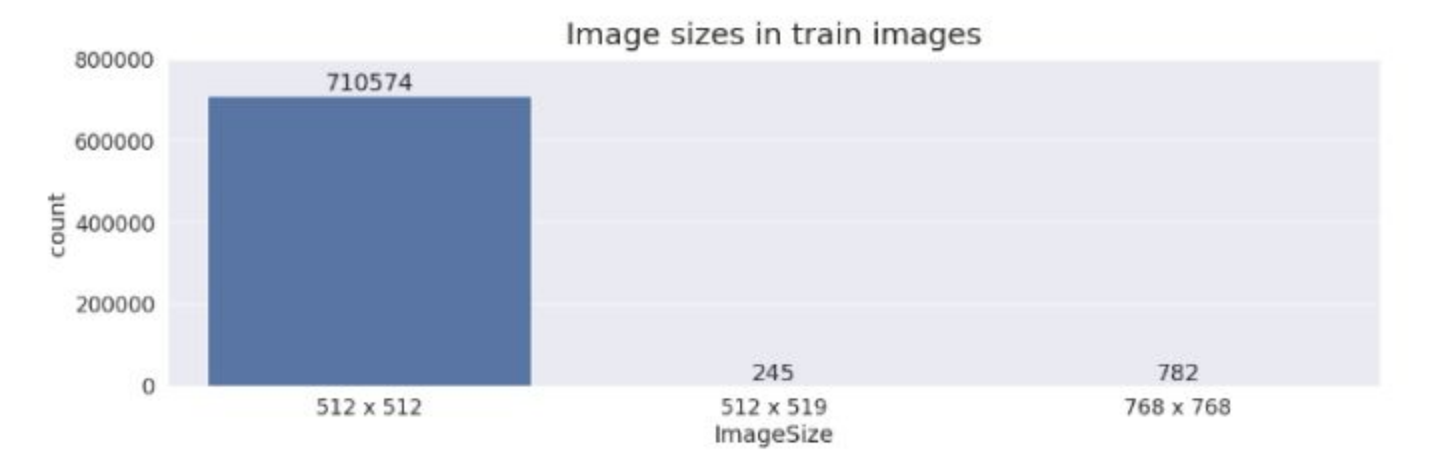
Data distribution (by vertebrae)

EXPLORATORY DATA ANALYSIS (EDA)



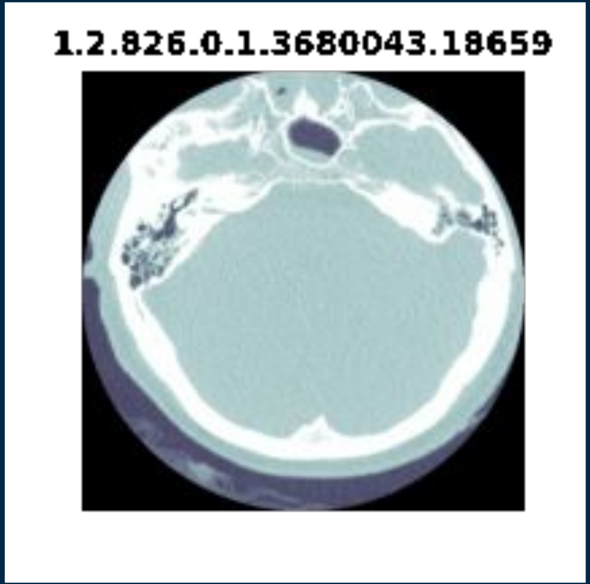
Number of fractures distribution

EXPLORATORY DATA ANALYSIS (EDA)



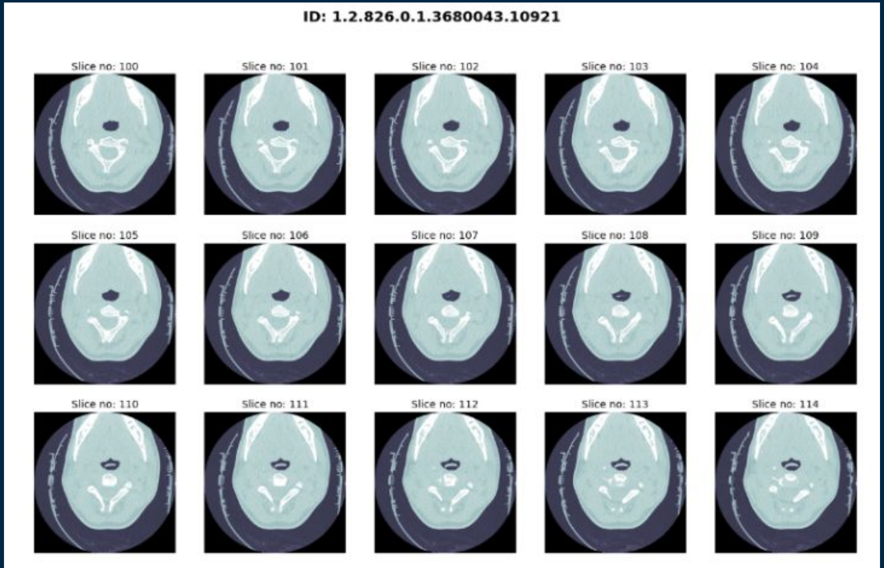
Number of fractures distribution

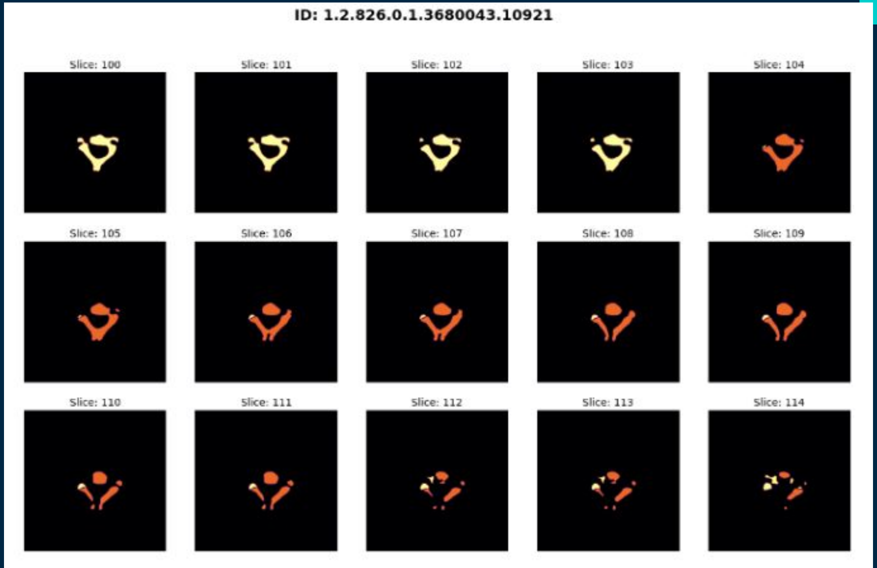
EXPLORATORY DATA ANALYSIS (EDA)



Example of all slice images of a case study

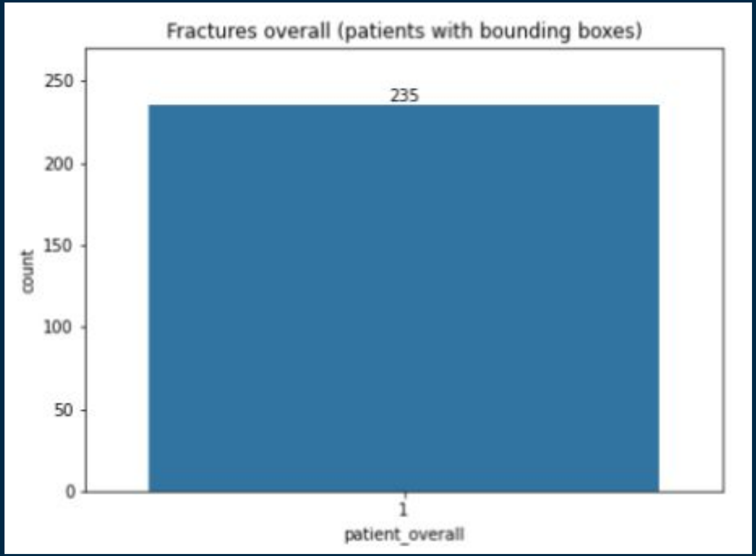
EXPLORATORY DATA ANALYSIS (EDA)



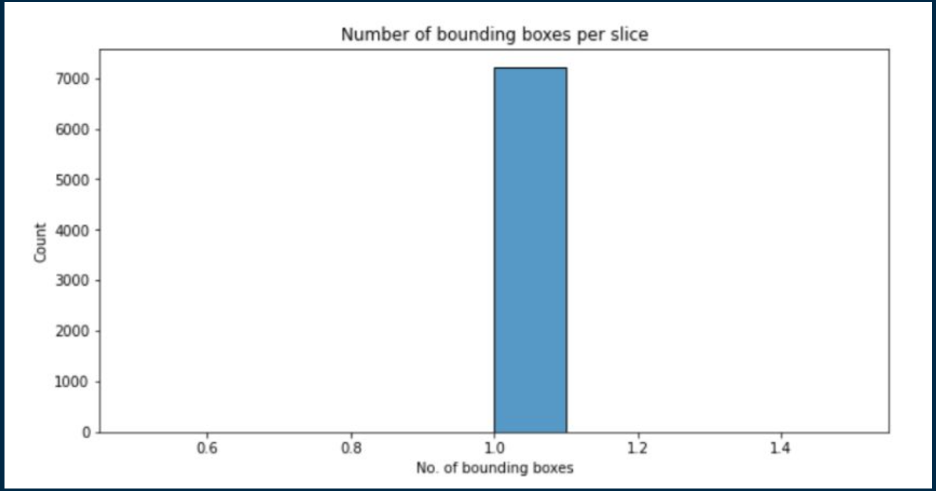


Example of 15 slice images (left) and their corresponding segmentation masks (right)

EXPLORATORY DATA ANALYSIS (EDA)

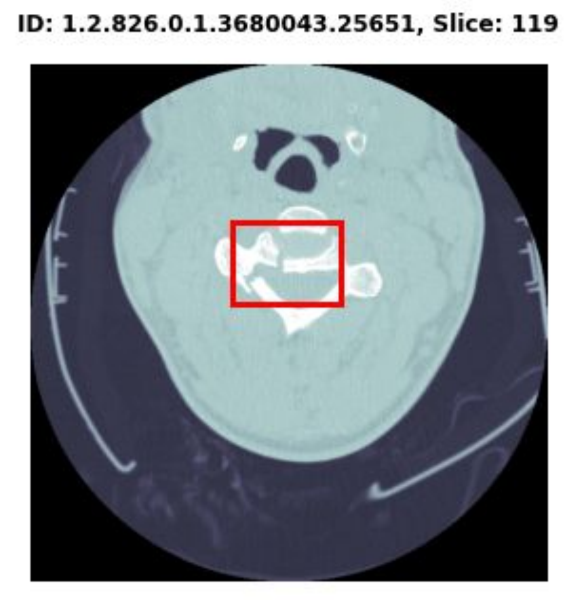


Number of fractures distribution (with cases with bounding boxes)



Number of bounding boxes per slice distribution

EXPLORATORY DATA ANALYSIS (EDA)



Example of a slice with bounding box

IMPLEMENTATION 03

EVALUATION METRIC

- Evaluated using a weighted multi-label logarithmic loss.
- The model is expected to predict the fracture probability of each vertebra (C1 - C7), as well as for the overall of the patient (patient_overall).
- Binary weighted log loss function for label j on exam i:

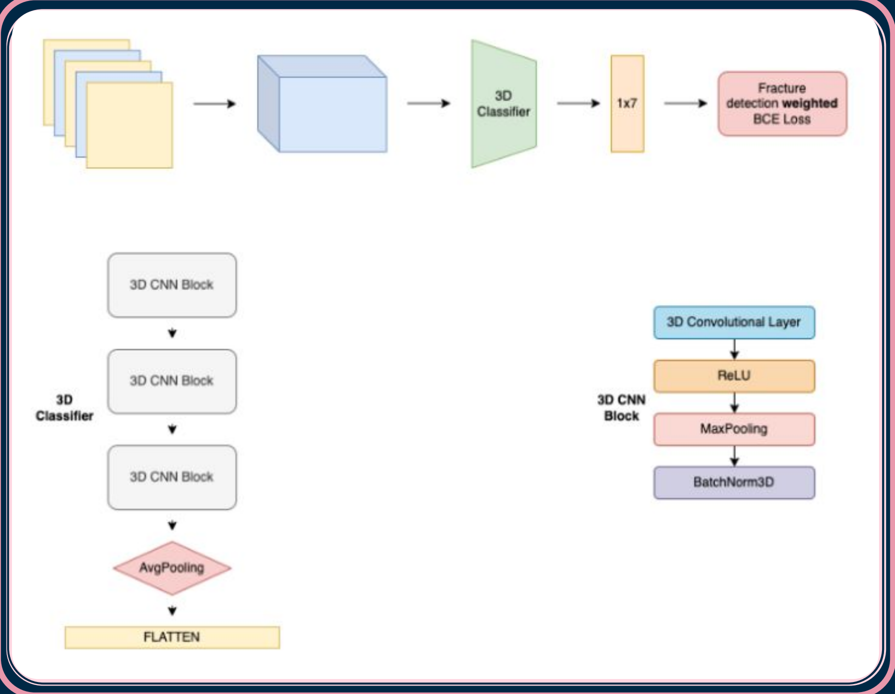
$$L_{ij} = w_{ij} \times [y_{ij} \times \log(p_{ij}) + (1 - y_{ij}) \times \log(1 - p_{ij})]$$

where the weights are:

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

DATA PROCESSING

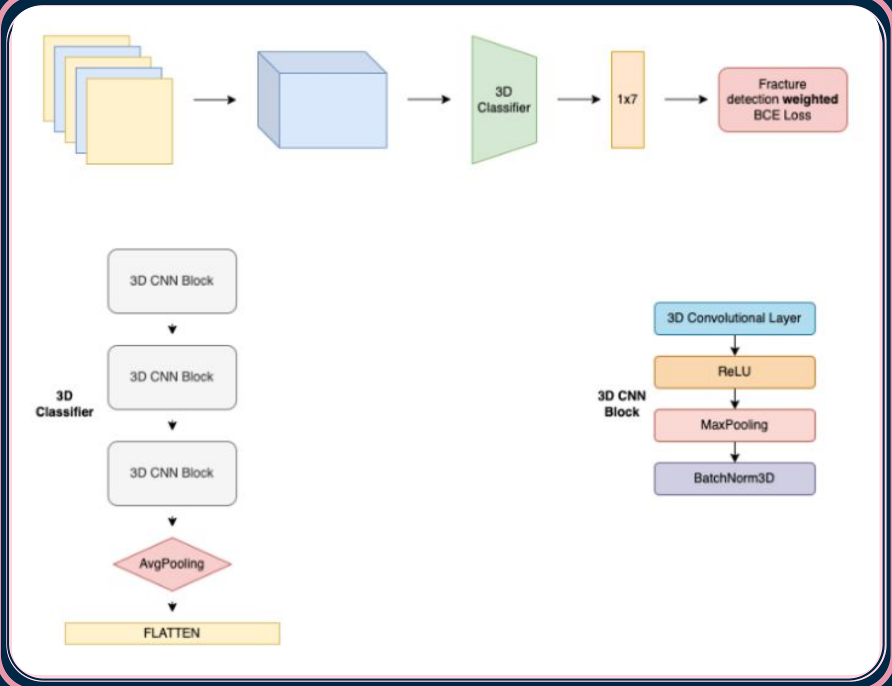
- Download images from directory provided by the contest and remove corrupted ones.
- Normalize and resize the data.
- For 3D data:
  - ➢ Stack 2D images to get 3D input.
  - ➢ Use Random Rotation and Random Horizontal Flip for augmentation.
  - ➢ Size 224x224x224.
- For 2D data: size 3x224x224.



3D CNN Classifier

3D CLASSIFICATION MODEL

- Pass 3D data through a model of 3 3D convolution blocks to obtain the feature map.
- The feature map would be passed through several Fully Connected layers to get the final output.
- Architecture of the convolution block:
  - ➢ A convolution layer
  - ➢ An activation layer
  - ➢ A pooling layer
  - ➢ A normalization layer
- The output of this model has 8 dimensions: 7 vertebrae (C1 - C7), patient_overall
- Use *AdamW* as optimizer and *CosineAnnealingLR* as scheduler.

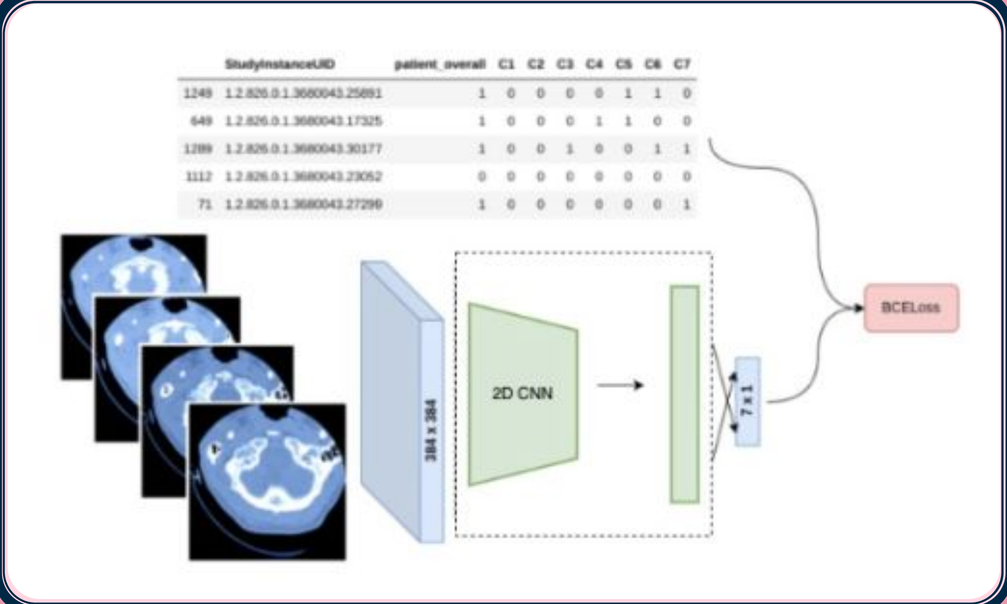


3D CNN Classifier

2D CLASSIFICATION

1. Single-head Model:

- First trained a CNN model (with ConvNeXt-Tiny as the backbone) with vertebrae labels extracted from segmentation mask provided by organizers (87 cases, a slice belongs to a class if >=1 pixel of that slice classified to that class)
- Trained via 5-fold cross-validation to get 5 models.
- Then, we inferred all training data (2019 cases) and average predictions of models, in order to get the pseudo vertebrae labels for the next model.

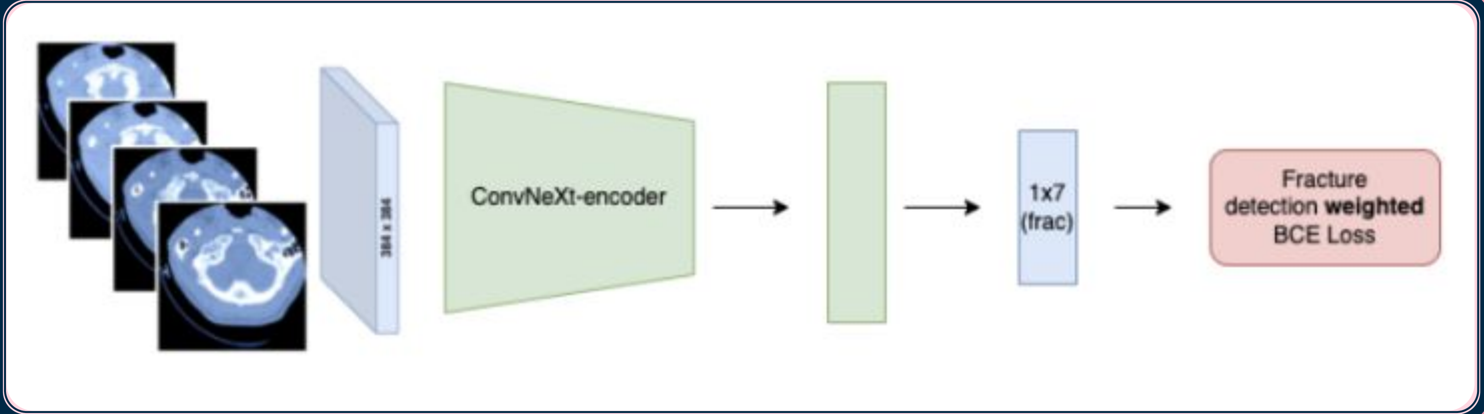


CNN model for vertebrae classification

2D CLASSIFICATION

1. Single-head Model:

- Passed training data with pseudo-labels through an another CNN model with ConvNeXt-Tiny as the backbone.
- Used multilabel loss function Binary Cross Entropy Loss with Logits from *pytorch* library.



Single-head approach for Cervical Spine Fracture Detection

2D CLASSIFICATION

1. Single-head Model:

- In the end: get a model detects fractures and visible C1 - C7 vertebrae using a single image.
- For each case study:
  - ➢ Aggregate prediction for each vertebra (C1 - C7).
  - ➢ Calculate patient_overall probability using the equation:

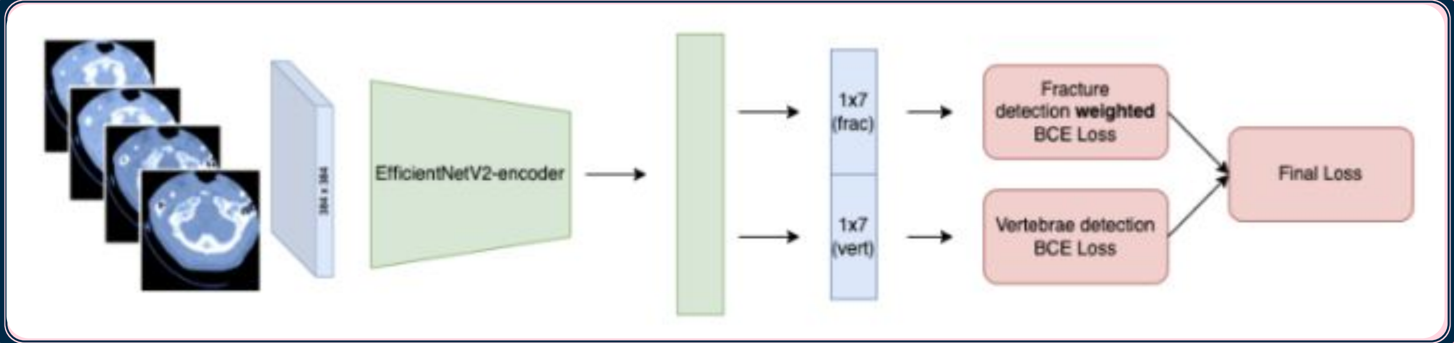
$$P_{\text{patient\_overall}} = \max_{i=1}^{7} P_{C_i}$$

- We split data into 5 folds using GroupKFold with “StudyInstanceUID” as group to avoid data leakage and trained 5 versions to get the ensemble model.

2D CLASSIFICATION

2. Multi-head Model:

- First, trained CNN models for vertebrae classification (as in Single-head approach).
- Data was passed to the pretrained encoder of EfficientNetV2S.
- The output of the encoder was flattened and put through 2 Fully Connected layers in parallel, to optimize 2 loss functions simultaneously.



Multi-head approach for Cervical Spine Fracture Detection

2D CLASSIFICATION

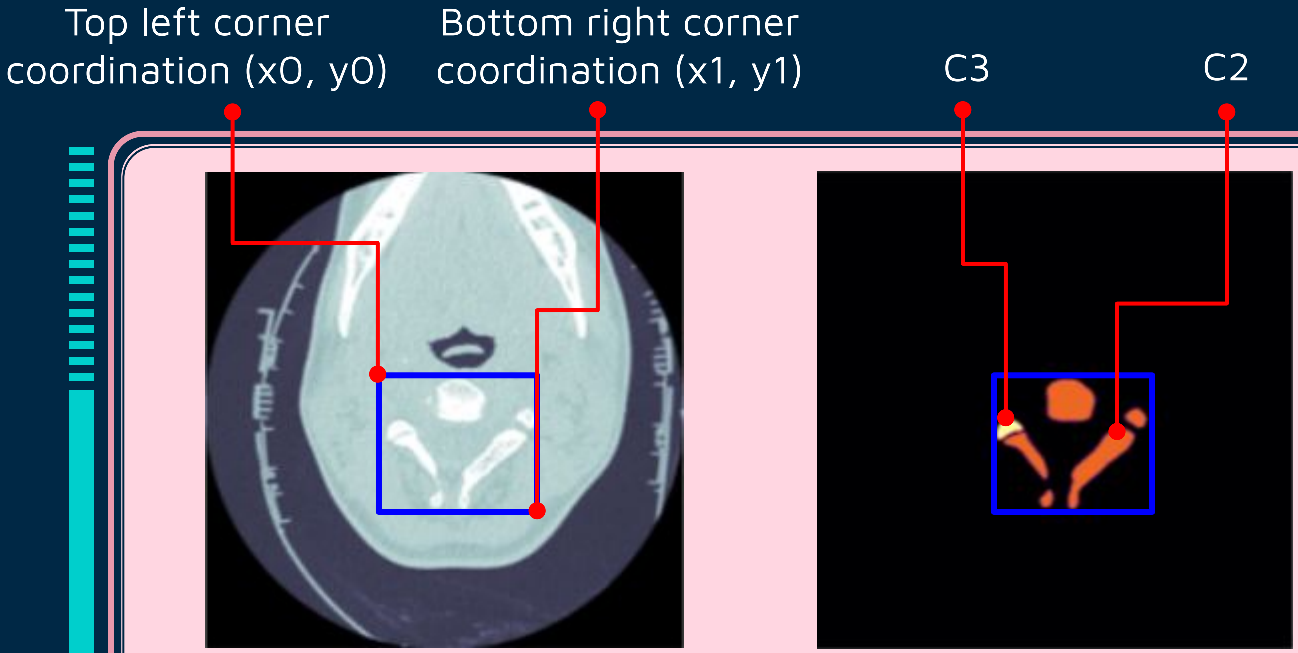
2. Multi-head Model:

- In the end: get a model classify fractures on a single image.
- The final predictions for each case study was obtained in the same way as in the previous approach.
- Unlike in Single-head approach, the patient_overall probability is calculated as:

$$P_{\text{patient\_overall}} = 1 - \prod_i (1 - P_{C_i})$$

- Splitted data into 5 folds, trained 5 versions to get the ensemble model.

PROPOSED METHOD (DATA PREPARATION)

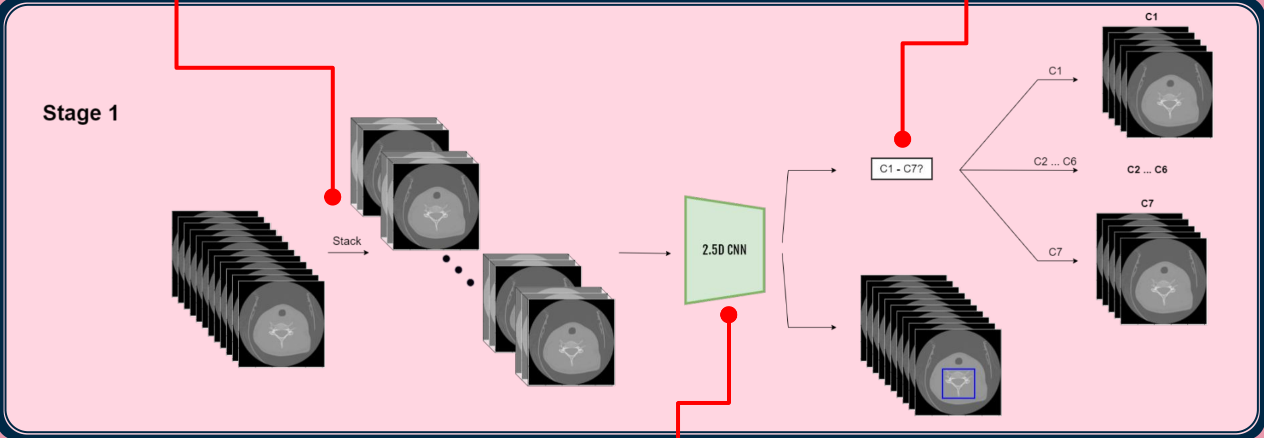


Available for 87/2019 patients only

PROPOSED METHOD (STAGE 1)

Stack 3 grayscale images to 1 RGB images (2.5D)

Choose from lists of slices to get 24 images using evenly spaced indices for each type of bone and each patient (Ex. 47 slices → 24 slices with index 0,2,4, …, 46)

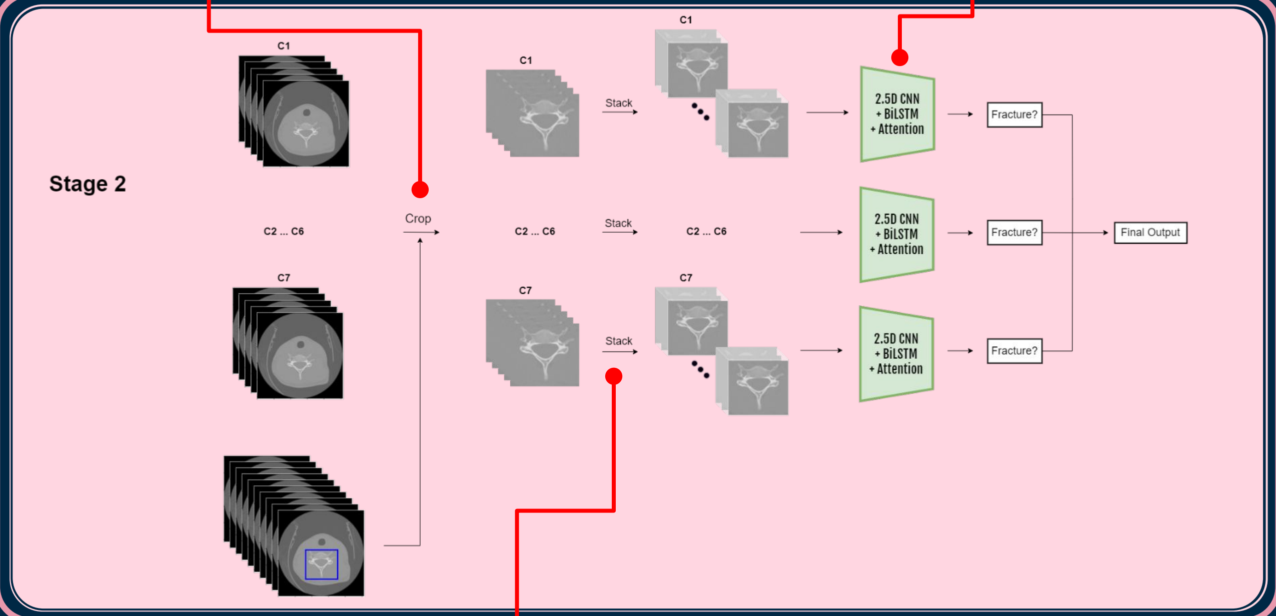


Train a CNN model to classify vertebrae and detect the bounding box of them on 87 cases with segmentation mask available, then infer on all 2019 cases

PROPOSED METHOD (STAGE 2)

Crop each study's cervical vertebrae using its bounding box

Train a CNN model with BiLSTM and Attention Layer for fracture detection



From the 24 chosen images each cervical vertebrae, stack each 3 images to 2.5D, stack all inputs to sequence of 8 2.5D images

PROPOSED METHOD (STAGE 2)

PATIENT OVERALL PREDICTION:

- The patient_overall is calculated as:

$$P_{\text{patient\_overall}} = 1 - \prod_{k=1}^{N}(1 - P_{C_k})$$

where

- N is the top N highest probability vertebrae
- N = 1: $P_{\text{patient\_overall}} = \max(P_{Ck})$
- P_{Ck} is the probability of vertebrae C_k

PROPOSED METHOD (CROSS VALIDATION)

$$P_{\text{patient\_overall}} = 1 - \prod_{k=1}^{N} (1 - P_{C_k})$$

Cross validation using competition metric:

N	Score	N	Score
1	0.4228	5	0.3691
2	0.3862	6	0.3676
3	0.3762	**7**	**0.3668**
4	0.3716		

RESULTS

04

RESULTS

Results comparison of 4 aforementioned methods

Model	Score
3D CNN	0.6048
Single-head	0.5813
Multi-head	0.5019
Our method	**0.3691**

RESULTS

Results comparison between our model and Kaggle top-2 solution. Inference time is calculated on full 2019 studies training data.

Model	Score	Time (h)
Top-2 method	**0.2389**	4.55
Our method	0.3691	**3.67**

FUTURE WORKS

- Experimenting with Transformer layers instead of LSTM.
- Training other backbone models.
- Trying models with bigger image size and longer sequence length

Do you have any questions?

THANKS