Final Write-Up: RSNA Cervical Spine Fractures Detection

Motivation

Our research question is "*How can we quickly detect and locate vertebral fractures in the cervical spine from CT scans to aid in the prevention of neurologic deterioration and paralysis after trauma?*"

Fractures of the cervical spine, or broken bones in the neck, are associated with around 56% of all spinal cord injuries (McMordie et al, 2022). Approximately 250,000 to 500,000 people suffer spinal cord injuries each year in the US alone according to WHO (2013). According to the American College of Radiology (Radiology Key, 2019), it is estimated that over one million patients with blunt trauma may sustain potential cervical injuries each year. These fractures can lead to paralysis, or even death. Given the high number of people impacted and the severe nature of these injuries, it is important to find ways to improve the detection and diagnosis of cervical fractures.

As shown by our literature review, there is a high rate of inefficiency and ineffectiveness in detecting these fractures through radiology. The use of artificial intelligence, especially with CT scans in the diagnostic process, can help reduce the relatively high diagnostic discrepancy between radiographic interpretation and the final read by board-certified radiologists due to human error. (Guermazi et al, 2021) This project aims to use machine learning to accurately detect cervical spine fractures, with the goal of increasing efficiency and reducing costs while matching the accuracy of a good radiologist.

Literature Review

Fracture detection is primarily carried out by one of two imaging techniques/methods; Radiography or Spiral Computed Tomography (SCT). Radiography uses X-rays to produce images of the inside of the body, including bones. This method can be used to visualize fractures and any other abnormalities in the bones. Spiral Computed Tomography (SCT) is a type of X-ray imaging that uses a computer to produce detailed, cross-sectional images of the body. SCT produces more detailed images and can be used to visualize fractures in greater detail. MRI is reserved for ligamentous and other soft tissue structures (Kumar et. al, 2016). Emergency physicians have two main tools available to them to determine the need for imaging techniques namely the Canadian C-spine rule and the National Emergency X-Radiography Utilization Study (NEXUS) criteria (Michaleff et. al, 2012). Essentially these tools aim to reduce the need for medical imaging due to its cost and efficiency.

Radiography is the more common method of the two, and yet it still leads to many false negatives in terms of fracture detection accounting for about 24% of harmful diagnostic errors in the emergency room (Guermazi et al, 2021). According to a retrospective study of 480 patients, artificial intelligence (AI) helped improve radiographic fracture detection by 10.4% without specificity reduction (Guermazi et al, 2021). For patients with moderate to high risk of cervical fractures, CT is generally regarded as the preferred and more economical strategy. (Blackmore et al, 1999). Therefore, in our project, we focus on using AI/ deep learning (specifically CNN or Convolutional Neural Networks) to further improve the accuracy of cervical fracture detection via CT scans.

Dataset

Data Source:

https://www.kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection/overview

The data we are working with consists of CT scans of cervical spines of patients and the goal is to identify whether a given vertebra and therefore the overall patient has a fracture. We got the data from Kaggle (*RSNA 2022 Cervical Spine Fracture Detection*, n.d.). The data consists of:

1. 2019 directories of train images. Each folder corresponds to a single study and has between 200 and 400 DICOM images which represent ≤ 1 mm image slices of the cervical spine (neck). The slices correspond to top-level X-rays of the neck from the base of the skull to the beginning of the upper back vertebrae. Randomly selected slices are shown in Figure 6.
2. A CSV file corresponding to each of the directories of the train images indicates whether each of the vertebrae C1-C7 has a fracture or not and whether the overall patient has a fracture. A detailed description of each of the variables can be found here and is shown in Table 1.
3. Three directories of test images containing about 1500 images and a CSV file for three directories indicating which vertebrae to predict on. However, the provided test directories do not correspond to the test CSV directory names. We, therefore, create a test dataset by randomly sampling five directories.
4. 87 segmentation files which correspond to masks of 87/2019 directories. The mask corresponds to pixel-based labels of the images. The masks are labeled 0 to 19 with 1 to 7 corresponding to masks of C1 to C7 vertebrae, 0 corresponding to no mask (pixels

surrounding image mask), and 8 to 19 corresponding to upper back vertebrae. Side-level masks are shown in Figure 7 and top-level masks are shown in Figure 8.

Methodology

One of the most challenging parts of this project was determining the best way to handle over 300 GB of data. We explored several options including finding a smaller JPEG version of the images (about 57 GB) and uploading these to Google Colab, buying an external hard drive with 5TB of space and running the models locally, and using Kaggle notebooks to load the data directly from Kaggle. We ended up using each of these options for different purposes. The local version was used to explore the images in-depth and generate videos that show the progression of the skulls overlaid on each other to generate a complete view of the neck. The Google Colab option was used to run an exploratory Analysis of the dataset as shown and discussed in the Analysis section below and to work on segmentation. And we used the Kaggle option to run the classification algorithms to determine the probability of whether a vertebra has a fracture or not.

The fracture detection process was a multi-step process. The first step was to use the 87 segmentation mask files and the corresponding directories to train the model for semantic segmentation. Semantic segmentation is a pixel-level classification where the output corresponds to a high-resolution image of the same size as the input image and assigns pixels that belong to each class based on the image (Lamba, 2019). In this case, the segmentation dataset is labeled and we are using semantic segmentation to train a model to classify what each of the vertebrae looks like. Figure 8 in the analysis section shows an image, its segmentation mask and the mask overlaid over the image. The labeled segmentation masks were provided by Kaggle and the goal

is to make it easier for fracture detection. Specifically, given a study with CT scans of the vertebrae, the model can identify which of the slices in the study correspond to which vertebrae. Then, with this information, we can train a fracture detection model and combine the results to determine which vertebrae have a fracture. For the segmentation, we use a UNET model which is an encoder-decoder model that was developed in 2015 for use in BioMedical Imaging (to cite). Being an end-to-end fully connected CNN, UNET is appropriate for segmentation because; it has no Dense layers (which means images of any size can be input), uses downsampling to learn all the necessary information about the image in terms of what makes up the image and uses upsampling to learn where each of the pixels of the images is located in a process known as precise localization (Lamba, 2019). The upsampling is the crucial bit of the UNET because a normal CNN will lose information after downsampling and pooling.

The second step is to train a classification model to identify the fractures. Given the complexity of the dataset, the difficulty in identifying fractures, and the great class imbalance, we went straight to deep learning for the classification tasks. With shallow learning methods, we would need to extract features from the images using a method such as PCA which means there would be some unexplained variance in the model (Chauhan et al., 2019). A lot of research has gone into why deep learning is better for image classification tasks and one of the main arguments for it is the use of hidden convolutional layers to match the pixel values of the images to a grid and identify where the image lies within the grid. This then helps the model learn the differences between images and identify each image based on the labels. This is a high-level summary of how CNNs work. The other major argument for CNNs and neural networks, in general, is the ability to update weights as the model gets new information and more data on how

the images should look like. Therefore, we went with a selection of 4 models: a basic CNN, and three pretrained transfer learning models: InceptionV3, ResNet50, and EfficientNetB5. Each of the pre-trained models was chosen based on research about the most commonly used models in medical image classification and especially when dealing with fractures (Emon et al., 2022). We also considered the efficiency metrics detailed in the Kaggle website in terms of the size of the model, the time it takes to run among others.

The basic CNN consists of a sequential model with multiple hidden convolution blocks and artificial Dense blocks as part of the output layers. The advantage of using this CNN is that it does not have many layers and it does not use transfer learning making it quite fast compared to other models. The disadvantage of this CNN is that it needs to learn the weights for the first time which means it starts very low and has significantly less variance and more bias toward the specific data. It is therefore more prone to overfitting the data. The remaining three pre-trained models share a similar concept although their architectures are different. Specifically, we use transfer learning when using them which means that these models have been trained on millions of different images and have had their weights trained. This makes the training and updating weights significantly more efficient and less prone to overfitting because the model has seen a wide variety of images. It also means that the models are able to quickly distinguish images because they have familiarity with image datasets. With that said, the pre-trained models have the big disadvantage of being very deep with layers ranging from 16 to over 500. This large depth means that the models take much longer to train relative to the basic CNN and especially without GPU acceleration. Therefore, when choosing the models to try, applicability and efficiency are very important.

The second model, InceptionV3 is a collection of CNNs and Pooling layers that have been optimized to reduce the number of parameters generated and to increase computational efficiency (Kurama, 2020). Specifically, the convolution layers are factorized, reduced in size and made to run in parallel while preserving the features of the images. Additionally an auxiliary classifier is placed between the layers for faster updates of the weights and to reduce the loss. The main disadvantage of this model is understanding how to fine-tune given the very many layers involved. The third model is the EfficientNetB5 model, was developed to handle scaling of CNNS while improving accuracy and efficiency based on a baseline neural network (Kumar, 2020). The result is a family of models that are wider, deeper and higher resolution while maintaining high time and computational efficiency and accuracy. Based on their initial analysis and test on the ImageNet dataset, the researchers found that EfficientNetB5 was using much lower parameters than previous models such as InceptionV3 while achieving higher accuracy. The disadvantage however was that the model is very deep and therefore takes a lot longer per epoch even though it achieves high accuracy. The last model, ResNet50, is built to improve efficiency by using residual mapping and fewer filters so that accuracy does not saturate and degrade ie overfitting (Kurama, 2020). The residual mapping allows for shortcut connections between layers and increases dimensions making the network more robust without adding parameters. Amongst the three models, ResNet50 has the shortest depth making it quite fast but not as robust as the EfficientNetB5. For all these models, we only train them for 10 epochs and use the validation loss for early stopping in the loss is not improving. Ideally, we should run for at least 20 epochs to give the models more time to learn but given each epoch takes about 12 minutes and we do not have GPU acceleration, we determine that 10 epochs are sufficient to make predictions and compare the models.

For the threshold metrics, we carefully considered metrics that would deal with the high-class imbalance identified in the analysis section. The F1 score and ROC score helps tells us whether the models are able to overcome the issue of class imbalance. The F1 score is the harmonic convergence of the recall and precision. The precision tells us the fraction correctly predicted positives and the recall tells us how well the positive class was predicted. Generally, a model with high F1 means that it is able to balance precision and recall and can clearly tell the positives from the negatives. It also means there are low false negatives and false positives (Brownlee, 2020). AUC score calculates the false positive and false negative rate and gives a value between 0 and 1 where a value near 1 means the classifier does a good job distinguishing between the positive and negative classes while near 0.5 means it is no better than random and anything below that is not useful (Brownlee, 2020). If these two metrics are high, then the model is able to overcome the class imbalance and identify the small subset of positives amidst a sea of negative classes.

Analysis

This section focuses on exploratory analysis of the metadata, images, and segmentation files. We include visuals and discuss the insights from each of them in relation to the analysis.

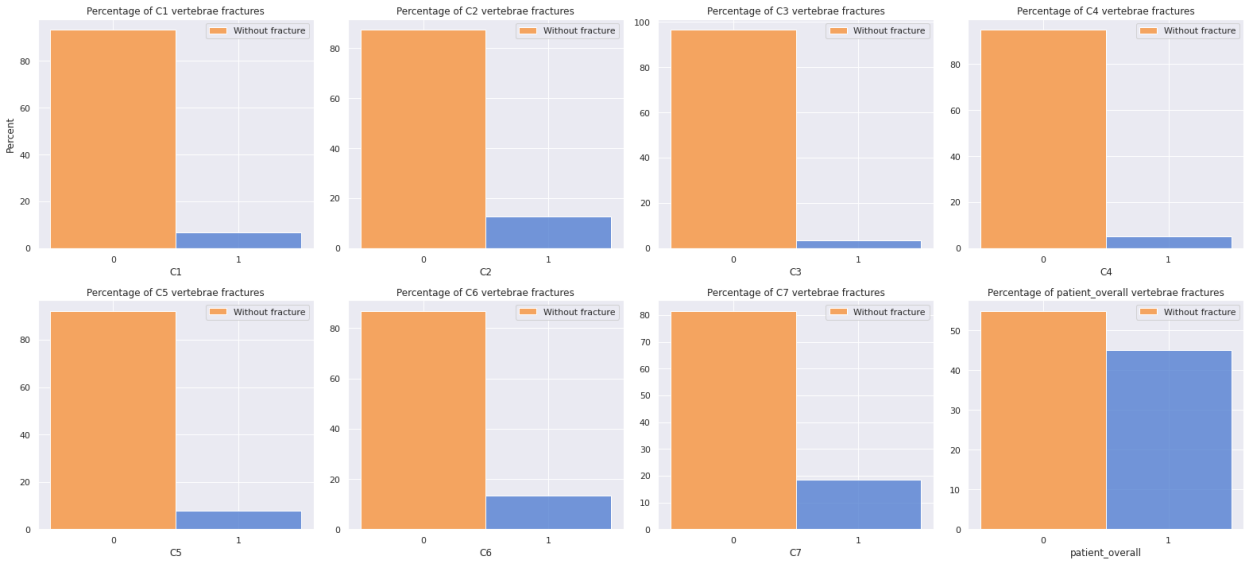

Figure 1: Percentage of fractures in each of the vertebrae and in the overall patient.

From Figure 1, we can see that C1, C3, C4, and C5 vertebrae have the lowest proportion of fractures. On the other hand, we can see that C2, C6, and C7 have comparatively higher proportions of fractures. C7 has the highest proportion of fractures compared to other vertebrae because it's the vertebrae that connect the neck with the upper back and is highly associated with back injuries. Back injuries are much more common than neck injuries which explain this difference. C3 and C4 have particularly low fractures because of the curvature of the neck which means that they are protected by the other vertebrae in the event of a neck injury. Figure 2 below shows the total number of fractures per vertebrae ordered from lowest to highest. It's a more compact side-by-side comparison that has the same insights as Figure 1.

The patient overall corresponds to whether or not a patient has a fracture in any one of their vertebrae. Therefore, as long as a patient has a fracture in one vertebra, then the overall

conclusion is that they have a fracture. This is why the patient overall column has significantly higher fractures than the individual vertebrae.

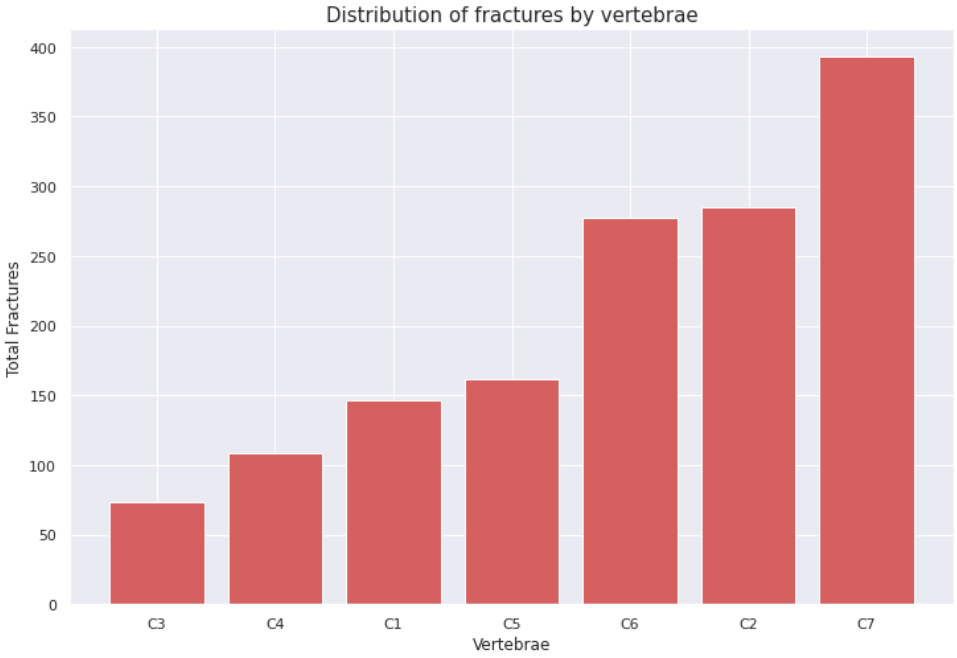

Figure 2: Total number of fractures per vertebrae across all studies

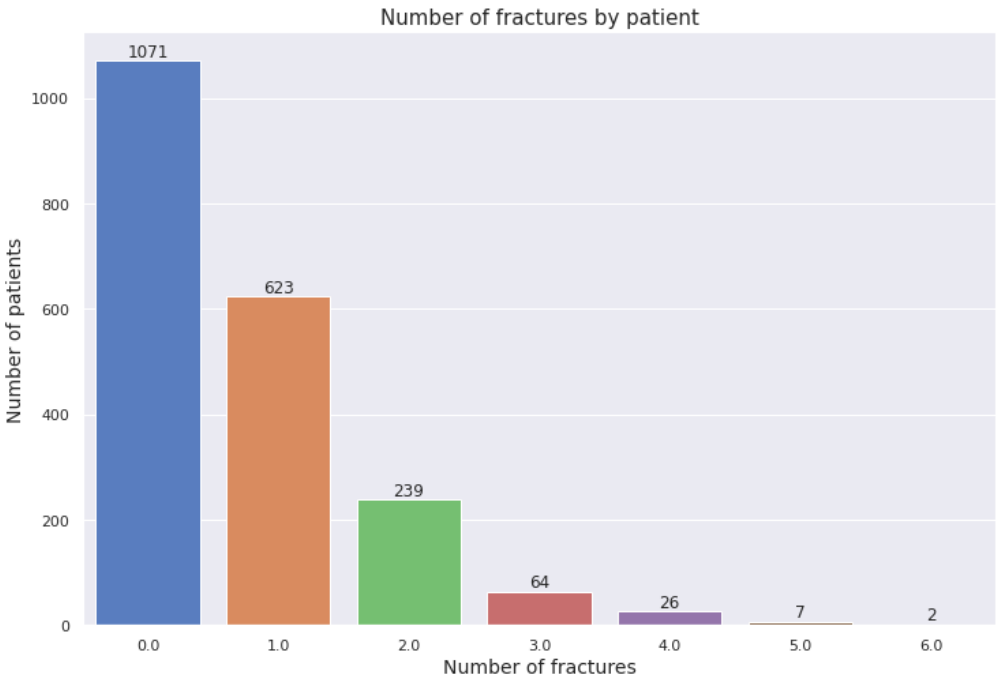

Figure 3: Number of fractures by number of patients (distribution of fractures by patients)

By looking at the distribution of the fractures by patients, we can see that most patients do not have fractures. That is out of the 2019 studies, about half of them do not have a single fracture while the remaining half have at least one fracture with the majority of patients having only one fracture. It's very rare to have 5 or more fractures as chances of survival with that many fractures are very low. A good proportion of patients (about 10%) have two fractures which make sense as seen in the correlation plot in Figure 4 below.

Figure 4: Correlation plot showing the relationship between the vertebrae

We can see that C7 us highly correlated with the patient overall. This makes sense because it is the vertebra with the highest number of fractures from patient data. That means that if a patient has a fracture on C7 then they have a fracture overall. However, this is not new information. What is interesting is the correlation between C3 and C4, C4 and C5, and C5 and C6. There is a moderate level of correlation between these vertebrae. This makes sense given how close they are to each other. If there is an injury in C3 there is likely to be an injury in C4. Interestingly, this moderate level of correlation is not observed with C1 and C2, C2 and C3, and C6 and C7 despite the vertebrae being close together. Again, this may be because of the curvature of the neck such that the middle vertebrae are harder to get fractured but when they do, they get fractured together. These moderate correlations can somewhat explain why a good proportion of patients have two fractures.

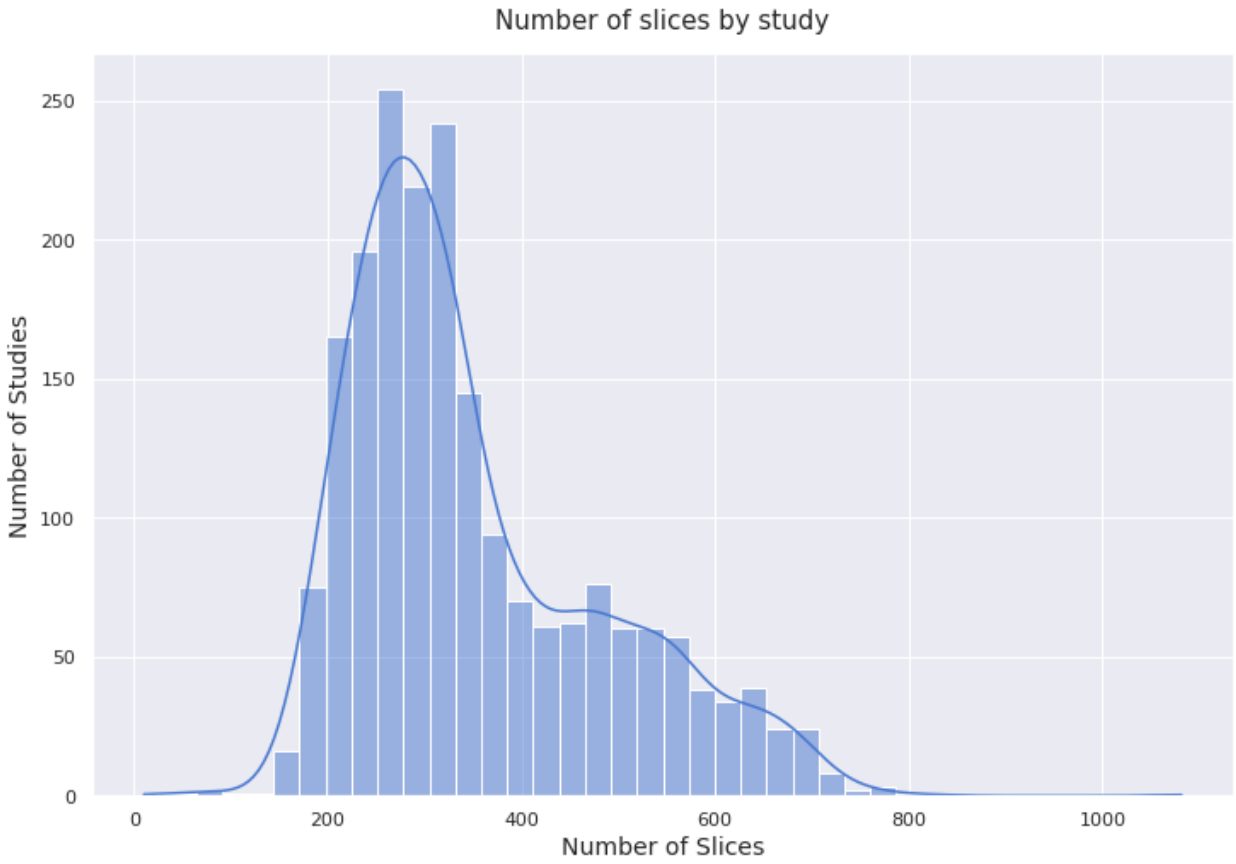

Figure 5: Distribution of slices by studies.

The distribution of slices across studies has a slight bimodal distribution and a right-skew. The median number of slices per study is 314 while the average number of slices per study is 352. This tells us that on average, there are about 350 slices in a given study and therefore we can expect a sufficient number of images to train the model despite the fact that there are only 2019 total studies provided.

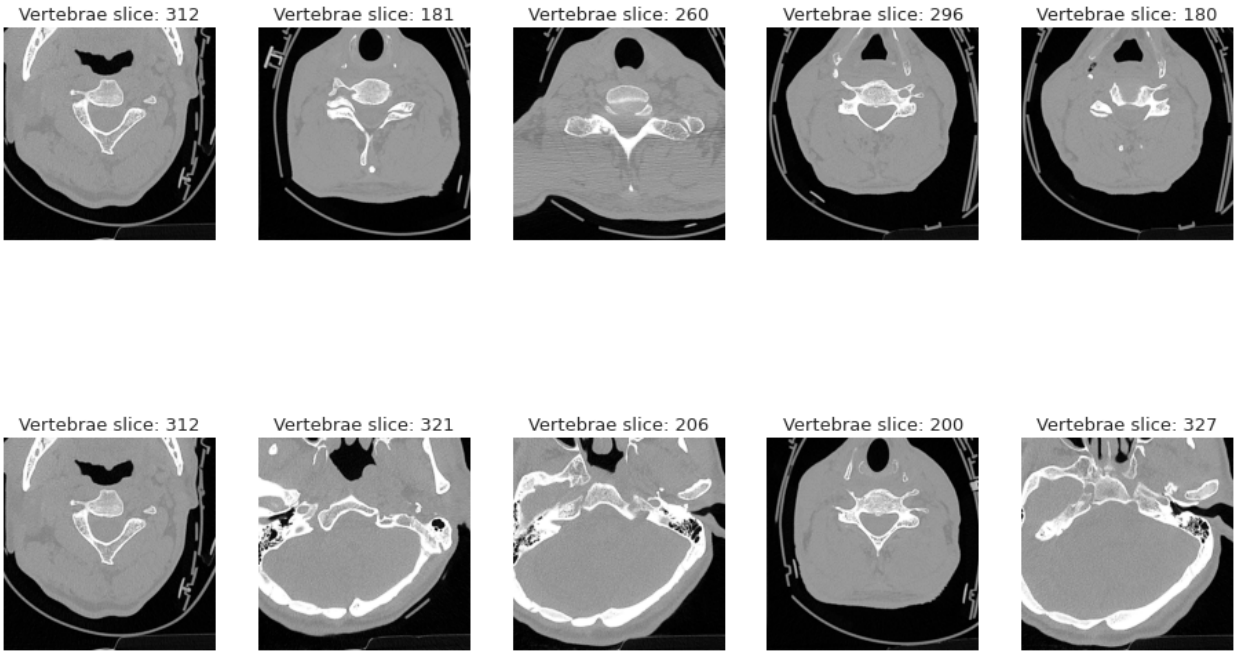

Figure 6: Randomly selected top-level slices of the cervical spine.

By randomly selecting images from one of the studies, we can see what some of the scans look like and start to identify cervical vertebrae as in Figure 6. The slices represent ≤ 1mm of images taken during the CT scan and overlaying the images on each other produces a holistic picture of the entire scan from the base of the skull to the upper back vertebrae as in this video. Some of these images may have fractures but it is hard to know if you are not a seasoned radiologist. It's also harder to distinguish which images correspond to which vertebrae if you do not have knowledge of this. The segmentation masks are shown in Figure 7 help with this process.

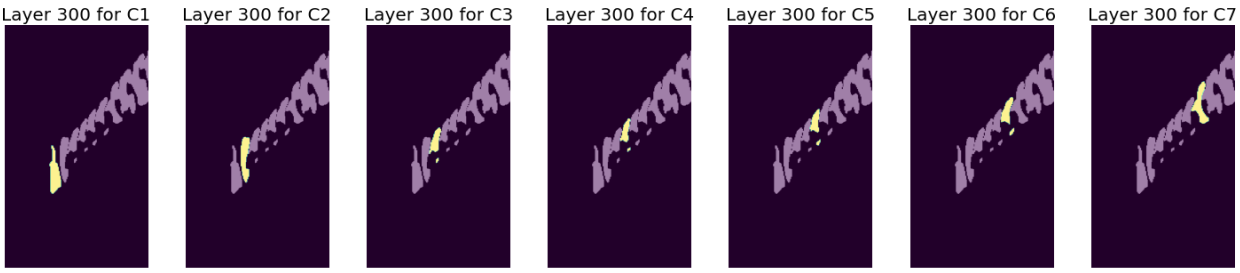

Figure 7: Side view of the neck showing the position of each of the vertebrae based on segmentation masks.

Figure 7 shows the segmentation masks from a slice of a layer of an image. These segmentations represent where each pixel of each vertebra is where yellow is the given vertebra and purple is all the other vertebrae. This is a neck-side view of all the vertebrae for a given slice at a given layer. We got this view by taking a layer of a 3D image from a randomly sampled image whose segmentation mask we have from the segmentation dataset. Given it is a 3D image, there are multiple views including the neck view shown, a top-level view, a vertical front, and back view, and the bottom-level view. This segmentation is important because we can teach the model what each of the vertebrae looks like from different perspectives. This process is called image segmentation and we will specifically use semantic segmentation (more on this later). This then helps us label all the training images before classifying whether they have a fracture or not. A clearer picture of the usefulness of segmentation is shown below.

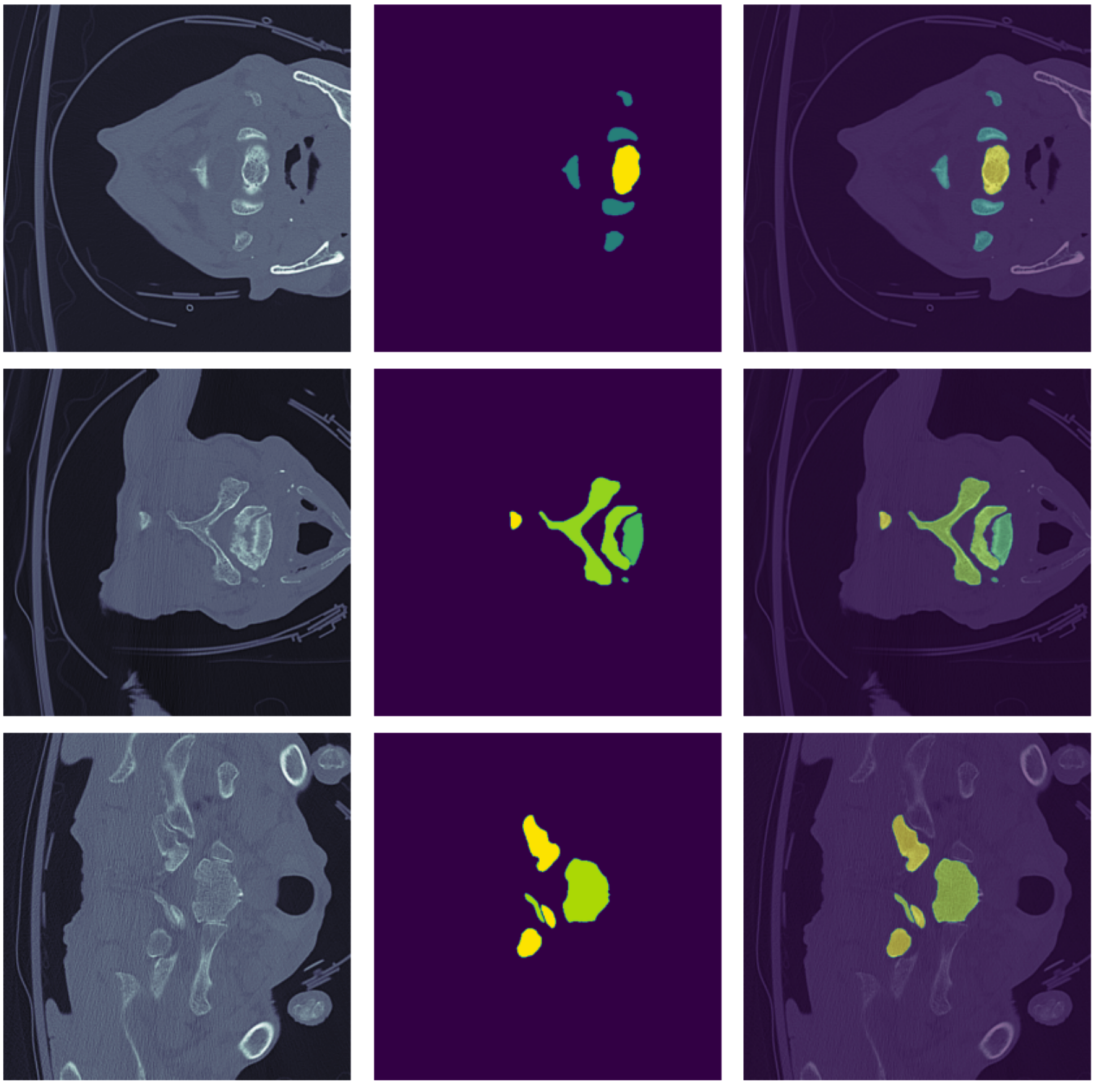

Figure 8: Random vertebrae images, the corresponding mask, and the image overlaid by the mask

Figure 8 illustrate the importance of image segmentation. The masks in the middle column have labels where each color corresponds to a different vertebra. The left column corresponds to the unlabeled image slices and the right column corresponds to the image slices overlaid with the mask to identify which vertebrae are in the image based on the labeled mask. A

video of the masks overlaid on the images can be found here. With these datasets available, we can train a model to identify what each of the vertebrae looks like and predict unseen vertebrae.

Results and Discussions

From the UNET model, we were not able to get any results. The first epoch took more than 1 hour to complete and when it did, the result was 0 accuracy and 0 loss. This means that the model would need a lot of epochs to get some results that would be somewhat reliable. It is also possible that we configured the data and model incorrectly which led to the model incorrectly processing the input and therefore zero accuracy. We can do without the segmentation module as the classification model can still learn the types of fractures given it will be fed similar-looking vertebrae with some having fractures and some not having. Eventually, the classification model learns what C1 with a fracture looks like and can predict its probability. The segmentation model is important in the long-term to improve the performance of the classification model and to make work easier for radiologists since they still need to go through the hassle of identifying which are the C1-C7 vertebrae.

After training all 4 models, we got the results shown in Table 2 (appendix). To our surprise, the basic CNN had the best performance besides having the lowest number of layers. It had the lowest loss and highest F1 and AUC scores. Additionally, it was also the fastest model to run and therefore the best overall. One of the potential reasons for this unexpected result is that we understood how to configure a basic CNN model and therefore optimized the configuration to match the needs of our image. The pre-trained models have weights and are much harder to customize because one has to understand what is going on. They would probably perform better

if we fine-tuned the hyperparameters such as changing the learning rate for the optimizer, changing the optimizer type, or using even more layers than present in the models. Another reason could be the fact that the pre-trained models are very deep with many layers and need at least 50 epochs to improve performance. However, as previously mentioned, the runtime for each epoch is too long without GPU acceleration and so we wanted to see how each would perform with the bare minimum tuning.

The accuracy and loss models are shown in Figures 9 to 12 paint a good picture of the performance of the models. It's important to note that accuracy is not a metric we consider in this case because of the class imbalance and so are sure the models will overfit and have high accuracy. The most interesting metrics are the F1 and AUC as mentioned before and the loss. The loss is important because it helps us know whether the models are updating the weights correctly. The lower the loss, the better the model will perform in classification. The high AUC of the basic CNN tells us that the model has a low false positive and false negative rate and the high F1 score tells us it does a very good job differentiating between fractured vertebrae and those without fractures. This makes it a very reliable model especially since we want to avoid false negatives ie we do not want to misclassify the patient as not having a fracture when they actually have a fracture. We also want to reduce false positives because we do not want patients to undergo expensive treatment for fractures they did not have. Interestingly, the EfficientNetB5 model had very poor performance: it had a very high loss and a very low F1 score. It also had low accuracy relative to the other models. Given it's a model developed to be efficient while being deeper and wider, we probably needed to run it for a lot more epochs and configure it differently to achieve better results.

After determining the basic CNN had the best performance, we evaluated the model on 5 directories of unseen test data to predict the probability that a given vertebra has a fracture. The results of these predictions are summarized in Table 4. Ideally, we should have improved on this model using k-fold cross-validation but training time and resources were not in our favor. The benefit of k-fold cross-validation would be to create a variance in the dataset and train the model on more features. This would make the model more robust to unseen data and able to accurately detect whether a vertebra has a fracture or not with high precision. The results show that the basic CNN model does a very good job of predicting the probability that a given vertebra has a fracture. This can be seen by comparing Table 3 which shows the true values and Table 4 which shows the predicted values.

Conclusions

Although we were not able to successfully get results from the UNET segmentation model, we were able to set it up for training and get results from one epoch. A future endeavor is getting the model to be more efficient and configuring it correctly so that it can run at least 10 epochs. With this model, we can combine it with the classification model to help radiologists determine which vertebrae have fractures.

From trying out different models, the best-performing model turned out to be the basic CNN model. As mentioned, there is room for improvement in the model using cross-validation but the metrics are already very impressive. The model also handled the class imbalance very well as measured by the AUC and F1 sores.

The combination of the classification and the segmentation models would save radiologists a lot of time because all they would need to do is given the model a scan and it would return a prediction of the type of vertebrae and which if any have fractures. The radiologists would not need to go through an average of 350 images per patient and would focus their efforts on diagnosis and treatment. To make their work even easier, an extension of this would be to create a website where radiologists simply upload DICOM files and they get back the predicted results. The cleaning and pre-processing steps for the DICOM would be very similar to what we have done. As radiologists add data, the data could be used to train the models, even more, to increase their accuracy while avoiding misdiagnosis.

References

Exploratory Analysis Assignment: https://docs.google.com/document/d/1KZn7WBejTkBxu6YnCYhGDSAwp3qOXfbymmxv6WWlypI/edit#heading=h.up2wakf2yq2g

Accuracy of the Canadian C-spine rule and NEXUS to screen for clinically important cervical spine injury in patients following blunt trauma: A systematic review—PMC. (n.d.). Retrieved December 16, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3494329/

Blackmore, C. C., Ramsey, S. D., Mann, F. A., & Deyo, R. A. (1999). Cervical Spine Screening with CT in Trauma Patients: A Cost-effectiveness Analysis. *Radiology*, *212*(1), 117–125. https://doi.org/10.1148/radiology.212.1.r99jl08117

Brownlee, J. (2020, January 7). Tour of Evaluation Metrics for Imbalanced Classification. *MachineLearningMastery.Com*. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

Chauhan, S., Vig, L., De Filippo De Grazia, M., Corbetta, M., Ahmad, S., & Zorzi, M. (2019). A Comparison of Shallow and Deep Learning Methods for Predicting Cognitive Performance of Stroke Patients From MRI Lesion Images. *Frontiers in Neuroinformatics*, *13*. https://www.frontiersin.org/articles/10.3389/fninf.2019.00053

Emon, M. M., Ornob, T. R., & Rahman, M. (2022a). Classifications of Skull Fractures using CT Scan Images via CNN with Lazy Learning Approach. *Journal of Computer Science*, *18*(3), 116–129. https://doi.org/10.3844/jcssp.2022.116.129

Emon, M. M., Ornob, T. R., & Rahman, M. (2022b). Classifications of Skull Fractures using CT Scan Images via CNN with Lazy Learning Approach. *Journal of Computer Science*, *18*(3), 116–129. https://doi.org/10.3844/jcssp.2022.116.129

Emon, M. M., Ornob, T. R., & Rahman, M. (2022c). Classifications of Skull Fractures using CT Scan Images via CNN with Lazy Learning Approach. *Journal of Computer Science*, *18*(3), 116–129. https://doi.org/10.3844/jcssp.2022.116.129

Guermazi, A., Tannoury, C., Kompel, A. J., Murakami, A. M., Ducarouge, A., Gillibert, A., Li, X., Tournier, A., Lahoud, Y., Jarraya, M., Lacave, E., Rahimi, H., Pourchot, A., Parisien, R. L., Merritt, A. C., Comeau, D., Regnard, N.-E., & Hayashi, D. (2022). Improving Radiographic Fracture Recognition Performance and Efficiency Using Artificial Intelligence. *Radiology*, *302*(3), 627–636. https://doi.org/10.1148/radiol.210937

Image segmentation using UNet. (n.d.). Retrieved December 16, 2022, from https://kaggle.com/code/mineshjethva/image-segmentation-using-unet

Kumar, D. V. (2020, June 19). *Implementing EfficientNet: A Powerful Convolutional Neural Network*. Analytics India Magazine. https://analyticsindiamag.com/implementing-efficientnet-a-powerful-convolutional-neural-network/

Kumar, Y., & Hayashi, D. (2016). Role of magnetic resonance imaging in acute spinal trauma: A pictorial review. *BMC Musculoskeletal Disorders*, *17*, 310. https://doi.org/10.1186/s12891-016-1169-6

Kurama, V. (2020, June 5). *A Guide to ResNet, Inception v3, and SqueezeNet*. Paperspace Blog. https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/

Lamba, H. (2019, February 17). *Understanding Semantic Segmentation with UNET*. Medium. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

Maynard-Reid, M. (2022, February 21). U-Net Image Segmentation in Keras. *PyImageSearch*. https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/

Munera, F., Rivas, L. A., Nunez, D. B., & Quencer, R. M. (2012). Imaging Evaluation of Adult Spinal Injuries: Emphasis on Multidetector CT in Cervical Spine Trauma. *Radiology*, *263*(3), 645–660. https://doi.org/10.1148/radiol.12110526

Pretrained Models for Transfer Learning in Keras for Computer Vision. (n.d.). DEV Community 👩‍💻👨‍💻. Retrieved December 16, 2022, from https://dev.to/amananandrai/pretrained-models-for-transfer-learning-in-keras-for-computer-vision-5eei

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation* (arXiv:1505.04597). arXiv. http://arxiv.org/abs/1505.04597

Rosebrock, A. (2017, March 20). ImageNet: VGGNet, ResNet, Inception, and Xception with Keras. *PyImageSearch*. https://pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

RSNA 2022 Cervical Spine Fracture Detection. (n.d.). Retrieved December 16, 2022, from https://kaggle.com/competitions/rsna-2022-cervical-spine-fracture-detection

Small, J. E., Osler, P., Paul, A. B., & Kunst, M. (2021). CT Cervical Spine Fracture Detection Using a Convolutional Neural Network. *AJNR: American Journal of Neuroradiology*, *42*(7), 1341–1347. https://doi.org/10.3174/ajnr.A7094

Spinal cord injury. (n.d.). Retrieved December 16, 2022, from https://www.who.int/news-room/fact-sheets/detail/spinal-cord-injury

Tan, M., & Le, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* (arXiv:1905.11946). arXiv. http://arxiv.org/abs/1905.11946

Team, K. (n.d.). *Keras documentation: Keras Applications*. Retrieved December 16, 2022, from https://keras.io/api/applications/

Themes, U. F. O. (2019, August 20). Spinal Trauma. *Radiology Key*. https://radiologykey.com/spinal-trauma-4/

Tutorial Keras: Transfer Learning with ResNet50. (n.d.). Retrieved December 16, 2022, from https://kaggle.com/code/suniliitb96/tutorial-keras-transfer-learning-with-resnet50

Appendix 1: Statement of Contribution

Haitham: Worked on methodology section (segmentation), analysis of image data (configured and displayed image slices and masks, produced videos of image slices), results and discussion (built UNET model, configured all images for loading into models)

Eisha: Worked on introduction, literature review, Blog Post, general research on pre-existing methodologies

Stevedavies: Worked on the dataset section, methodology section (classification models), analysis of metadata (bar graphs and correlation plots), results and discussion (classification model setting, optimizing datasets for the model, evaluation of models and prediction on the unseen dataset), conclusion and appendices.

Appendix 2: Important Resources

Main Github: https://github.com/Hadavand-s-Minions/rsna-cervical-spine

Analysis Notebooks:

https://github.com/Hadavand-s-Minions/rsna-cervical-spine/tree/main/notebooks

Assignment Directory:

https://github.com/Hadavand-s-Minions/rsna-cervical-spine/tree/main/assignments/3_Final_Project

Appendix 3: Images and Tables

Variable Name	Label/Description	Variable Codes	Variable Format	Value Indicating Missing Data
StudyInstanceUiD	Study ID for each patient scan (ID corresponding to folders containing multiple images)	1.2.826.0.1.3680043.[int_no_of_images]	Float	None
patient_overall	Whether the patient has a fracture	0 = No Fracture Overall 1 = Has Fracture	Binary: numeric	None
C1	Whether C1 (first vertebrae) has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None
C2	Whether C2 has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None
C3	Whether C3 has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None

C4	Whether C4 has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None
C5	Whether C5 has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None
C6	Whether C6 has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None
C7	Whether C7 (last vertebrae) has a fracture	0 = no fracture 1 = has fracture	Binary: numeric	None

Table 1: Description of the vertebrae

Model	**Loss**	**AUC**	**F1-score**
Basic CNN	0.0049	1.0000	0.8381
InceptionV3	0.0512	0.9802	0.4783
EfficientNetB5	1.0992	0.9482	0.0660
ResNet50	0.7423	0.9639	0.3874

Table 2: Accuracy metrics

Study	Overall	C1	C2	C3	C4	C5	C6	C7
1	1	1	0	0	0	0	0	1
2	1	1	0	0	0	0	0	1
3	1	1	0	0	0	0	0	1
4	1	1	0	0	0	0	0	1
5	1	1	0	0	0	0	0	1

Table 3: Test set provided to best performing Model

Evaluation on test data

	0	1	2	3	4	5	6	7
0	0.999960	0.999999	2.901001e-07	3.384608e-08	4.057587e-06	9.828905e-08	2.654984e-07	0.999999
1	1.000000	1.000000	1.230189e-09	2.878411e-10	1.603627e-10	1.330558e-12	9.379061e-11	1.000000
2	0.999826	0.999999	1.692328e-07	1.198899e-07	1.329357e-06	4.446853e-08	5.053161e-07	0.999995
3	0.999869	0.999999	5.090977e-08	1.101224e-07	1.140147e-04	2.192029e-09	7.228550e-08	0.999993
4	0.999980	1.000000	4.921936e-08	8.768658e-09	2.488766e-07	8.310396e-09	6.154199e-08	0.999999

Table 3: Evaluation and prediction results of CNN models: probability the given class has a fracture where 0 corresponds to overall patient and 1 - 7 correspond to C1-C7

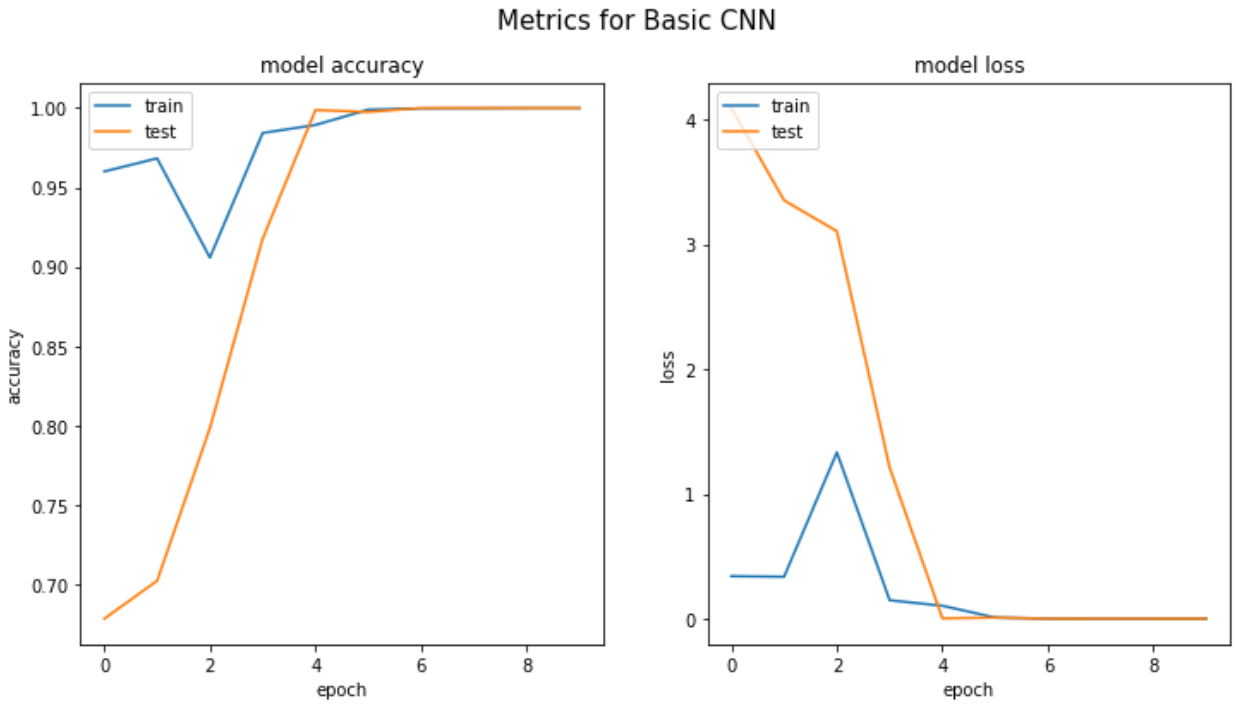

Figure 9: Accuracy and loss for Basic CNN with 10 epochs

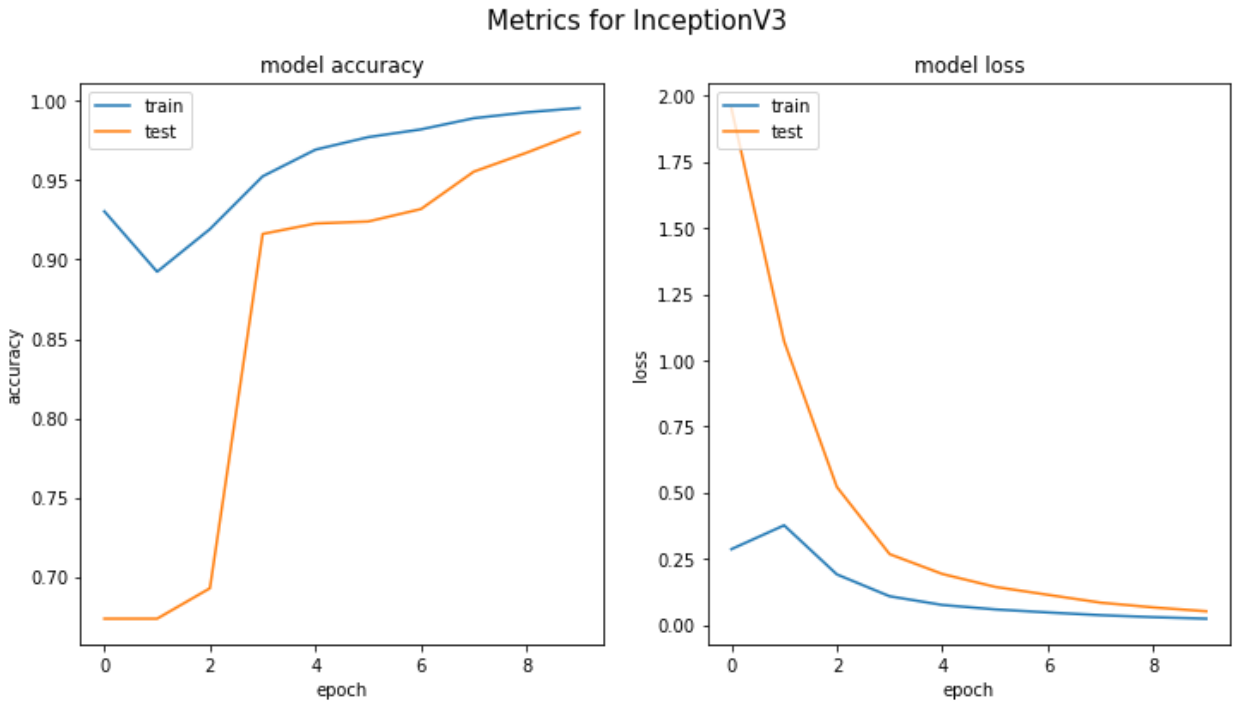

Figure 10: Accuracy and loss for InceptionV3 with 10 epochs

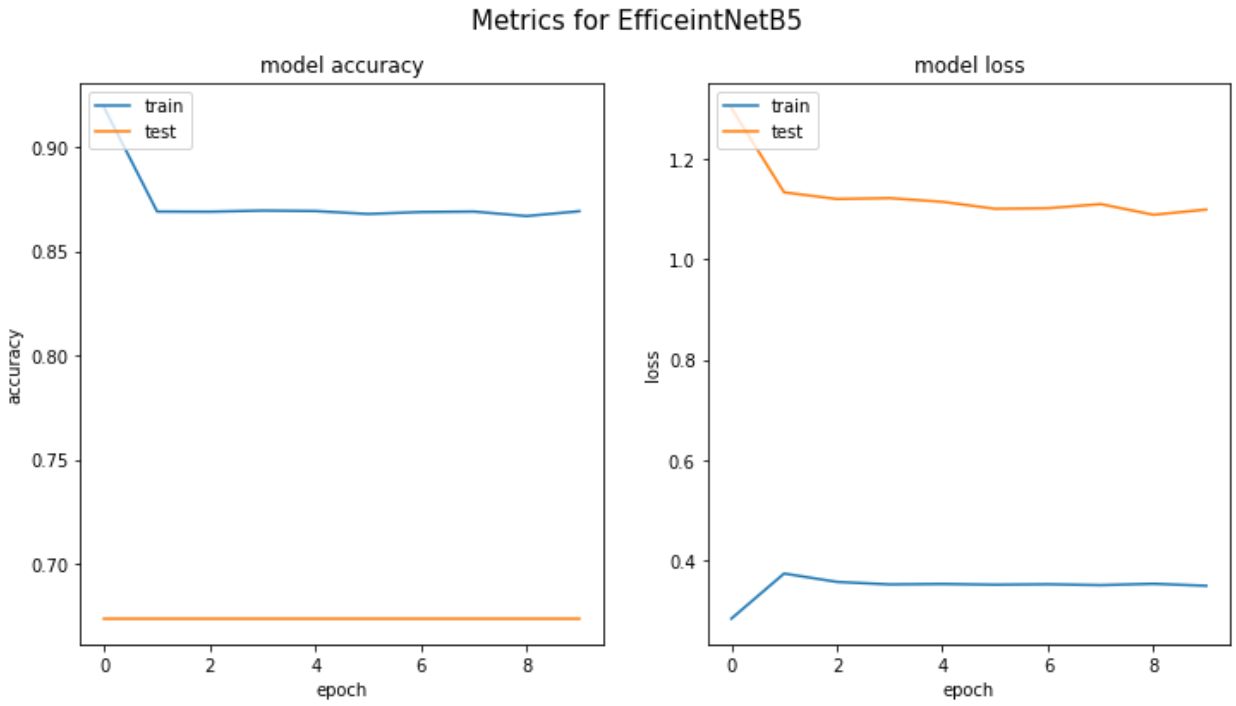

Figure 11: Accuracy and loss for ResNet50 with 10 epochs

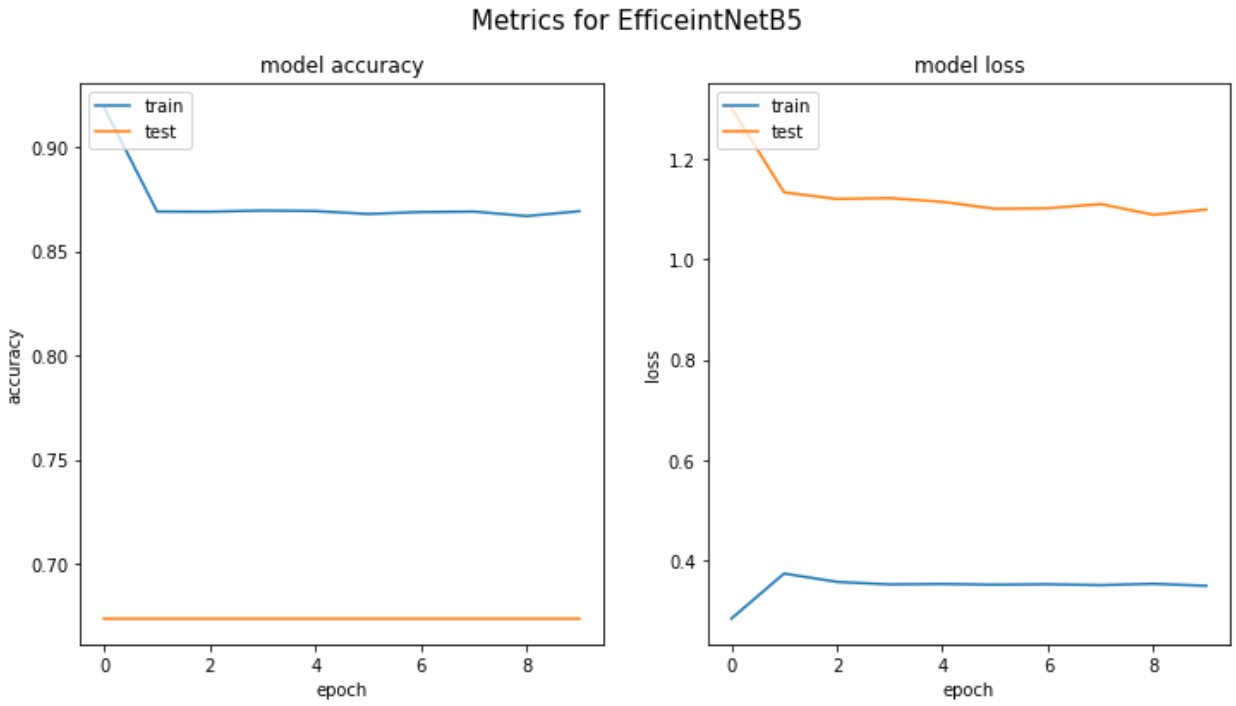

Figure 12: Accuracy and loss for EfficientNetB5 with 10 epochs