



ABNORMALITY DETECTION IN BONE X-RAYS WITH DEEP LEARNING ARCHITECTURES

Deep Learning - Project 2

Sofia Baltzi, AUEB

Contents

Introduction.....	2
Data Preprocessing.....	3
Model Evaluation	4
Architecture 1 – Custom CNN	5
Architecture 2 – Pretrained Network.....	6
Experimental Results	7
Forearm	7
Elbow	7
Finger	8
Humerus	8
Hand	8
Shoulder	9
Wrist	10
Discussion	10
References.....	11

Introduction

The purpose of this document is to report the results for the 2nd Assignment of class Deep Learning, February – April 2021. The objective is to create a classifier, using deep learning architectures, that decides if a study, consisting of bone X-rays, shows abnormalities or not.

The dataset used is the MURA (musculoskeletal radiographs) [1], a large dataset of bone X-rays. The X-rays come from 7 body parts: wrist, shoulder, humerus, elbow, finger, forearm, and hand. We are given a train set of 36,808 images and a validation one of 3,197 images. Both sets are organized by patient and study. Below there can be seen ten image examples.



Figure 1: Image examples with labels

As it can be observed in the following graphs, some classes consist of more images than others. In the training set, for all body parts, we are given more images belonging to negative class than positive. The validation set is more balanced.

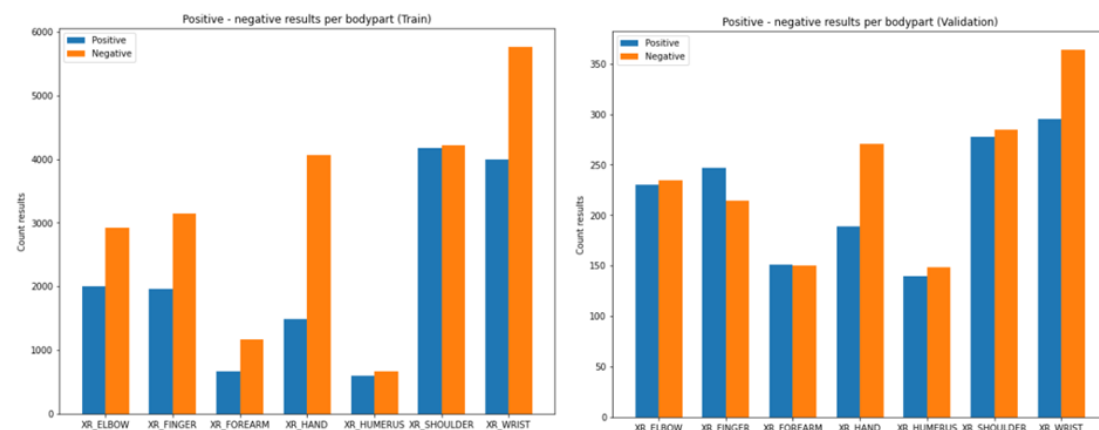


Figure 2: Class balance (on the left: train set, on the right: validation set)

*Code for the project can be found [here](#).

Data Preprocessing

Since all images are of different sizes, as a first preprocessing step, we resize them to (224,224). Below, are some examples of the resized images.

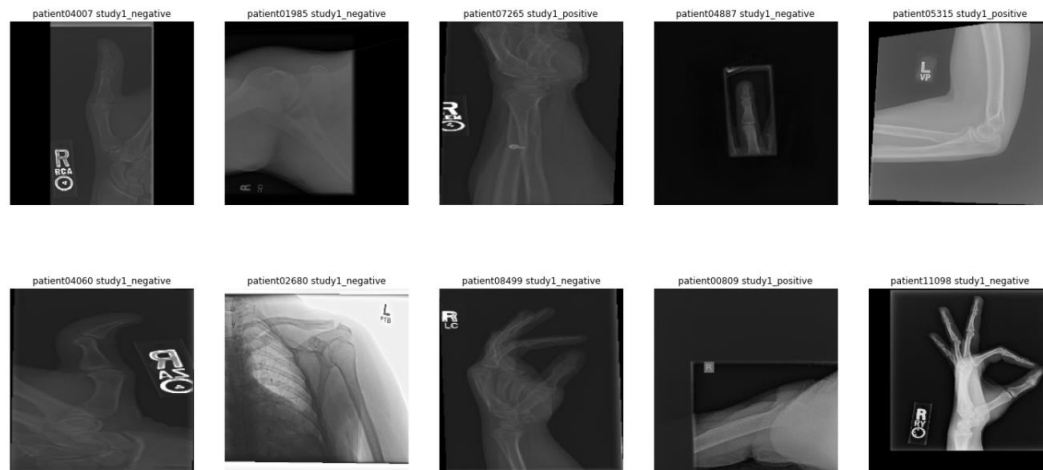


Figure 3: Resized images

For each body part, we keep 20% of training images to validate our models. The rest 80% is used for training. The old validation set is used as the test set, to evaluate our models.

Following that step, we use ImageDataGenerator class to create the following real-time image augmentations:

- Rotation: we randomly rotate images from -5 to 5 degrees
- Horizontal flip: images are flipped horizontally
- Zoom: images are zoomed randomly from 90% to 110% of the originals
- Shift: images are shifted randomly up to 10% to any direction

We create one generator for each one of training, validation, and test sets. The augmentations are performed only on the training set. For the training and validation sets and only for the pretrained model we also use the predefined preprocessing input function. For the custom CNN model, we rescale each pixel value by dividing it by 255.

The generators as well as the `flow_from_dataframe` method are used to feed our images to our networks in batches of 32 at the training phase. At the testing phase we pass the images in batches of 64.

The method, by default, validates the image filenames, a process that is time consuming. Therefore, we disable that functionality by setting the corresponding input to False.

Model Evaluation

For the evaluation of our models we use the following metrics:

- Accuracy: ratio of correctly predicted observation to the total observations

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

- F1 score: weighted average of Precision and Recall

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- Precision score: ratio of correctly predicted positive observations to the total predicted positive observations

$$Precision = \frac{TP}{TP + FP}$$

- Recall score: ratio of correctly predicted positive observations to all the observations in actual positive class

$$Recall = \frac{TP}{TP + FN}$$

Both precision and recall, focus more on the positive class. In our case, negative class has almost always (5 out of 7 body parts) more observations than positive class. Therefore, we additionally use Cohen's kappa coefficient as an evaluation metric.

Cohen's kappa is a score that expresses the level of agreement between two annotators on a classification problem [2]. The first annotator will always be the true class – the class assigned by the creators of the dataset. The second annotator will be our classifier. We are interested in examining whether our predictions are correct due to chance or not.

$$\kappa = \frac{P_{obs} - P_{exp}}{1 - P_{exp}}$$

Evaluation is performed on two levels:

- First, we evaluate based on the answer per image, and
- Second, we evaluate based on the reply per study. That comes from the majority vote of the answers for all images per patient and study and is the answer that would be given to a patient.

Accuracy, F1, Precision and Recall range from 0 to 1. Kappa ranges from -1 to 1. The maximum value means complete agreement between annotators, while 0 or lower means that agreement between annotators is due to chance. That metric is more useful than accuracy in cases of high imbalance between classes, where our predictor could always predict the majority class.

Architecture 1 – Custom CNN

The first architecture that we test consists of multiple convolutional layers. Our input is a batch of images of size (224, 224, 3). The first layer is a convolutional layer, with 16 (3x3) kernels and ReLU as activation function, followed by (2x2) MaxPooling with stride 2 and 20% dropout rate. Then, follows another convolutional layer, with 32 (3x3) kernels and ReLU as activation function, followed by (2x2) MaxPooling with stride 2 and 20% dropout rate. The final convolutional layer, has 64 (3x3) kernels and ReLU as activation function, followed by (2x2) MaxPooling with stride 2 and 20% dropout rate. The next layer is a Dense layer of 256 units and 40% dropout rate with ReLU as activation function. Finally, the output is a Dense layer of 1 unit and sigmoid as activation function. The output layer gives the probability for an image to belong to class 1 – positive.

Layer (type)	Output Shape	Param #
Input (InputLayer)	[(None, 224, 224, 3)]	0
Conv1 (Conv2D)	(None, 224, 224, 16)	448
MaxPool2D1 (MaxPooling2D)	(None, 112, 112, 16)	0
Dropout1 (Dropout)	(None, 112, 112, 16)	0
Conv2 (Conv2D)	(None, 112, 112, 32)	4640
MaxPool2D2 (MaxPooling2D)	(None, 56, 56, 32)	0
Dropout2 (Dropout)	(None, 56, 56, 32)	0
Conv3 (Conv2D)	(None, 56, 56, 64)	18496
MaxPool2D3 (MaxPooling2D)	(None, 28, 28, 64)	0
Dropout3 (Dropout)	(None, 28, 28, 64)	0
Flatten (Flatten)	(None, 50176)	0
Hidden1 (Dense)	(None, 256)	12845312
Dropout4 (Dropout)	(None, 256)	0
Output (Dense)	(None, 1)	257
Total params: 12,869,153		
Trainable params: 12,869,153		
Non-trainable params: 0		

Figure 4: Custom CNN

The model is trained in batches of 32 for 30 epochs, monitoring validation accuracy in a split equal to 20% of the training set. Adam optimizer is used to minimize the loss function that is binary crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs.

Architecture 2 – Pretrained Network

The second architecture we explore is one that takes advantage of pretrained models. We choose to use DenseNet-121, a DenseNet of depth 121. DenseNet (Dense Convolutional Network) is an architecture that focuses on making deeper networks that are efficient to train, by using shorter connections between the layers [3], [4]. Essentially, each layer is connected to all the other deeper layers allowing for maximum information flow from layer to layer.

Our model consists of a custom input of shape (224, 224, 3), followed by the pretrained network with non-trainable weights, leading to a Dense layer of 256 nodes with ReLU as the activation function and finally to a Dense layer of 1 node with sigmoid activation function. The output gives the probability for an image to belong to class 1 – positive.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
densenet121 (Functional)	(None, 1024)	7037504
dense (Dense)	(None, 256)	262400
dense_1 (Dense)	(None, 1)	257
Total params: 7,300,161		
Trainable params: 262,657		
Non-trainable params: 7,037,504		

Figure 5: Pretrained model

The model is trained in batches of 32 for 30 epochs, monitoring validation accuracy in a split equal to 20% of the training set. Adam optimizer is used to minimize the loss function that is binary crossentropy. Early stopping is used as a callback, terminating the training process if a better validation accuracy is not achieved for 10 consecutive epochs.

The same model is trained for all body parts. In the following paragraphs, we present the achieved results.

First, we present the results per input image. Then, we group the answers per patient and per study. By averaging the predicted results and rounding to the closest integer (0 or 1) we end up with the final prediction.

Experimental Results

Forearm

For the study of the forearm we have 1,460 training images, 365 validation images and 301 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	36.22 %	50.17 %
<i>Negative</i>	63.78 %	49.83 %

The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
<i>Metrics</i>	<i>Per image</i>	<i>Per patient</i>	<i>Per image</i>	<i>Per patient</i>
<i>Accuracy</i>	60.13 %	57.14 %	77.08 %	83.46 %
<i>F1</i>	46.43 %	43.56 %	73.56 %	81.36 %
<i>Precision</i>	71.23 %	59.46 %	87.27 %	88.89 %
<i>Recall</i>	34.44 %	34.38 %	63.58 %	75.00 %
<i>Kappa</i>	0.2	0.13	0.54	0.67

Custom model is probably usually labeling the images as negative, since the accuracy is close to negative class balance percentage and kappa score is relatively low. Pretrained model is performing much better, given very high kappa scores.

Elbow

For the study of the elbow we have 3,944 training images, 987 validation images and 465 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	40.68 %	49.46 %
<i>Negative</i>	59.32 %	50.54 %

The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
<i>Metrics</i>	<i>Per image</i>	<i>Per patient</i>	<i>Per image</i>	<i>Per patient</i>
<i>Accuracy</i>	58.49 %	60.76 %	78.71 %	79.11 %
<i>F1</i>	47.70 %	47.46 %	76.71 %	75.19 %
<i>Precision</i>	63.31 %	53.85 %	83.59 %	74.63 %
<i>Recall</i>	38.26 %	42.42 %	70.87 %	75.76 %
<i>Kappa</i>	0.17	0.17	0.57	0.57

Observe that even though pretrained network works well on this body part, our custom network performs poorly. Based on precision and recall scores, we are probably predicting negative class most of the times.

Finger

For the study of the finger we have 4,084 training images, 1,022 validation images and 461 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	38.54 %	53.58 %
<i>Negative</i>	61.46 %	46.42 %

The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
<i>Metrics</i>	<i>Per image</i>	<i>Per patient</i>	<i>Per image</i>	<i>Per patient</i>
<i>Accuracy</i>	63.56 %	70.29 %	68.76 %	70.86 %
<i>F1</i>	55.79 %	60.61 %	64.36 %	62.22 %
<i>Precision</i>	79.70 %	81.63 %	82.80 %	80.77 %
<i>Recall</i>	42.91 %	48.19 %	52.63 %	50.60 %
<i>Kappa</i>	0.29	0.39	0.39	0.40

Humerus

For the study of the humerus we have 1,017 training images, 255 validation images and 288 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	47.09 %	48.61 %
<i>Negative</i>	52.91 %	51.39 %

The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
<i>Metrics</i>	<i>Per image</i>	<i>Per patient</i>	<i>Per image</i>	<i>Per patient</i>
<i>Accuracy</i>	65.62 %	68.89 %	82.64 %	85.93 %
<i>F1</i>	60.56 %	68.18 %	81.34 %	85.93 %
<i>Precision</i>	68.47 %	69.23 %	85.16 %	85.29 %
<i>Recall</i>	54.29 %	67.16 %	77.86 %	86.57 %
<i>Kappa</i>	0.31	0.38	0.65	0.72

Compared to the rest of the body parts, our predictions are better on humerus images. That is one of the most balanced body part sets.

Hand

For the study of the hand we have 4,434 training images, 1,109 validation images and 460 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	26.77 %	41.09 %
<i>Negative</i>	73.23 %	58.91 %

The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
<i>Metrics</i>	<i>Per image</i>	<i>Per patient</i>	<i>Per image</i>	<i>Per patient</i>
<i>Accuracy</i>	62.17 %	64.07 %	72.17 %	73.05 %
<i>F1</i>	16.35 %	16.67 %	56.46 %	55.45 %
<i>Precision</i>	89.47 %	100.0 %	79.05 %	80.00 %
<i>Recall</i>	8.99 %	9.09 %	43.92 %	42.42 %
<i>Kappa</i>	0.10	0.11	0.38	0.39

Observe that while precision is high, recall is very low. That means that we tend to classify more images to negative class. That probably comes from the fact that training set is very imbalanced. Our false predictions also reflect on our kappa score.

Shoulder

For the study of the shoulder we have 6,703 training images, 1,676 validation images and 563 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	49.74 %	49.38 %
<i>Negative</i>	50.26 %	50.62 %

The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
<i>Metrics</i>	<i>Per image</i>	<i>Per patient</i>	<i>Per image</i>	<i>Per patient</i>
<i>Accuracy</i>	69.98 %	68.56 %	68.56 %	69.07 %
<i>F1</i>	66.53 %	66.30 %	71.03 %	72.73 %
<i>Precision</i>	74.01 %	69.77 %	65.17 %	64.00 %
<i>Recall</i>	60.43 %	63.16 %	78.96 %	84.21 %
<i>Kappa</i>	0.40	0.37	0.37	0.39

Shoulder is a balanced body part and therefore, we expect better predictions. That is the case for the custom network. Shoulder is one of the best performing body parts. That is not the case for the pretrained network though. Even though the performances are about the same between the two models, that is one of our worst body parts considering the pretrained network.

Wrist

For the study of the wrist we have 7,801 training images, 1,951 validation images and 659 testing images.

In the following table, we present the class balance in training and test sets (validation set was created maintaining the class balance of training set):

	<i>Train</i>	<i>Test</i>
<i>Positive</i>	40.88 %	44.76 %
<i>Negative</i>	59.12 %	55.24 %





























The achieved results are presented below:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
Metrics	Per image	Per patient	Per image	Per patient
<i>Accuracy</i>	72.08 %	75.53 %	76.18 %	79.32 %
<i>F1</i>	60.00 %	62.34 %	70.54 %	72.63 %
<i>Precision</i>	84.64 %	84.21 %	78.99 %	79.27 %
<i>Recall</i>	46.78 %	49.48 %	63.73 %	67.01 %
<i>Kappa</i>	0.41	0.46	0.51	0.56

That is our best body part concerning custom model performance.

Discussion

Let's observe the kappa scores of all our models and make some general remarks:

	<i>Custom CNN</i>		<i>Pretrained network</i>	
Body part	Per image	Per patient	Per image	Per patient
<i>Forearm</i>	 0.20	 0.13	 0.54	 0.67
<i>Elbow</i>	 0.17	 0.17	 0.57	 0.57
<i>Finger</i>	 0.29	 0.39	 0.39	 0.40
<i>Humerus</i>	 0.31	 0.38	 0.65	 0.72
<i>Hand</i>	 0.10	 0.11	 0.38	 0.39
<i>Shoulder</i>	 0.40	 0.37	 0.37	 0.39
<i>Wrist</i>	 0.41	 0.46	 0.51	 0.56

- Pretrained network performs better on all body parts, which is expected since it is previously trained on large datasets.
- Per patient/ study predictions are almost always better than per image predictions, an expected result, since they aggregate the answers on many images.
- Performance is different between different body parts.
- Models perform better when training set is balanced between positive and negative classes.
- The fact that a model is good for a specific body part that does not mean it will be a good fit for all body parts.

References

- [1] <https://stanfordmlgroup.github.io/competitions/mura/>
- [2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html
- [3] <https://keras.io/api/applications/densenet/#densenet121-function>
- [4] <https://towardsdatascience.com/creating-densenet-121-with-tensorflow-edbc08a956d8>