ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Deep Learning

Abnormality Detection in Bone X-Rays

Professor

Makis Malakasiotis

Project Report By

Gkountoumas Filippos (f3352105)
Papadopoulos Nikolaos (f3352118)

Contents

Introduction

As our final project for the course of Deep Learning, we were asked to implement a set of deep learning models, on Stanford's ML Group MURA Datataset[1].MURA, is a large dataset of musculoskeletal radiographs containing 40,561 images from 14,863 studies, where each study is manually labeled by radiologists as either normal or abnormal. To evaluate models robustly and to get an estimate of radiologist performance, additional labels from six board-certified Stanford radiologists we collected on the test set, consisting of 207 musculoskeletal studies. On this test set, the majority vote of a group of three radiologists serves as gold standard. The initial comparison be

The purpose of this project was to structure the models in order to decide if the X-Rays under study were normal or abnormal. A basic requirement was to implement at least 2 different models (CNN, Pretrained model). Although our resources were limited, even more so on the pretrained models (since google banned us from using GPU several times during the implementations), we were able to fulfill the project's requirements. The differences of the two implementations will be discussed on the following parts.

Quick overview of the implementation process

1. Access to the Mura Dataset
2. Imports of the available data
3. Data Frame creation from the training and test data utilizing specific columns
4. Data Preprocessing in order to extract the information needed for specific columns (i.e., Results regarding abnormality)
5. Data Exploration (X-Ray prints, research regarding the image sizes, class balance)
6. Function creation for future plotting
7. Split the train data to train and validation set to tune the hyperparameters, while keeping the test set untouched for the final evaluation
8. Pixel normalization
9. Use of ImageDataGenerator
10. Data Augmentation and proper resizing of the images
11. Model Creation (last steps are the same all the models)
12. Fit the training data to the model and evaluate using the validation set
13. Predict for the test and fine tune according to the results
14. Error Analysis

[1] MURA (musculoskeletal radiographs) is a large dataset of bone X-rays. Algorithms are tasked with determining whether an X-ray study is normal or abnormal. Musculoskeletal conditions affect more than 1.7 billion people worldwide, and are the most common cause of severe, long-term pain and disability, with 30 million emergency department visits annually and increasing. MURA is one of the largest public radiographic image datasets.

Exploring the Data

The preprocessing of our data started by visualizing random parts of the dataset to have a better view of what the data look like, noting also the difficulty in many cases to annotate the abnormalities. The following figure represents a random selection of X-Rays, from both classes (normal-abnormal).

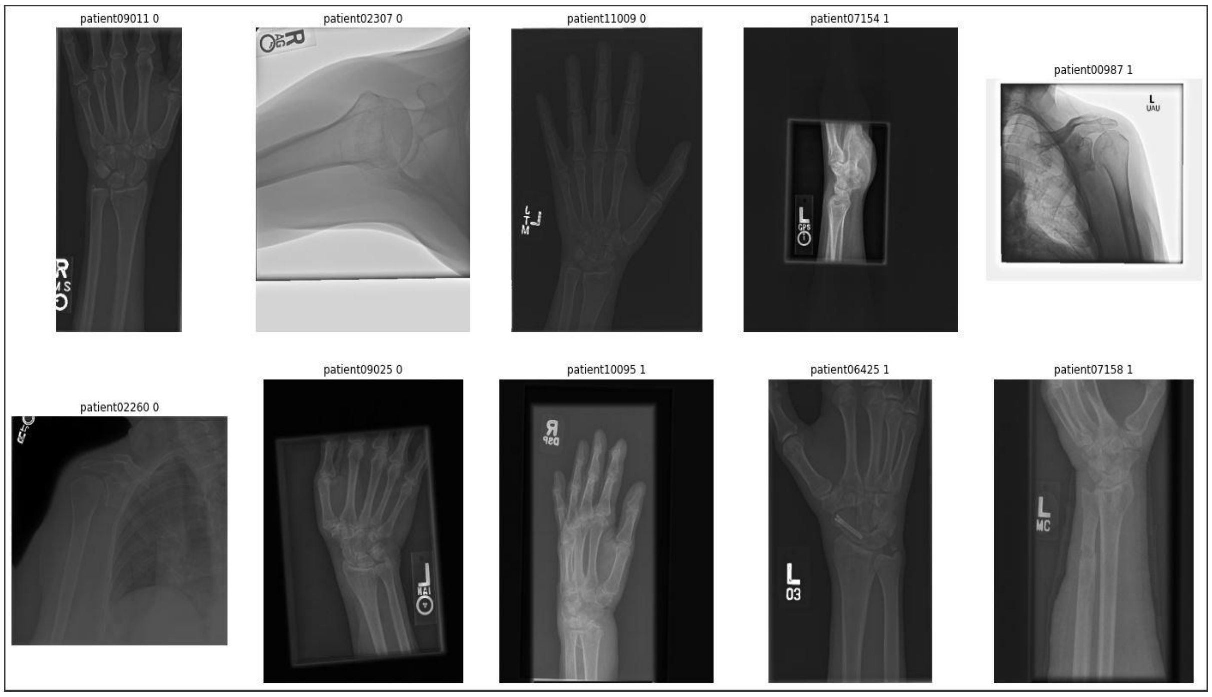


Figure 1 X-Rays from both classes

After reading the data and visualizing some random X-rays it was time to check the balance of the classes in both the train and test set. The first reason to plot this figure is because we should consider at this part implementing a stratified split to maintain the class balance, in case the X-Rays are not equally divided in the set categories.

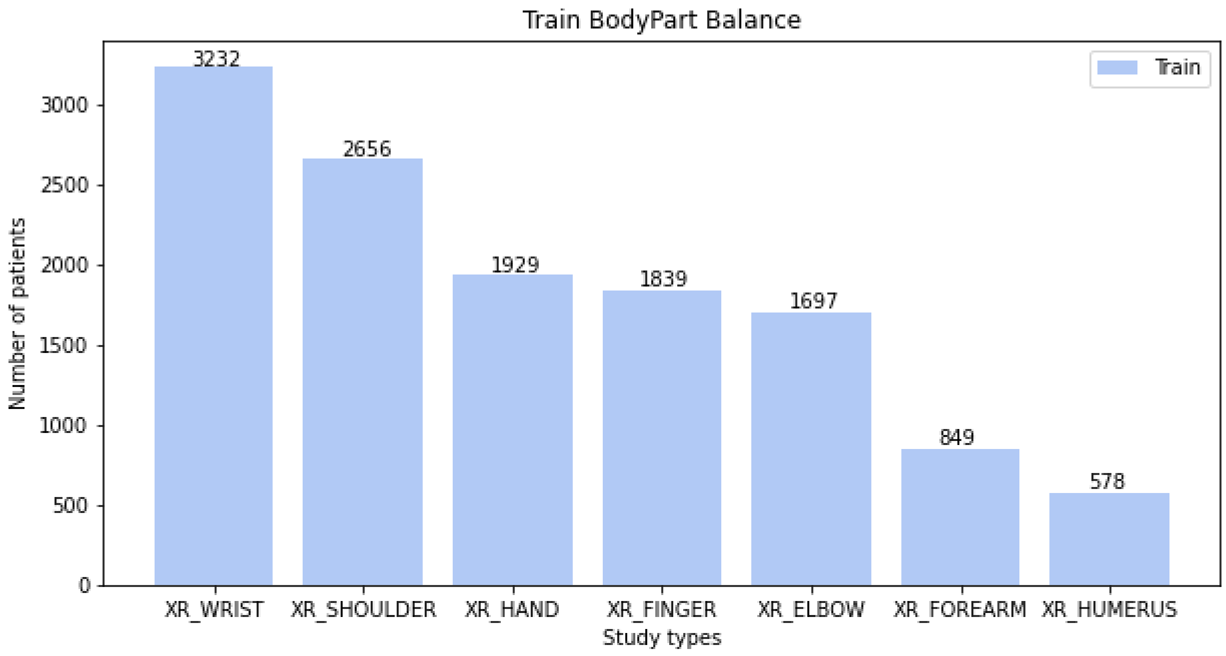


Figure 2 Training Data Class Balance

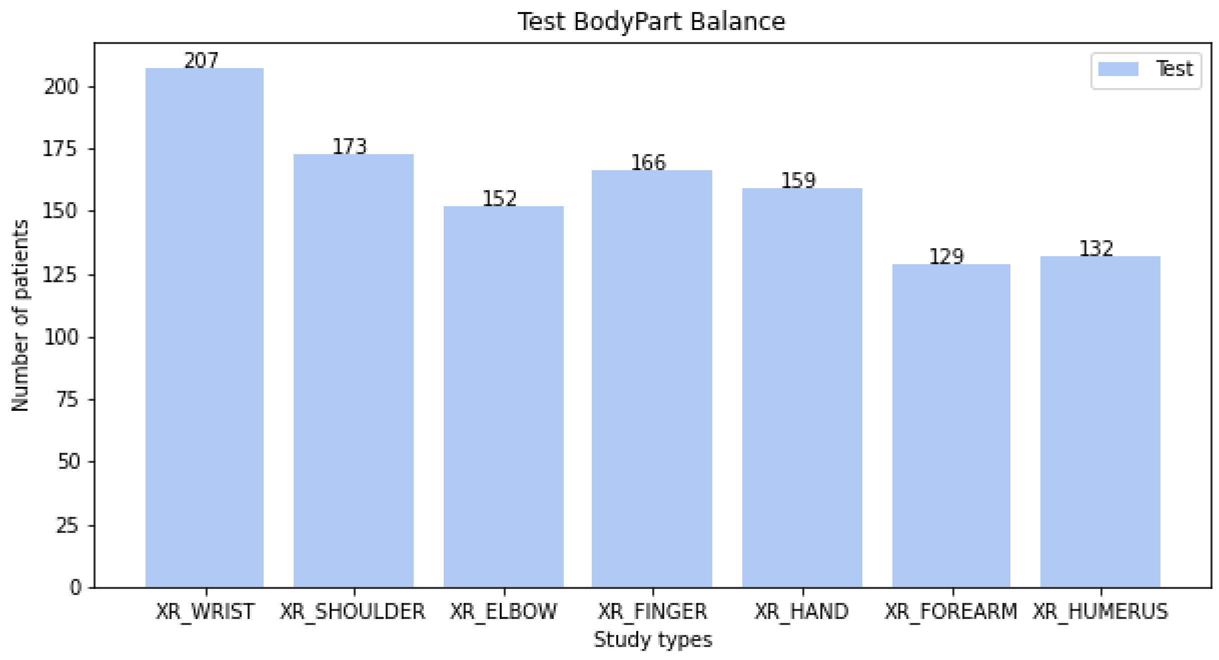


Figure 3 Test Data Class Balance

As can be inferred from the above figures, the dataset is, in fact unbalanced. The number of X-Rays for "Wrist" for example are 6 times the number of X-Rays for "Humerus". This imbalance will most likely lead our models into classifying some body parts better than other. It remains to be seen whether our claims were correct or not, in the following error analysis.

Although we gained some valuable insights as to what our expectations should be, regarding the models, we wanted to be meticulous, therefore we decided to explore the dataset further. First, we plot the balance of the classes in terms of normal and abnormal, per body part.

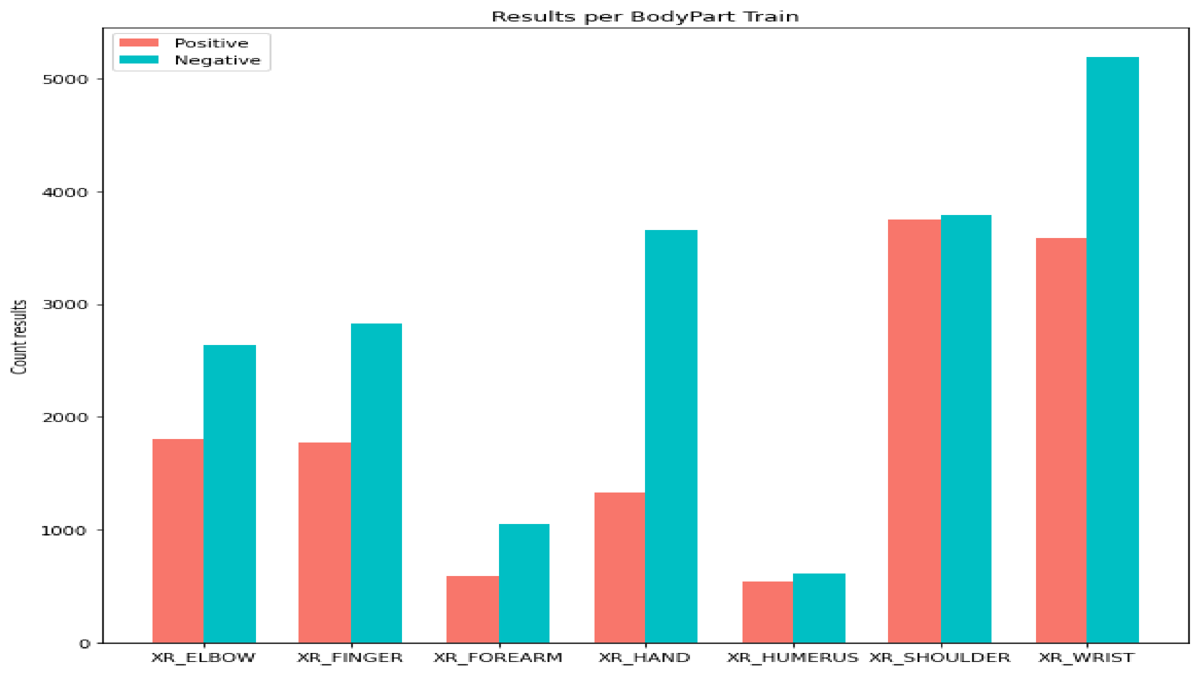


Figure 4 Train Data Class Balance (normal, abnormal)

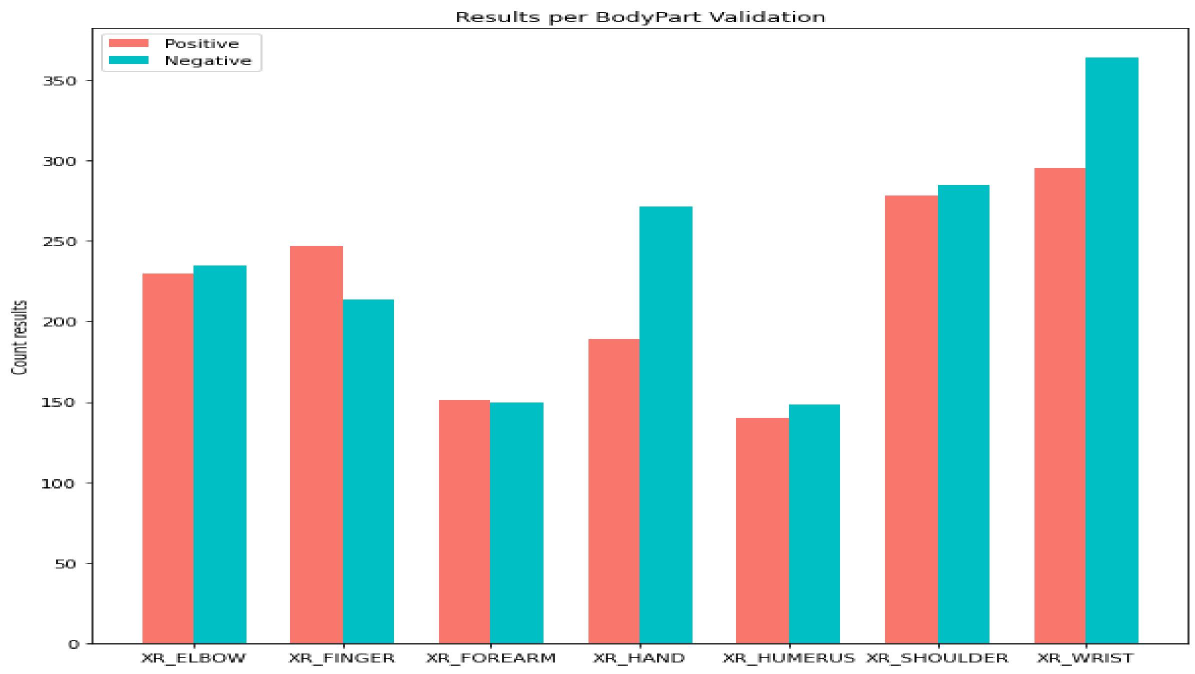


Figure 5 Test Data Class (normal, abnormal)

What can be inferred from the above figures is that in general the balance between normal and abnormal X-Rays are balanced enough, so that the models can train seeing enough of both classes. The only exception being the "Hand" where most of the training data, are of X-Rays from healthy people.

Consequently, a second plot that could provide some valuable insights, is the number of views per study[2] as shown in the figure below[3].

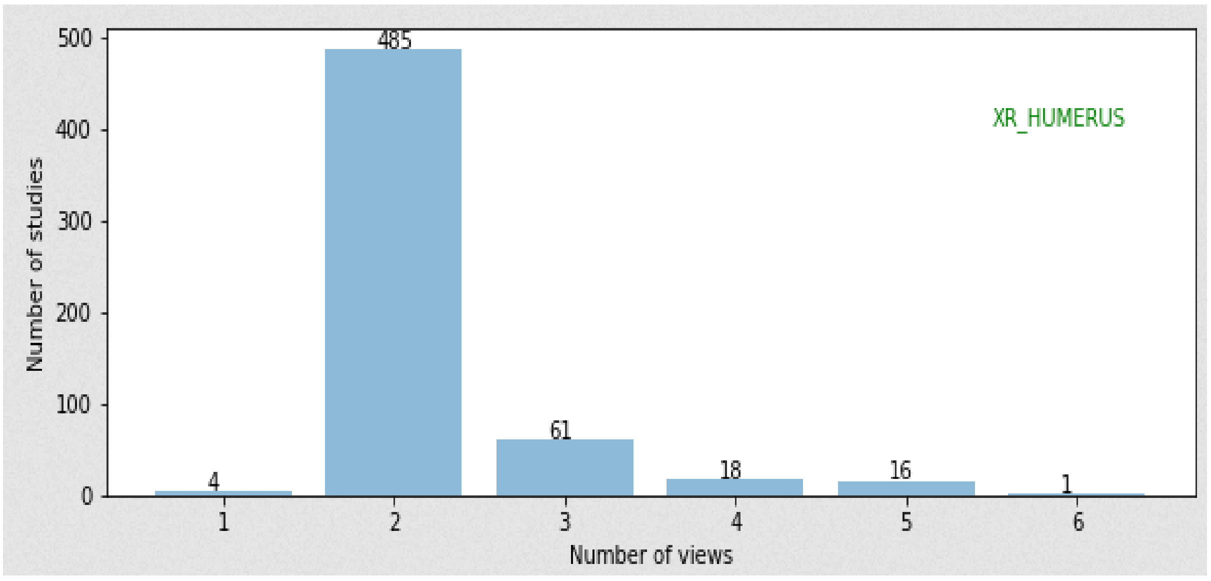


Figure 6 Number of Studies for Humerus

[2] This number of studies is practically the number of different studies that had to be taken into account in order for the triads of radiologist to classify it as normal or abnormal.

[3] We show only one plot regarding "Humerus". The rest of the body parts can be seen in the available notebook.

Figure 6 shows that for specific body parts (i.e., Humerus), one study was not enough, and in most cases at least 2 studies were necessary. This shows that many of the X-Rays do not immediately indicate the existence of some kind of anomaly, thus making it even harder for a deep learning model to classify such observation.

Preprocessing & Data Augmentation

After sufficiently exploring the data, we continued the process, now proceeding to the preprocessing and data augmentation. Firstly, we decided to keep the original validation set intact and use it as test set. Therefore, we took 20% of the train set to use as validation, in order to validate and possibly tune any hyperparameters.

The MURA dataset was for the most part, well structured, making it easy to use. The only point worth mentioning was the fact that many X-rays were of different sizes. In order to fit the images into our models, we resized them (the resizing differentiated in some occasions so that it was exactly the size that the pretrained model needed). This process was done simultaneously with the data augmentation using the ImageDataGenerator Class of Keras. This class generates batches of tensor image data with real-time data augmentations. It only loads batches of image during the training process in order to avoid overloading the memory. The data were augmented using a series of different augmentations such as:

- Rotation: Randomly rotate the images 30 degrees
- Horizontal flip: Images are flipped horizontally, maintaining the full range of information
- *Zoom: Images are zoomed randomly 20% from the original (90% of the size or 110%)
- *[4]Shift: images are sifted randomly into different directions
- Predefined pre-process function (only on the pretrained models)

Although the augmentations were applied only on the training set, 3 generators were made (one for each set). The training set images were also shuffled in order to make the model update its weights, by using as input different kinds of X-Rays.

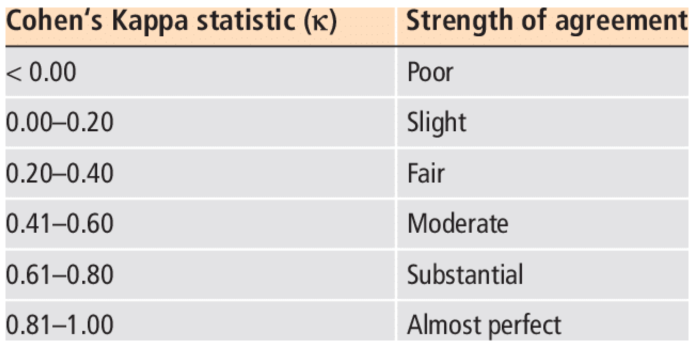
As a test bench we used a non-augmented copy of the training data first, resulting in undesirable results, so we decided to use only the augmented training data for the rest of the process.

Evaluation Metrics

Right before the model building phase of the project, it was important to set which evaluation metrics would be used to measure the model's performance. The accuracy, loss, precision, recall, F1 score, were the first metrics set, but there was also one specific metric that was used from Stanford's ML Group for the particular dataset. This was Cohen's Kappa. Both precision and recall, focus more on the positive class, but in our case negative class almost always has more observations that the positive counterpart. Hence this metric is a

[4] * Those 2 augmentations were reversed on the last run of the models, because they degraded the model's overall performance.

score (ranging from -1 to 1), that expresses inter-annotator agreement on classification problems. The minimum value represents complete disagreement between annotators, whilst the maximum value represents perfect agreement. When the score is zero, it indicates agreement by chance.



Cohen's Kappa statistic (κ)	Strength of agreement
< 0.00	Poor
0.00–0.20	Slight
0.20–0.40	Fair
0.41–0.60	Moderate
0.61–0.80	Substantial
0.81–1.00	Almost perfect

Figure 7 Cohen Kappa Strength of Agreement

Model Building

After the first valuable steps of the process, it was time to start building models that could use our processes data to return the desired results. In this part we had to choose which implementations we could use so we decided to try many of the techniques that were taught during the course. As per the project's requirements, we implemented a CNN (similar to the one from the previous project), which was mostly used in order to compare it with the second implementation which was a pretrained model. In our case the pretrained model that was selected for the purposes of this project was the Densenet201 from keras applications[5]. Once more the constant Google restrictions on GPU usage from Google, refrained us from using multiple pretrained with ensemble, in order to achieve better results.

Convolutional Neural Network (CNN)

The first approach to solving this problem was training a Convolutional Neural Network. The model took as input the images rescaled to 224x224 in 3 channels (RGB). The model structure consisted of 4 convolutional blocks, each of which contained:

- Convolutional Layer: The starting layer has 32 filters and rest of the layers have 64. The padding was set to "same" so as not to change the filtered image size and a ReLU as an activation function for the layer.
- Batch Normalization Layer: To achieve a stable distribution of activation values throughout training
- Max Pooling Layer: Using a max pooling operation (Stride= (2,2)), to help avoid over-fitting by providing an abstracted form of the representation. This operation selects the maximum element from the region of the feature map covered by the filter.
- Dropout Layer: The dropout was set to 0.25 for all 4 layers. Dropout has the effect of reducing the capacity or thinning the network during training and at the same time prevents over-fitting.

[5] Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction and fine-tuning.

The output of the above convolutional layers was then flattened and passed to a dense layer of 128 units. Lastly, the output layer consisted of a single unit layer with a sigmoid activation function so as to provide a value between 0 and 1. We used the Adam optimizer with a learning rate of 1e-4 and utilized the binary cross-entropy as our loss function[6].

The CNN model had 1.700.545 parameters in total of which only 704 were not trainable. This number is less than 10% of the pretrained models that will be used later on.

Pre-trained model (Densenet201)

As our second implementation we had to research the literature to find which pre-trained model we could use. We had to take into account, once more, the limited resources and the training time it would take for a pretrained model to finish (since Google restrict the GPU usage, when a model is training for over 3 hours straight).

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. Embracing this observation, we decided to try Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections - one between each layer and its subsequent layer, DenseNet network has L(L+1)/2 direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNet has several compelling advantages: it alleviates the vanishing-gradient problem, strengthens feature propagation, encourages feature reuse, and substantially reduces the number of parameters from other pretrained models. DenseNet is also able to achieve high performance whilst requiring less computational power than other pre-trained models.

The course of actions we needed to take in order to use this model, was to resize the images to (224,224,3), which was the optimal size for the aforementioned model. It was also imperative to reduce the batch size (from 32 to 8). Apart from our own augmentation we also used the Densenet's image preprocessing.

As per the feature extraction part, we excluded the fully connected layer at the top of the pretrained model and we applied max pooling to extract features. The idea of using max pooling is in order to emphasize sharp features of the X-rays (e.g., prosthetic placements, medical screws etc.).

The pre-trained model had 18.323.905 parameters in total, of which only 1921 parameters were not trainable. In comparison to the initial CNN this model has 1100% more parameters, making it a far superior model in terms of capacity.

[6] The models analytical summary can be seen in the notebook files attached with this report.

Model Evaluation & Error Analysis

After building our models, it was time to train them and evaluate the yielded results of each model separately. Inevitably we also compared the results of the models. A key point that we need to mention is that we used to methods during training in every model.

Early Stopping: We used the early stopping method to specify an arbitrary large number of training epochs (100 in our CNN and 20 in our pretrained model) and stop training once the model performance stops improving on a holdout validation dataset.

Reduce LR on Plateau: We used this method to reduce the learning rate when a metric had stopped improving. Models often benefit form reducing the learning rate by a factor of 2e-10 once learning rate stagnates. This scheduler reads a metrics quantity (Cohen Kappa) and if no improvement is seen for a 'patience' number of epochs (2 in our case), the learning rate is reduced.

Convolutional Neural Network Evaluation

The process started with us using the resized data as a baseline in order to see the results of the CNN with the bare minimum of effort from our part. Using the non-augmented data and training for 100 epochs, using early stopping, we finally achieved a Cohen Kappa score of 0.317 and an Accuracy score of 0.598.

We retrained the CNN, only this time using the augmented data, expecting an increase in performance. The results were improved as expected with the model yielding now an overall Cohen Kappa score of 0.384 and an Accuracy score 0.694. Therefore, we decided to use only the augmented trained data from now on[7], in every model.

CNN Learning Curves

After the metric scores we plotted the learning curves of the model regarding loss and Cohen Kappa as shown in the figures below (figures 7,8).

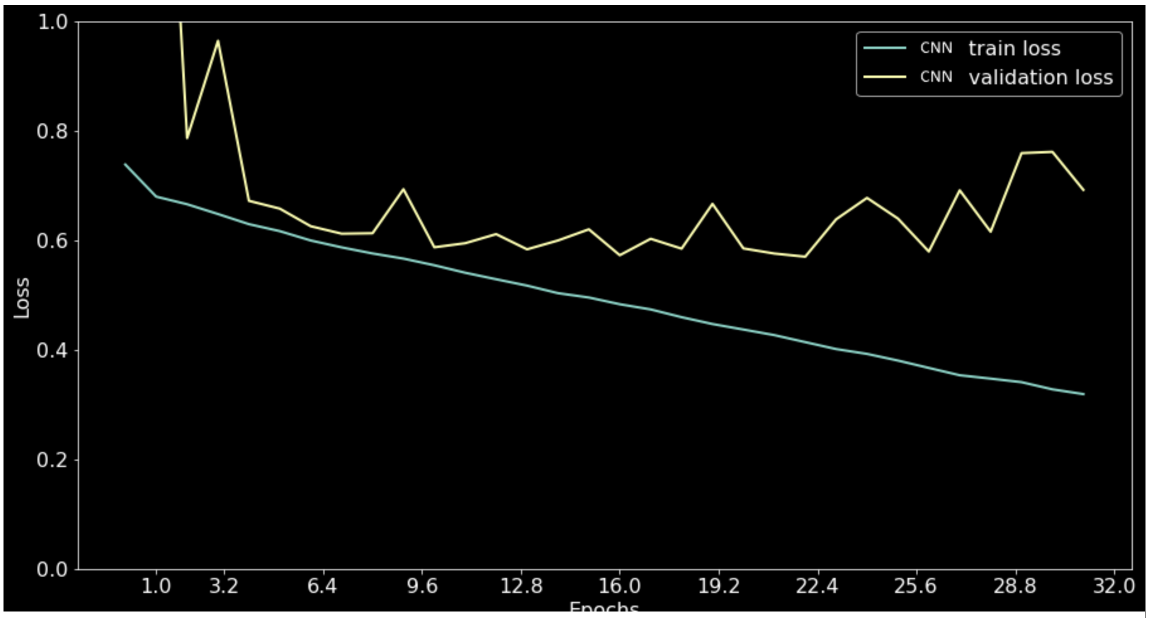


Figure 8 CNN Learning Curve of Loss

[7] In the attached code you will find the models ran with the augmented data.

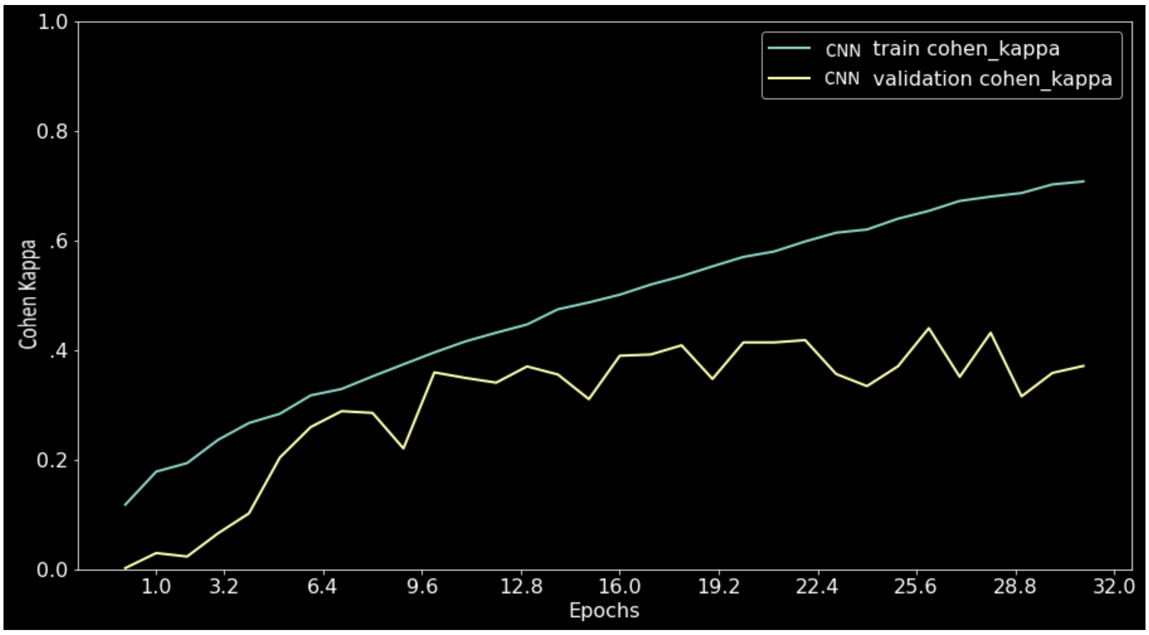


Figure 9 CNN Learning Curve of Cohen Kappa Score

CNN Classification Report

To further investigate and evaluate our models results, we decided to create a set of classification reports, one for each separate body part and we created a table containing the most valuable metrics from each one (figure 9).

Part/Metrics	CNN Classifier Model							
	Precision		Recall		F1 score		Accuracy	Kappa
	Abnormal	Normal	Abnormal	Normal	Abnormal	Normal	WeightAvg	Score
Wrist	0,72	0,74	0,65	0,8	0,68	0,77	0,73	0,45
Shoulder	0,63	0,71	0,77	0,55	0,69	0,62	0,66	0,33
Elbow	0,72	0,68	0,63	0,76	0,68	0,72	0,7	0,4
Finger	0,79	0,63	0,58	0,82	0,67	0,71	0,69	0,39
Hand	0,71	0,67	0,38	0,89	0,49	0,77	0,65	0,29
Forearm	0,7	0,64	0,59	0,75	0,64	0,69	0,67	0,34
Humerus	0,73	0,71	0,66	0,77	0,7	0,74	0,72	0,44

Figure 10 Table containing all metrices form CNN

As we can see from the table above, the “Wrist” category scores the highest which is expected since it had by far the most training examples. What is counterintuitive is the shoulder category with the second highest number of observations in the training set, in which our CNN had the second lowest Kappa score. On the hand, in the categories with the lowest number of X-Rays (Forearm, Humerus), our model achieved over 70% average in all categories and a 0,44 Cohen Kappa Score (Moderate Agreement) in the “Humerus”, which was unexpected. On the “Forearm” the model didn’t manage to recognize the abnormalities really good, but it was expected in some part since there were only 577 training observations.

Overall, the category that our model was the least able to spot the abnormalities was the “Hand”, with a Cohen Kappa score of 0.29.

In order to better understand the model’s behavior, we printed some of the test results as they were classified from the model. In the first figure (figure 10) is a compilation of X-Rays from the category that out CNN performed it’s best (“Wrist”). On the second figure below (figure 11), we plotted a compilation of X-Rays in our two worst performing categories (“Shoulder”, “Hand”).

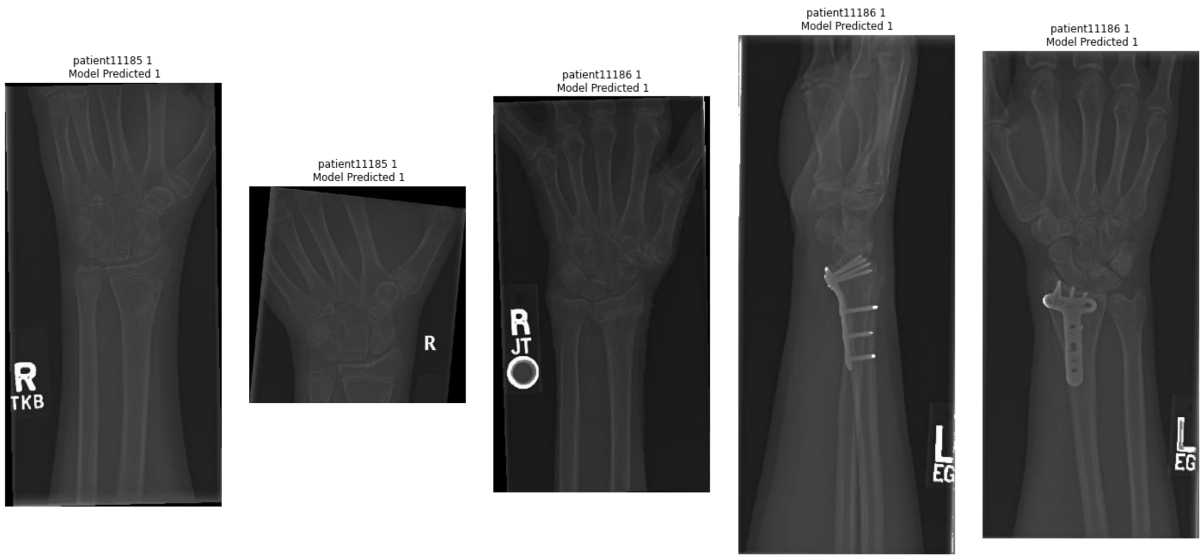


Figure 11 Compilation of correctly predicted abnormalities in the “Wrist” category.

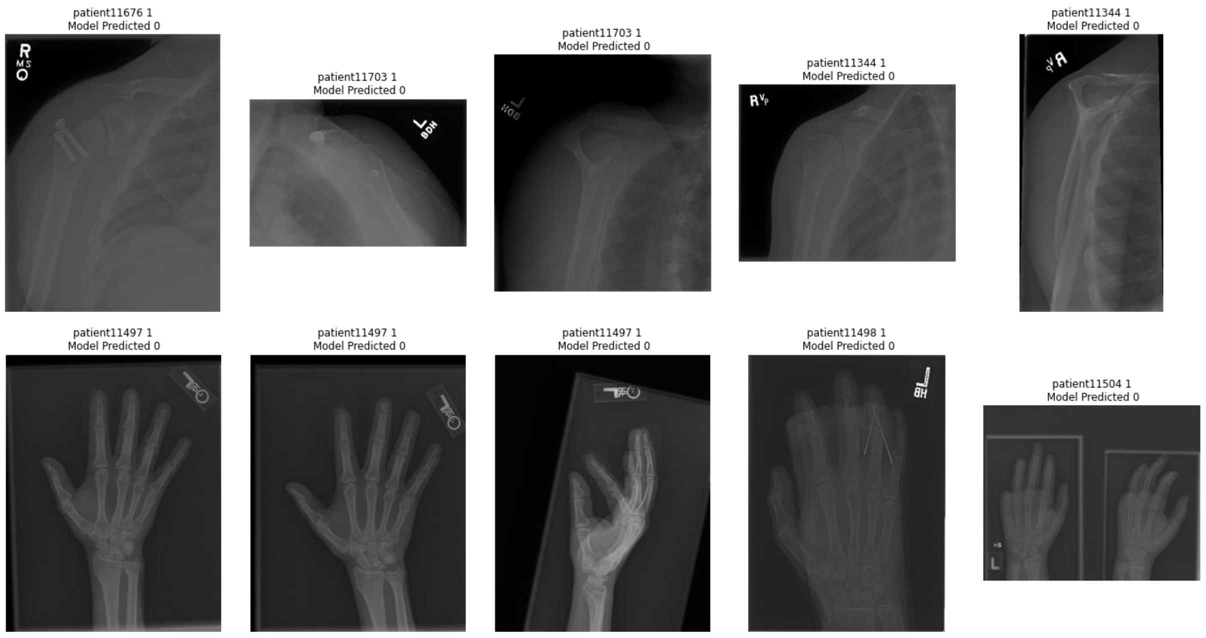


Figure 12 Compilation of wrong prediction in the categories “shoulder” (above), “Hand” (below)

What can be inferred from the figures above is that in the “Wrist” category the model can detect the prosthetic placements, whilst on the other two categories, it missed two major abnormalities that could even be detected by the naked eye of an untrained annotator.

Overall, the CNN didn’t achieve a stellar performance, still for a model of this size, without pretraining, it performed adequately in some categories.

Pre-trained Model (Densenet201) Evaluation

Since the overall performance of the CNN, did not yield the desired results, we proceeded to a pre-trained model. After researching the literature to find the optimal pre-trained model, we resorted to the DenseNet. In this process we started with the new generated and augmented data, with the smaller batch size (batch size=8). During the first training of the model, we froze the whole convolutional base, in order to prevent weight updates, during training. We used once more a simple classification top level(head) consisting of a dropout layer and a dense layer with sigmoid activation function.

As expected, the pre-trained model far exceeded the performance of the CNN (the frozen pre-trained model scored a Cohen Kappa score of 0.47 and an overall accuracy of 0.71. Although this would be enough in many cases, we decided to proceed one step further. In an attempt to increase the performance of the model, we unfreezed the entire convolutional base that was previously frozen, thus making the whole model trainable. A point worth mentioning here, was that we set a very low learning rate (1e-5), in order to avoid catastrophic forgetting. We also used early stopping and Reduce LR on Plateau to avoid overfitting.We retrained the Densenet, using the augmented data for a total of 20 epochs (due to the overwhelming training time), and when the training was finished, this modeled yielded our best results, thus far. The results were far superior than any of the previous attempts with a Cohen Kappa score of 0.582, an accuracy of 0.792 and an overall loss of 0.49. After achieved those scores it was time to research the model, in order to find any insights that could help us increase its overall performance even more.

DenseNet Learning Curves

After the metric scores we plotted the learning curves of the model regarding loss and Cohen Kappa as shown in the figures below (figures 12,13).

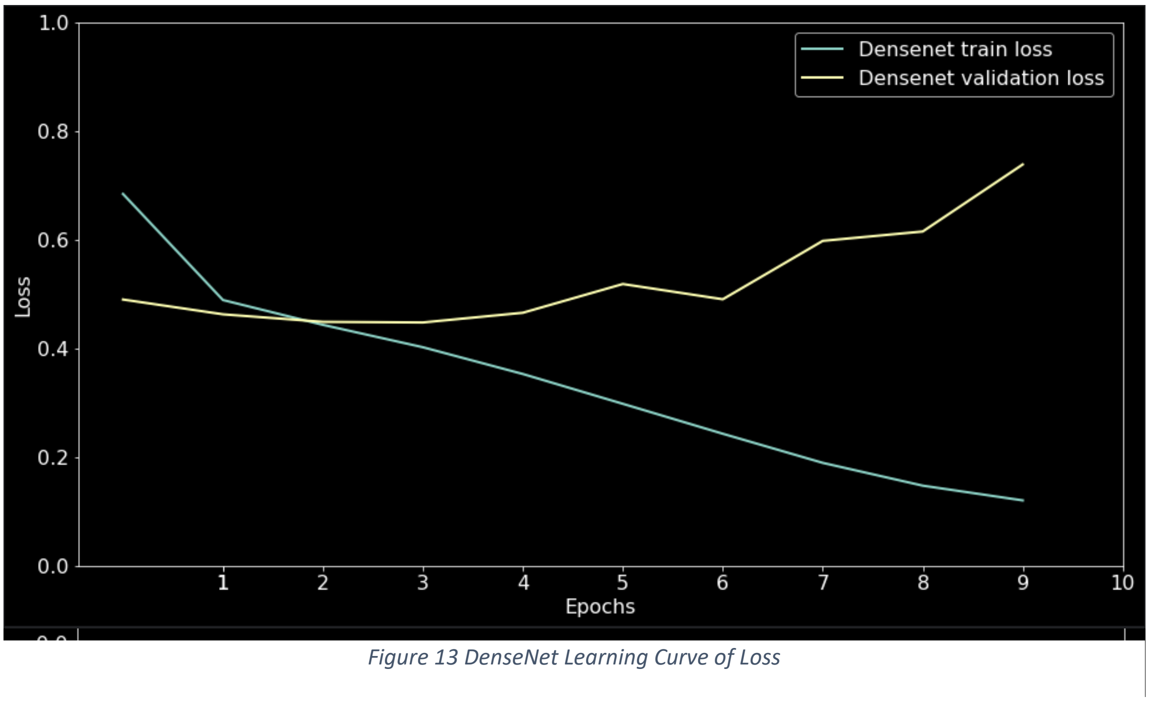
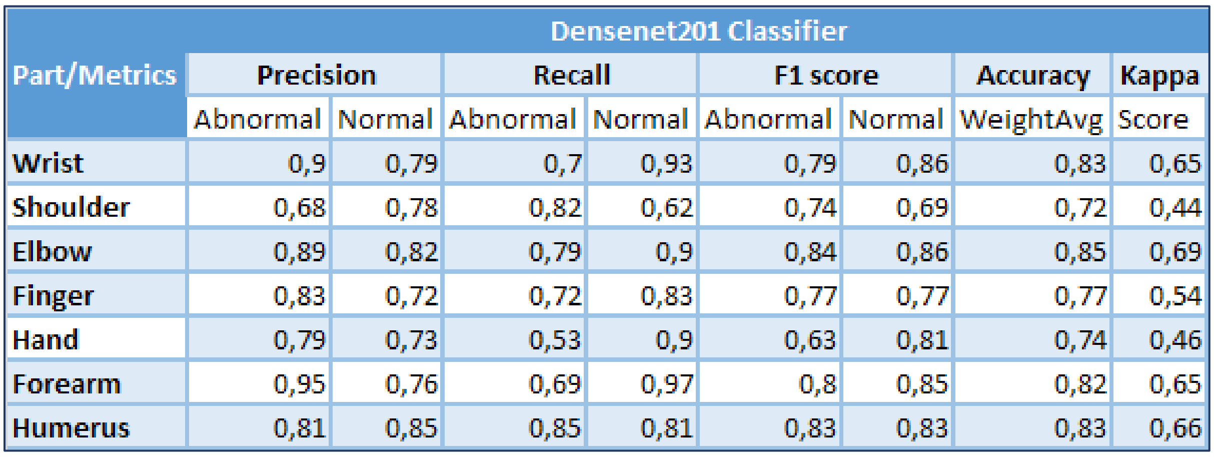


Figure 13 DenseNet Learning Curve of Loss

DenseNet Classification Report

To further investigate and evaluate our models results, we decided to create a set of classification reports, one for each separate body part and we created a table containing the most valuable metrics from each one (figure 14).



Densenet201 Classifier								
Part/Metrics	Precision		Recall		F1 score		Accuracy	Kappa
	Abnormal	Normal	Abnormal	Normal	Abnormal	Normal	WeightAvg	Score
Wrist	0,9	0,79	0,7	0,93	0,79	0,86	0,83	0,65
Shoulder	0,68	0,78	0,82	0,62	0,74	0,69	0,72	0,44
Elbow	0,89	0,82	0,79	0,9	0,84	0,86	0,85	0,69
Finger	0,83	0,72	0,72	0,83	0,77	0,77	0,77	0,54
Hand	0,79	0,73	0,53	0,9	0,63	0,81	0,74	0,46
Forearm	0,95	0,76	0,69	0,97	0,8	0,85	0,82	0,65
Humerus	0,81	0,85	0,85	0,81	0,83	0,83	0,83	0,66

Figure 14 Table containing all metrices from DenseNet201

As we can see from the table above, the "Elbow" category scores the highest outscoring even "Wrist". This worth mentioning since "Elbow" was the category with the third lower number of X-Rays in our training Set. Once more the "Shoulder" category with the second highest number of observations in the training set, was also the one with the lower weighted average accuracy and the lower Cohen Kappa score. On the hand, in the categories with the lowest number of X-Rays (Forearm, Humerus), the pre-trained model achieved a weighted accuracy of over 0.8 in both categories and Kappa Cohen score >0.65. Overall, this pretrained model achieved a Substantial Agreement (Cohen Kappa score 0.61-0.80), in 4 out of 7 categories, which is a remarkable achievement. The rest of the categories were on Moderate Agreement scale (Cohen Kappa 0.41-0.60).

Wanting to visualize the performance of the model we plotted once more a set of pictures from the best performing category ("Elbow") in the following figure (figure 15), but we also plotted the 2 worst performing categories ("Shoulder", "Hand") to try to understand what could be wrong with those categories (figure 16).

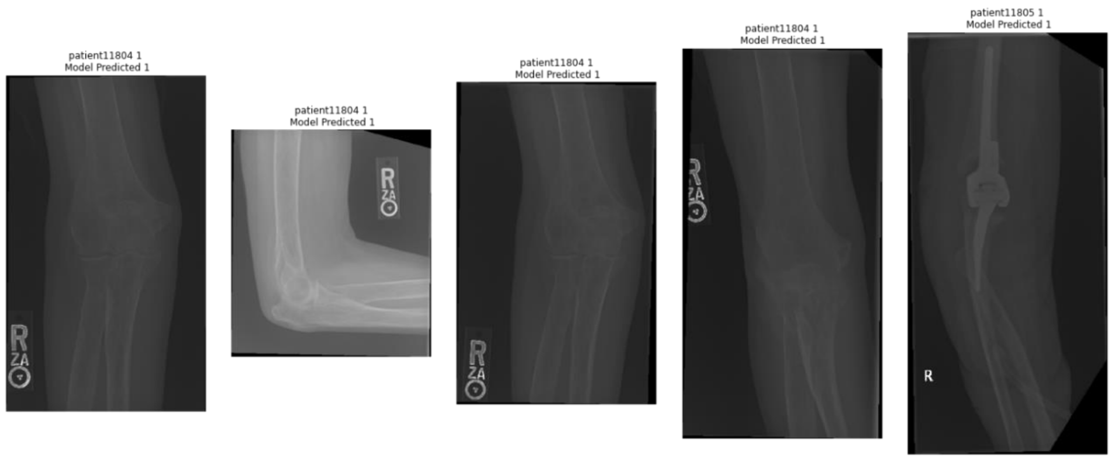


Figure 15 Compilation of correctly predicted abnormalities in the "Elbow" category.

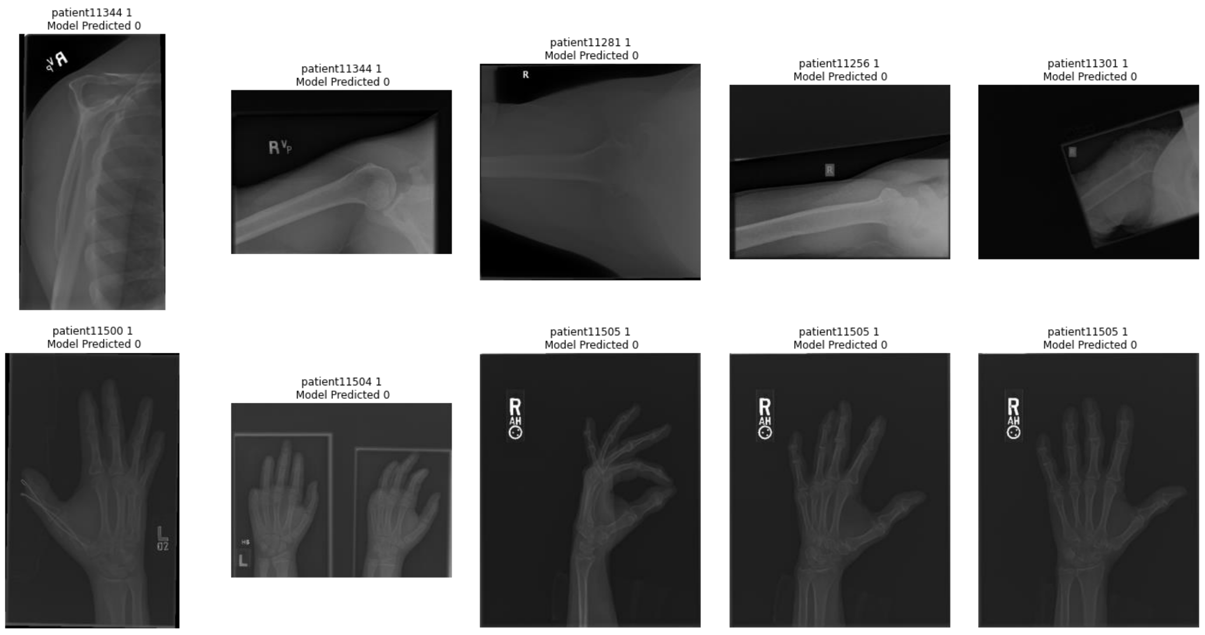


Figure 16 Compilation of wrong prediction in the categories "shoulder" (above), "Hand" (below)

What can be inferred from the above figures is that, the pretrained model, manages to spot the abnormalities (e.g., prosthetic parts) in the best category, whilst underperforming in the same two categories as before. The fact that the worst performing categories remain the same in both models, might indicate that there might be a problem with the X-Rays belonging to them or that these categories may be harder to evaluate, even for a trained professional.

Future Work

Despite the positive results of this stage of our study project, there are still some areas to improve the algorithm's accuracy. The first of these is the ongoing development of the model's training data. The MURA dataset's data quality and diversity have a negative impact on the model training results to some extent. Because deep learning relies so heavily on data, the quality of the training data is extremely important. The whole rescaling process in order to help the model training might be helpful, but overall, the greater resolution could reveal tiny abnormalities, lost from the squeezed resolution in our dataset.

The size of the training set, plays a key part in the performance of the models, therefore, to gain better results the train dataset should be no less than 10 000 samples, this will allow to get precise forecast about the fracture localization.

The other problem that we faced during this stage is the big variety of different clinical cases. Thus, a great diversity of different types of the abnormalities in the combination with the list of different body parts used in the dataset creation could be the cause of a decrease in success rates of the algorithm.

The overall complexity of the models could be exponentially increased, using different models, and combining their results using different techniques (e.g., ensembling). In order to do this, a TPU or computer with high computational pawer is necessary (as we noted a few times by now, the GPU limits from Google on Google Colab, make the training of such models almost impossible).

We could also increase the complexity of pre-trained models by adding a bigger network on top of the already trained model instead of just the final layer. For example, we could have added a Convolutional Neural Network on top of our DenseNet instead of the output layer.

Cleaning the images of any text box would be another attempt to try and increase the quality of our training data. Although Keras-OCR is a great tool for removing such boxes, we decided not to clean our images for this project because the procedure proved to be too time consuming.

Conclusion

This project showcases bone anomalies detection methods developed using Stanford's MURA dataset. Detecting musculoskeletal abnormalities is a crucial step in making a clinical diagnosis, but it can be hampered by human biases like fatigue, a heavy workload, a lack of knowledge, cases of bone fractures and musculoskeletal abnormalities that are difficult to see, lack of concentration, and multitasking on the part of the radiologist. Computer-aided diagnosis systems should be implemented into the working instruments of the practical specialist in order to eliminate this factor and to make the abnormality identification procedure faster and more reliable. The ability to identify the zone of the abnormality and further building of the prioritization list of the patients according to the degree of the danger of the possible diagnoses will significantly increase the speed of medical care and will decrease the unnecessary work load for the doctors. Such computer-aided technologies won't entirely replace radiologists, but they will greatly improve the caliber of their work. Our model performed well in detecting bone anomalies, however it was solely trained with the binary problem classification (existence or absence of the abnormality). The model should be further trained to be able to discern between various abnormalities with a stable high level of accuracy in order to make it usable for practical experience.