Deep Learning

Bone X-Ray abnormality detection

AUEB MSC 2023

Phivos Dadamis f3312214

Sofia Eleftheriou 210

Introduction

The purpose of this study is to build a deep learning model that decides if a study containing X-Ray images is normal or abnormal. To do so we used the MURA dataset to train and validate our proposed methods.

MURA (**mu**sculoskeletal **ra**diographs) 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.

1. CODE ACCESSIBILITY

The code of this study is open source available via github in the following public repository: https://github.com/sofiaele/bone_x-ray.git .

2. DATASET

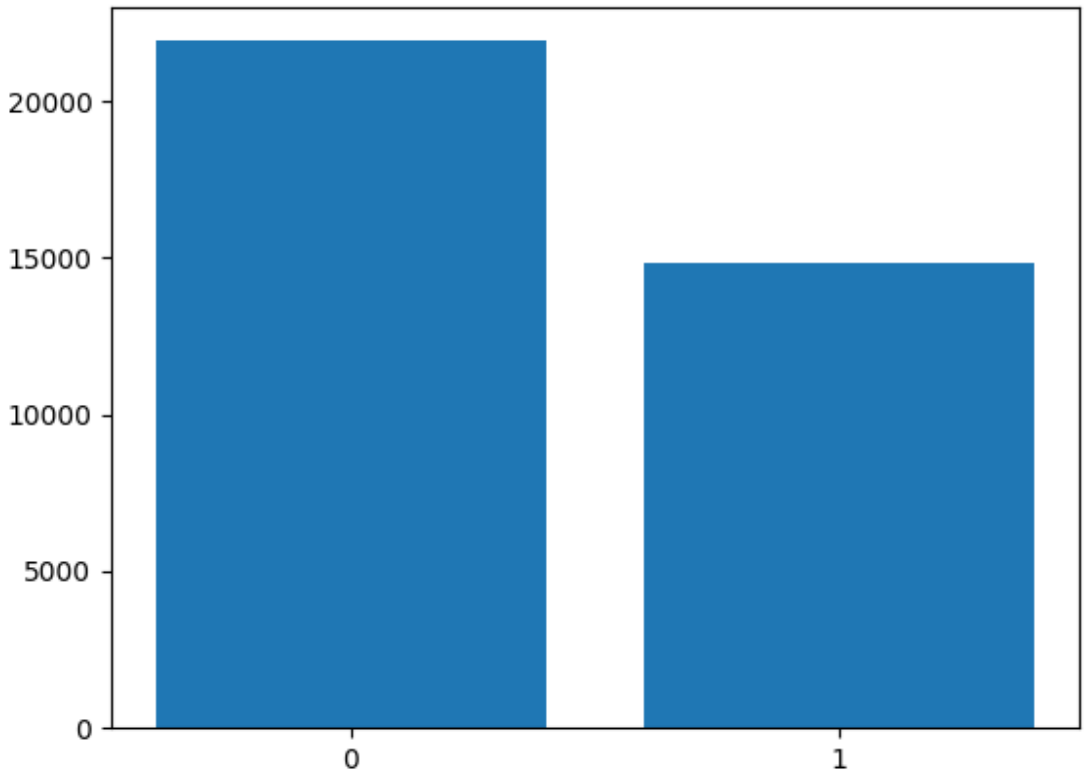
The dataset consists of radiographic png images which depict x-ray studies and is structured as follows. The samples are firstly grouped by the body part they refer to and secondly by the patient they were taken from. For each body part/patient combination, a study is provided, which consists of one or more images and is labeled as positive or negative. Positive means that an abnormality is detected, while negative expresses the absence of any noticed abnormality. Furthermore, the studies are split in a train and a validation set, while the actual test set is not provided.

2.1 Dataset Split

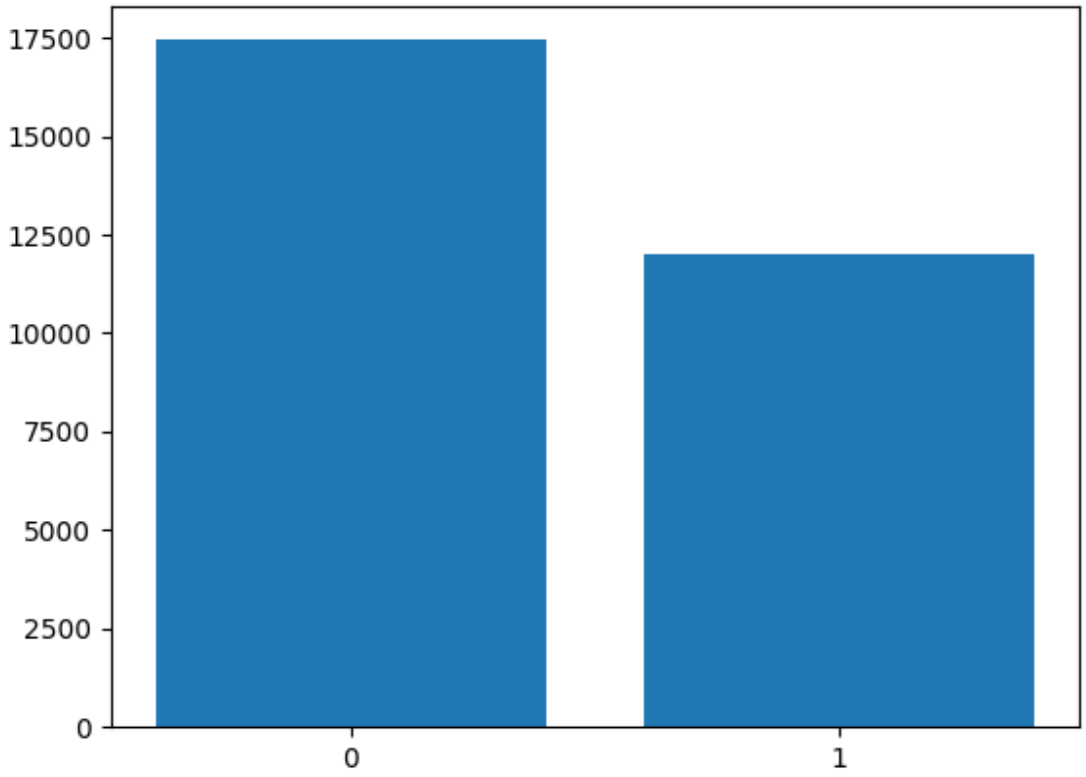
Due to the fact that there was not any test set provided, we decided to use the provided validation set as a test set. Therefore, we split the provided train set into the actual train set that was eventually used for the training and a new validation set. The new train set contains 80% of the provided train set, while the validation set contains the remaining 20%. During the splitting process, each study is exclusively assigned to either the training set or the validation set. This ensures that all the images from a particular study are utilized for either training or validation, but not both. In other words, a study's images are not divided between the training and validation sets; they are completely allocated to one set to maintain the integrity and independence of the data used for training and validation.By ensuring that a patient's data appears in only one set, any unique characteristics or specific attributes present in their bones do not introduce bias or unfairly influence the results. In order to achieve the desired result sklearn 's method "GroupShuffleSplit" was used with a grouping by the patients' id's as can be seen:

```
splitter = GroupShuffleSplit(test_size=.20, n_splits=2, random_state=7)
split = splitter.split(df, groups=df['id'])
```

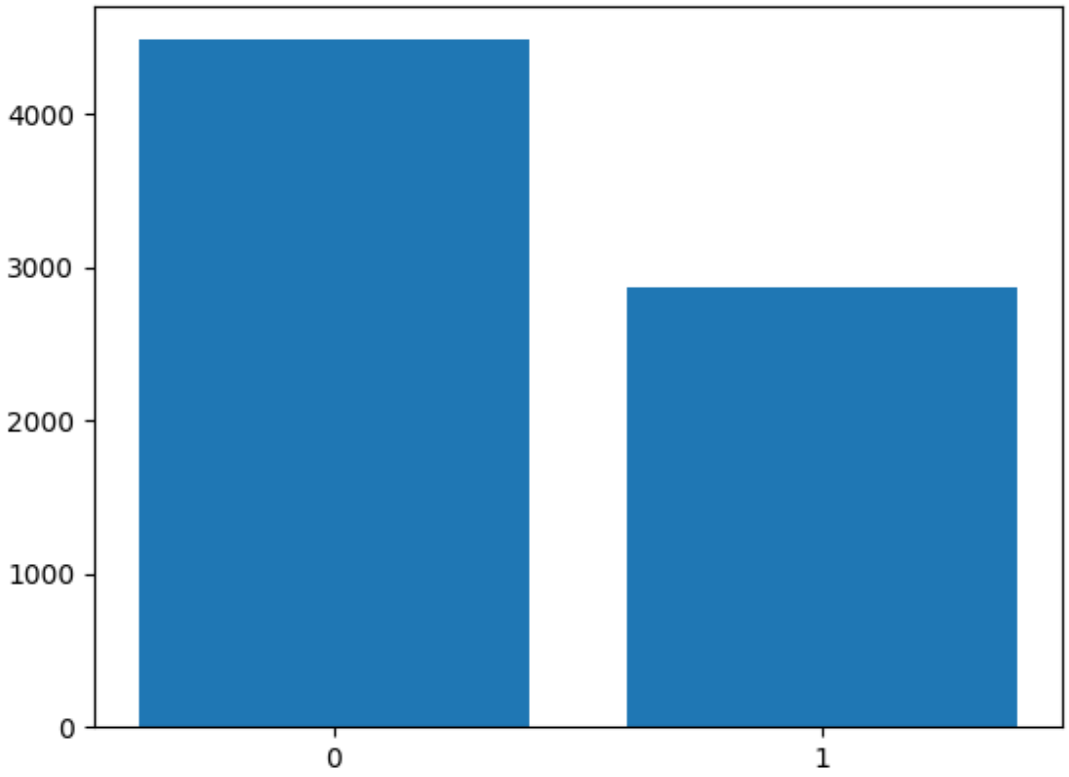
Having noticed that the labels and the extremities/body parts to which the studies refer are not uniformly distributed, a depiction of the actual distributions of the initial train set and the produced train and validation sets was used in order to make sure that the initial distribution was preserved. This can be confirmed through the following graphs. The preservation of the distributions was of course expected due to the shuffling and random picking of the splitter.



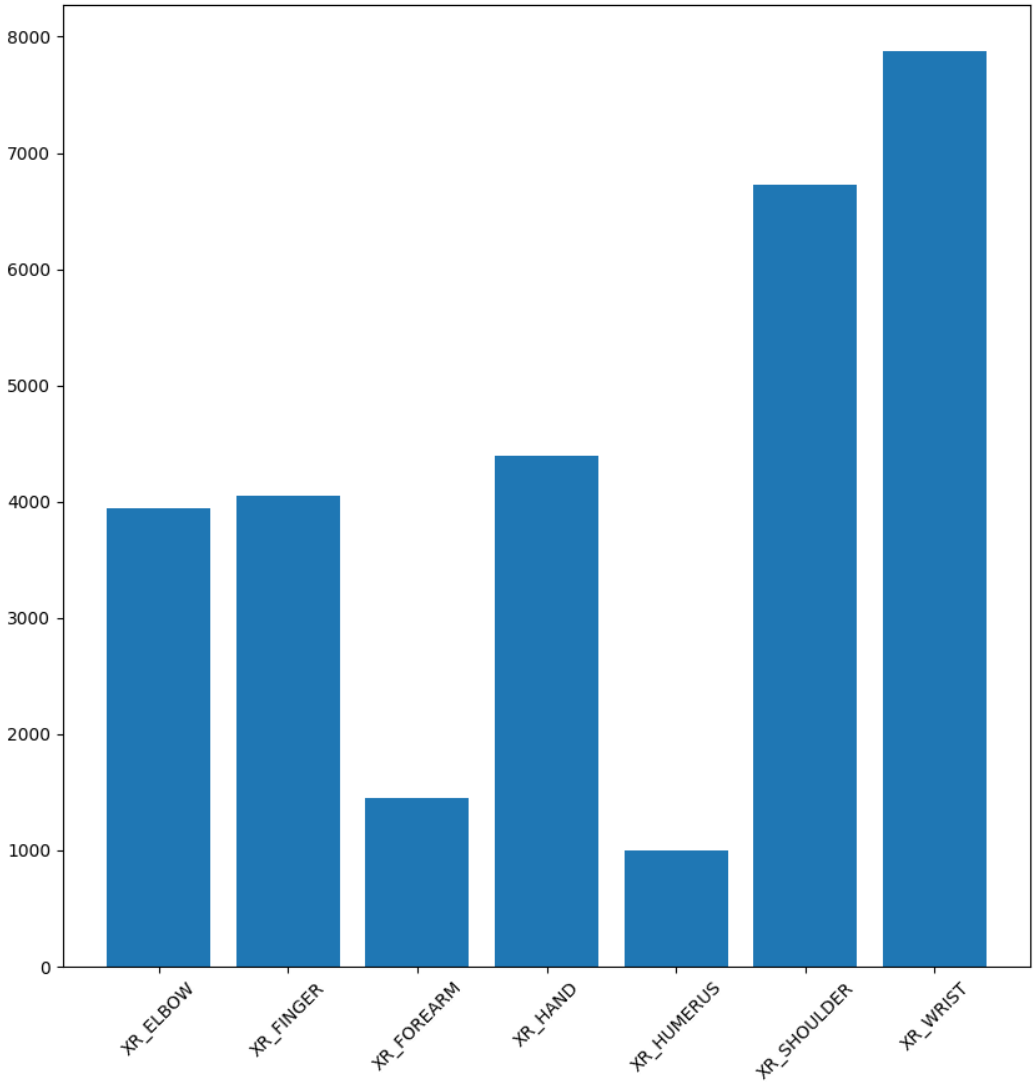
Initial labels distribution



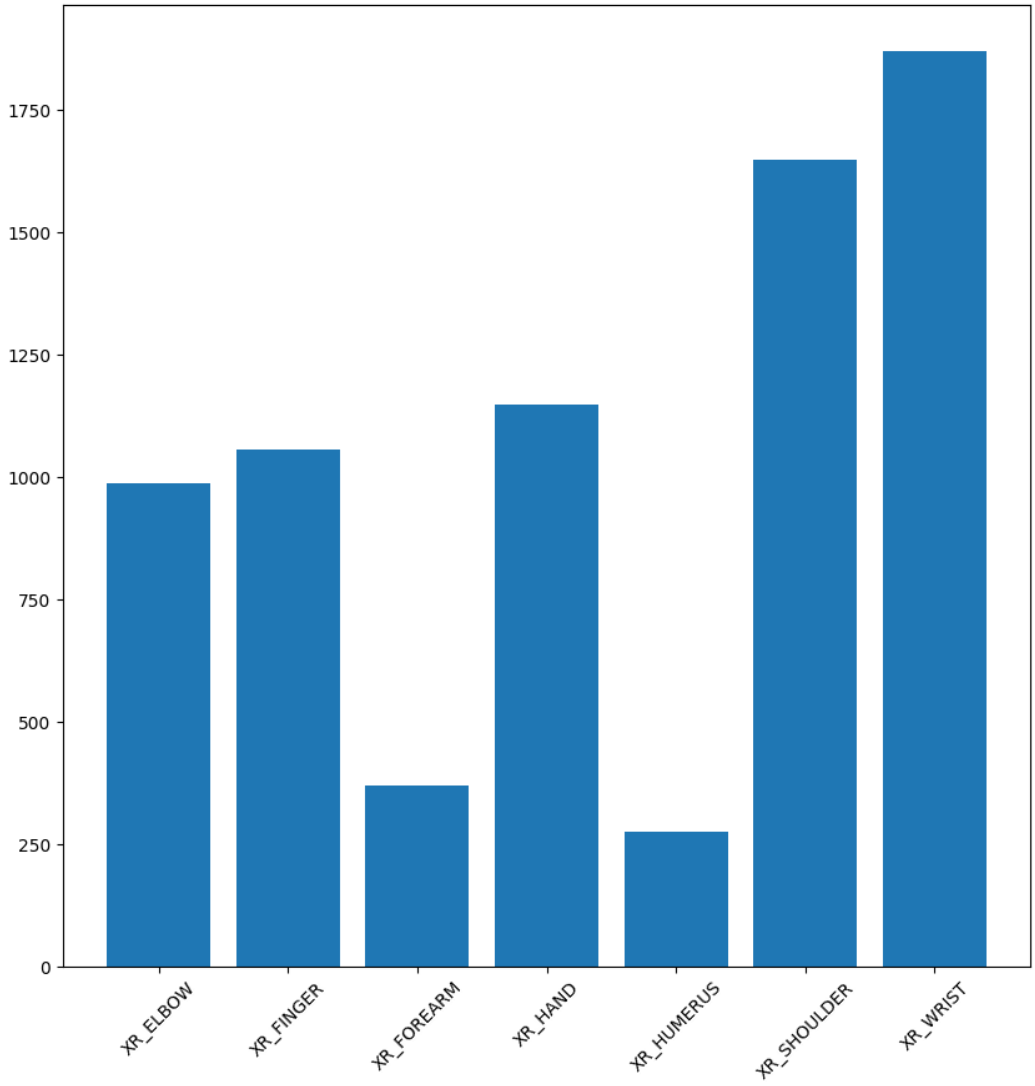
train labels distribution



Validation labels distribution



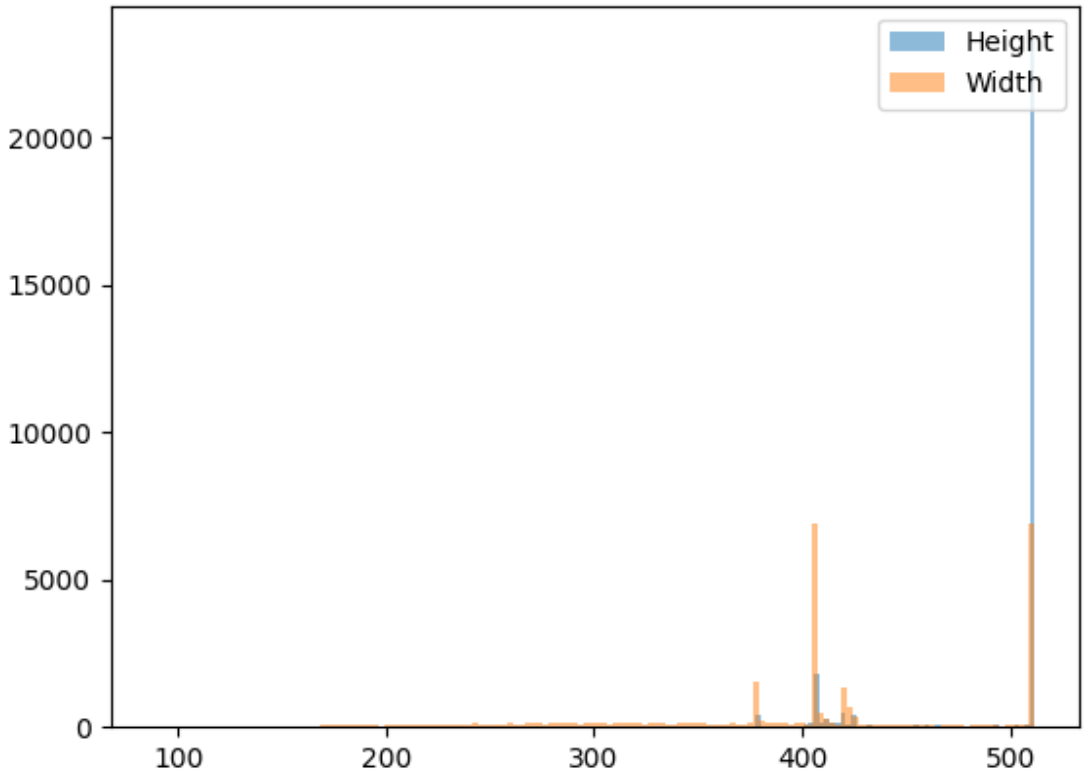
Train extremities distribution



validation extremities distribution

2.2 Image sizes

The dimensions of the png images comprising the dataset were not constant. As can be observed in the following histogram the width varies a lot between the samples, with most of the samples concentrated to either 400 or 512 values, while the height is mainly 512.



Height-width histogram

In order to train the network all the images should have a constant size, so they were resized to be of equal dimensions. In general, shrinking is preferred against enlargement since the latter produces much more noise and some information can be lost. In addition, larger images are more difficult to fit into memory, making it harder and less efficient to train a model. Hence, all the images were shrunk to 224x224 pixels. The specific value of 224 was chosen after several tries with higher values not fitting in memory while an

alternative value of 256 was observed as not being efficient. Furthermore, a pretrained Densenet-121 network was tested which is pre-trained to that kind of image sizes.

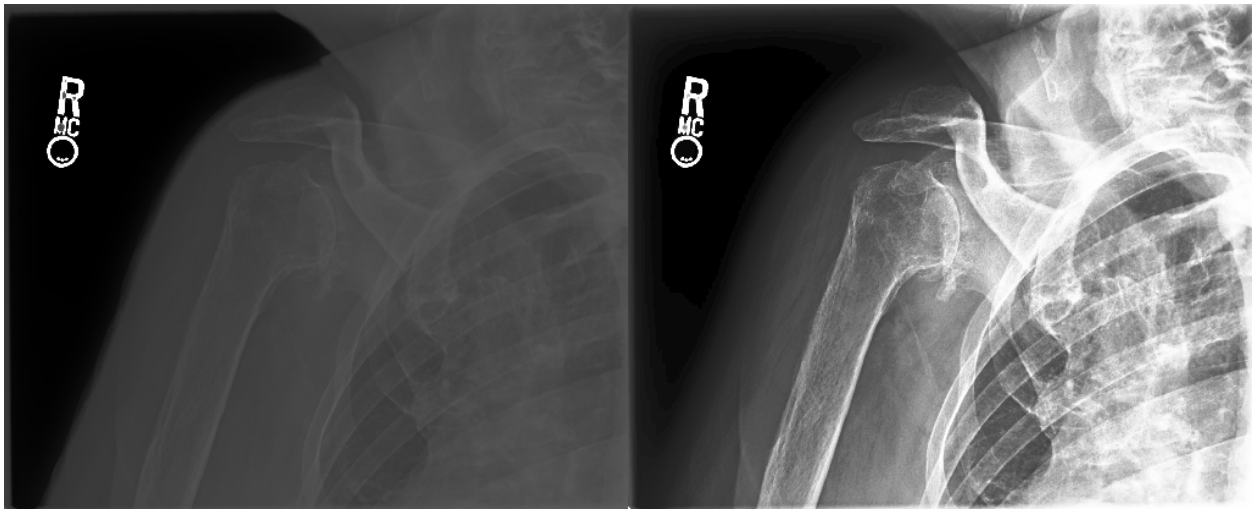
2.3 Preprocessing methods

The dataset in general appeared to be challenging because of the nature of the images and the heterogeneity between them. It was noticed that many times the main parts of the radiographs are rotated and also there is a black frame surrounding them in the majority of the images. In order to facilitate the networks and expect improved results, three different preprocessing methods were tested which were applied before the resizing.

Histogram equalization

https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html

This method improves the contrast of the given images. Since the images are x-rays, it was anticipated that regions of interest would be easier detected by the models resulting in improved performance.

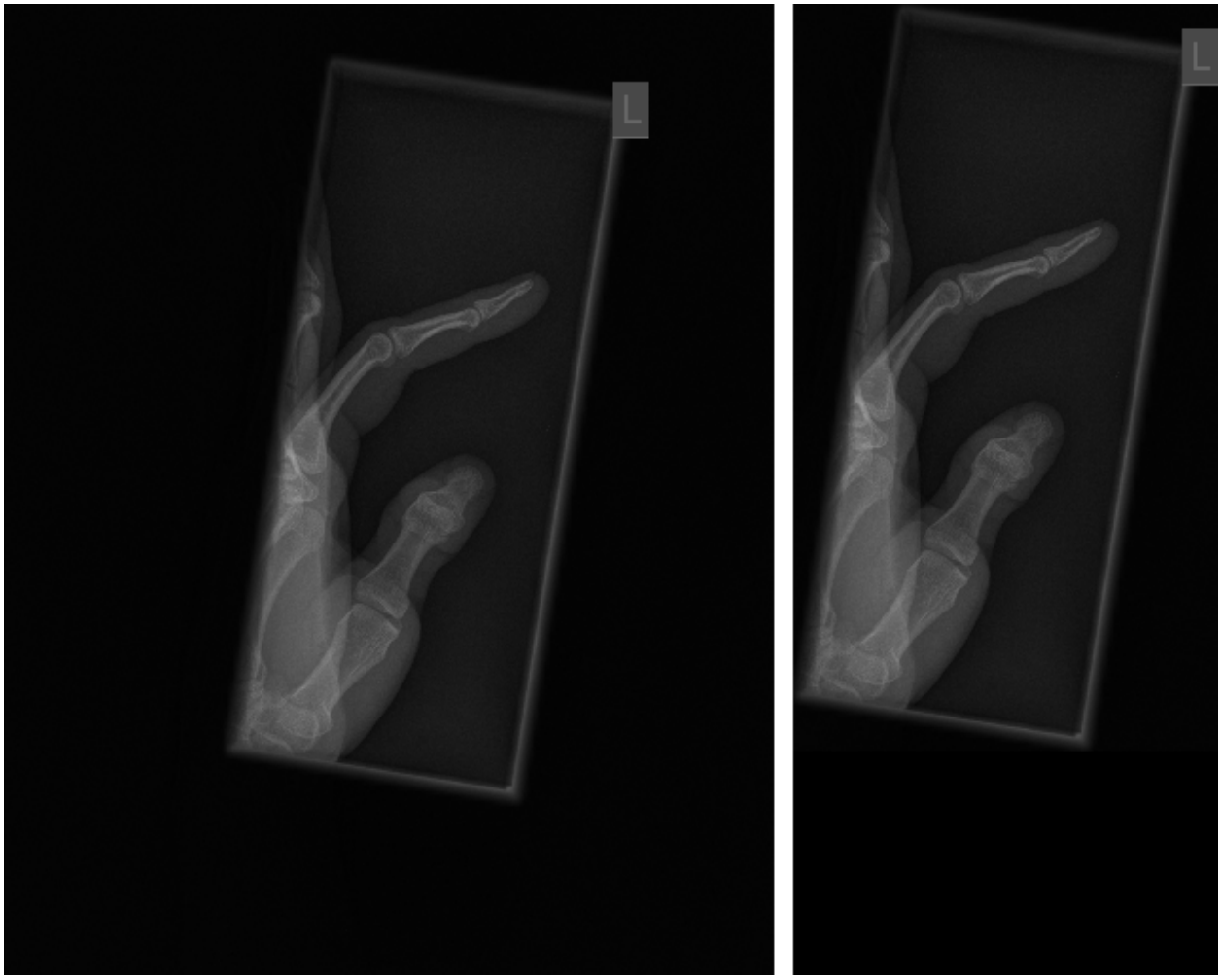


Histogram equalization example

Black Frame cropping

Upon observation, it was noted that numerous images exhibited the x-rays positioned atop a scanner, leading to a black frame surrounding the region of interest. Since this black area

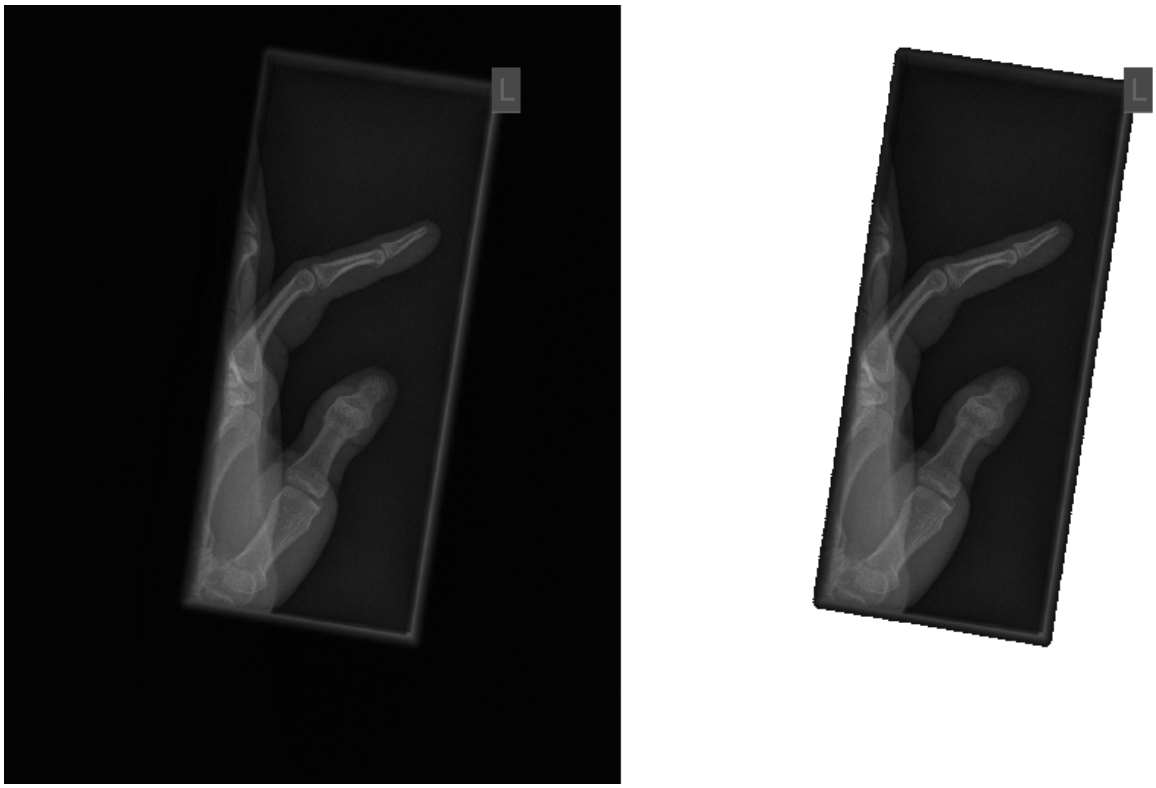
does not contribute to the predictions, it was removed from the images. Regardless, the cropping process always results in a parallelogram-shaped region, irrespective of the angle of the x-ray. Consequently, a small black area may remain surrounding the x-ray within the cropped region.



Black frame cropping example

Black Frame removal

In continuation of the previous method, the complete removal of the black region around the actual x-ray was tested, which however led to rotated images.



Black frame removal example

3. AGGREGATION METHODS

As already mentioned, the entire dataset consists of studies which include in general more than one image. The labels, though, characterize an entire study and not individual images. Thus, an aggregating method should be used, in order to decide what prediction must be

assigned to each study, taking into consideration the predictions for each image comprising the study. Therefore, three different methods of aggregating predictions were tested.

Majority voting

Each study's images participate in a voting process to determine the final prediction. The prediction that receives the majority of votes from the images is assigned to the entire study. However, if an equal number of votes are received for two different classes, the final prediction is chosen randomly.

Average probability

Using the posteriors of the model's output and applying softmax, this method employs a weighted voting process across the images within each study. It calculates the average probability for each predicted class and selects the one with the highest value as the final prediction for the entire study.

Positive max

Revisiting the posteriors, this method focuses solely on the positive class. It examines the posteriors of each image within the study and selects the image with the highest posterior value as the decisive factor for the prediction. The image that exhibits the greatest confidence in its decision among all other images determines the prediction class. If the probability assigned to the positive class exceeds 50%, the prediction is set as positive; otherwise, it is assigned as negative.

4. BASELINE METHOD

For the purpose of having a baseline which can be used to compare the results of the experiments conducted, a dummy classifier that always predicts the highest frequent class was used. The reported f1 macro metric for the predictions on the test set was **0.34** with respect to the class distribution in the train set.

5. CUSTOM CNN

The first series of experiments that were conducted were based on a custom CNN, that was built from scratch. The tested network consisted of stacked cnn layers, each of them followed by a batch normalization layer, a relu activation function, a max-pooling layer and eventually a dropout layer. After these stacked layers, two dense layers follow, with the second one producing the two desired outputs for the binary classification. The first cnn layer accepts one channel as input (grayscale format) and produces a multiple of 8 output channels. The kernel size is always 3, the stride is 1 and the padding also 1.

The network can be parameterized according to the number of the stacked layers, the number of convolutional channels, the probability of dropout and lastly the number of units that comprise the input of the last linear layer. In order to choose the best combination of these parameters the **optuna** tuner of pytorch framework was used.

6. PRETRAINED NETWORK

The second series of experiments were conducted on a densenet-121 network, pretrained on the ImageNet dataset. The network was fine-tuned, utilizing the train and validation sets. The fine tuning procedure was conducted by either only training the last linear layer, training the last 5 layers or training the whole network.

7. RESULTS

7.1 Custom CNN

To train the network, a total of 50 epochs were utilized. Early stopping was employed with a patience of 10, alongside the ReduceLROnPlateau scheduler. The validation f1 macro score was utilized for tuning purposes. In experiments involving the optuna tuner, the following parameters were fine-tuned: the number of convolutional layers (ranging from 1 to 4), the number of channels (ranging from 1 to 4, with the first layer multiplied by 8 and each subsequent layer further multiplied by 2), the dense nodes of the linear layer (ranging from 1 to 5, with a value of 2 raised to the power of (value + 4)), the dropout rate (ranging from 0.1 to 0.9), the optimizer options ("Adam", "RMSprop", "SGD"), and the learning rate (ranging from 1e-5 to 1e-1).

The first experiment in the table corresponds to the best model obtained after optuna tuning, with the following parameter values: 2 convolutional layers, 24 channels in the first layer, dropout rate of 0.22, Adam optimizer, and a learning rate of 0.0006.

In the second experiment, predefined parameter values were initially set empirically. However, it was found that the network did not demonstrate the desired learning ability. These predefined values included: 5 convolutional layers, 1 channel, 5 dense nodes, and a dropout rate of 0.5.

Subsequently, in the last experiment, the first preprocessing method mentioned earlier was employed to assess if it could enhance the results. However, no conclusive improvements have been established thus far.

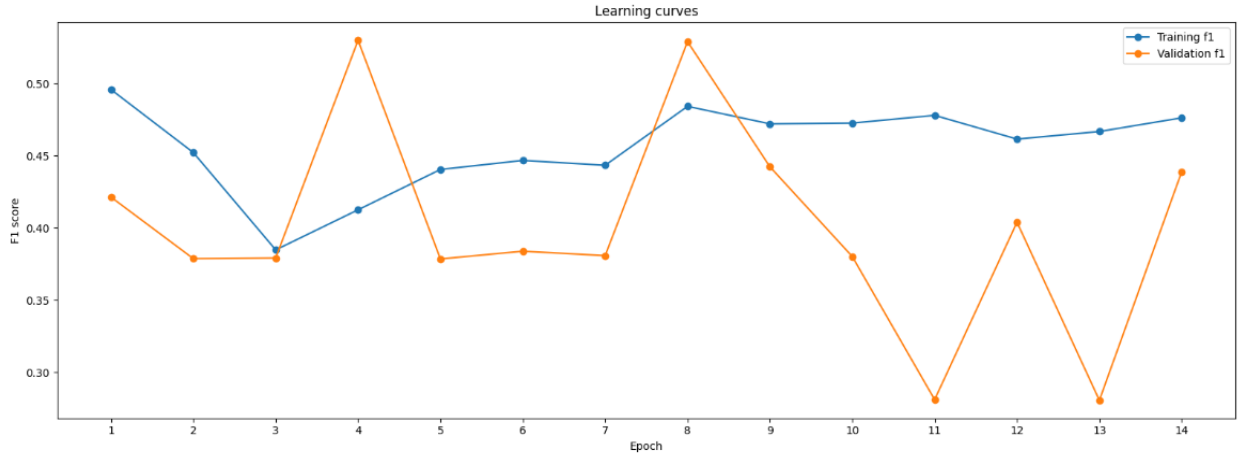
	Preprocessing method	**Tuner**	**F1 validation**	**F1 test**	**Aggregation Method**
1.	No	Optuna	0.53	0,54	No
				0,54	Majority vote

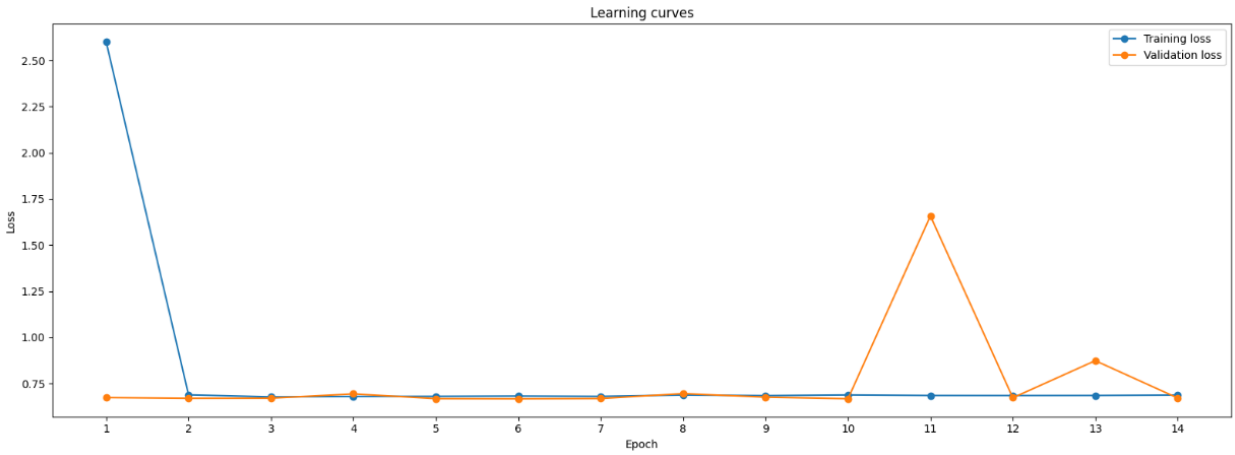
				0,57	Average probability
				0,56	Positive max
2.	No	No	0.38	-	-
3.	Equalization	Optuna	0.53	0,52	No
				0,52	Majority vote
				0,53	Average probability
				0,51	Positive max

Table 1: Custom CNN Experiments

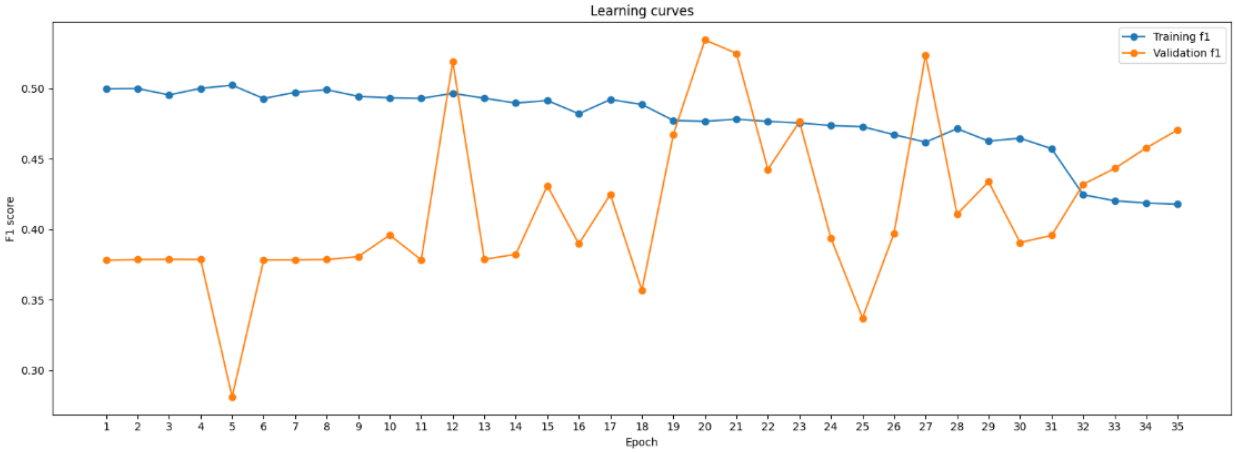
The learning curves for experiments 1 and 3 are displayed below for reference.

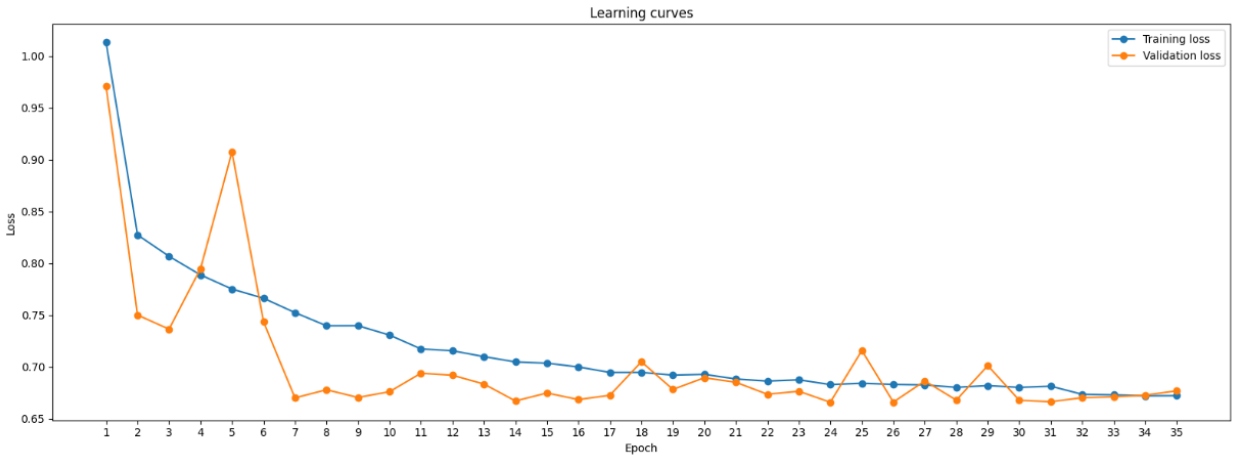
Experiment 1:





Experiment 2:





As previously indicated, in experiment 1, it appears that the network was unable to learn effectively, as the loss on both the training and validation sets had already converged by the second epoch. Consequently, the model exhibited a significant underfitting issue. Conversely, in the third experiment, the utilization of equalization preprocessing resulted in noticeably improved graph distributions, making them more normal in nature.

7.2 Fine-tuned densenet-121

In the following experiments we used SGD as optimizer with starting lr=0.001, and momentum=0.9 (the same used in the pretraining of densenet). We also used ReduceLROnPlateau scheduler and tuned the network on the validation f1 macro score.

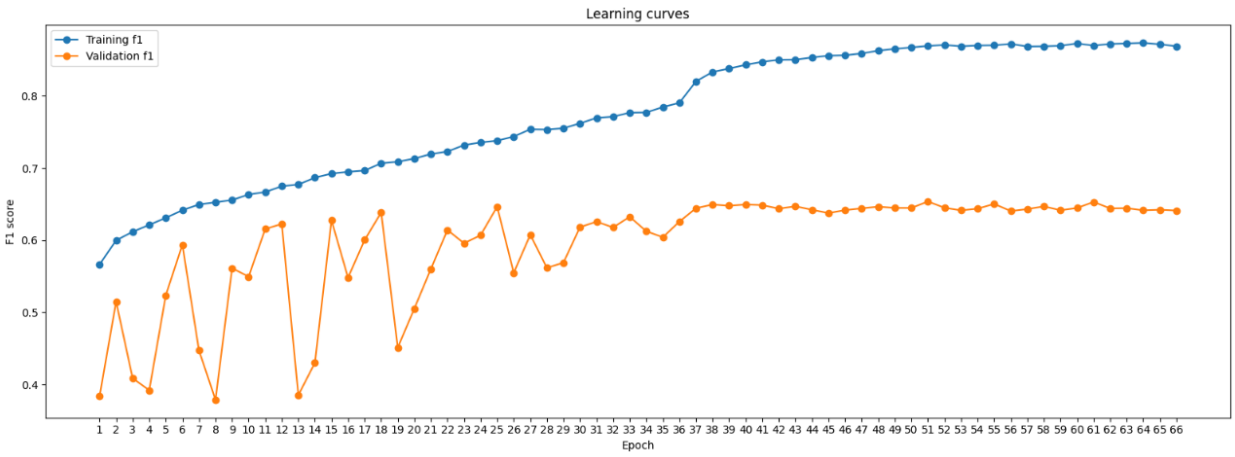
	Preprocessing method	**F1 validation**	**F1 test**	**Aggregation Method**	**#unfrozen layers**	**Epochs**	**Patience**
1.	No	0.53	0.54 Cohen's kappa: 0.09	No	1	100	15
			0.55 Cohen's kappa: 0.10	Majority vote			
			0.56 Cohen's kappa: 0.11	Average probability			
			0.55 Cohen's kappa: 0.11	Positive max			
2.	Equalization	0.59	0.59	No	1	50	15
			0.6	Majority			

				vote			
			0.61	Average probability			
			0.6	Positive max			
3.	Cropping	0.5	0.49	No	1	100	15
			0.49	Majority Vote			
			0.52	Average probability			
			0.56	Positive max			
4.	Removal	**0.65**	0.62	No	1	100	15
			0.63	Majority Vote			
			0.65	Average probability			
			0.66	Positive max			
5.	Removal + Equalization	**0.65**	0.62 Cohen's kappa: 0.25	No	1	100	15
			0.65 Cohen's kappa: 0.30	Majority Vote			
			0.65 Cohen's kappa: 0.32	Average probability			
			0.63 Cohen's kappa:	Positive max			

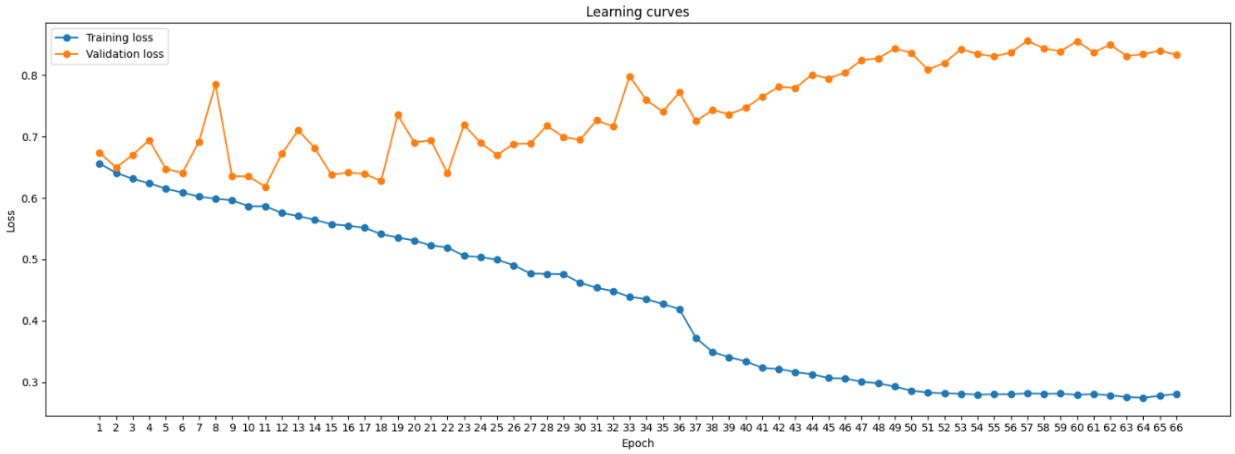
			0.27				
6.	No	0.52	0.50	No	5	50	15
			0.50	Majority Vote			
			0.51	Average probability			
			0.55	Positive max			
7.	No	0.55	0.55	No	all	100	15
			0.55	Majority Vote			
			0.57	Average probability			
			0.58	Positive max			

Table 2: Densenet-121 Experiments

The learning curves of the best method (removal) are shown below:



F1-curves for model 4.



Loss curves for model 4.

The utilization of pretrained DenseNet models consistently produced improved results in most of the experiments, irrespective of the number of unfrozen layers. Among the preprocessing methods employed, the removal method outperformed all others, demonstrating superior performance in terms of both validation and test f1 metrics, with f1 reaching as high as 0.66. Although the utilization of equalization led to improved performance compared to the non-processed dataset in the initial experiment, incorporating it alongside the best preprocessing method (removal) did not yield superior results.

Extremity Analysis

In the best experiment, which involved the combination of the removal preprocessing method and positive max technique, we conducted a comprehensive analysis to assess the performance for each extremity type. The results obtained are as follows:

- XR_ELBOW: f1=0.65, Cohen's kappa=0.31

- XR_FINGER: f1=0.73, Cohen's kappa=0.45

- XR_FOREARM: f1=0.69, Cohen's kappa=0.38

- XR_HAND: f1=0.63, Cohen's kappa=0.26

- XR_HUMERUS: f1=0.68, Cohen's kappa=0.36

- XR_SHOULDER: f1=0.52, Cohen's kappa=0.10

- XR_WRIST: f1=0.66, Cohen's kappa=0.33

Among the extremity types, the highest performance metric was observed for XR_FINGER, while the shoulder extremity exhibited the lowest prediction metrics. These findings provide valuable insights for future work, suggesting the potential utilization of more advanced machine learning methods and models. Specifically, focusing on enhancing the performance of each extremity type through approaches such as ensemble modeling or reinforcing the models targeting the weakest extremity type. By addressing the specific challenges posed by individual extremities, we can aim to improve the overall performance and accuracy of the system.

8. CONCLUSION

The conclusions derived from this study are listed below:

1. The MURA dataset poses a considerable challenge due to the inclusion of diverse image formats. However, through a thorough examination of preprocessing methods, it has been determined that these techniques are applicable and effective for enhancing performance on this specific dataset and task. The implementation of these methods has resulted in improved performance. The best method turned out to be the last one (removal), while the first one (equalization) also helped to performance enhancement.
2. The performance of the system varied depending on the aggregation method employed. Interestingly, the last two methods, namely average voting and positive max, exhibited superior performance and contributed to further enhancements in f1 score.
3. The models developed from scratch demonstrated limited proficiency in learning the specific task within this dataset. Conversely, the utilization of transferred knowledge from pretrained models appeared to yield better results and performance.
4. The chosen metric for tuning the model was f1 score. However, a significant number of relevant papers primarily reported the Cohen's kappa statistic, making it challenging to compare the results directly. As a consequence, the reported kappa statistics did not yield satisfactory outcomes.