Cervical Spine Fracture Detection Using Deep neural network

CSCI-417 Project

Fall 2023

Submitted by:

Nadeen Nassif Mahmoud	19105325
Abdallah Mahmoud Elsayed	19100718
Mohamed Salah Elhag	19100903
Omar Ahmed Gamal	19104756
Omar Khaled Mohamed	19105630

Under the Supervision of Prof. Ghada Khoriba

Table of Contents

Cervical Spine Fracture Detection Using Deep neural networks

Abstract

Numerous research has been done to determine how well artificial intelligence (AI) performs in identifying fractures. On radiographs, AI has been used to identify hip, humeral, distal radius, wrist, hand, ankle, and thoracic fractures, and fractures of the lumbar spine on dual x-ray absorptiometry. AI has also been utilized to identify calcaneal10 and thoracic and fractures of the lumbar vertebral bodies 11–13 on CT. To the best of my knowledge, automatic detection of cervical spine fractures still needs enhancement. Research is still working on this because it will save effort and time and help in detection faster than humans. This research depends on the Kaggle competition, the RSNA has teamed with the American Society of Neuroradiology (ASNR) and the American Society of Spine Radiology (ASSR) to conduct an AI challenge competition exploring whether artificial intelligence can be used to aid in the detection and localization of cervical spine fractures. The most common fracture in the 2nd and the 7th fracture between all patients

1. Introduction

More than three million patients a year are evaluated for a cervical spine injury, which is a frequent condition. According to the National Cancer Institute about the anatomy of the Spinal Cord “The spine is made up of bones, muscles, tendons, nerves, and other tissues that reach from the base of the skull near the spinal cord (clivus) to the coccyx (tailbone). The vertebrae (back bones) of the spine include the cervical spine (C1-C7) thoracic spine (T1-T12), lumbar spine (L1-L5), sacral spine (S1-S5), and tailbone. Each vertebra is separated by a disc. sc. The vertebrae surround and protect the spinal cord.”. Nowadays much research is done to use AI in detecting fractures in the spinal cord, however much work is needed to enhance detection accuracy. The aim of this research is to apply the methods of AI on a 3d convolutional neural network with n layers to detect which vertebrae have fractures to avoid neurologic deterioration and paralysis after recovery.

2. Methodology

By applying the 3^{d} convolutional neural network with n layers, we found that there are some libraries like k-segmentation, tqdm.auto, glob,scipy.ndimage, and tensorflow that help in cleaning the data and helps in detecting if there is a fracture in the vertebra or not and in which one.

2.1. Dataset

Our data contains Dicom files of CT scans of the cervical spine (neck). According to "The American society of spine radiology" the dataset contains 3 files segmentation, test images, and train images. Segmentation and test image files contain scans for every patient with his unique ID. Every human has 7 vertebrae c1, c2, c3, c4, c5, c6, and c7. From the segmentation file, we found in file train.csv that the common fracture in 1st and 2nd vertebrae and to know if any fracture is there or not by finding in "patient overview" 0&1, 0 means no fracture, and 1 means there is a fracture. From the segmentation, we can detect in each patient which vertebra has a fracture.

2.2. Segmentation

First, we are trying to visualize the whole file and clean the data. To make visualization of the data we use fracture prediction to train and find how many vertebrae have been fractured, then if we found two vertebrae have been fractured.

Figure 1 shows the workflow of AI in radiology applications.

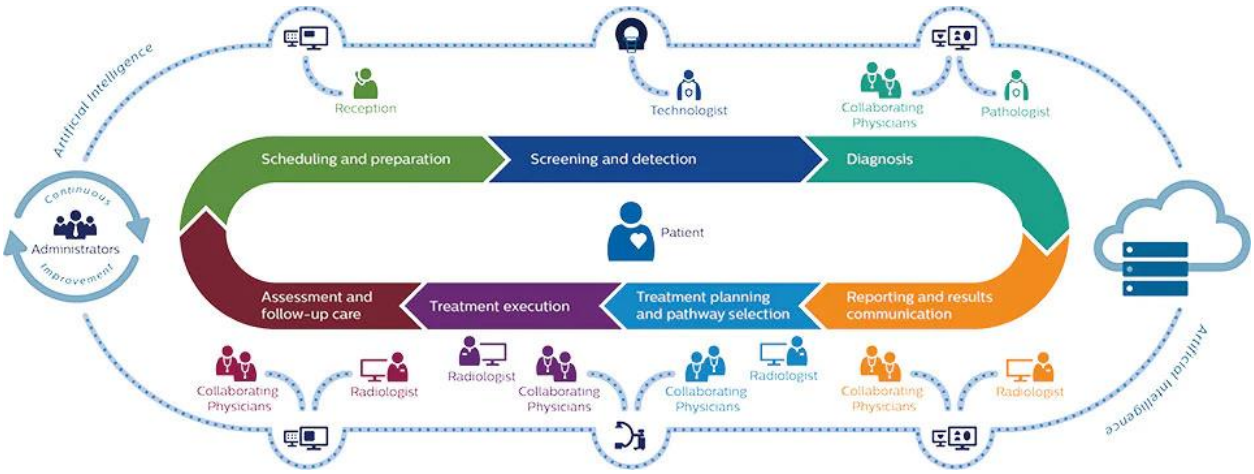


Figure 1:workflow of AI in radiology applications (Perkuhn, 2018)

In segmentation, we print the data for each patient. "Patient_overall" contains all the cervical vertebrae and shows if there is a fracture in the vertebra or not and which one has the fracture.

In figure 2, according to Sabaghian et. al.[71] fully automatic 3D segmentation of the thoracolumbar spinal cord and the vertebral canal from T2-weighted MRI using the K-means clustering algorithm. Spinal Cord 58, 811–820 (2020). "The framework includes:

(1) recognition of region of interest (ROI) based on mutual information as a similarity measure;
(2) applying a canny filter to extract edges and Hough line transform in order to remove the extra parts from ROI in the anterior-posterior direction;

(3) computing the centerline of the spinal cord by using Hough circular transform; (4) resorting to the circles by selecting a candidate circle close to the AP line with radius [7,8,9] to find the spinal cord curvature and (5) applying an anisotropic diffusion filter to remove noises and segmentation of spinal cord and canal by a k-means clustering algorithm to classify intensities."

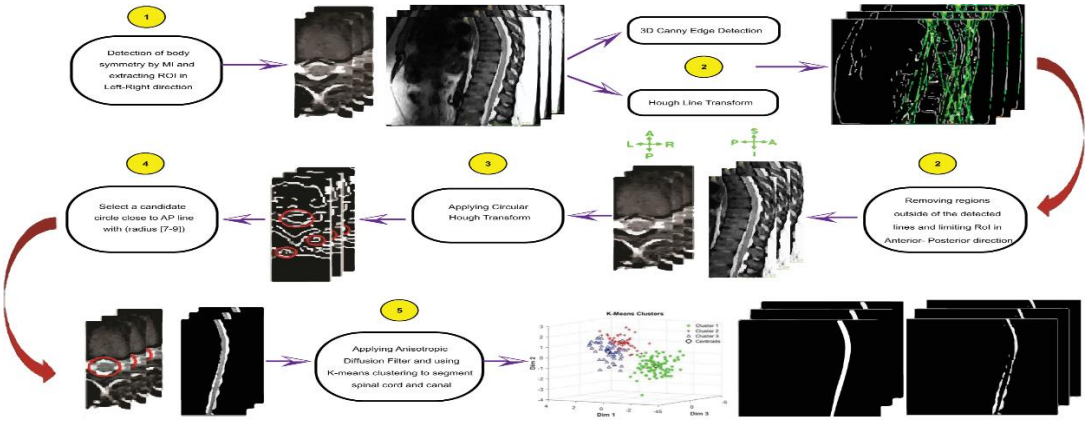


Figure 2: The framework of the K-Seg method (Sahar Sabaghian, 2020)

2.3. Cervical fracture detection

According to Salehinejad et.al. [1] in deep sequential learning for cervical spine fracture detection on computed tomography imaging "A cervical spine CT has N number of axial image slices along the craniocaudal axis where we represent each image with a vector x_n. Therefore, we can model the cervical spine as the set of input images X = (x1, x2, ..., xN) with the corresponding image level labels y = (y1, y2, ..., yN), where yn = 1 means the image contains at least one fracture and yn = 0 is the opposite. We can also define a case level label y ∈ {0, 1}, where if y = 1 at least one image contains a fracture and if y = 0 none of the images contain a fracture. Figure 1 shows the different steps of the proposed model. It has two major steps, which are prepossessing of the input images X and learning a mapping function from the preprocessed images to the target y. Different steps are discussed as follows" 5 feb 2021 so we know from 1 or 0 is there a fracture or not.

2.4. Preprocessing:

Fractures of the cervical spine are heterogeneous in their location, type, and composition.

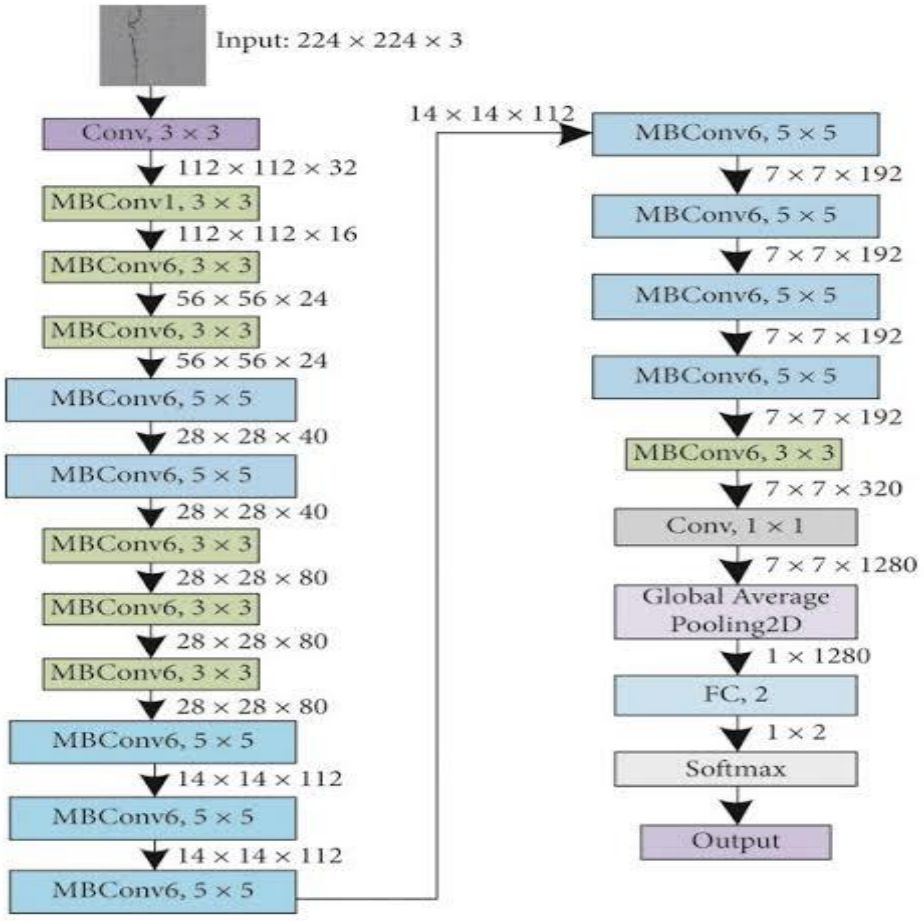
Malalignment of the cervical spine can be caused by a fracture and/or a strained ligament.

Localized bleeding and soft tissue inflammation can be linked to fractures and ligament damage.

This process will flatten the output and transfer the image from pixel to hu (Hounsfield Unit) to rescale the image and appear it as a real photo instead of dark or black photo.

2.5. Training classifiers

EffNetB0:

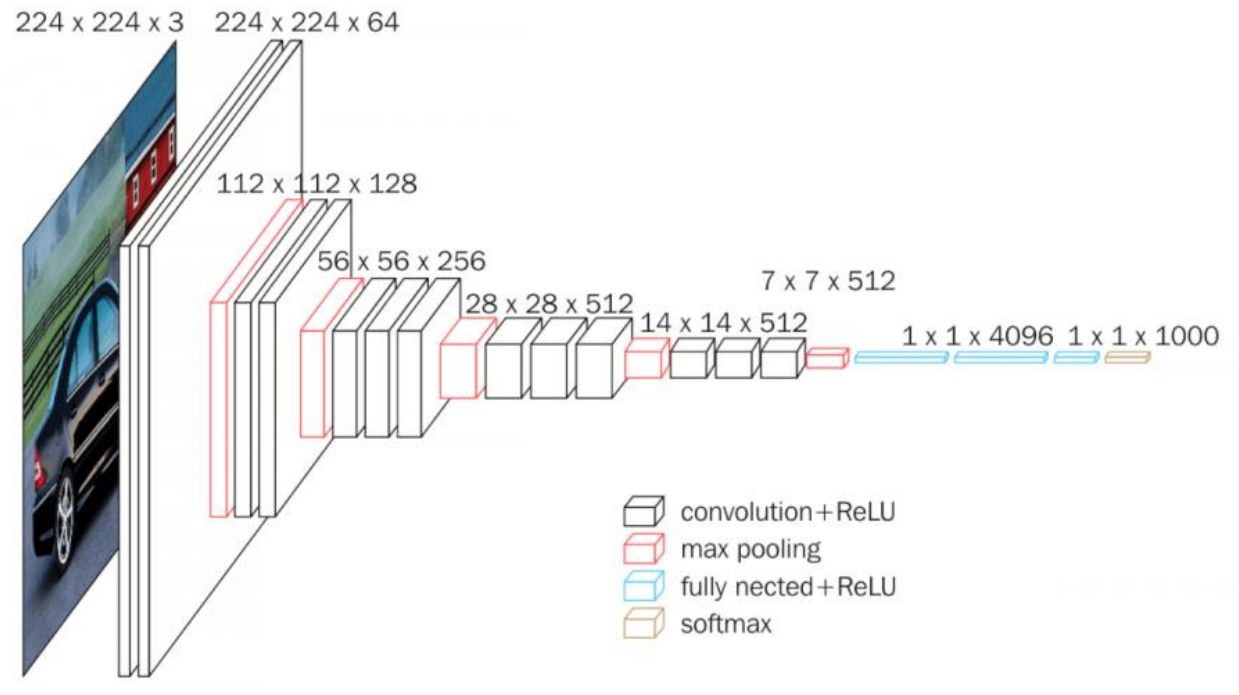


EffNetB0 model architecture

We used the EfficientNetB0 pre-trained model coupled with ImageNet weights without the top and with max pooling, and disabling all layers from being trained except the final 10, the model takes the input as the shape of the images and runs the model then connects it with a 128-neuron dense layer with a ReLU activation function, then a 7-neuron dense layer with a sigmoid

activation function and a 50% dropout. The model is compiled with a binary cross-entropy loss function, the ADAM optimizer with a learning rate of .0001, and binary accuracy as a metric.

VGG16:



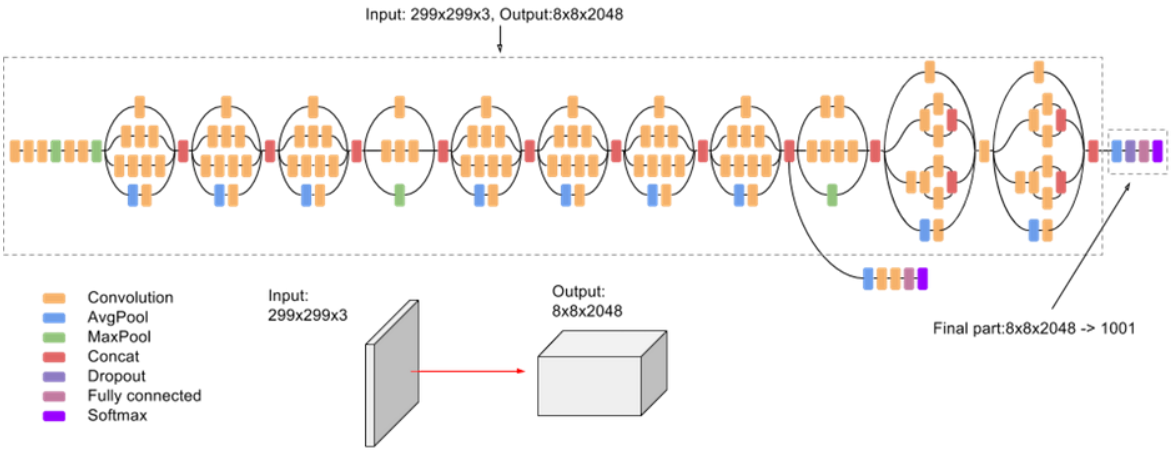
VGG16 model architecture

In this model we split the data into train and test, then use VGG16 model by using pre-training on ImageNet and using and then told the model not to use the last layer in the model and use the layer I'll give him and then give the model the shape of the photo. I add more about 3 layers relu and 1 sigmoid, relu to classify them into 0&1 and sigmoid to classify from 0 to 1.

We use optimizer Adam as it is the most used optimizer

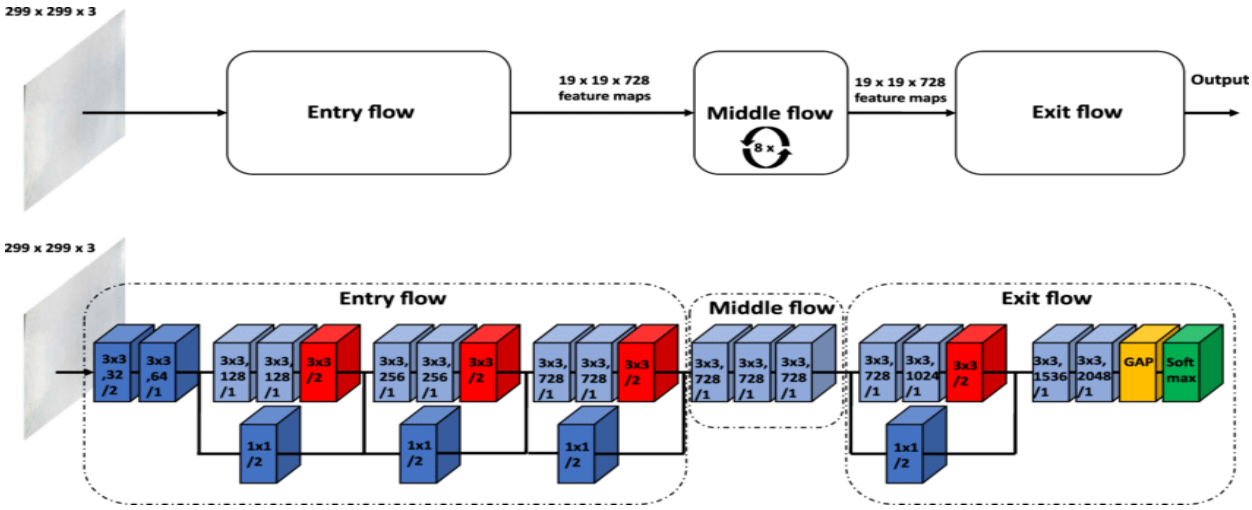
Then I apply 50 epochs to get the best accuracy but when the model find that the loss increase every time stop after two epochs from the best one.

InceptionV3:



We used Inceptionv3 pretrained model with ImageNet weights without including the top. The layers for this model were untrainable except the last 10 layers. We did 2 runs for this model, once where it takes input as the shape of the images and runs the model then connects it with 7 neuron sigmoid dense output layer and another where it takes input as the shape of the images and runs the model then connects it with 128 neuron sigmoid dense layer then dropout layer then connects it with 7 neuron sigmoid dense output layer. We have the loss as binary cross entropy and Adam optimizer.

XCeption:



XCeption model architecture

We used the XCeption pre-trained model coupled with ImageNet weights without the top and with max pooling, and disabling all layers from being trained except the final 10, the model takes the input as the shape of the images and runs the model then connects it with a 7-neuron dense layer with a sigmoid activation function. The model is compiled with a binary cross-entropy loss function, the ADAM optimizer with a learning rate of .0001, and binary accuracy as a metric.

Nerual Network:

```
def get_model_02():
    model = keras.Sequential([
        layers.Dense(128, activation='relu', input_shape=[128,128,3]),
        layers.Dropout(0.4),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.6),
        layers.Dense(64, activation='relu'),
        layers.Flatten(),
        layers.Dense(7, activation='sigmoid'),
    ])

    model.compile(loss="binary_crossentropy",
                  optimizer = tf.keras.optimizers.Adam(learning_rate = 1e-4),
                  metrics=[tf.keras.metrics.BinaryAccuracy()])
    model.summary()

    return model
```

We made a basic Neural Network with an input layer that takes the shape of the images, passes it into a 128 Dense layer with an ReLU activation function, then another 128 Dense with a ReLU

activation function and 40% dropout, then a 64 Dense layer with a ReLU activation function and 60% dropout, then flattens the input into a linear list and passes it into the final 7-neuron Dense layer with a sigmoid activation function. The model is compiled with a binary cross-entropy loss function, the ADAM optimizer with a learning rate of .0001, and binary accuracy as a metric.

CNN #1:

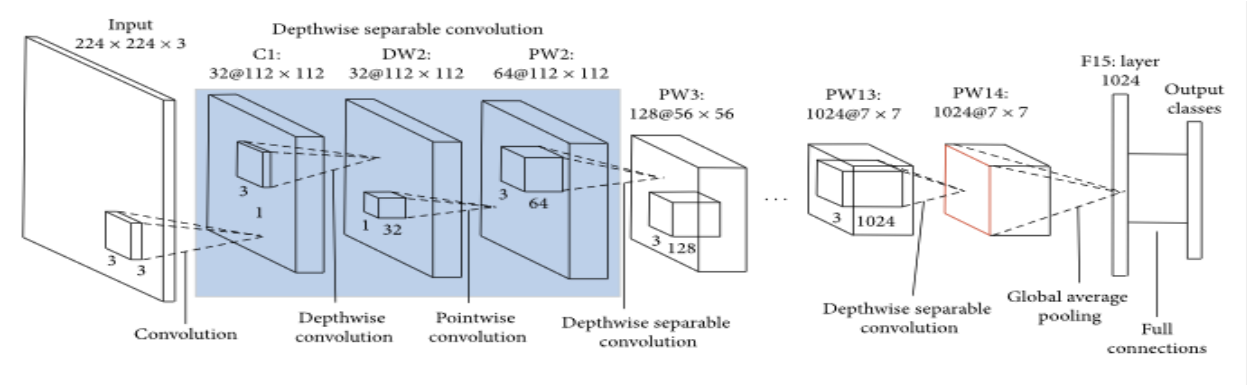
We have a CNN with input layer that takes the shape of the image, then convolution layer with 16 filters and kernel size of 3 and Relu activation, then max pooling layer, then another convolution layer with 32 filters and kernel size of 3 and Relu activation, then max pooling layer, then another convolution layer with 64 filters and kernel size of 3 and Relu activation, then max pooling layer, then flatten the image, then 128 neuron dense layer, then 64 neuron dense layer, then dropout layer then 7 classes output layer. We have the loss as binary crossentropy and Adam optimizer.

CNN #2:

```
def get_model_A2():
    inp = tfl.Input((128, 128 ,3))
    x = tfl.Conv2D(32, (3, 3), activation='relu')(inp)
    x = tfl.MaxPooling2D((2, 2))(x)
    x = tfl.Conv2D(64, (3, 3), activation='relu')(inp)
    x = tfl.MaxPooling2D((2, 2))(x)
    x = tfl.Conv2D(128, (3, 3), activation='relu')(inp)
    x = tfl.MaxPooling2D((2, 2))(x)
    x = tfl.Flatten()(x)
    x = tfl.Dense(128, 'relu')(x)
    x = tfl.Dropout(0.5)(x)
    out = tfl.Dense(7, 'sigmoid')(x)

    model = tf.keras.models.Model(inp, out)

    model.compile(loss="binary_crossentropy",
                  optimizer = tf.keras.optimizers.Adam(learning_rate = 1e-4),
                  metrics=[tf.keras.metrics.BinaryAccuracy()])
    model.summary()

    return model

get_model_A2()
```

We made a CNN with an input layer that takes the shape of the images, convolves it with 32 filters with a 3x3 kernel and a ReLU activation function, then down-samples with a 2x2 MaxPooling window, then convolves it with 64 filters with a 3x3 kernel and a ReLU activation function, then down-samples with a 2x2 MaxPooling window, then convolves it with 128 filters with a 3x3 kernel and a ReLU activation function, then down-samples with a 2x2 MaxPooling window, then flattens the input into a linear list and passes it into a 128-neuron Dense layer with a ReLU activation function, and finally passes it into a 7-neuron Dense layer with a sigmoid activation function and 50% dropout. The model is compiled with a binary cross-entropy loss function, the ADAM optimizer with a learning rate of .0001, and binary accuracy as a metric.

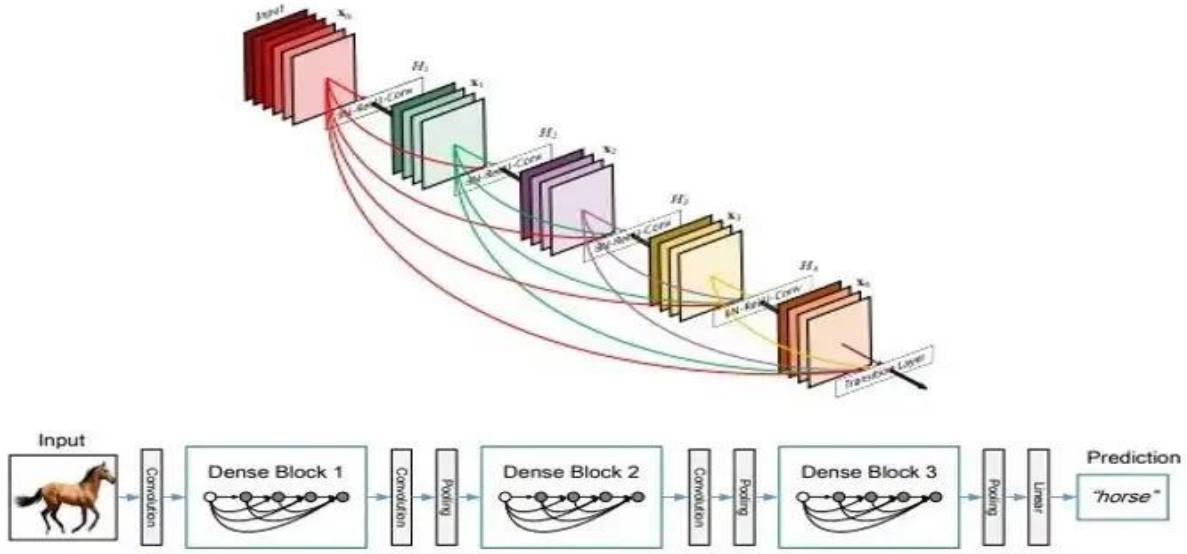
MobileNet:



MobileNets, a class of light weight deep neural networks that are vastly smaller in size and faster in performance than many other popular models. We used pretrained model MobileNet instance with ImageNet weights without including the top, the last 10 layers are trainable. First, we run the model with shape of the images as input. Second, connecting it with 7 neuron sigmoid dense output layer and another where it takes input as the shape of the images and runs the model then connects it with 128 neuron sigmoid dense layer then dropout layer then connects it with 7

neuron sigmoid dense output layer. We have the loss as binary cross entropy and Adam optimizer.

DenseNet121:



We used pretrained model DenseNet121 instance with ImageNet weights without including the top, the last 10 layers are trainable. First, we run the model with shape of the images as input. Second, connecting it with 7 neuron sigmoid dense output layer and another where it takes input as the shape of the images and runs the model then connects it with 128 neuron sigmoid dense layer then dropout layer then connects it with 7 neuron sigmoid dense output layer. We have the loss as binary cross entropy and Adam optimizer.

3. Results:

First, to visualize our data we make a list to print our files we found that we have 7 files in the data:

['sample_submission.csv', 'train_images', 'train_bounding_boxes.csv', 'segmentations', 'train.csv', 'test.csv', 'test_images']. Then, in figure 1 we print all the vertebrae in the human body with which human has a trauma and in which vertebra. We found that the common between most of the patients is second vertebra and in the column patient_overall contain if the patient have a trauma or not 0 means “No” and 1 means “yes”

[6]:

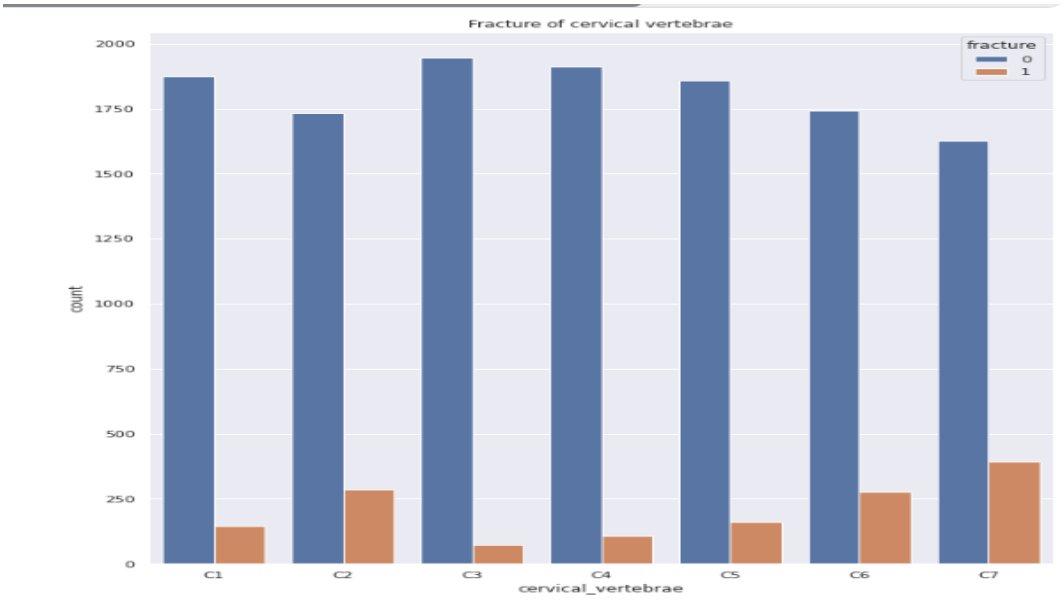
(2019, 9)

	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

Figure 3

In figure 4, we make a segmentation to know which vertebrae has trauma and it shows that c7&c2 is the most common two vertebrae that patients suffer from. About 125 patients suffer from trauma in 1st vertebra, 300 patients suffer from trauma in 2nd vertebra, few patients suffer from trauma in 3rd vertebra and this one is the least vertebra that patients suffer from. The second least vertebra is the 4th one which less than 125 patients suffer from trauma in it. 200 patients suffer from trauma in the 5th vertebra. In the 6th vertebra more than 250 patients suffer from trauma and the 7th one is the common vertebra in trauma between patients.

Figure 4



In figure 5, we make resize for the photo and preprocessing by flatten the photo and transfer it into Hounsfield unit (hu) to remove all the black in it after finding the result of the intensity 0 while the count is between 2000 and 3000

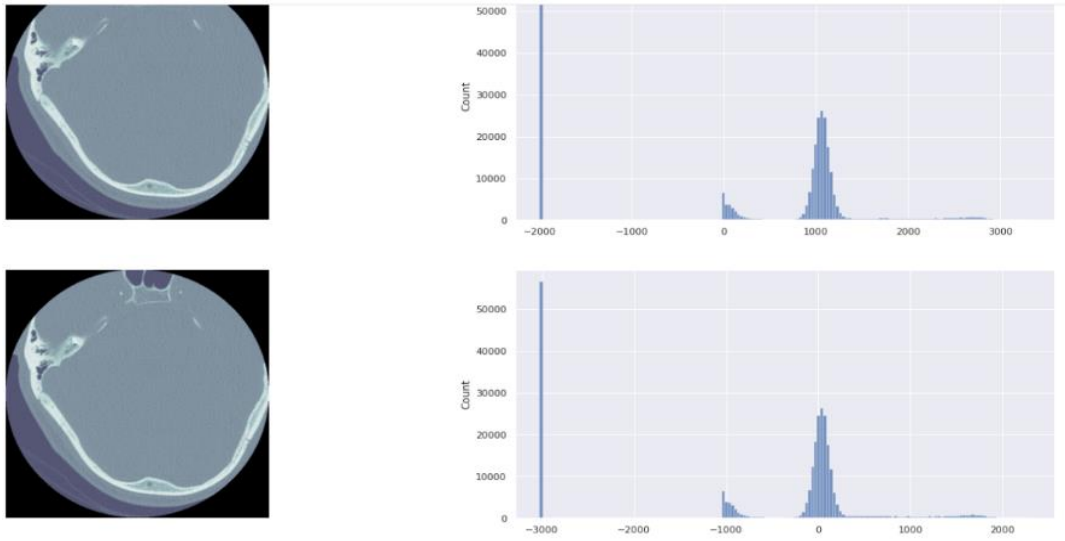


Figure 5

In figure 6, there are some slices for one patient from the data.

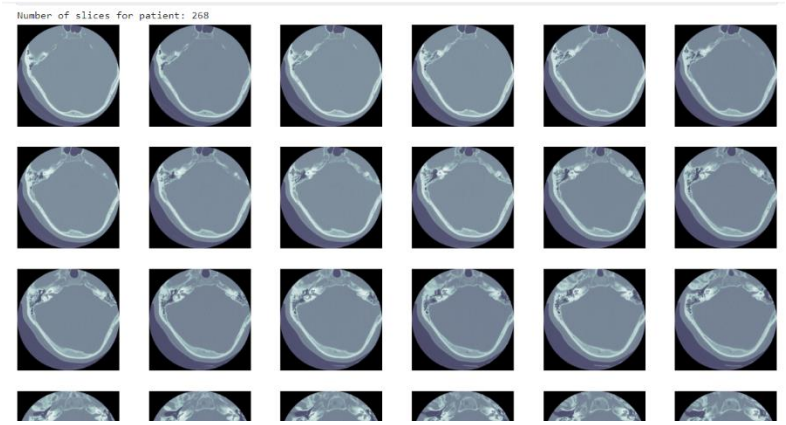


Figure 6

```
df_train[df_train.patient_overall==1].shape[0]
```

```
961
```

Figure 7 visualize how many patients

In fig7 we visualize the data and see how many patients' fracture have through see how many patients include through actual label (patient overall) as finding how many patient have 1 in patient overall

VGG16:

```
model_tr=get_model_tr_VGG16()
model_tr.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_18 (Dense)             (None, 128)               1048704
_________________________________________________________________
dense_19 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_20 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_21 (Dense)             (None, 1)                 33
=================================================================
Total params: 15,773,761
Trainable params: 1,059,073
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 8

In Fig 8, now we visualize our model, and this is the parameter for each layer.

```
Epoch 1/10
21/21 [==============================] - 13s 628ms/step - loss: 4.1635 - accuracy: 0.7180 - val_loss: 1.1988 - val_accuracy: 0.
4547
Epoch 2/10
21/21 [==============================] - 12s 598ms/step - loss: 2.3949 - accuracy: 0.6012 - val_loss: 3.1192 - val_accuracy: 0.
0000e+00
Epoch 3/10
21/21 [==============================] - 13s 672ms/step - loss: 1.1688 - accuracy: 0.1815 - val_loss: 0.8602 - val_accuracy: 0.
1469
Epoch 4/10
21/21 [==============================] - 12s 618ms/step - loss: 0.7309 - accuracy: 0.5179 - val_loss: 0.9509 - val_accuracy: 0.
4156
Epoch 5/10
21/21 [==============================] - 38s 2s/step - loss: 0.9735 - accuracy: 0.7805 - val_loss: 0.0773 - val_accuracy: 1.000
0
Epoch 6/10
21/21 [==============================] - 34s 2s/step - loss: 0.4203 - accuracy: 0.8527 - val_loss: 0.3292 - val_accuracy: 0.925
0
Epoch 7/10
21/21 [==============================] - 32s 2s/step - loss: 0.7269 - accuracy: 0.5216 - val_loss: 0.6640 - val_accuracy: 0.615
6
Epoch 8/10
21/21 [==============================] - 36s 2s/step - loss: 0.3426 - accuracy: 0.8058 - val_loss: 0.2939 - val_accuracy: 0.928
1
Epoch 9/10
21/21 [==============================] - 35s 2s/step - loss: 0.0045 - accuracy: 1.0000 - val_loss: 1.5857 - val_accuracy: 0.760
9
Epoch 10/10
21/21 [==============================] - 36s 2s/step - loss: 1.2415 - accuracy: 0.6949 - val_loss: 0.7478 - val_accuracy: 0.400
0
```

Figure 9

In figure9, we can find the output for the vgg16 as we found the accuracy for validation

XCeption:

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 128, 128, 3)]     0
_________________________________________________________________
xception (Functional)        (None, 2048)              20861480
_________________________________________________________________
dense (Dense)                (None, 7)                 14343
=================================================================
Total params: 20,875,823
Trainable params: 4,765,191
Non-trainable params: 16,110,632
```

Figure 10

In Fig 10, we visualize the XCeption based CNN model.

Basic Neural Network from scratch:

```
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128, 128, 128)     512
_________________________________________________________________
dropout (Dropout)            (None, 128, 128, 128)     0
_________________________________________________________________
dense_2 (Dense)              (None, 128, 128, 128)     16512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128, 128)     0
_________________________________________________________________
dense_3 (Dense)              (None, 128, 128, 64)      8256
_________________________________________________________________
flatten (Flatten)            (None, 1048576)           0
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 7340039
=================================================================
Total params: 7,365,319
Trainable params: 7,365,319
Non-trainable params: 0
```

Figure 11

In Fig 11, we visualize the basic Neural Network model.

```
x_val['prediction'] = val_pred
```

```
x_val['prediction']=x_val.prediction.map(lambda x: 1 if x>0.5 else 0)
```

```
x_val
```

	index	StudyInstanceUID	patient_overall	C1	C2	C3	C4	C5	C6	C7	prediction
0	1556	1.2.826.0.1.3680043.17403	1	0	0	0	0	0	1	1	0
1	526	1.2.826.0.1.3680043.1286	0	0	0	0	0	0	0	0	0
2	393	1.2.826.0.1.3680043.8744	0	0	0	0	0	0	0	0	0
3	1789	1.2.826.0.1.3680043.23847	0	0	0	0	0	0	0	0	0
4	433	1.2.826.0.1.3680043.23611	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
662	522	1.2.826.0.1.3680043.15979	1	0	0	0	1	1	1	1	0
663	513	1.2.826.0.1.3680043.3850	0	0	0	0	0	0	0	0	0
664	81	1.2.826.0.1.3680043.5812	0	0	0	0	0	0	0	0	0
665	1368	1.2.826.0.1.3680043.9506	0	0	0	0	0	0	0	0	0
666	1103	1.2.826.0.1.3680043.6506	1	0	0	0	0	0	0	1	0

667 rows × 11 columns

```
from sklearn.metrics import accuracy_score
val_scores = accuracy_score(x_val['patient_overall'].values, x_val['prediction'].values)
print("Validation Accuracy: %.2f%%"%(val_scores * 100))
```

```
Validation Accuracy: 52.02%
```

Now we add the row of prediction to train it (0,1) and get the output as a probability to compare it with the actual label to test the accuracy of the prediction then we get the accuracy = 52.02%

4. Discussion:

The accuracies for all our data

A1: 0.87, A2: 0.98,

M1: .74, M2: 0.9,

OA1: 0.95, OA2: 0.91

OK1: 0.57, OK2: 0.96, OK3: 0.924

N1: 0.9, N2: 0.95

The overall accuracy was 0.883971732676029

The best model that fits our data is A2 as its model from scratch:

Conv2d -> Maxpooling -> Flatten -> Dense 128 -> Dropout -> Dense

Its accuracy was 0.98.

5. Conclusion

The task of fracture detection in cervical spine CT images is extremely difficult. We presented a machine learning model based on our data, and after getting the result for all patients, we found that the common fracture in the 2^{nd} and 7^{th} vertebra between all patients. We encourage the research community to work actively on finding a real study to use AI in detecting the fracture.

5. References:

[1] Hojjat Salehinejad, Edward Ho, Hui-Ming, Priscila Crivellaro, "DEEP SEQUENTIAL LEARNING FOR CERVICAL SPINE FRACTURE DETECTION ON COMPUTED TOMOGRAPHY IMAGING" 2021

[2] John H Bland and Dallas R Boushey, "Anatomy and physiology of the cervical spine," in Seminars in arthritis and rheumatism. Elsevier, 1990, vol. 20, pp. 1–20

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[4] Hojjat Salehinejad, Shahrokh Valaee, Tim Dowdell, and Joseph Barfett, “Image augmentation using radial transform for training deep neural networks,” in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 3016–3020.

[5] Hojjat Salehinejad, Sumeya Naqvi, Errol Colak, Joseph Barfett, and Shahrokh Valaee, “Cylindrical transform: 3d semantic segmentation of kidneys with limited annotated images,” in 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2018, pp. 539–543.

[6] Hojjat Salehinejad, Errol Colak, Tim Dowdell, Joseph Barfett, and Shahrokh Valaee, “Synthesizing chest xray pathology for training deep convolutional neural networks,” IEEE transactions on medical imaging, vol. 38, no. 5, pp. 1197–1206, 2018.

[7] Anil Kalra, “Developing fe human models from medical images,” in Basic Finite Element Method as Applied to Injury Biomechanics, pp. 389–415. Elsevier, 2018.

[8] Sunil L Bangare, Amruta Dubal, Pallavi S Bangare, and ST Patil, “Reviewing otsu’s method for image thresholding,” International Journal of Applied Engineering Research, vol. 10, no. 9, pp. 21777–21783, 2015.

[9] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” arXiv preprint arXiv:1602.07261, 2016.

[10] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.

[11] Anna-Lena Robinson, Anders Moller, Yohan Robinson, ¨ and Claes Olerud, “C2 fracture subtypes, incidence, and treatment allocation changes with age: a retrospective cohort study of 233 consecutive cases,” BioMed research international, vol. 2017, 2017.