Image Classification for Cervical Spine CT Scans

Final Presentation

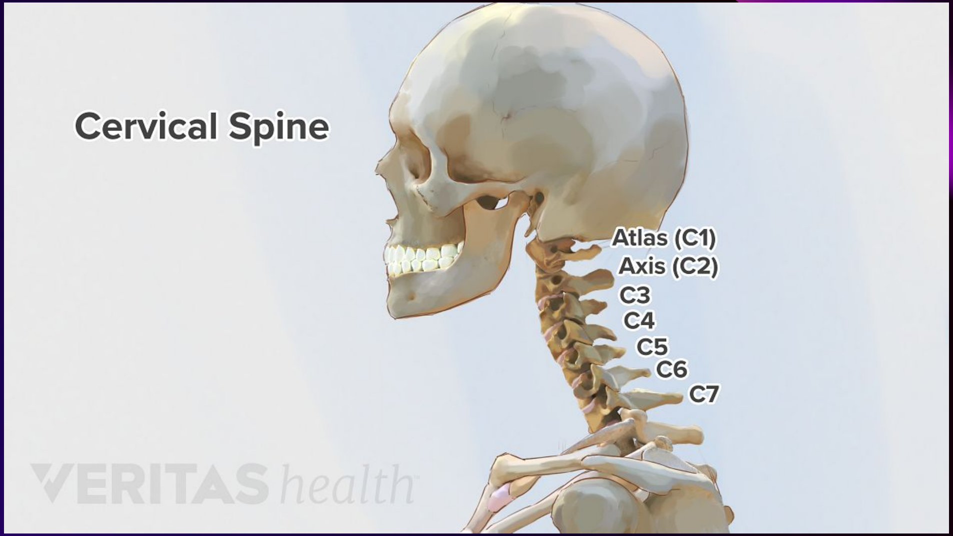
Aniket Lachyankar & Satwik Kamarthi
CS7150 - Professor David Bau
December 5, 2022

Table of **Contents**

01
Introduction

Our **motivation**

- An Image is Worth 16x16 Words: Transformers For Image Recognition At Scale
- Cervical Spinal Fractures domain
- Quick detection is essential to prevent neurological deterioration and paralysis after trauma



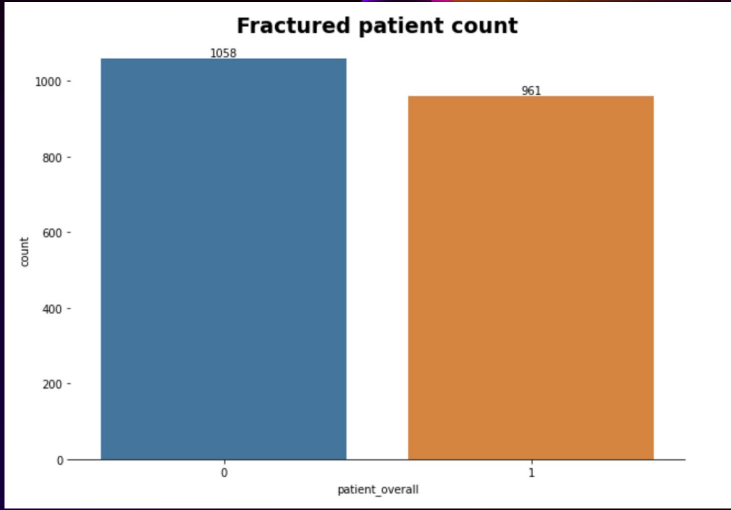
Project Goal

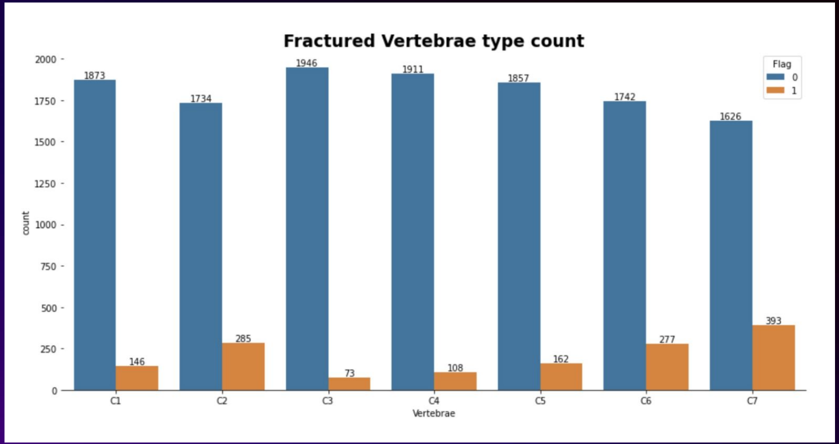
Understand the performance of top models in the medical imaging domain by comparing a ResNet model and Vision Transformer model and evaluate how well they can handle a CT scan classification task.

02
Data

Data **overview**

- ❖ The dataset we are using comes from RSNA 2022 Cervical Spine Fracture Detection Kaggle competition.
- ❖ Dataset comprised of Cervical Spine CT scans taken from 12 sites around the world by the Radiological Society of North America (RSNA)
- ❖ Training Data
  - ➢ 2000 patients
    - ■ 300-600 digital imaging files (.dcm) representing axial scans of fractures
  - ➢ Distribution of fractures
  - ➢ Fractured Vertebrae count





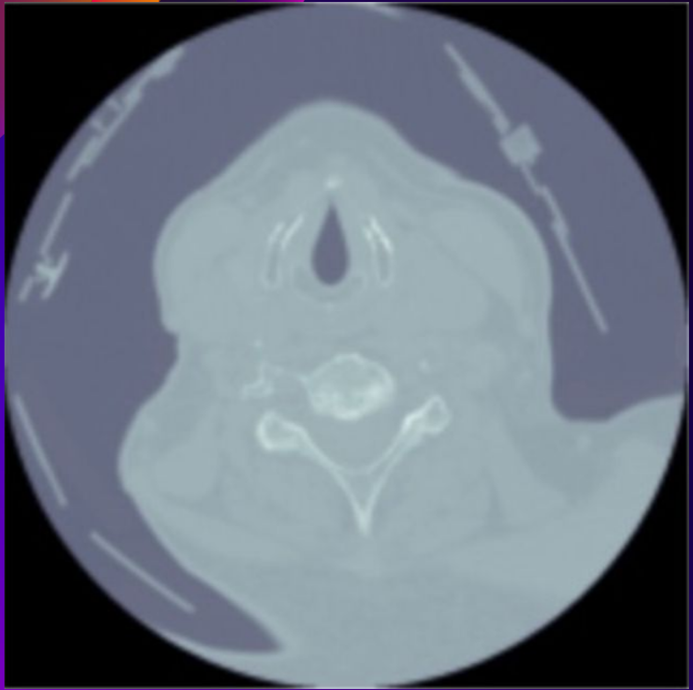
Data **Preprocessing**

- After meeting with a medical professional, we were advised that a better view to detect spinal fractures with our model may be the sagittal view (from the side).

- Since original dataset has all the images in axial view (top-down), we needed to convert all the axial images to sagittal images.

- Using methods adapted from pydicom library documentation:
  - Accessing DICOM image metadata
  - Create 3D array filled with original images from file
  - Manipulate & transform array to get the sagittal aspect and view from 3D array

- From 300-600 axial images we generated 25 sagittal images per patient in training set, and 100 sagittal images per patient in test set.
  - Total ~16K Train And Validation Images
  - Sample mainly from the middle of each scan as that is where the majority of each vertebrae can be seen.



03

Model Selection

ResNet

- Huggingface implementation
- Microsoft pre-trained weight checkpoint
- ResNet-50
  - Balance between performance and training time

```
==========================================================================================
Layer (type:depth-idx)                                  Output Shape              Param #
==========================================================================================
ResNetForImageClassification                            [1, 1000]                 --
├─ResNetModel: 1-1                                      [1, 2048, 1, 1]           --
│    └─ResNetEmbeddings: 2-1                            [1, 64, 56, 56]           --
│    │    └─ResNetConvLayer: 3-1                        [1, 64, 112, 112]         9,536
│    └─ResNetEncoder: 2-2                               --                        (recursive)
│    │    └─ModuleList: 3-4                             --                        (recursive)
│    └─ResNetEmbeddings: 2-3                            --                        (recursive)
│    │    └─MaxPool2d: 3-3                              [1, 64, 56, 56]           --
│    └─ResNetEncoder: 2-4                               [1, 2048, 7, 7]           --
│    │    └─ModuleList: 3-4                             --                        (recursive)
│    └─AdaptiveAvgPool2d: 2-5                           [1, 2048, 1, 1]           --
├─Sequential: 1-2                                       [1, 1000]                 --
│    └─Flatten: 2-6                                     [1, 2048]                 --
│    └─Linear: 2-7                                      [1, 1000]                 2,049,000
==========================================================================================
Total params: 25,557,032
Trainable params: 25,557,032
Non-trainable params: 0
Total mult-adds (G): 4.09
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 177.83
Params size (MB): 102.23
Estimated Total Size (MB): 280.66
==========================================================================================
```

ResNet cont.

- Model hyperparameters inspired by PyTorch blogpost
- Small batch size
  - GPU memory size
- Ability to increase with given compute resources
- Because we have different number of labels, we use BCELoss for multilabel output
  - Use Sigmoid on each output class rather than softmax for all classes

Hyperparameter/Optimizer	Value
Loss Function	Binary Cross Entropy
Learning Rate	0.5
Learning Rate Sceduler	Cosine Annealing
Learning Rate Warmup Epochs	5
Learning Rate Warmup Method	linear
Learning Rate Warmup Decay	0.01
Batch Size	24
Optimizer	SGD (Stochastic Gradient Descent)
SGD Momentum	0.9
Weight Decay	2e-05
Epochs	100

Vision **Transformers**

- Huggingface implementation

- Google and Huggingface pre-trained weight checkpoint
  - ➢ Avoid vanilla model
  - ➢ Trained on ImageNet-21k

- Stages and parameter sizes of model for 1 input image on right

```
==========================================================================================
Layer (type:depth-idx)                             Output Shape              Param #
==========================================================================================
ViTForImageClassification                          [1, 1000]                 --
├─ViTModel: 1-1                                    [1, 197, 768]             --
│    └─ViTEmbeddings: 2-1                          [1, 197, 768]             152,064
│    │    └─ViTPatchEmbeddings: 3-1                [1, 196, 768]             590,592
│    │    └─Dropout: 3-2                           [1, 197, 768]             --
│    └─ViTEncoder: 2-2                             [1, 197, 768]             --
│    │    └─ModuleList: 3-3                        --                        85,054,464
│    └─LayerNorm: 2-3                              [1, 197, 768]             1,536
├─Linear: 1-2                                      [1, 1000]                 769,000
==========================================================================================
Total params: 86,567,656
Trainable params: 86,567,656
Non-trainable params: 0
Total mult-adds (M): 201.58
==========================================================================================
Input size (MB): 0.60
Forward/backward pass size (MB): 162.19
Params size (MB): 345.66
Estimated Total Size (MB): 508.46
==========================================================================================
```

Vision Transformers cont.

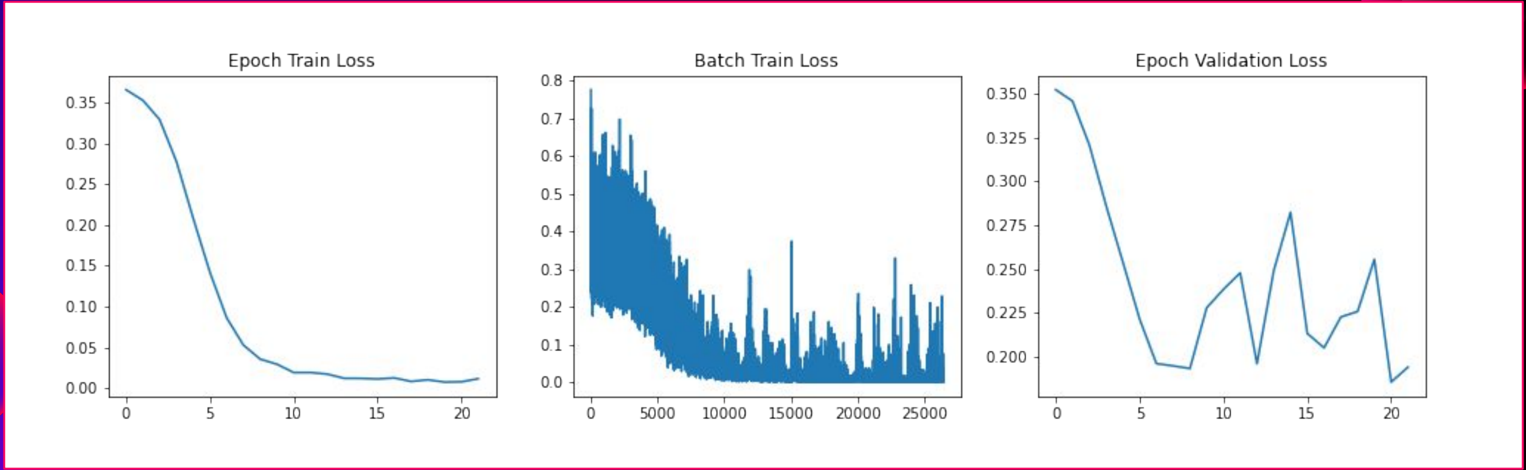
- Adjust input images to (3,384, 384)
  - No models for images of size 512
- Similar to ResNet, we use BCELoss for calculating gradient and Sigmoid activation on output layer.

Hyperparameter/Optimizer	Value
Loss Function	Binary Cross Entropy
Learning Rate	3e-05
Batch Size	12
Optimizer	Adam
Adam Betas	(0.9, 0.999)
Adam Epsilon	1e-08
Epochs	10
Learning Rate Sceduler	Linear

04

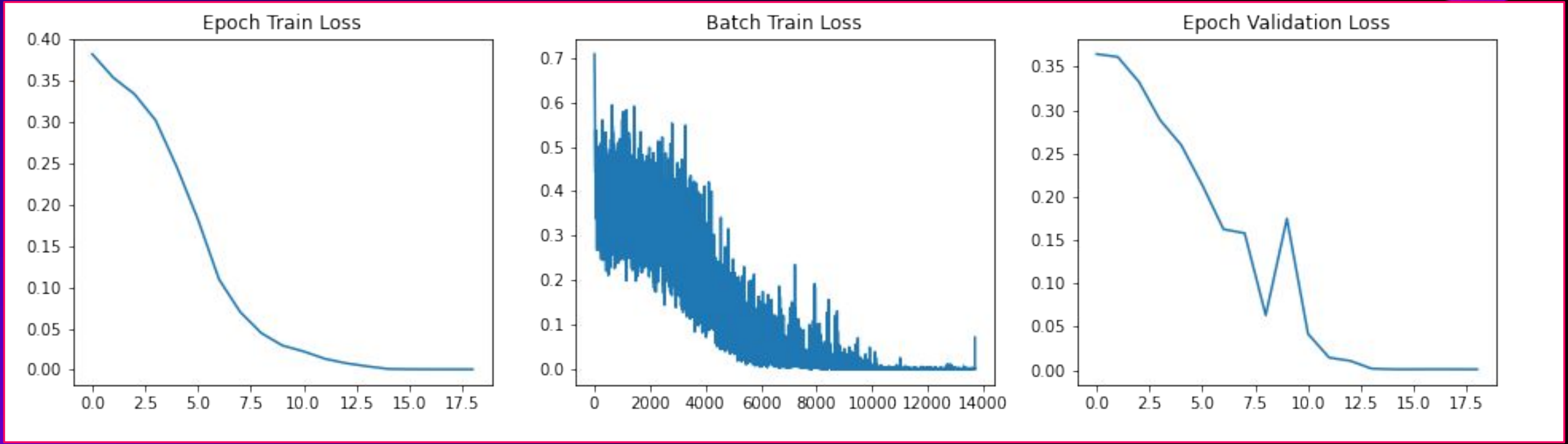
Results & Future Work

ViT Results



- We see that the model converges on training data after approximately 10 epochs.
  - Jagged validation loss suggests that the model may be overfitting.

ResNet Results



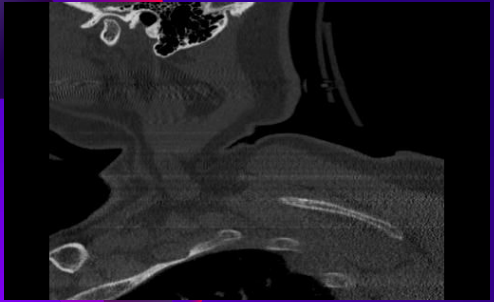
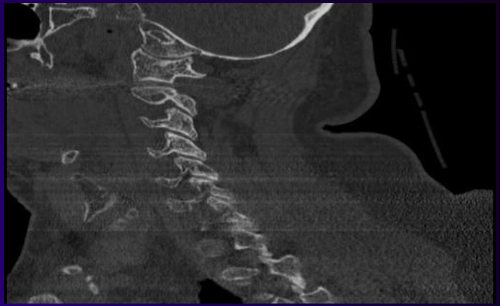
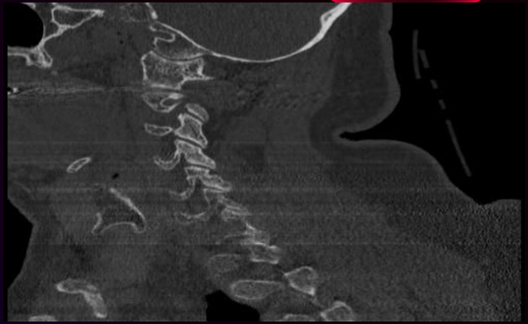
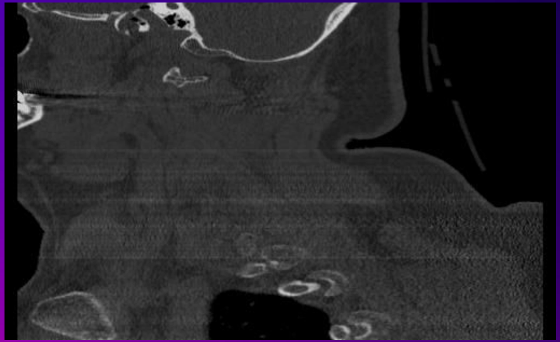
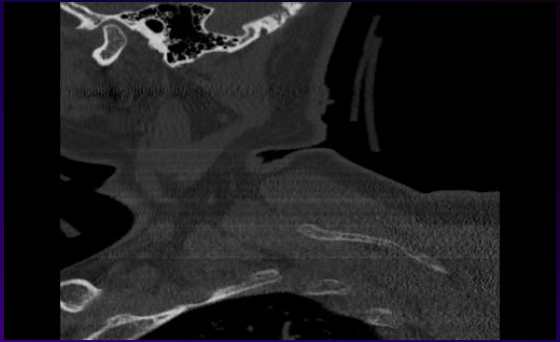
- ❖ Similar convergence after 10 epochs.
  - ➢ Seems like less overfitting based on validation loss.

Results Contd.

- Next, we tested the best trained model (lowest validation loss) on a test holdout set
  - 301 Sagittal Samples, all labeled with a C1 fracture.
- ResNet did not predict and correct samples.
  - Either dataset was not large enough, or we overfit the model on the training dataset
- ViT predicted 5 correct predictions out of 301 samples in the test set:
  - Low performance suggests overfitting in this case as well based on the loss graphs we see previously.

Results Contd.

❖ Below are the test samples that ViT correctly predicted with the C1 fracture label.

Analysis

A few takeaways from this project:

- Dataset selection/preprocessing is EXTREMELY important
  - Choices made regarding selecting data for training were the bulk of the project.
  - In our case, the used dataset size may have been too small for the complexity of the task.
  - A key task of our data preprocessing was selecting slices from the scan, which we did arbitrarily. Because of this and the nature of the labels, we cannot guarantee that
- Pre-Trained models may not contain trained parameters that work best for a domain vastly different from the pretraining.
  - Repurposed models that are usually for single label prediction didnt necessarily work well for complex multilabel prediction.

Future Work

- Improve input data
  - One major problem with the dataset provided is that we have no way of guaranteeing whether all the slices in a scan contain the fracture that is in the label.
  - Dataset came with bounding boxes for axial slices, maybe possible to use these for sagittal slices.
- Utilize models that can handle sequential data.
  - If we can pass all the slices of the scan as part of a single sequence of inputs, the model may have a better chance at analyzing multiple slices .

QUESTIONS?