FPT Education

FPT UNIVERSITY

UNDERGRADUATE THESIS

Cervical Spine Fracture Detection via Computed Tomography scan

Author:

Tran Duc Tuan

Le Quang Hung

Nguyen Trong Hieu

Supervisor:

Dr. Phan Duy Hung

A thesis submitted in fulfillment of the requirements

for the degree of Bachelor of Artificial Intelligence

in the

Department of Information Technology

December 21, 2022

FPT UNIVERSITY

Abstract

Department of Information Technology

Bachelor of Artificial Intelligence

Cervical Spine Fracture Detection via Computed Tomography scan

The application of artificial intelligence in image processing and decision support in the medical field has received increasing attention recently in the community. In this work, we did experiments with multiple machine learning models to find the one that matches radiologists' performance in detecting and locating fractures on the seven vertebrae of the cervical spine via a Computed Tomography scan. Among our experiments, the model, which consists of two stages using deep convolutional networks with RNN and Attention layers to classify whether a patient has a cervical spine fracture, achieved the highest performance.

Keywords: Convolutional Neural Network, CNN, Attention, cervical spine fracture classification.

Acknowledgements

We want to express our gratitude to Dr. Phan Duy Hung, our lecturer, for his tolerance, his time, and his enthusiastic instruction and guidance. Second, we would like to thank FPT, our university, for providing us with a favorable atmosphere in which to learn and develop over the years. Finally, we will never forget the encouragement and support of our family. They inspire us to get better every day, thus we want to express our sincere gratitude to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem & Motivation

Patients with spine fractures often have a lot of difficulties in moving their body, which prevent them from working and daily routines. The cause of spine fractures can be due to accidents or old age. According to [1], there have been over 1.5 million cases suffered from spine fractures annually in the United States alone, leading to about 18000 spinal cord injuries, and these cases are usually seen in elderly people. The early detection and localization of spine fractures can play an essential role in preventing neurologic deterioration and paralysis after trauma. However, 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.

Recently, more and more AI-based technologies have emerged to automate various tasks that often require human intelligence to perform. To deal with images, these technologies often use deep learning methods, which perform quite well compared to traditional ones and sometimes even better than what humans could do. However, these methods are not always giving accurate results every time they are in use, which is why there have been many competitions revolving around them in order to find the most accurate method while still achieving the maximum time to be performed. As a result, a competition on Kaggle, namely RSNA 2022 Cervical Spine Fracture Detection [1], was held to find the best AI-based method to support the early detection and localization of cervical spine fracture, which is the most common site of spine fracture.

1.2 Related Works

U-Net [22] was first proposed as an deep learning approach for medical image segmentation, which is the task of classifying each pixel in an image. It outperforms the sliding-window convolutional network which is the prior best method in terms of both score and speed, and from then it becomes a popular approach for image segmentation in general. Along with the strong use of data augmentation, U-Net can produce fine segmentation results while training on a few images and it not only works well with 2D images but also with 3D ones.

CNN serves as backbone in a variety of computer vision tasks such as image classification, detection, segmentation, etc. It has existed from more than decades, starting from the introduction of AlexNet [15] to the more effective architectures namely ResNet [11], EfficientNet [29], ConvNeXt [17], etc. These architectures are well-known in the computer vision community for achieving a lot of great success in terms of both accuracy and speed.

In the specific problem that is cervical spine fracture detection, there is some previous research on deep learning models introduced. In [24], they proposed a deep convolutional neural network (DCNN) with a bidirectional long-short term memory (BLSTM) layer for the automated detection of cervical spine fractures in CT axial images. Besides, another work introduced a 3D convolutional sequence to sequence model for vertebral compression fractures identification in CT in [3]

Within the contest, there are several solutions from several top teams of [1] which can detect fractures in cervical spines quite effectively. Most of the top teams in the competition use an architecture that includes at least two models: a segmentation model and a classification model. Qishen Ha in [8] developed a 2-stage method for fracture detection. This method first trained an U-Net model with resnet18d or efficientnet-v2s for 3D semantic segmentation to generate 3D masks for all training data, then a 2D CNN (ConvNeXt) model followed by a LSTM module was trained for final classification. Similarly, Harshit Sheoran's method [9] consists of 2 stages, in which U-Net models were trained for both sagittal and bone segmentation, and EfficientNet CNN with RNN model was trained for classification afterwards. In the classification stage, images were put into 2.5D format, which is concatenating three consecutive slices into a single image, and two bidirectional GRU layers with attention and Conv1D layer were used for RNN model. [23] used an U-Net model for 2.5D segmentation and a CNN with bidirectional GRU layers and attention was trained for classification. In the second stage, a SpatialDropout layer was added to the model; therefore, this gave a slight improvement in the overall classification. However, the architectures mentioned above all take a long time or a large amount of resources for training (or also pretrain). Therefore, we did experiments to find a model that is more timely efficient and still be able to achieve acceptable results.

1.3 Objectives

In this study, we aim to contribute practical experiments to developing an automated tool that can help doctors make quick and accurate decisions, bringing the most benefit to patients and society. We experimented with two approaches to the above-mentioned problem, which are 3D classification and 2D classification. In the 2D approach, we implement two famous backbone models, ConvNeXt and EfficientNet, with different training methods. Finally, we utilized convolutional neural network (CNN) model and data processing techniques to obtain a model which consists of 2 stages: detecting vertebrae bounding boxes and classifying whether a cervical spine was fractured. At both stages, we used 2.5D input by stacking three slices and ConvNeXt for the backbone model. The final model achieve an acceptable results and lower inferrence time with limited training resources. The final result is equivalent to the top 25 of the contest. Details of experiments, data processing, and parameters will be covered in chapter 4.

Chapter 2

Background

2.1 Image classification

Image classification is a fundamental task that attempts to comprehend an entire image as a whole. The objective is to label the image in order to categorize it.

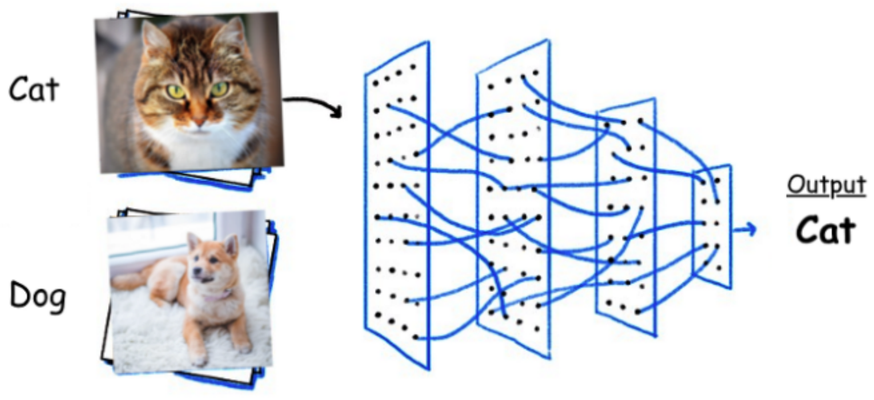


FIGURE 2.1: Image Classification

A convolutional neural network (CNN) is a commonly used paradigm in image processing, which is the concept behind recent developments in the field of computer vision. Among the different types of neural networks (others include recurrent neural networks (RNNs), long short-term memory (LSTMs), artificial neural networks (ANNs), and more), CNNs is the most common type. The most notable structures of a convolutional network in image processing are: Convolutional Layer, Pooling Layer, ...

2.1.1 Convolutional Layer

The foundation of a CNN is a convolutional layer. It has a number of filters (or kernels), whose settings must be learned over the course of training. Typically, the filters' size is smaller than the original image. Each filter produces an activation map after it convolves with the image.

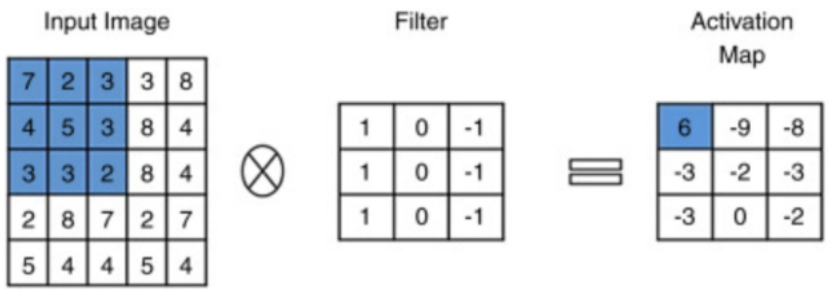


FIGURE 2.2: Convolutional Layer

Due to the local connectivity of the convolutional layer, the network is forced to learn filters with the highest response to a specific local region of the input. [18]

2.1.2 Pooling Layer

Pooling layers are used to decrease the dimensions of the feature maps. As a result, it lessens the quantity of network computation and the number of parameters that must be learned. The feature map created by a convolution layer's feature pooling layer summarizes the features that are present in a certain area. Therefore, instead of precisely positioned features produced by the convolution layer, further operations are conducted on summarized features. As a result, the model is more resistant to changes in the features' positions in the input image.

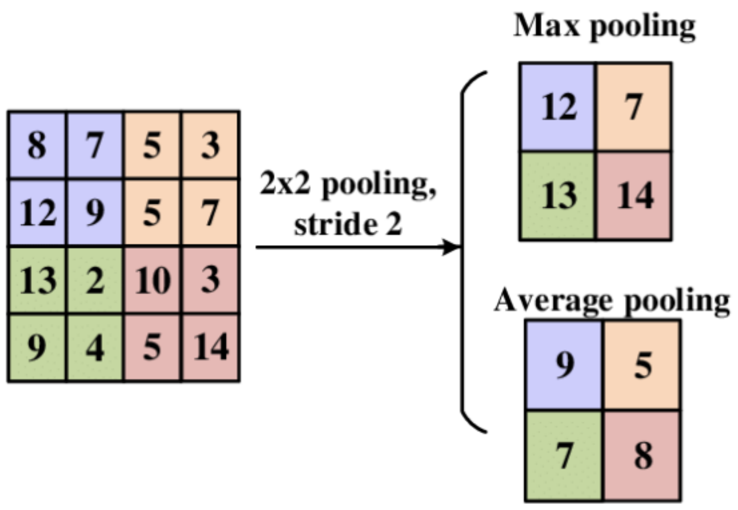


FIGURE 2.3: Pooling Layer

2.2 3D Image Classification

Deep learning models are being used more frequently in the medical field as a result of the quick development of machine learning, graphics processing technology, and the accessibility of medical imaging data. Using 3D deep learning, it is now possible to analyze three-dimensional (3D) medical pictures like CT, DTI, fMRI, ultrasound, and MRI scans thanks to falling computational costs and the availability of powerful graphics processing units (GPUs). These scans provide in-depth, three-dimensional images of human organs and can be used to look for anomalies in blood vessels and organs as well as infections, malignancies, and traumatic injuries. In this work, we also experiment with 3D classification to classify whether a vertebra was fractured or not. [26]

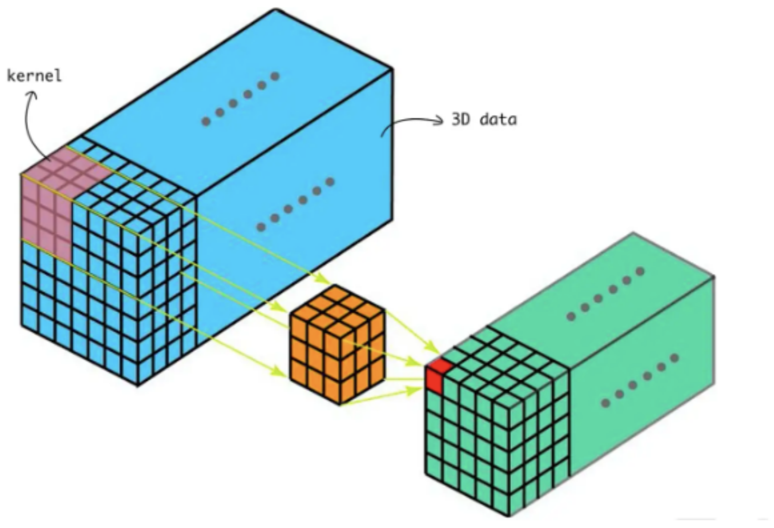


FIGURE 2.4: 3D Convolutional Layer

2.2.1 3D Convolution Layer and 3D Pooling Layer

The convolution layer is one of the key distinctions between 3D and 2D image classification. Difference between 2D and 3D convolutions applied on a set of frames. 2D convolutions use the same weights for the whole depth of the stack of frames (multiple channels) and result in a single image. 3D convolutions use 3D filters and produce a 3D volume as a result of the convolution, thus preserving temporal information of the frame stack.

Another distinguishing feature between the two approaches is the pooling layer. Similar to the convolution layer, the 3D pooling layer also uses a 3D filter when implementing the pooling algorithm.

2.3 Image Classification Models

2.3.1 EfficientNet

EfficientNet is a convolutional neural network design and scaling technique that uses a compound coefficient to consistently scale all depth, breadth, and

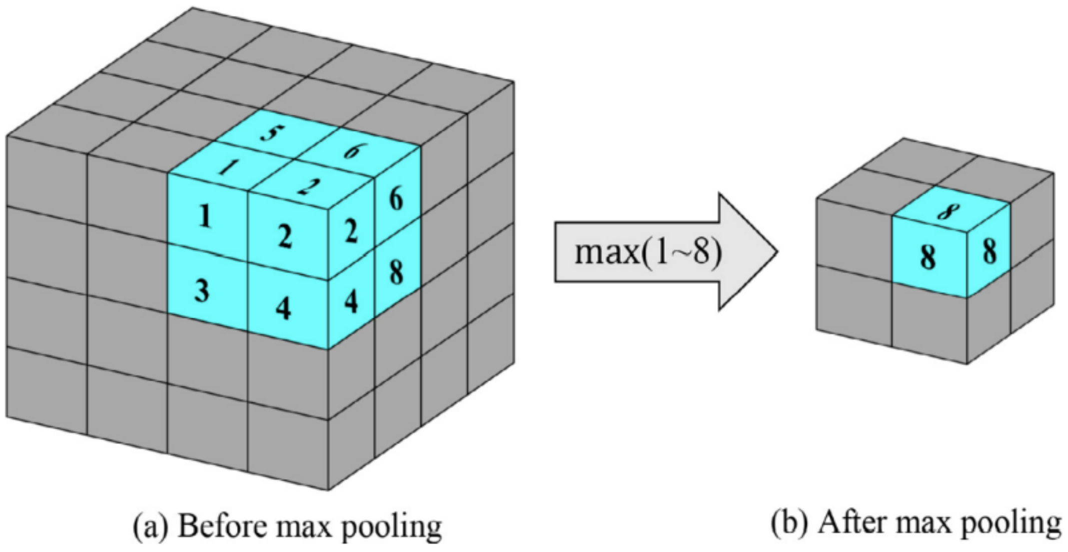


FIGURE 2.5: 3D Pooling Layer

resolution dimensions. The EfficientNet scaling method uniformly scales network width, depth, and resolution using a set of fixed scaling coefficients, in contrast to standard practice, which scales these variables arbitrarily. Not only did it achieve 84.4% top-1 accuracy in the ImageNet classification task but it did it with many times less parameters than the earlier state-of-the-art models. [28]

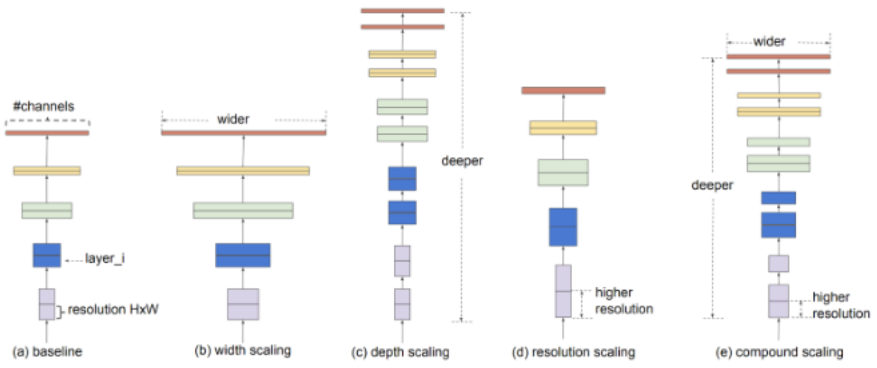


FIGURE 2.6: Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) EfficentNet's method that uniformly scales all three dimensions with a fixed ratio.[28]

Stage i	**Operator $\hat{\mathcal{F}}_i$**	**Resolution $\hat{H}_i \times \hat{W}_i$**	**#Channels $\hat{C}_i$**	**#Layers $\hat{L}_i$**
1	Conv3x3	224×224	32	1
2	MBConv1,k3x3	112×112	16	1
3	MBConv6,k3x3	112×112	24	2
4	MBConv6,k5x5	56×56	40	2
5	MBConv6,k3x3	28×28	80	3
6	MBConv6,k5x5	14×14	112	3
7	MBConv6,k5x5	14×14	192	4
8	MBConv6,k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

TABLE 2.1: EfficientNet-B0 baseline network

There are multiple sized EfficientNet, which are named from B0 to B7 in an ascending order based on the complexity of the network. EfcientNet-B0 is the baseline model that is then scaled to obtain EfficientNet B1-B7. The fundamental blocks of the EfficientNet-B0 network are the squeeze-and-excitation blocks and the inverted bottleneck residual blocks of MobileNetV2. [28]

The scaling method makes it possible to efficiently scale up a baseline ConvNet to any target resource restrictions in a more principled manner. With an order of magnitude less parameters and FLOPS, the mobile size EfficientNet model can be scaled up very effectively and surpass state-of-the-art accuracy due to this compound scaling strategy. [29]

2.3.2 EfficientNetV2

EfficientNetV2 is a type of convolutional neural network that has faster training speed and better parameter efficiency than the original EfficientNet.[28]

EfficientNetV2-S achieved 83.9% top-1 accuracy in the ImageNet classification task.[28]

Stage	Operator	Stride	#Channels	#Layers
1	Conv3x3	2	24	1
2	Fuse-MBConv1,k3x3	1	24	2
3	Fuse-MBConv4,k3x3	2	48	4
4	Fuse-MBConv4,k3x3	2	64	4
5	MBConv4,k3x3,SE0.25	2	128	6
6	MBConv6,k3x3,SE0.25	1	160	9
7	MBConv6,k3x3,SE0.25	2	256	15
8	Conv1x1 & Pooling & FC	-	1280	1

TABLE 2.2: EfficientNetV2-S architecture

2.3.3 ConvNeXt

With the development of Transformer architecture and Attention mechanism, image processing models also achieve a new step with the combination of Transformer with classic CNN architecture. One of the models created as a result of this pairing, ConvNeXt, has shown promise in image classification.

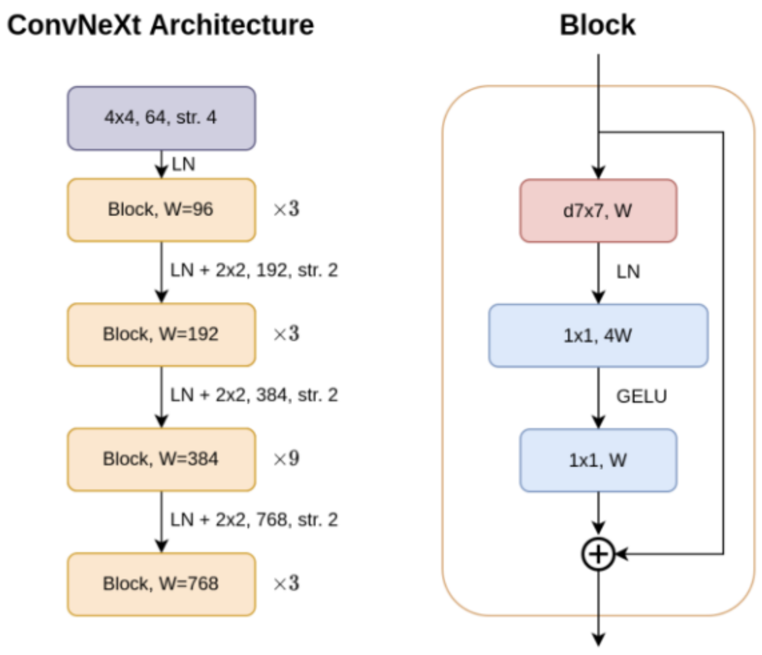


FIGURE 2.7: ConvNeXt Architecture

Constructed entirely from standard ConvNet modules, ConvNeXts compete favorably with Transformers in terms of accuracy and scalability, reaching 87.8% ImageNet top-1 accuracy while keeping the simplicity and effectiveness of standard ConvNets. [17]

2.4 Image Segmentation

2.4.1 3D vs 2D vs 2.5D segmentation

3D segmentation

3D image segmentation, according to [32], is a computational technique to predict the element-level labels of objects in a 3D volume. That means 3D segmentation is able to directly generate segmented masks for whole 3D volumes. Therefore, this technique usually applies for 3D volumes of data.

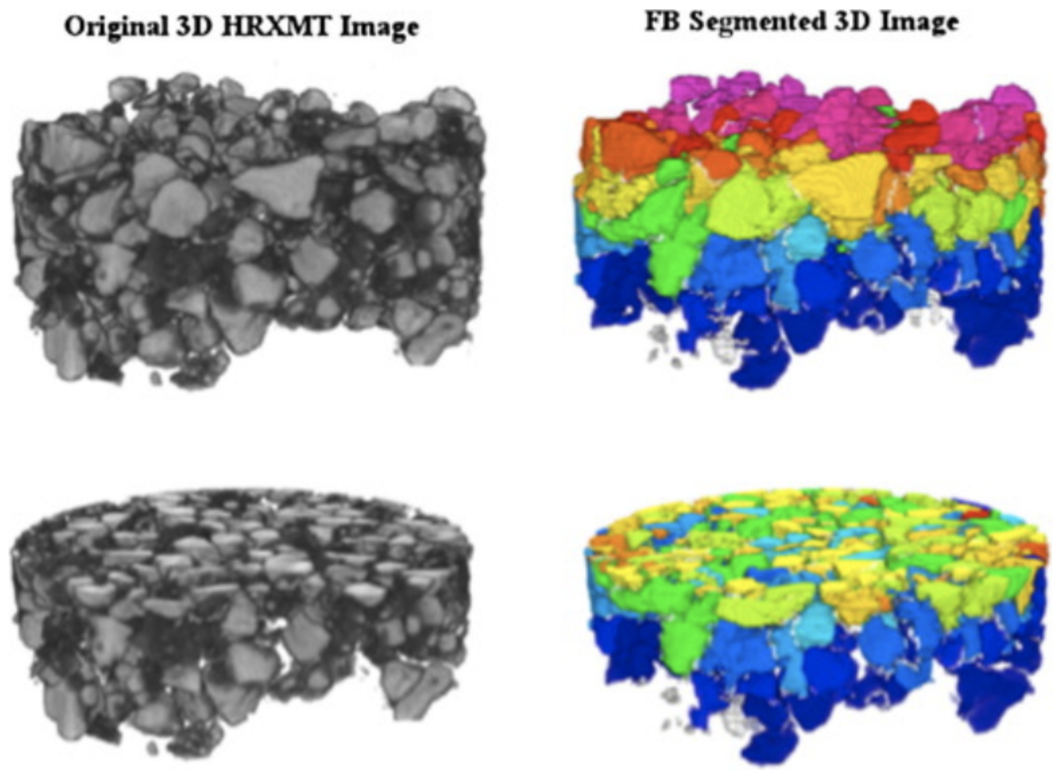


FIGURE 2.8: 3D image segmentation (source: [14])

2D segmentation

Similar to 3D segmentation, 2D segmentation is another type of image segmentation techniques, but applies for only 2D images, rather than 3D volumes. As a result, 2D segmented masks are commonly less comprehensively understandable than 3D masks, due to dimensional limitations.

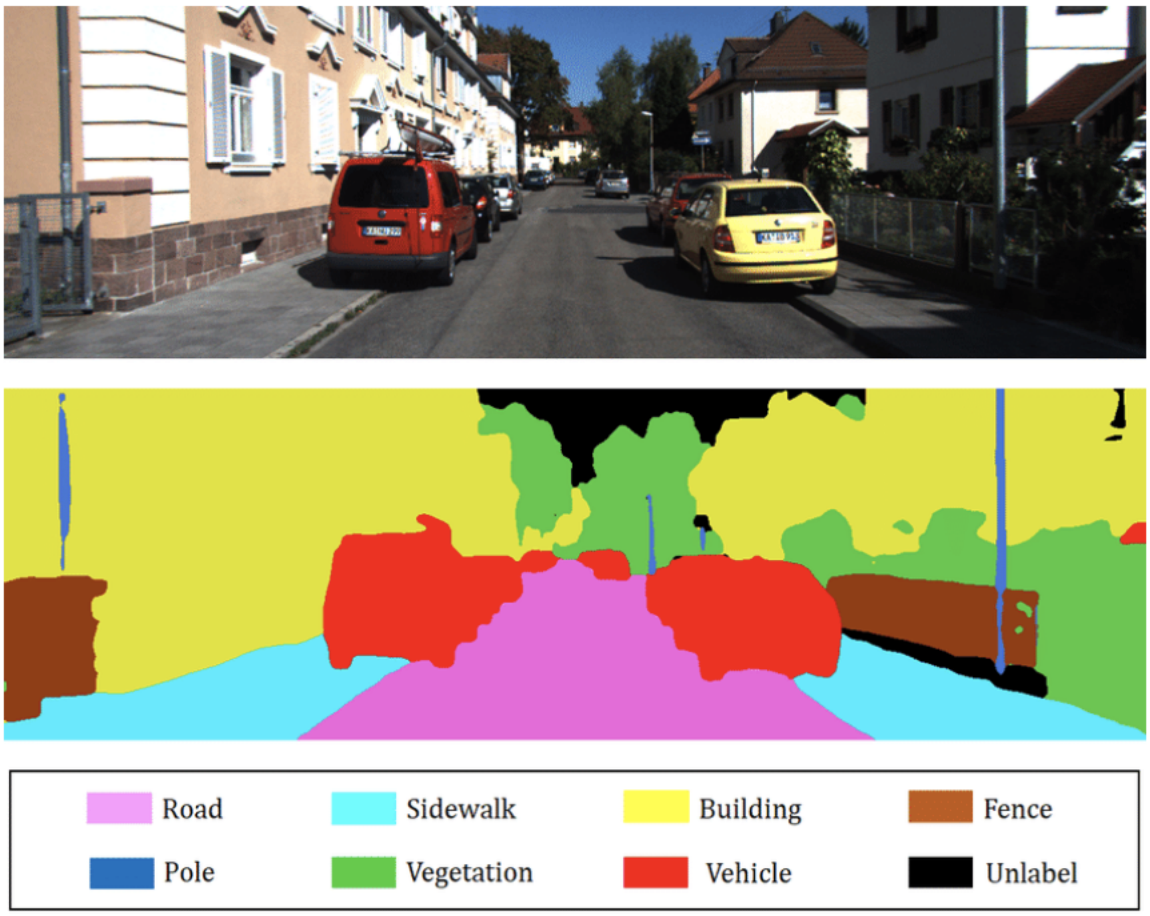


FIGURE 2.9: An example of 2D image segmentation (source: [33])

2.5D segmentation

This type of technique is considered in between 2D and 3D segmentation types. This technique is done by putting stacks of 2D images, which are usually slices from 3D volumes, into 2D models [2]. Therefore, this technique is able to overcome the dimensional limitations of 2D segmentation techniques, as well as reduce computational costs, which is one of the biggest disadvantages of 3D segmentation techniques.

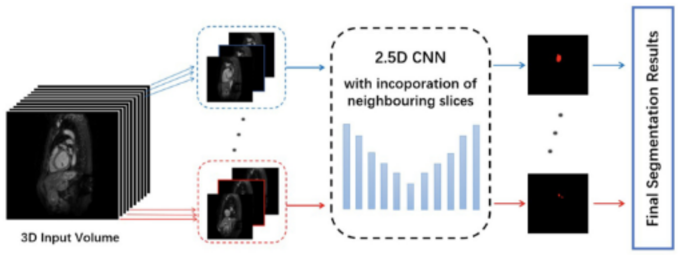

2.5D CNN
with incoporation of neighbouring slices
3D Input Volume
Final Segmentation Results

FIGURE 2.10: An example of 2.5D image segmentation (source: [33])

Differences between 2D, 3D and 2.5D segmentation techniques

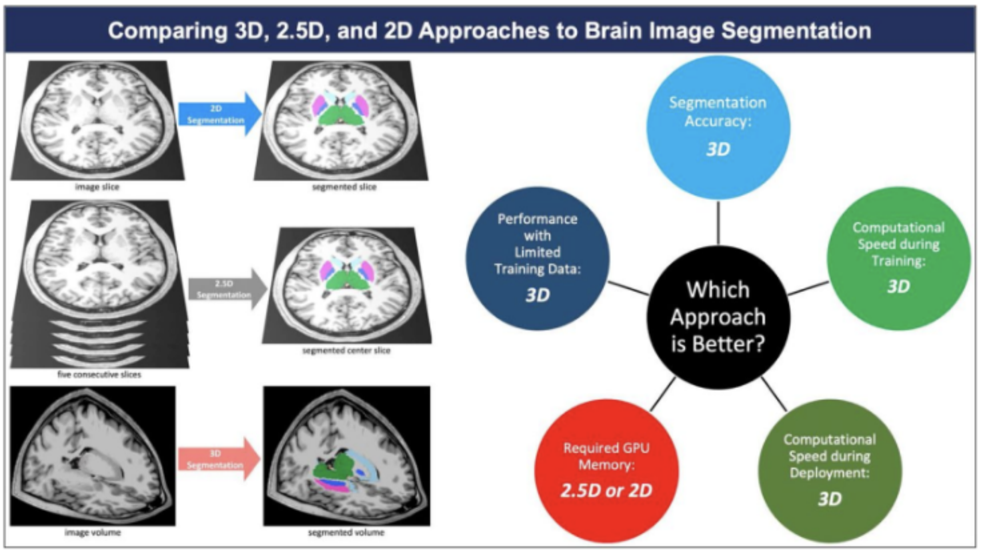

Comparing 3D, 2.5D, and 2D Approaches to Brain Image Segmentation
2D Segmentation
image slice
segmented slice
2.5D Segmentation
five consecutive slices
segmented center slice
3D Segmentation
image volume
segmented volume
Segmentation Accuracy: 3D
Performance with Limited Training Data: 3D
Computational Speed during Training: 3D
Which Approach is Better?
Required GPU Memory: 2.5D or 2D
Computational Speed during Deployment: 3D

FIGURE 2.11: Brief comparison between 2D, 2.5D and 3D segmentations

Figure 2.11 briefly describes the differences between 3 techniques. As 3D segmentation applies on whole 3D volumes, this technique is able to perform quite fast and accurately on training. However, this technique is extremely computational expensive and only suitable for highly computational resource conditions, as well as with limited training data given. In contrast, 2D and 2.5D segmentations could be less computational cost, as well as require less GPU memory and be compatible with larger training datasets. In addition, 2.5D segmentation could be better than 2D in several domains, such as in medical problems due to being more deeply informative in their masks. Last but not least, while 3D segmentation puts the whole 3D volume as input, 2.5D segmentation puts consecutive slices as input, and the consecutive slices, when stacked together, may result in a single 3D volume. Therefore, 2.5D segmentation techniques are able to perform efficiently with 3D data, despite a different way (as consecutive slices), but only require less amount of resources than 3D segmentation techniques (as they only use 2D CNNs rather than 3D ones).

2.5 Recurrent Neural Network

Machine learning focuses on creating algorithms that automatically improve by practice. The learning algorithm should ideally get better the more times it is used. The learning algorithm's job is to build a classifier function from the provided training data. The effectiveness of this built-in classifier is then evaluated by using data that had never been seen before. Artificial neural networks (ANN) are rough models of biological learning systems that draw inspiration from them. Neuronal networks in biological learning systems are intricate and linked. A vector of real-valued inputs and a single realvalued output are all that neurons are capable of processing. Feed-forward neural networks are the most prevalent type of conventional neural network. One input layer, one output layer, and at least one intermediate hidden layer are used to organize the sets of neurons in this system. Static classification tasks are the only ones that feed-forward neural networks can handle. They can only offer a static mapping between input and output as a result. Feedforward neural networks can be expanded to support dynamic classification.We must feed signals from earlier timesteps back into the network in order to gain this attribute. Recurrent Neural Networks are these networks with recurrent connections. [27]

Long short-term memory (LSTM)

Despite such advantages, the maximum amount of time that RNNs can look back is ten timesteps [12], [19]. This occurs as a result of the fed-back signal either disappearing or blowing up. Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) were used to address this problem [6], [13]. Depending on the complexity of the created network, LSTM networks can learn more than 1,000 timesteps and are somewhat biologically plausible [20].

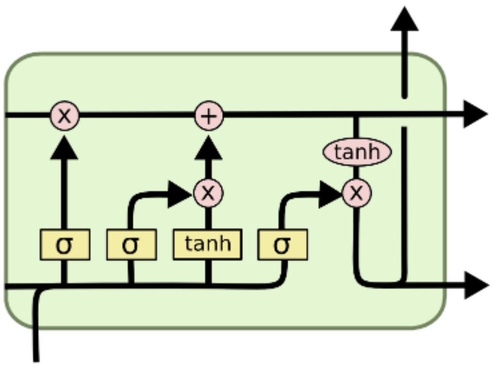


FIGURE 2.12: LSTM Architecture

Attention

The attention mechanism was included to enhance the machine translation encoder-decoder model's performance. By combining all of the encoded input vectors in a weighted fashion, with the most pertinent vectors receiving the highest weights, the attention mechanism was designed to allow the decoder to employ the most pertinent portions of the input sequence in a flexible manner. In addition, self-attention and Transformer model, which was proposed in [31], achieved state-of-the-art performance in natural language processing and became popular in other research field. A lot of new architectures in image processing were introduced based on Transformer architecture such as: ViT [4], Swin Transformer [16], ...

Chapter 3

Data

3.1 Dataset Discovery and Explanation

The dataset used for this work is the dataset obtained from RSNA 2022 Cervical Spine Fracture Detection [1] competition at Kaggle. This dataset has 3 main folders: train_images, test_images, and segmentations. The train_images folder contains training images, and this folder contains 2019 subfolders. Each subfolder is for a specific patient or case study, and it contains multiple slice images of the corresponding case. Therefore, those subfolders were named as the UID of their corresponding case study. Every image in those two folders is in the DICOM file format, which has slice thickness of under 1 mm, as well as in the axial orientation and bone kernel [1], and has a .dcm extension in each one. Meanwhile, the segmentations folder contains annotation masks and those masks are stored in NIFTI files (each file for a single patient). Target labels of the training data are given in the train.csv file. This file contains a column for case IDs, patient level labels (binary) and labels for a specific vertebra. In addition, this dataset provides a training_bounding_boxes.csv file which stores information about bounding boxes, such as anchor coordinates, width and height of bounding boxes.

According to [1], in the "train.csv" file, labels for training images are:

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

TABLE 3.1: Dataset overview. Number of test studies of the contest is 1500, but we only evaluate models on 72% of the dataset (private hidden set)

- patient_overall: overall target label, when any of the vertebra is fractured.
- C1 - C7: seven additional labels, whether a specific vertebra is fractured. Each vertebra is located in Figure 3.1 below.

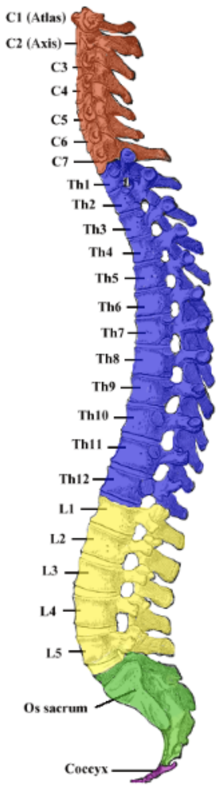


FIGURE 3.1: Vertebral column, cervical spines are C1 to C7 (top)

DICOM files, according to [10], in this dataset have 2 uses. One of them relates to the data in each file. These files, typically, are loaded via a function named *dcmread* in a library named *pydicom* which is a Python library used for accessing DICOM-format files. After loading images, there are four attributes that can be retrieved: *ImageOrientationPatient*, *ImagePositionPatient*, *PatientID*, *PatientName*. *PatientID* indicates the ID of the patient, the case study itself. *PatientName* shows the name of the patient; however, in this dataset, the name of the patient is the ID itself. *ImageOrientationPatient* indicates the orientation of the patient, while *ImagePositionPatient* indicates the position itself. *ImagePositionPatient* stores a list of 3 values, which is the position in 3-dimension space (x-axis, y-axis, z-axis). In addition, [10] revealed that unlike other competitions, z-axis value is the position of the slice image in the sagittal plane, rather than the timestamp. Sagittal plane, according to [30], is a vertical plane that creates two sections on either side of the body. However, all images in this dataset, according to [10], are in axial orientation, that means there is only one bone to observe for each image. Despite that, z-axis value can be referenced for position in the sagittal plane.

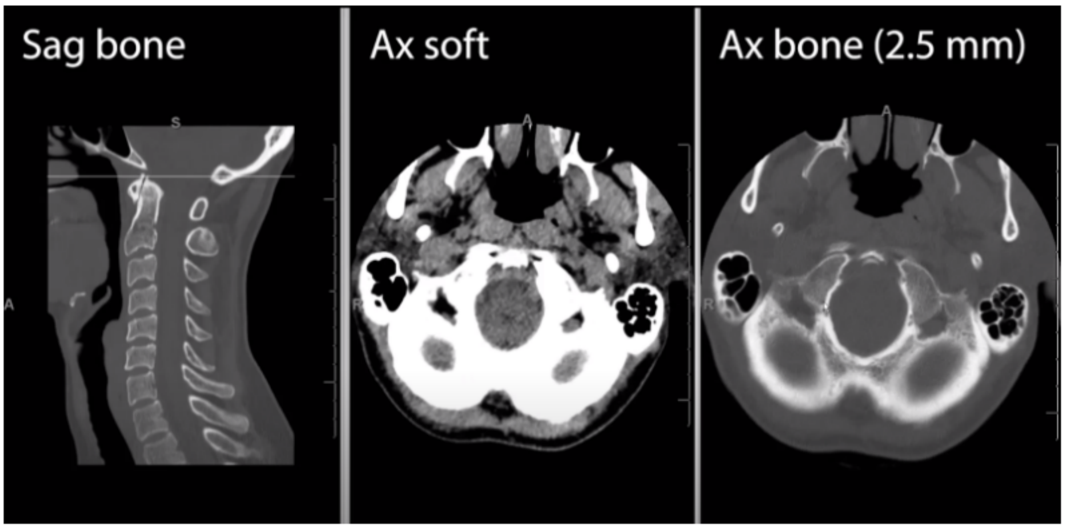


FIGURE 3.2: Bone images in sagittal plane and axial plane (source: [10])

In Figure 3.2, the left side is the bone image in the sagittal plane. Next, the middle is the axial view of bones. This view is quite soft; however, this type of data is not available in this dataset. Similarly, the right size is the axial view, but 2.5mm deep. The images in this dataset are only under 1mm deep, that means they are sharper than the right size of Figure 3.2.

Additionally, one thing needs to be considered on the left side of Figure 3.2 is the thin line. That line is the line that indicates the bone in axial view, and z-axis coordinate shows where the line is, as well as the corresponding position of the axial view in the sagittal plane.

However, there is a problem: although the sagittal view is available in this dataset, there is no way to check where the image is corresponding to which bone. As a result, data in the segmentation folder comes into action. This folder contains 87 files, and each file is for a specific case and named as the UID of that case. Unlike slice images, segmentations are in NIFTI format; therefore, another Python library named *nibabel* is used to load this kind of data. Moreover, the segmentations loaded are in 3D format (height, width, num_images), and num_images is the same number of slices of corresponding case study in the train_images folder. As there are differences about format between *NIFTI* segmentations and *DICOM* images, while *DICOM* files are segmented in the axial plane, *NIFTI* files are in the sagittal plane. This also allows us to choose the proper orientation so that the *DICOM* pictures and segmentation match using the *NIFTI* header information; otherwise, there is a chance that the segmentations will be mirrored in the x-axis and flipped in the z-axis. That is the guide to load segmentations properly. Figure 3.3 and 3.4 shows the slice image and its corresponding segmentation slice (in the case loaded properly).

After loading segmentations properly, we inspected the values in each segmentation slice, then we discovered that unique values would indicate which bone on each slice. For example, the unique values for the slice in Figure 3.4 are 0 and 6; and while number 0 indicates the background, number 6 refers to which bone, in this case, it is bone C6.

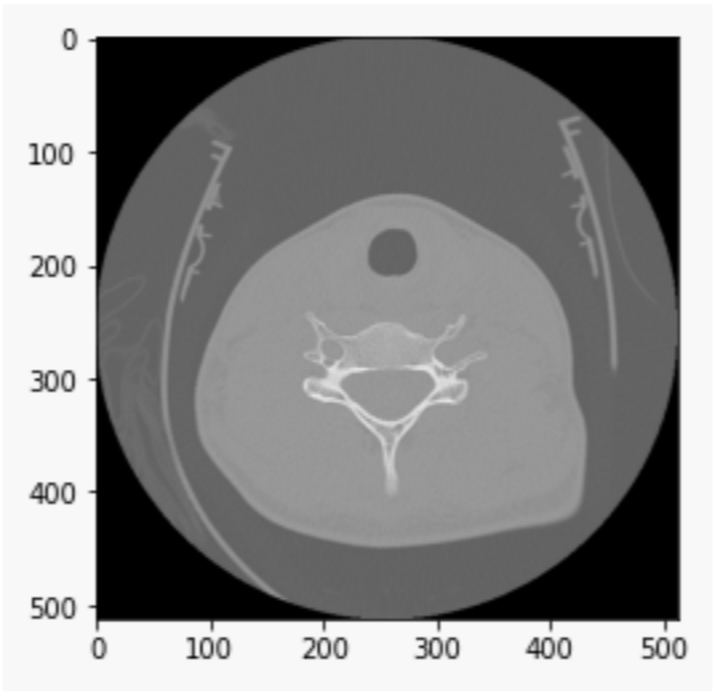


FIGURE 3.3: The sample image (after loading from DICOM format) (source: [10])

Figure 3.2, not only describes the vertebral column, but also is like the reference table of which value in segmentation corresponds to which bone. For example, value 1 is bone C1, value 2 is bone C2, value 8 is bone Th1 and so on.

3.2 Voxel

As the segmentation files in this dataset are in 3D format, every element in the segmentation cannot be called as pixels as in regular 2D images. Instead, they are called *voxels*.

A voxel in 3D computer graphics represents a value on an even three-dimensional grid. Voxels often do not explicitly encode their position (for example, coordinates) with their values, similar to how pixels in a 2D bitmap do. Instead, rendering systems infer a voxel's position based on how it is situated in relation to other voxels. Then, about the voxel size, [21] discusses that slice

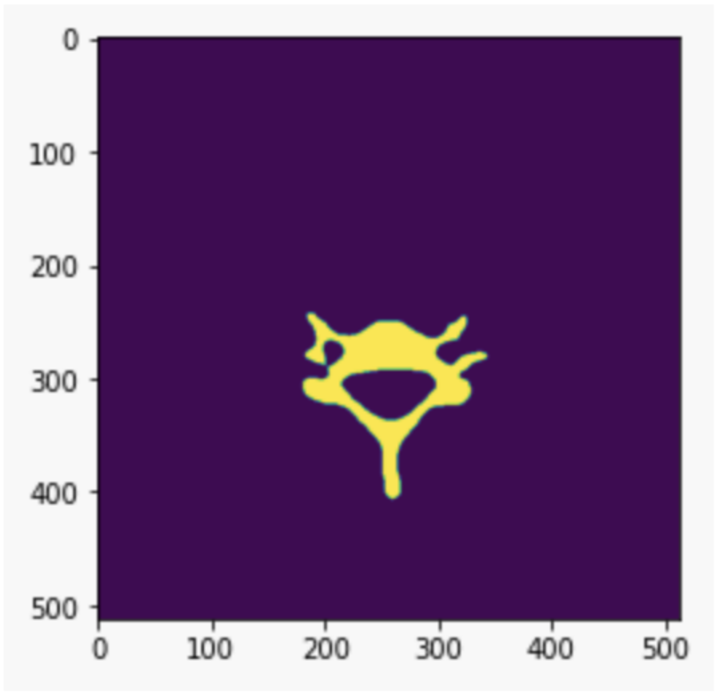


FIGURE 3.4: The corresponding segmentation of the image in Figure 3.3 (source: [10])

thickness and pixel size both influence voxel size; moreover, both the field of view and the picture matrix affect pixel size. In addition, [21] depicts that the higher the spatial resolution, the smaller the pixel size.

In CT simulations, Goertzen et. al in [7] illustrated that voxelized approach could be used in X-ray CT system simulations. In contrast, Goertzen discovered a drawback of this method, which would result in inherent errors due to the phantom voxelization.

On the other hand, voxels data can be used for deep learning. Gao et. al in [5] developed a deep learning based detection method for voxel-wise mapping of lumbar spine modic changes. As a result, the model obtained a sensitivity of 0.71 (±0.072), specificity of 0.95 (±0.022), and a Cohen's kappa score of 0.63 in 85.7% of samples in the unseen test set. This demonstrated major concurrence with radiologists and this method would be used to strengthen inter-rater reliability for modic changes assessments.

Due to the characteristics of segmentation data in this dataset, *nibabel* library is used for loading this type of data in Python.

3.3 Exploratory Data Analysis

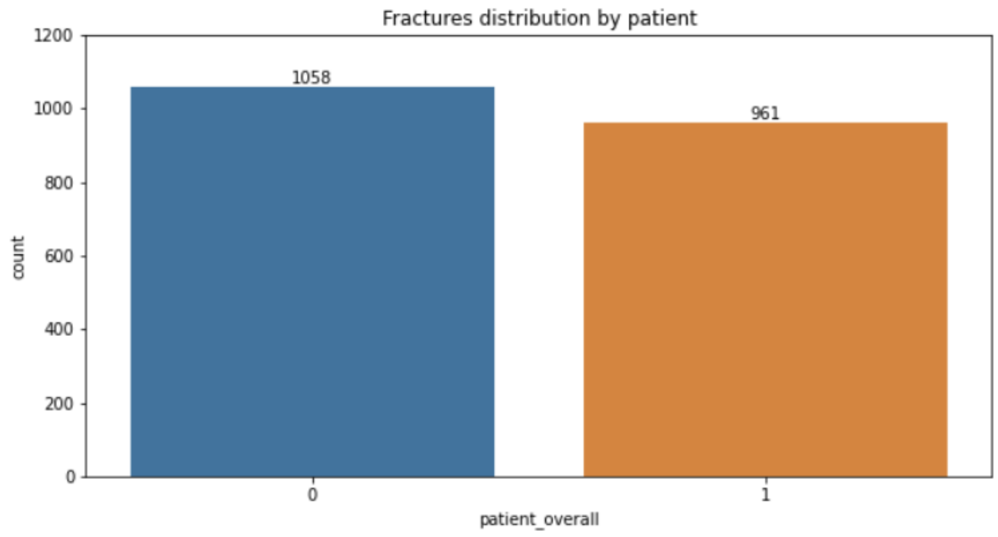


FIGURE 3.5: Dataset distribution (overall))

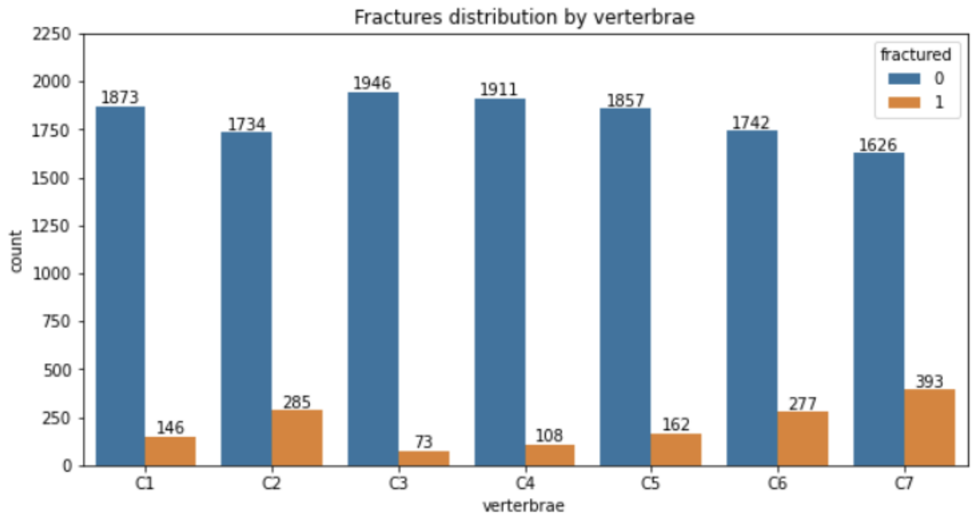


FIGURE 3.6: Fractures distribution by vertebrae

Figure 3.5 illustrates that the dataset, in overall, is roughly balanced (961 patients have fractures compared to 1058 normal ones); however, Figure 3.6

shows that distribution of fractures based on vertebrae seems to be imbalanced. Meanwhile in Figure 3.7, most of the fractured patients have fractures on only 1 single vertebra, that means some of patients have fractures on multiple vertebrae (no one has fractures on all 7).

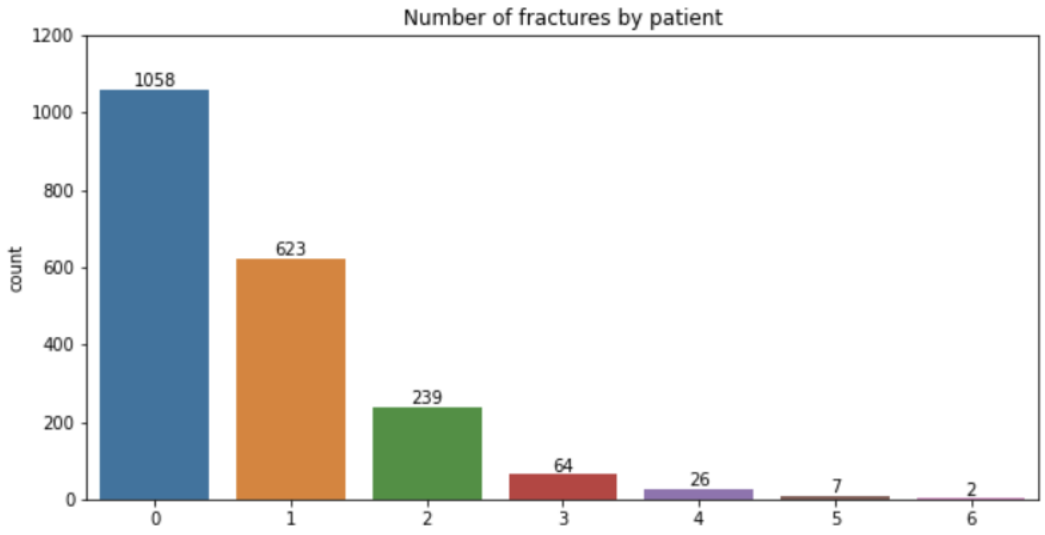


FIGURE 3.7: Number of fractures distribution

As can be seen from Figure 3.8, most of the training images have the size of 512 x 512 pixels, and some others have a little bit bigger sizes. As a result, we will scale all of the training images to the size 512 x 512 to unify the size for training processes.

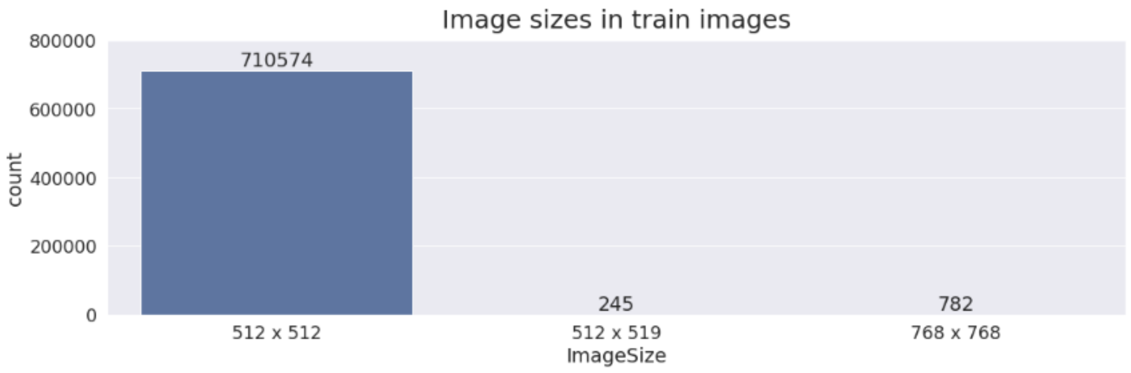


FIGURE 3.8: Image sizes distribution [25]

Figure 3.9 and 3.10 are examples of slice images and their corresponding segmentation masks of a particular case. Although there are a total of 2019 cases in the training dataset, only 87 of them have corresponding segmentation masks.

However, there is a difference about the dimension between images and segmentation masks. While all images are two-dimensional, segmentation masks are three-dimensional masks. Slices visualized in Figure 3.10 are just the position of 2D slices in the whole 3D data. As a result, while each case study in the train_images folder is a sub-directory which stores 2D images, each case in the segmentations folder is just a single file. This indicates that 3D segmentation techniques might be suitable for this type of data.

In addition, this difference can be eliminated by stacking consecutive slice images of each case, which results in a single 3D volume for the case. Therefore, this opens up opportunities for deep learning methods which use 3D volumes as input, such as 3D CNNs. Moreover, as 3D images are formed by stacking consecutive slices, 3D volumes can be in a variety of sizes; therefore, this might fit various 3D CNNs for fracture detection.

Similar to segmentations, only a subset of the training set have bounding boxes measurements (235 cases out of 2019), and Figure 3.11 depicts that all of those cases are fractured.

Moreover, Figure 3.12 illustrates that all of the images that have bounding boxes, only have 1 single bounding box for each. Last but not least, Figure 3.13 is the example of a slice image with a bounding box drawn on it.

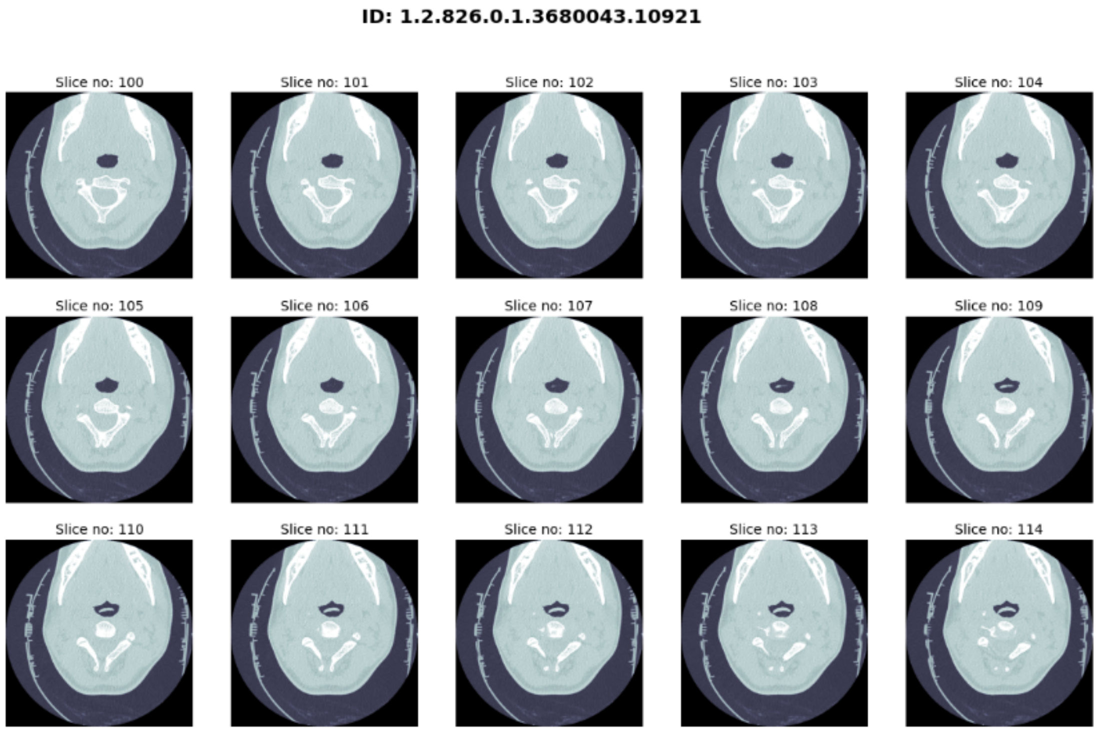


FIGURE 3.9: Example of 15 slice images

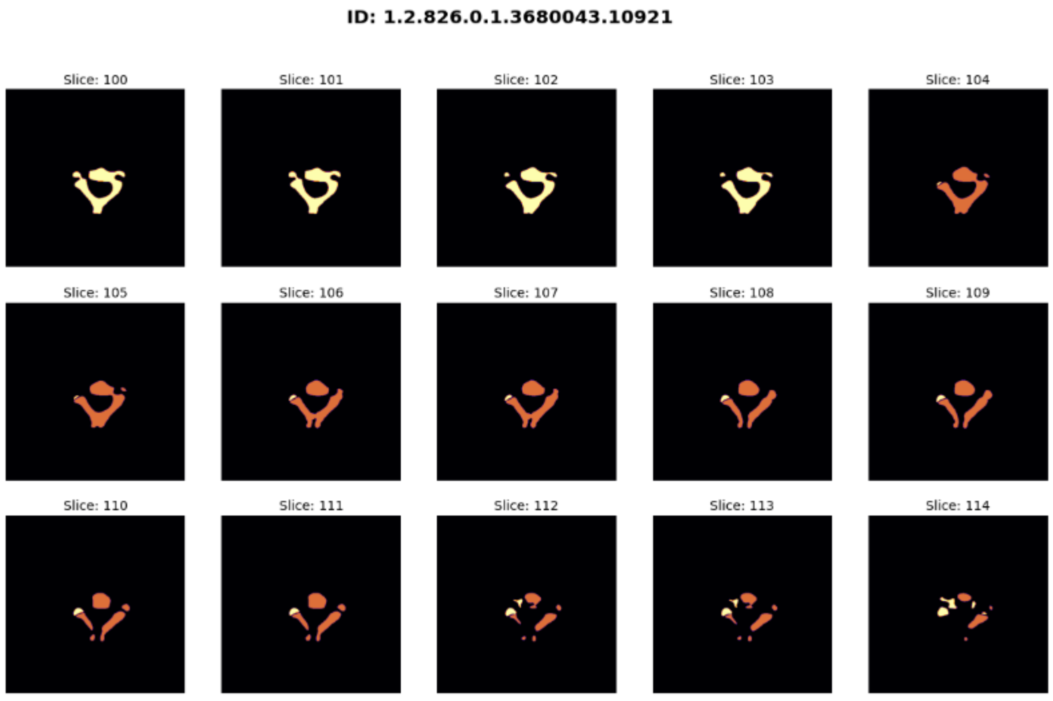


FIGURE 3.10: Example of 15 segmentation masks

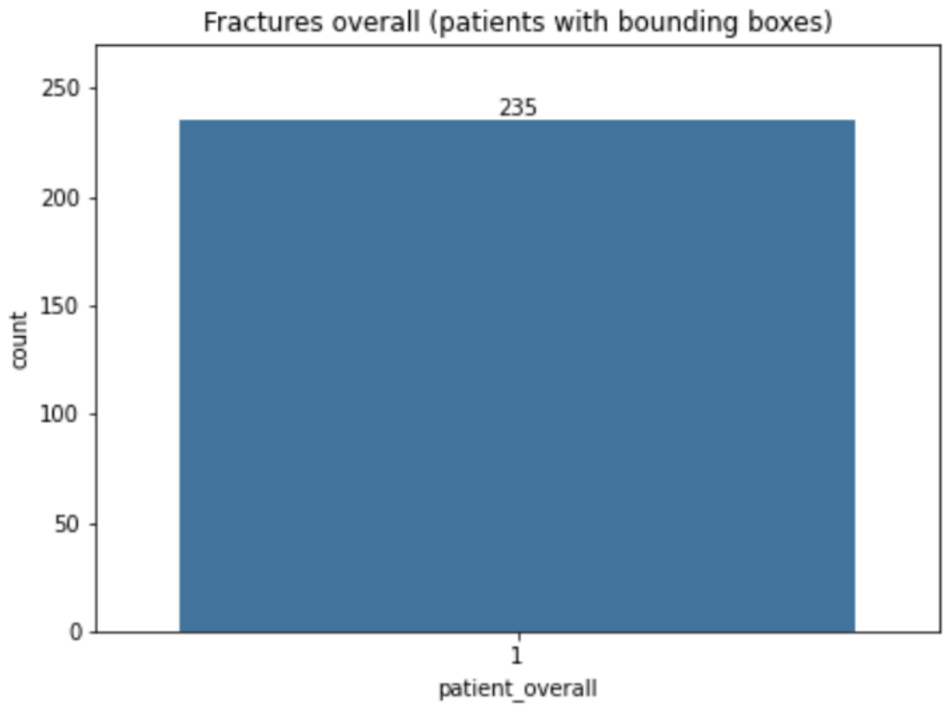


FIGURE 3.11: Example of 15 segmentation masks

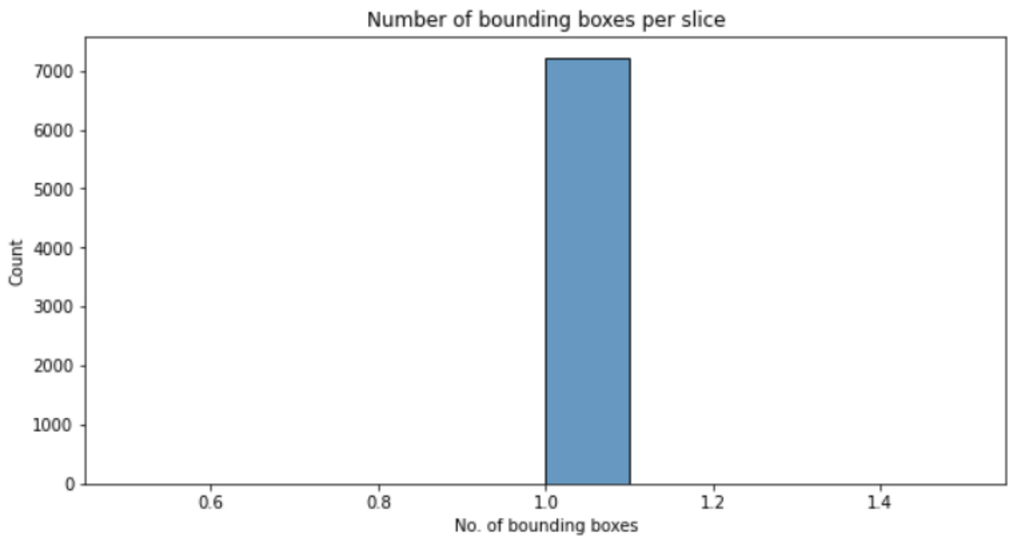


FIGURE 3.12: Distribution of number of bounding boxes per slice image

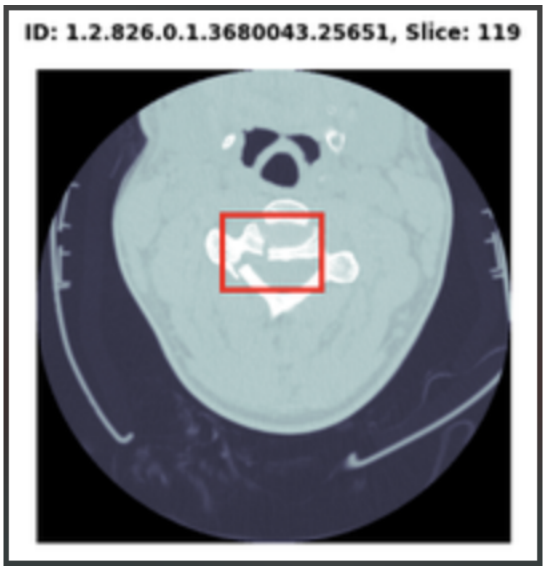


FIGURE 3.13: Example of a slice image with bounding box

Chapter 4

Implementation and Analysis

4.1 Evaluation Metric

Model performance is evaluated using a weighted multi-label logarithmic loss. Each fracture sub-type is its own row for every exam, and the model is expected to predict a probability for a fracture at each of the seven cervical vertebrae designated as C1, C2, C3, C4, C5, C6 and C7. There is also an "any label", patient_overall, which indicates that a fracture of any kind described before exists in the examination. Fractures in the skull base, thoracic spine, ribs, and clavicles are ignored. The "any label" is weighted more highly than specific fracture level sub-types.

The binary weighted log loss function for label j on exam i is specified as:

$$L_{ij} = w_{ij} \times [y_{ij} \times \log(p_{ij}) + (1 - y_{ij}) \times \log(1 - p_{ij})] \tag{4.1}$$

where the weights given by:

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

Finally, loss is averaged across all rows.

4.2 Data Processing

We first download the images from the directory provided by the contest and then remove the corrupted images. The data after being loaded will be normalized and resized. As for the data for the 3D model, the 2D images are stacked to obtain the 3D input and then are augmented by Random Rotation and Random Horizontal Flip. With input for 2D model, image will be 3x384x384 size, while input for 3D model will be 224x224x224.

4.3 3D Classification Model

Our first experiment was the 3D classification model, which has proven its performance in medical image data processing. [26] After being converted to 3D, the data will be passed through a model of 3 3D convolution blocks to calculate the feature map. Then the output matrix will be passed through several Fully Connected layers to get the final output. The architecture of a convolutional block consists of a convolution layer, an activation layer, a pooling layer, and a normalization layer.

The output of this model will have 8 dimensions, corresponding to the percentage of 8 layers: vertebrae C1-C7, patient overall. The goal of the model is to optimize the loss function given by the contest organizer, which has been described above.

We use *AdamW* as optimizer and *CosineAnnealingLR* as scheduler.

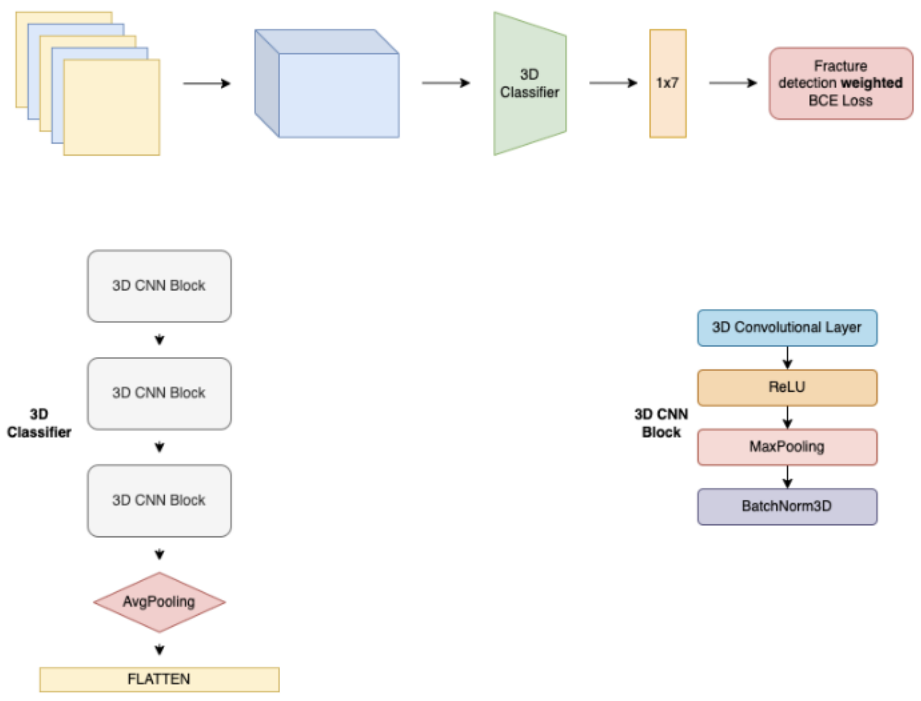
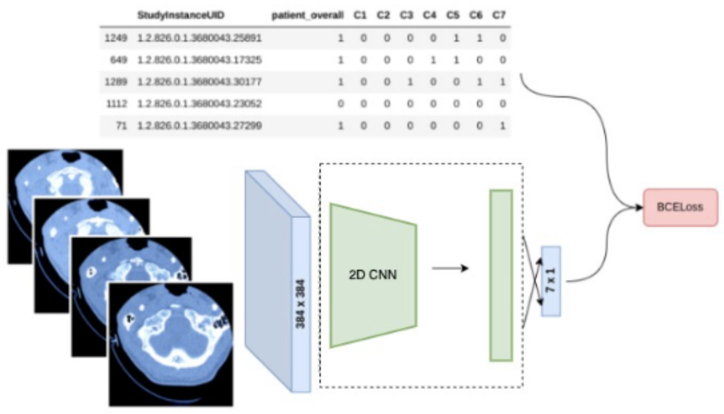


FIGURE 4.1: 3D CNN Classifier

4.4 2D Classification

4.4.1 Single-head Model

In this approach, we first trained a CNN model with ConvNeXt-Tiny as backbone on vertebrae labels provided by the organizers. There are only 87 studies in the training data that have the segmentation labels; a slice is determined to belong to a class if there is at least one pixel of that slice classified to that class. The data is split into 5 folds, 5 models then trained on 4 folds of them and evaluated on the others. After that, we used 5 models to infer all



	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7
1249	1.2.826.0.1.3680043.25891	1	0	0	0	0	1	1	0
649	1.2.826.0.1.3680043.17325	1	0	0	0	1	1	0	0
1289	1.2.826.0.1.3680043.30177	1	0	0	1	0	0	1	1
1112	1.2.826.0.1.3680043.23052	0	0	0	0	0	0	0	0
71	1.2.826.0.1.3680043.27299	1	0	0	0	0	0	0	1

FIGURE 4.2: CNN model for Vertebrae Classification

the training data to get the pseudo vertebrae labels to train the next model, predictions of the models are averaged to get the final prediction.

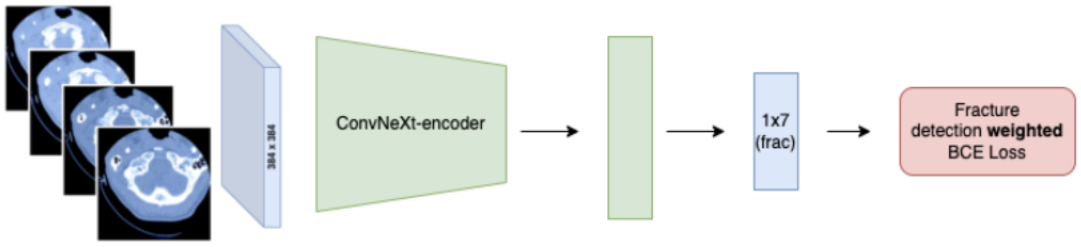


FIGURE 4.3: Single-head approach for Cervical Spine Fracture Detection

After getting the pseudo vertebrae labels, we multiplied vertebrae label and fracture label provided by the contest of each image to get the final fracture label. Next, we simply passed training data with fracture binary label through CNN model (ConvNeXt-Tiny) as a binary classification model. We used the multilabel loss function Binary Cross Entropy Loss with Logit from the library *pytorch*.

In the end we get a model that detects fractures and visible C1-C7 vertebrae using a single image. After that, for each `StudyInstanceUID` we first aggregate predictions for each of C1-C7 vertebrae by getting the max fracture probability predictions among slices. We use a simple formula to derive `patient_overall` fracture probability. `patient_overall` is the probability of any vertebra being fractured. It is equal to the maximum fracture probability of a vertebral component. Under assumption of independence of vertebrae fractures we can derive the following simple equation:

$$P_{\text{patient\_overall}} = \max_{i=1}^{7} P_{C_i} \tag{4.2}$$

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. From this point forward, k-folds splitting is understood to be applied to the grouped-by-study data.

4.4.2 Multi-head Model

At the first stage, we trained a CNN model to vertebrae classification with aforementioned architecture in 4.4.1 and inferred all training data. For this method, we also used EfficientNetV2 for backbone of this model.

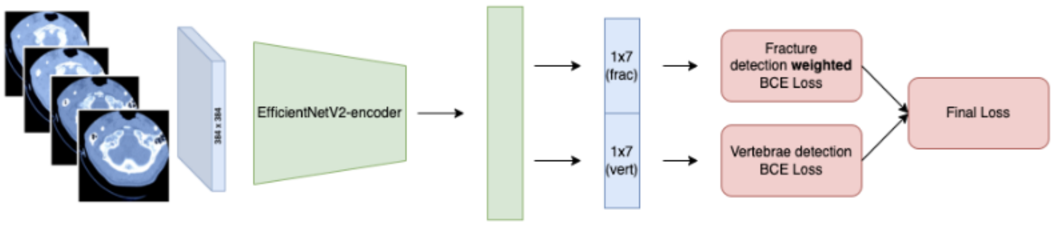

384 x 384
EfficientNetV2-encoder
1x7 (frac)
1x7 (vert)
Fracture detection weighted BCE Loss
Vertebrae detection BCE Loss
Final Loss

FIGURE 4.4: Multi-head model for Cervical Spine Fracture Detection

Data is passed to a pre-trained encoder of EfficientNetV2S. The final classification layer of the EfficientNetV2S was ignored, because its shape is irrelevant given the current task. The output of the encoder is then flattened and is passed to 2 Fully Connected Layers parallelly to optimize 2 loss functions contemporaneously. Note that we use logits in the loss function to improve numerical stability.

Vertebrae fracture targets are loaded from training data of the contest, while vertebrae detection targets are pseudo-label which comes from Stage 1.

Finally, we also had a model that classify fractures on single image. The final output of each `StudyInstanceUID` was obtained in the same way with the previous approach. The sole difference is the `patient_overall` fomular:

$$P_{\text{patient\_overall}} = 1 - \prod_{i}(1 - P_{C_i}) \tag{4.3}$$

We also split data into 5 folds and trained 5 versions of EfficientNetV2 models to obtain an ensemble model, which has slightly improved performance.

4.5 2.5D CNN+RNN Model

Overall, our final experiment consists of two stages: cervical vertebrae segmentation and fracture classification. At stage 1, we trained a CNN model using 2.5D images from 87 samples provided by the organizers to get the segmentation mask of cervical vertebrae and ratio of each cervical vertebra in all train data. After that, we cropped out all 7 cervical vertebrae of each study separately and trained a fracture classification model using these outputs. The model at stage 2 is a convolution network followed by bidirectional LSTM layers with attention.

Label	Cervical vertebrae
1	C1
2	C2
...	...
7	C7
8	T1-T12

TABLE 4.1: Labeling rules

4.5.1 Stage 1

First, we used the 87 studies of segmentation which were provided labels. We recreated the class labels as the Table 4.1.

We used 2.5D inputs with 3 channels of image data: the original image i and its sides (i-1, i+1). The detailed data augmentation section is in Table 4.2. In preparation for this stage, from the segmentation mask provided by organizer, we need to determent the bounding box of the bone. To do this phase, we first label connected regions of the mask, then measure the morphological properties of these connected regions and keep the bounding boxes. For each connected region, a bounding box of the form (x_min, y_min, x_max, y_max) is returned. For this stage's model, we used ConvNeXt-Tiny as the backbone of the encoder. In terms of the training process, we used AdamW as optimizer and CosineAnnealingWarmRestarts as the scheduler. Our model in this stage has two output: bounding box of the vertebra and the ratio of the vertebra's type. Note that the ratio is calculated by dividing the maximum number of pixels belonging to C_k class in all slices by the number of pixels belonging to C_k class in the current slice.

After training the stage 1 model, we inferred all 2019 studies with the inputs preprocessed using the same method mentioned earlier. We keep a list of categorized slices for each patient and each type of bone. Note that a slice can be more than one vertebra (for example 60% C1 and 70% C2), and we save

Augmentation methods
HorizontalFlip
RandomBrightnessContrast
HueSaturationValue
ShiftScaleRotate
Cutout

TABLE 4.2: Augmentation methods

images of vertebrae with more than 50% of that kind to the vertebra's image list. We individually intented to crop each study's seven cervical vertebrae. A fracture label from `train.csv` is attached to each cervical vertebrae. Our EDA estimated that the majority of studies have 200–300 slices per study, so that each vertebra typically has 30 slices. We chose from lists of slices to get 24 images using evenly spaced indices for each type of bone. For cervical vertebrae with less than 24 slices, we evenly duplicated some slices to get enough images.

4.5.2 Stage 2

After getting the bounding box from stage 1, we cropped the images following to the bounding boxes. Once all the cervical vertebrae have been cropped, the data shape of our input was (batch_size, number_of_channels, image_size, image_size), we then stacked three slices into one to get the 2.5D inputs, 24 images chosen as earlier mentioned method turned into a sequence of 8 2.5D images, it is also the sequence length of LSTM layers. We used the same augmentation with Stage 1. For the model we also utilized ConvNeXt-Tiny from the *timm* library for backbone model. Additionally, we included an Attention Layer, which also provides a slight improvement.

The `patient_overall` is calculated as:

$$P_{\text{patient\_overall}} = 1 - \prod_{k=1}^{N}(1 - P_{C_k}) \tag{4.4}$$

where:

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

P_{C_k} is the ratio of `vertebrae` C_k

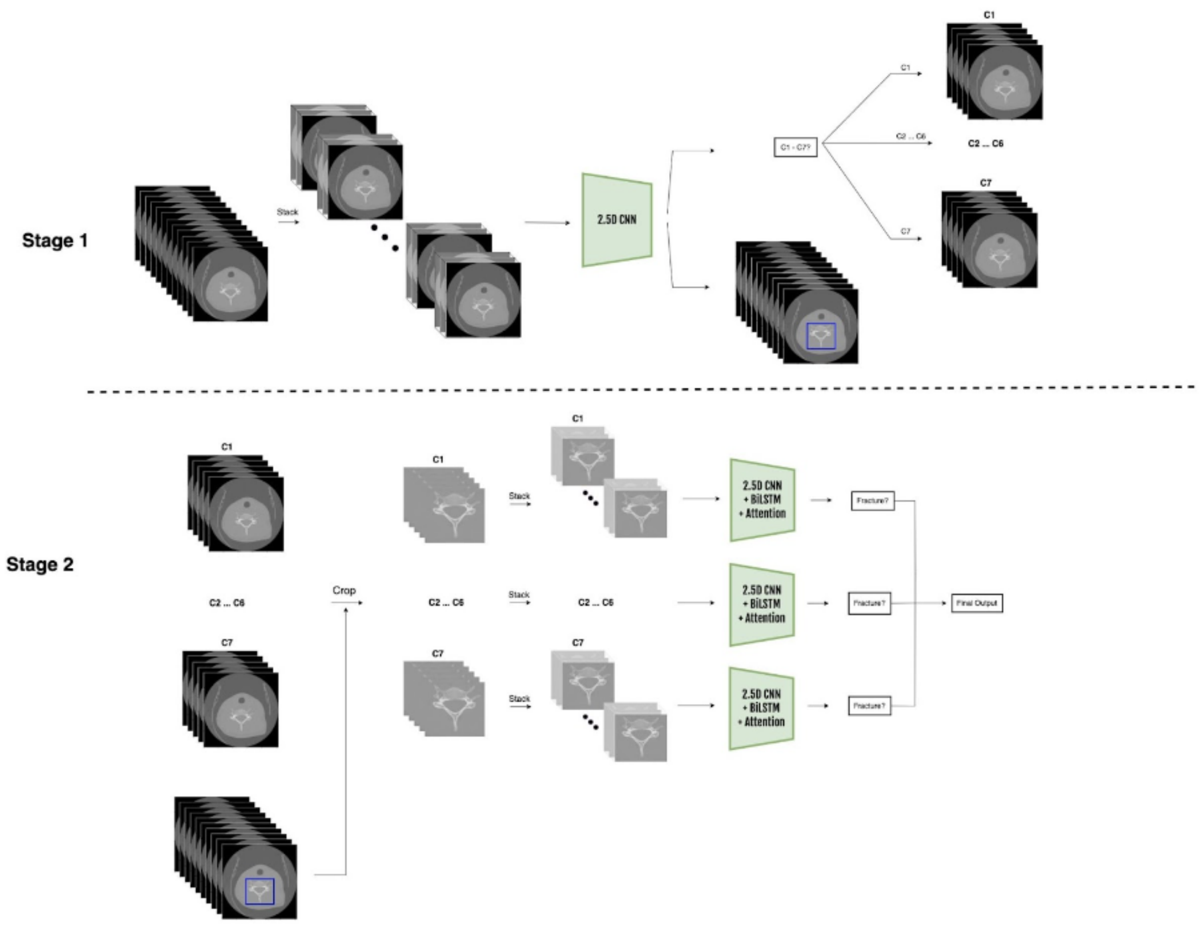


FIGURE 4.5: Two-stage Model Architecture

We ran several times to choose the suitable N. The result of each run is indicated in table 4.3. At this stage, we split the data into 5 folds for training and ensembled the results of 5 models for final prediction.

All aforementioned experiments were implemented on resources that are free and available for everyone: Tesla T4 GPU on Google Colab and P100 GPU on Kaggle Notebook. More detail on our experiments can be found at: `https://github.com/trantuan4132/kaggle-RSNA-Fracture-Detection`

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		

TABLE 4.3: *N* choosing. Score is cross-validate contest metric.

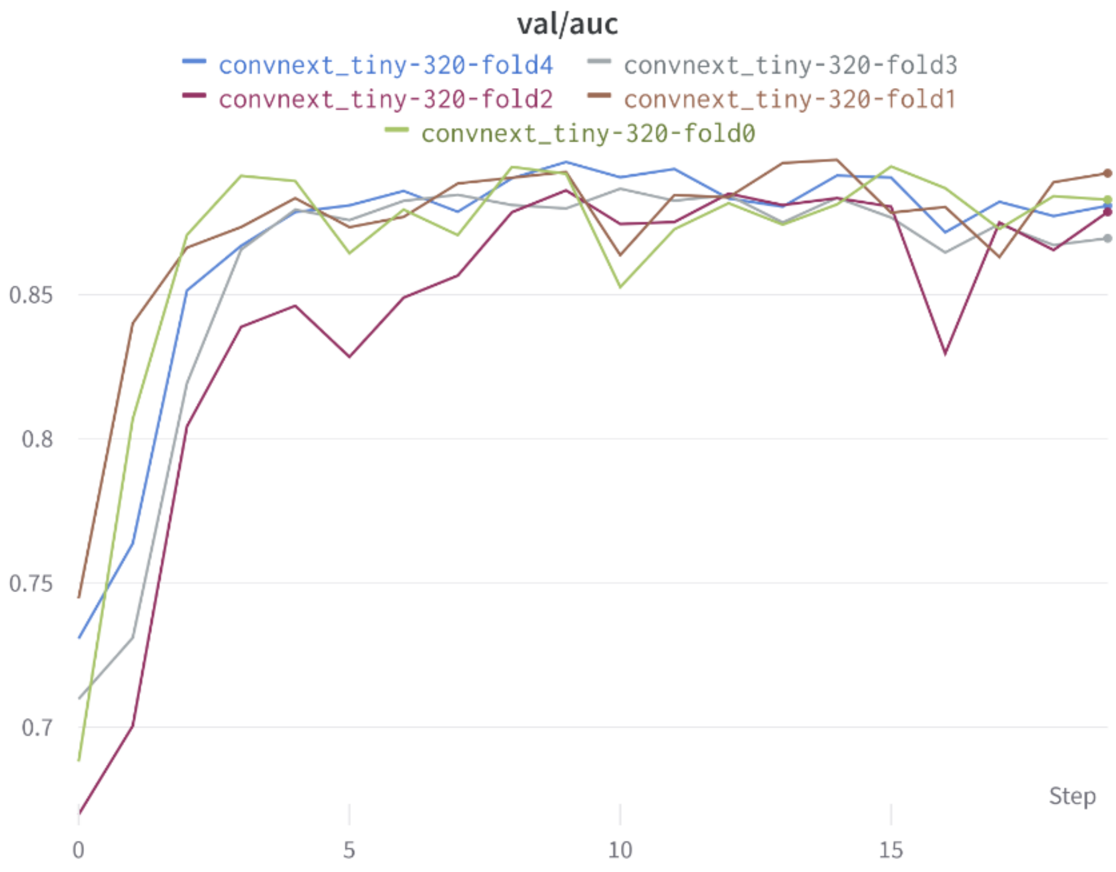


FIGURE 4.6: Stage-2 Model's AUC on validation dataset

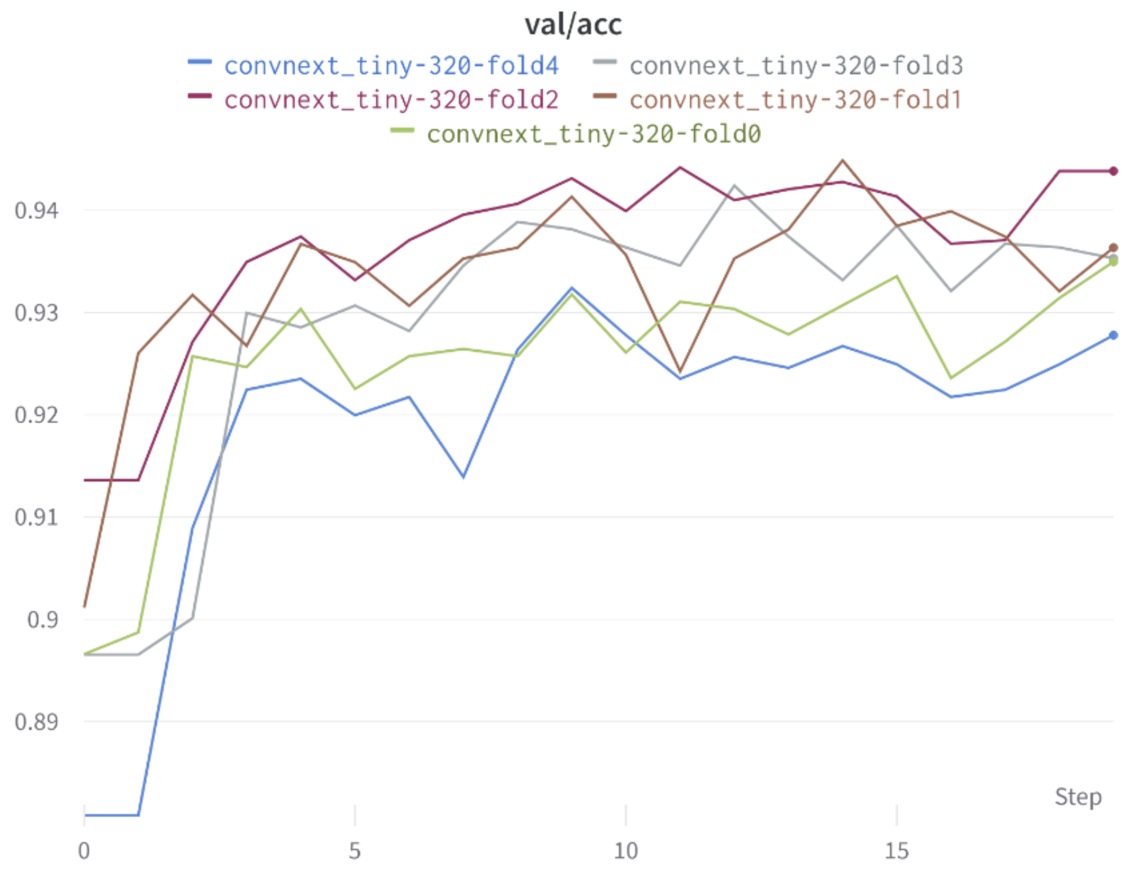


FIGURE 4.7: Stage-2 Model's Accuracy on validation dataset

Chapter 5

Results and Discussion

5.1 Results

We compared different approaches for cervical spine fracture detection. Table 5.1 shows the performance of each approach on the private test dataset of the contest with the representative metric mentioned earlier.

Regarding solutions in the Kaggle contest, our final implementation is inspired by the top-2 method [23], which also uses a segmentation model along with a classification model. The critical difference in our model is that its segmentation component is smaller in size than the top-2 one; therefore, our model has a shorter inference and training time and requires fewer resources. Our final work's achievement is equivalent to the score of top 24 and 25 of the contest on the public and private test dataset respectively.

Model	**Score**
3D CNN	0.6048
Single-head	0.5813
Multi-head	0.5019
2.5D CNN + RNN	**0.3691**

TABLE 5.1: Results on hidden test dataset provided by the organizer

Model	**Score**	**Stage-1 inference time (h)**
Kaggle Top 2	**0.2389**	4.55
2.5D CNN + RNN	0.3691	**3.67**

TABLE 5.2: Comparison with Top-2 solution. Score is contest metric on hidden test set. Inference time is calculated on full 2019 studies training data.

5.2 Discussion and Future Works

In this work, we experimented multiple approaches for cervical spine fracture detection. The final model 2.5D CNN + RNN demonstrated higher efficiency in time and resources than the previous methods and achieved a reasonable result. This model still has various parts that can be optimized to get higher performance. With the benefits of Transformer's recently proven string data such as faster training time due to parallelization, better performance with self-attention techniques, our next steps will be experimenting with Transformer layers instead of LSTM for sequence data processing, training another backbone model, trying models with bigger image size and longer sequence length, etc.

References

[1] Errol Colak FelipeKitamura HCL-kanishkaa Hui Ming Lin JeffRudie John Mongan Katherine Andriole Luciano Prevedello Michelle Riopel Robyn Ball Sohier Dane Adam Flanders Chris Carr. *RSNA 2022 Cervical Spine Fracture Detection*. 2022. URL: https://kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection.

[2] Arman Avesta et al. "Comparing 3D, 2.5D, and 2D Approaches to Brain Image Segmentation". In: *medRxiv* (2022). DOI: 10.1101/2022.11.03.22281923.

[3] David Chettrit et al. *3D Convolutional Sequence to Sequence Model for Vertebral Compression Fractures Identification in CT*. 2020. DOI: 10.48550/ARXIV.2010.03739.

[4] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. DOI: 10.48550/ARXIV.2010.11929.

[5] Kenneth T. Gao et al. "Automatic detection and voxel-wise mapping of lumbar spine Modic changes with deep learning". In: *JOR SPINE* 5.2 (2022), e1204. DOI: https://doi.org/10.1002/jsp2.1204.

[6] F.A. Gers, J. Schmidhuber, and F. Cummins. "Learning to forget: continual prediction with LSTM". In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[7] Andrew Goertzen, Freek Beekman, and Simon Cherry. "Effect of voxel size in CT simulations". In: vol. 3. Feb. 2000, 20/93 –20/97 vol.3. ISBN: 0-7803-6503-8. DOI: 10.1109/NSSMIC.2000.949326.

[8] @haqishen. *1st Place Solution*. 2022. URL: https://www.kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection/discussion/362607.

[9] @harshitsheoran. *8th Place Solution*. 2022. URL: https://www.kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection/discussion/362669.

[10] @harshitsheoran. *Explaining Data and Submission in detail*. 2022. URL: https://www.kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection/discussion/340612.

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[12] Sepp Hochreiter. "Untersuchungen zu dynamischen neuronalen Netzen". In: (Apr. 1991).

[13] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[14] Jongmin Jeong, Tae Sung Yoon, and Jin Bae Park. "Towards a Meaningful 3D Map Using a 3D Lidar and a Camera". In: *Sensors* 18.8 (2018). ISSN: 1424-8220. DOI: 10.3390/s18082571.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[16] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. DOI: 10.48550/ARXIV.2103.14030.

[17] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. DOI: 10.48550/ARXIV.2201.03545.

[18] Sakib Mostafa and Fang-Xiang Wu. "Chapter 3 - Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI

images". In: *Neural Engineering Techniques for Autism Spectrum Disorder*. Ed. by Ayman S. El-Baz and Jasjit S. Suri. Academic Press, 2021, pp. 23–38. ISBN: 978-0-12-822822-7. DOI: https://doi.org/10.1016/B978-0-12-822822-7.00003-X.

[19] Michael C Mozer. "Induction of Multiscale Temporal Structure". In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson, and R.P. Lippmann. Vol. 4. Morgan-Kaufmann, 1991.

[20] Randall C. O'Reilly and Michael J. Frank. "Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia". In: *Neural Comput.* 18.2 (2006), 283–328. ISSN: 0899-7667. DOI: 10.1162/089976606775093909.

[21] radiopaedia.org. *Radiology Reference Article*. 2022. URL: https://radiopaedia.org/articles/voxel-size-1.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[23] @ryanrong. *2nd Place Solution*. 2022. URL: https://www.kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection/discussion/365115.

[24] Hojjat Salehinejad et al. *Deep Sequential Learning for Cervical Spine Fracture Detection on Computed Tomography Imaging*. 2020. DOI: 10.48550/ARXIV.2010.13336.

[25] @samuelcortinhas. *RSNA Fracture Detection - in-depth EDA*. 2022. URL: https://www.kaggle.com/code/samuelcortinhas/rsna-fracture-detection-in-depth-eda.

[26] Satya Singh. "3D Deep Learning on Medical Images: A Review". In: *Sensors* 20 (Sept. 2020). DOI: 10.3390/s20185097.

[27] Ralf C. Staudemeyer and Eric Rothstein Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019. DOI: 10.48550/ARXIV.1909.09586.

[28] Mingxing Tan and Quoc Le. “EfficientNetV2: Smaller Models and Faster Training”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 10096–10106.

[29] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. cite arxiv:1905.11946Comment: Published in ICML 2019. 2019.

[30] teachmeanatomy.info. *Anatomical Planes*. 2022. URL: https://teachmeanatomy.info/the-basics/anatomical-terminology/planes/.

[31] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762.

[32] Y. Wang, C.L. Lin, and J.D. Miller. “Improved 3D image segmentation for X-ray tomographic analysis of packed particle beds”. In: *Minerals Engineering* 83 (2015), pp. 185–191. ISSN: 0892-6875. DOI: https://doi.org/10.1016/j.mineng.2015.09.007.

[33] Yichi Zhang et al. *Bridging 2D and 3D Segmentation Networks for Computation Efficient Volumetric Medical Image Segmentation: An Empirical Study of 2.5D Solutions*. 2020. DOI: 10.48550/ARXIV.2010.06163.