Classification of Musculoskeletal Abnormalities with Convolutional Neural Networks

Guilherme Tiaki Sassai Sato *Leodécio Braz da Silva Segundo*
Zanoni Dias

Relatório Técnico - IC-PFG-20-14
Projeto Final de Graduação
2020 - Agosto

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Classification of Musculoskeletal Abnormalities with Convolutional Neural Networks

Guilherme Tiaki Sassai Sato **Leodécio Braz da Silva Segundo**
Zanoni Dias

Institute of Computing
University of Campinas

Abstract

Computer-aided diagnosis has the potential to alleviate the burden on medical doctors and decrease misdiagnosis, but building a successful method for automatic classification is challenging due to insufficient labeled data. In this work, we investigate the usage of convolutional neural networks to diagnose musculoskeletal abnormalities using radiographs (X-rays) of the upper limb and measure the impact of several techniques in our model. While these techniques are overall well-established, some did not generalized to out setting. We achieved the best results by utilizing an ensemble model that employs a support vector machine to combine different models, resulting in an overall AUC ROC of 0.8791 and Kappa of 0.6724 when evaluated using an independent test set.

1 Introduction

Musculoskeletal conditions are extensively present in the population, affecting over 1.3 billion people worldwide [1]. These conditions often cause long-term pain, directly and indirectly reducing the quality of life of those suffering from it and their household [2, 3]. In this setting, medical imaging as X-rays plays an essential role as one of the main tools for abnormality detection.

Insufficient medical staff, along with the complexity of diagnosis, creates a system prone to errors. False-negative diagnosis leads to untreated injuries and symptoms such as chronic pain and further complications in the long term, and false-positives diagnosis leads to unnecessary treatment. Computer-Aided Diagnosis (CAD) systems are used to counteract these problems, improve diagnostic accuracy, and assist decision-making, alleviating the burden on radiologists.

Computer-aided diagnosis has been a topic of research since the 1960s and has significantly evolved, due to advances in medicine itself and in computer science. From a task standpoint, CAD has found application in a wide variety of medical disorders. A few of the innumerous works include breast cancer [4, 5], lung cancer [6], and Alzheimer's [7]. X-ray classification has been particularly prevalent for the chest area [8, 9, 10]. Under the same scope of this work (musculoskeletal abnormalities in the upper limb), 70 works were submitted under a competition[1] that took place using the same dataset as this work. The reported achieved Cohen's Kappa range from 0.518 to 0.843. However, no further information, such as methodology, is generally available for these works.

Methodology-wise, some of the most successful classifiers include k-nearest neighbors (KNN) [11, 12], support vector machines (SVM) [13], random forests [14, 15], and neural networks [16, 17]. In this work, we will evaluate the use of a neural network classifier due to its promising performance, producing state-of-the-art results in many other applications [18, 19].

A wide range of techniques, such as deep learning, image processing, and computer vision, are applied to interpret radiographic images automatically. However, deep learning models' success is highly dependent on the amount of data available, creating a particular challenge for medical images due to privacy concerns and time-consuming labeling requiring experts. In this work, we present a method to classify normal and abnormal X-rays from the upper limb by applying convolutional neural networks and several machine learning techniques aiming to improve the classification and offset the lack of data. Concretely, we will assess the performance implication of employing transfer learning, pre-processing images, manual annotation, data augmentation, and ensembling.

The rest of this report is organized as follows. In Section 2, we review a few concepts that are necessary for this work. Section 3 describes the settings of this work and all the experiments executed to reach a final classifier, and Section 4 evaluates this classifier using an independent test set and discusses the results obtained, as well as alternative scenarios. Section 5 concludes the report.

2 Theoretical Fundamentals

This section reviews some of the foundations of this work. Subsections 2.1 through 2.3 explain concepts that are applied during the experiments, while Subsection 2.4 defines the evaluation metrics used.

2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) is a class of artificial neural network, frequently used in computer vision applications. Neural networks are inspired by the operation of neurons in the biological brain, so each neuron receives an input, performs some computation, and passes on the output to neurons in the next layer [20].

A CNN takes a multidimensional tensor as input, unlike regular neural networks. This characteristic allows the CNN architecture to be far better optimized to process images, whereas the regular neural network would require a huge number of parameters, and be prone to overfitting.

[1] `https://stanfordmlgroup.github.io/competitions/mura/`

The architecture of a CNN consists of an input layer, several hidden layers (including convolutional, activation, and pooling layers) and an output layer. Figure 1 illustrates this typical structure.

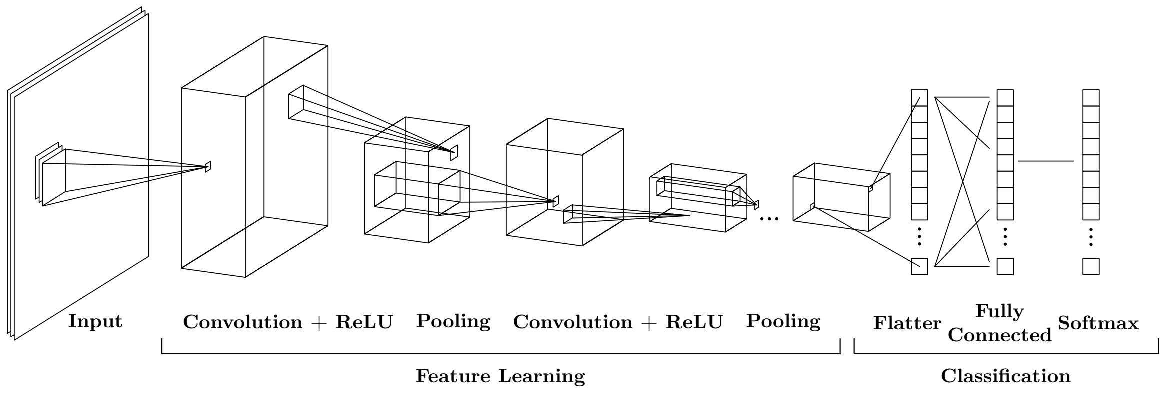
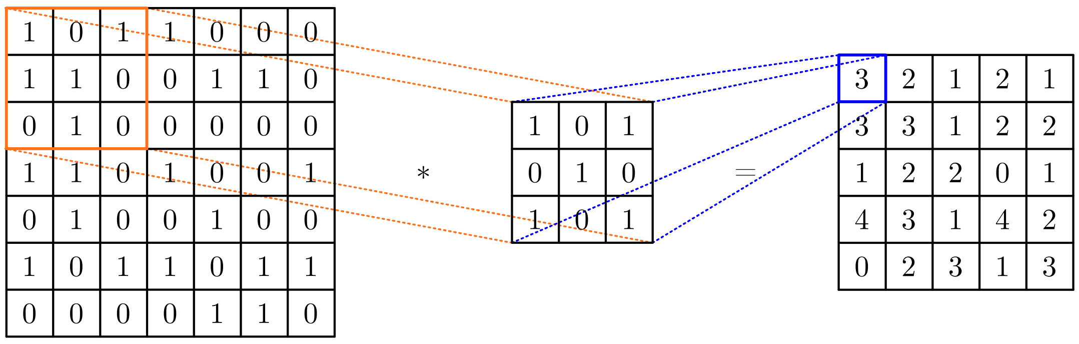


Figure 1: Typical architecture of a convolutional neural network. The input, a three-channel image, is processed by a series of convolutional, activation, and pooling layers to create the feature layer. The feature layer is then flattered to become the input of a fully connected layer. The fully connected layer output is transformed into a probability distribution by the softmax function, giving the classifier prediction probability for each of the classes. Adapted from MathWorks [21].

2.1.1 Convolution Layer

The convolution layer uses a kernel that is convolved across the whole image performing element-wise multiplication between kernel values and the input matrix, outputting a new matrix. This process is illustrated in Figure 2.



$$\begin{bmatrix}1&0&1&1&0&0&0\\1&1&0&0&1&1&0\\0&1&0&0&0&0&0\\1&1&0&1&0&0&1\\0&1&0&0&1&0&0\\1&0&1&1&0&1&1\\0&0&0&0&1&1&0\end{bmatrix} * \begin{bmatrix}1&0&1\\0&1&0\\1&0&1\end{bmatrix} = \begin{bmatrix}3&2&1&2&1\\3&3&1&2&2\\1&2&2&0&1\\4&3&1&4&2\\0&2&3&1&3\end{bmatrix}$$

Figure 2: Convolution operation on a 7×7 matrix using a 3×3 kernel, resulting in a 5×5 matrix.

The optimal value for the kernel is learned during the training phase of the network. This operation takes into account the spacial arrangement of the pixels, allowing the model to learn features that depend on neighbor pixels. Moreover, unlike regular neural networks, it avoids connecting all neurons from one layer to all neurons from the next layer.

2.1.2 Activation Layer

The activation layer adds non-linearity to the model and is typically performed by the ReLU (Rectified Linear Units) function, defined as:

$$f(x) = max(0, x) \tag{1}$$

which transforms all $x < 0$ into 0 and leaves $x \geq 0$ untouched. It is usually placed right after a convolution layer [22].

2.1.3 Pooling Layer

The pooling layer performs a downsample of the image, reducing the number of parameters, and hence reducing computation needed and overfitting, while retaining critical information.

In this operation, the values inside a region of the map generated by the convolutional layer are replaced by a metric over this region. Among these metrics are average, min, and max, max being the most common, where the values in a region are replaced by the maximum value in it [22]. Figure 3 illustrates the max pooling operation.

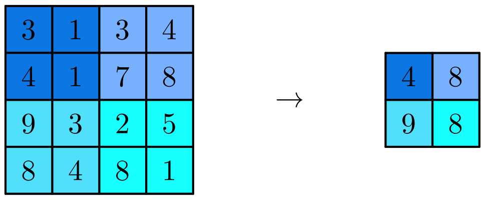


Figure 3: Max pool operation using a 2×2 filter, which eliminates 3 in every 4 pixels. The image is divided into regions the size of a filter, and for each region, only the highest value in the region is outputted.

2.2 Transfer Learning

Properly training a neural network might require large amounts of data, which is not always available. Transfer learning is a machine learning technique in which a previously trained model can be reused in another similar task, instead of starting training from scratch, resulting in reduced training time and increased accuracy [23].

2.3 Data Augmentation

Another technique employed to overcome the limited sized dataset is data augmentation. Data augmentation consists of generating more samples from the existing samples by applying geometric transformations (e.g., flipping, cropping, and rotation) or color transformations (e.g., contrast, saturation, and hue). This technique will lead to a more diverse dataset, reducing data overfitting and increasing the model's performance.

2.4 Evaluation Metrics

The results obtained by this work will be measured quantitatively using four metrics: accuracy, balanced accuracy, Cohen's Kappa, and Area Under the Receiver Operating Characteristic Curve (AUC ROC). All metrics are evaluated using binary labels.

		Prediction	
		Positive	Negative
Input	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 4: Typical confusion matrix for binary classification

2.4.1 Accuracy

The most straightforward metric, accuracy, simply measures the proportion of correct predictions among all samples. Given the confusion matrix in Figure 4, it is defined by:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

2.4.2 Balanced Accuracy

When the classes' distribution in the test set is unbalanced, the accuracy may mislead us with a good result. Balanced accuracy takes into account this imbalance and is defined as the average between the sensitivity (or true positive rate) and specificity (or true negative rate) of the model. Given the confusion matrix in Figure 4, it is calculated by:

$$\text{Balanced Accuracy} = \frac{\dfrac{TP}{TP + FN} + \dfrac{TN}{TN + FP}}{2} \tag{3}$$

2.4.3 AUC ROC

The AUC ROC score is equivalent to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [24]. The score can be obtained by plotting the true positive rate vs. the false positive rate at varying classification thresholds, resulting in a plot similar to Figure 5. The AUC ranges from 0 to 1, where 1 means a perfect classifier, 0.5 a random classifier, and 0 a completely wrong classifier.

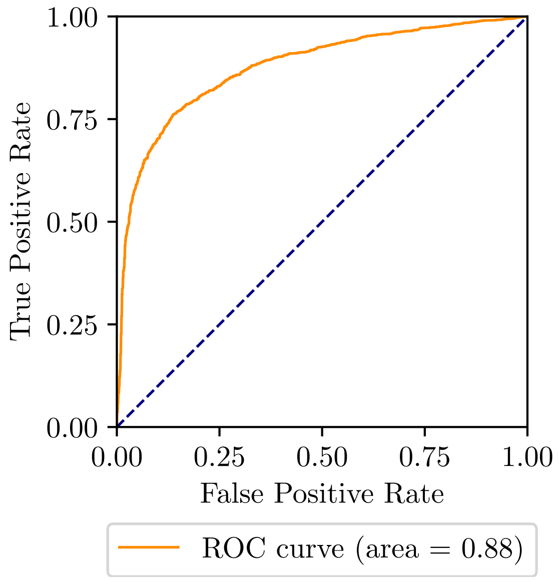


Figure 5: Plot of a receiver operating characteristic curve

2.4.4 Cohen's Kappa

The Cohen's Kappa coefficient (κ) provides a more robust metric, as it aims to measure the degree of agreement between the input and the predictions, excluding the agreement by chance [25]. Given the confusion matrix in Figure 4, it is defined by:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{4}$$

where p_0 is the proportion of agreement, the same as the accuracy above, and p_e is the expected proportion of agreements by chance:

$$p_e = \frac{(TP + FN) \times (TP + FP) \times (TN + FP) \times (TN + FN)}{(TP + TN + FP + FN)^2} \tag{5}$$

The web page of the dataset used in this work contains a leaderboard listing the Kappa coefficient achieved by each of the previously submitted models.[2] Since this data provides a basis of comparison, the ultimate goal of this work is to maximize the Cohen's Kappa on the test set, where each study is a sample, and each sample should be classified between normal and abnormal.

3 Methodology

This section goes over the dataset used in this work and details the experiments executed. Over the experiments, we explore different scenarios by applying machine learning and deep learning techniques and measure the impact of the proposed changes when comparing to previously tested scenarios, in a path to maximize the classifier robustness.

[2]`https://stanfordmlgroup.github.io/competitions/mura/`

3.1 Dataset

In this work, we used the *MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs* [26] dataset, which contains 40,005 radiographic images labeled by radiologists. The MURA dataset is divided into 14,656 studies from the body upper extremities – shoulder, humerus, elbow, forearm, wrist, hand, and finger. Each study is labeled as either normal or abnormal.

The available data is divided into validation and training sets. For testing, a third set was created to be used in a competition and, therefore, not publicly available. To offset this fact and provide a realist measurement of our model's performance in a real-world scenario, we split the training data to create a test set. Our goal was to create a test set the same size as the validation set. As the provided validation set has 8.6% of the size of the training set, the new test set was created by moving, for each body part, 8.6% of the studies from the training to the test set. The number of items in each class is described in Table 1.

Table 1: Distribution of the number of studies (images) contained in each class for the three sets.

	Train		Validation		Test	
	Normal	Abnormal	Normal	Abnormal	Normal	Abnormal
Shoulder	1242 (3838)	1331 (3833)	99 (285)	95 (278)	122 (373)	126 (335)
Humerus	293 (608)	247 (549)	68 (148)	67 (140)	28 (65)	24 (50)
Elbow	997 (2677)	601 (1841)	92 (235)	66 (230)	97 (248)	59 (165)
Forearm	543 (1069)	257 (595)	69 (150)	64 (151)	47 (95)	30 (66)
Wrist	1993 (5237)	1218 (3670)	140 (364)	97 (295)	201 (528)	108 (317)
Hand	1365 (3702)	475 (1354)	101 (271)	66 (189)	132 (357)	46 (130)
Finger	1161 (2834)	600 (1805)	92 (214)	83 (247)	119 (304)	55 (163)

3.2 Experiments

The models used for classification were developed in Python, mainly using the PyTorch framework [27].

Every sample from the dataset has its target class defined among 14 classes (7 body parts, normal or abnormal). Before inputting the images to the network, we normalized each image's pixels values to the mean and standard deviation of the ImageNet dataset [28] and resized to 224×224 pixels.

The normalization operation is performed by applying for each pixel in the three channels the following operation:

$$p \leftarrow \frac{p - mean}{std} \tag{6}$$

where *mean* is 0.485, 0.456, and 0.406 and *std* is 0.229, 0.224, and 0.225 for the red, green, and blue channels, respectively.

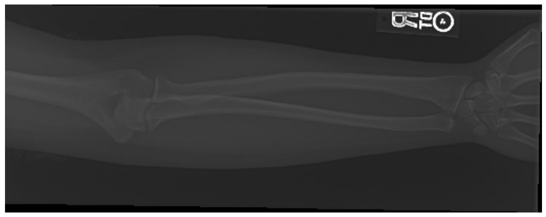
We trained each model over 40 epochs, with a batch size of 25. The samples are initially shuffled and reshuffled before each epoch. Upon each epoch, the network performance is measured with the scikit-learn library package [29], by the metrics listed in Subsection 2.4. To measure the binary classification performance, the 14 class output is condensed into two by ignoring the body part information. To determine the output of a study consisting of several images, the probability distribution output for each image in the study are averaged.

3.2.1 Experiment I: Fit, Pad, or Stretch?

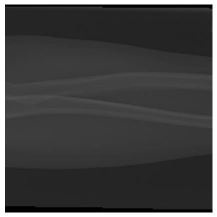
Our goal is to use pre-trained networks as a baseline. However, pre-trained networks expect square inputs, and our images have a variable aspect ratio, so before testing an assortment of networks, we need to decide how to transform our images: fit, pad, or stretch.

"Fit" crops the larger dimension to match the smaller one, leaving the central square. "Pad" adds a black border on the longer side, transforming the image into a square without erasing any region. "Stretch" directly changes the image aspect ratio to a square, stretching the shorter side until it matches the longer. Figure 6 illustrates the three transformations.

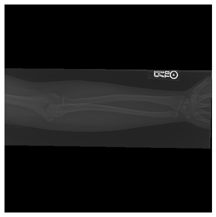
All three transformations come with pros and cons. "Fit" and "Pad" preserve the image aspect ratio, while "Stretch" distorts it. On the other hand, "Fit" loses some amount of information, which might be considerable if the image is very tall or long, "Pad" fills part of the input with irrelevant information, while "Stretch" includes the entire image.



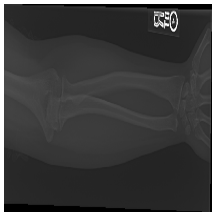
(a) Original image [26]



(b) Fit



(c) Pad



(d) Stretch

Figure 6: Three transformations to change a rectangular image into square. *Fit* does not deform but loses information. *Pad* also does not deform but adds irrelevant information. *Stretch* includes all pixels but deforms the image.

To decide on the best method, we trained the ResNet-18 network on the three options and compared the performance. Table 2 shows the results obtained. All of them were pretty similar, but “Stretch” was the best. Furthermore, “Stretch” was the fastest operation on our tests. Using three separated cloud instances to avoid data cached from one experiment speeding up another, “Stretch” loaded the training set 10% faster than “Pad” and 31% faster than “Fit.” Therefore, we chose the “Stretch” operation to be used in the subsequent experiments.

Table 2: Performance on the validation data for each of the transformations.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
Fit	0.8232	0.8140	0.8720	0.6373
Pad	0.8165	0.8054	0.8723	0.6222
Stretch	0.8274	0.8175	0.8617	0.6453

3.2.2 Experiment II: Pre-Trained Networks

The use of pre-trained models provides us with a consolidated and validated architecture. These models have demonstrated good results in many similar tasks [30]. In addition, it reduces the training time required compared to the training necessary to achieve similar results without pre-training.

Pre-trained models may, however, have a few caveats. The models were trained using the ImageNet dataset, containing colored images that do not resemble medical images. Therefore the layer structure of the networks may be suboptimal for this task. Moreover, the ImageNet images contain three channels. Consequently, our inputs must be reshaped to conform to this restriction. [31]

We tested several networks among the best-performing ones, including DenseNet [32], EfficientNet [33, 34], Inception-v3 [35], Inception-v4 [36, 37], Inception-ResNet-v2 [36, 37], ResNet [38], and VGG [39]. Table 3 presents the results obtained on each model. All the models performed very similarly, with a Kappa coefficient between 0.6450 and 0.6671.

Table 3: Performance of each network on the validation data.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
DenseNet-121	0.8365	0.8279	0.8874	0.6649
DenseNet-161	0.8374	0.8293	0.8892	0.6671
EfficientNet-B7	0.8274	0.8182	0.8873	0.6458
Inception-v3	0.8290	0.8197	0.8769	0.6491
Inception-v4	0.8315	0.8226	0.8783	0.6546
Inception-ResNet-v2	0.8324	0.8229	0.8864	0.6559
ResNet-18	0.8274	0.8175	0.8617	0.6453
ResNet-152	0.8299	0.8220	0.8780	0.6519
VGG-16	0.8357	0.8254	0.8877	0.6621
VGG-19	0.8282	0.8155	0.8840	0.6450

3.2.3 Experiment III: Transfer Learning from Another Medical Task

A possible issue with the previous approach was the optimization of the parameters to classify the ImageNet dataset, which is substantially different from the MURA dataset. To reduce the discrepancy between the ImageNet samples (compose of vehicles, objects, and animals, for instance) and MURA (composed of X-ray images), we first trained the model using the CheXpert dataset [40], then fine-tune it to the MURA dataset. The CheXpert dataset is also composed of X-ray images, but chest instead of the upper limb, and reasonably larger than the MURA dataset, 223,648 in contrast to 40,005 images.

As our goal was to simply update the network parameters with images closer to MURA, and not actually classify chest pathologies, we adapted and simplified the CheXpert labeling, which originally included not mutually exclusive labels, to classify only the view (lateral or frontal) and whether it had any abnormality labeled, totaling four mutually exclusive classes, similar to the MURA dataset. We used the DenseNet-161 network, training over 40 epochs using a batch of 60 images.

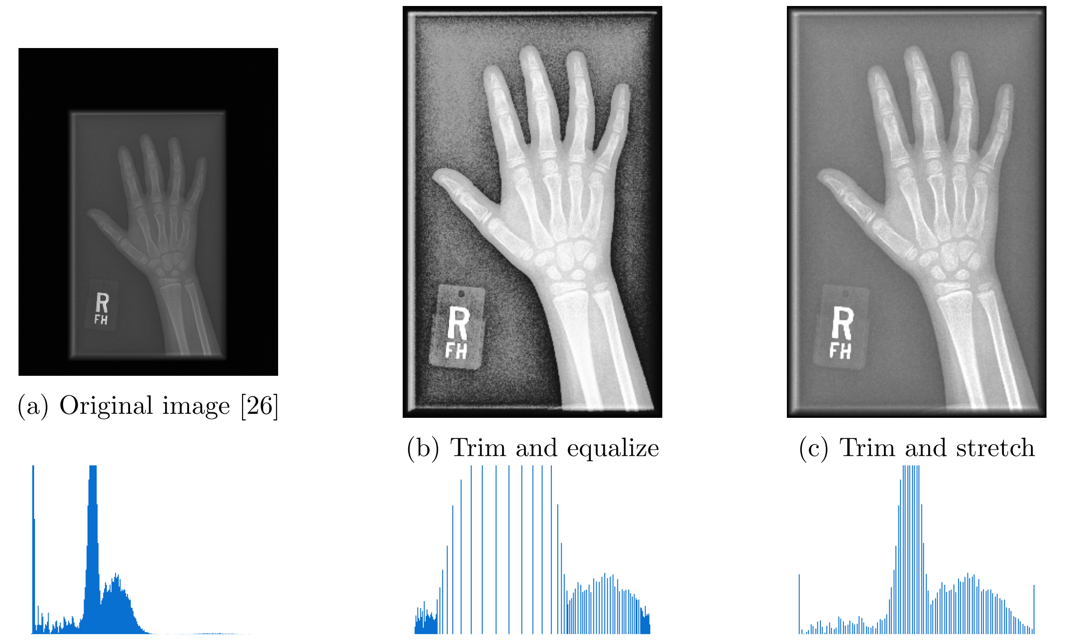
Table 4 shows the results of this experiment. The pre-training using another X-ray dataset did not improve the performance of the model, possibly due to poor performance of the CheXpert classification.

Table 4: Performance of DenseNet-161 on the validation data. The pre-training *From ImageNet* uses the default weights provided by PyTorch, and *From CheXpert* uses initially the same weights, and then is trained using the CheXpert dataset.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
From ImageNet	0.8374	0.8293	0.8892	0.6671
From CheXpert	0.8332	0.8235	0.8906	0.6574

3.2.4 Experiment IV: Pre-Processing

Many of the X-rays images in the dataset present poor contrast. To improve the visibility and exploit the entire gray spectrum, we experimented with pre-processing the images with two operations of contrast increase: histogram equalization and histogram stretching. Moreover, some images include borders, wasting an area of the input, so we can crop these out. All of these processings are afforded by the ImageMagick image editor [41], using operators *equalize*, *linear-stretch*, and *trim*. An example of this pre-processing is shown in Figure 7.



(a) Original image [26]

(b) Trim and equalize

(c) Trim and stretch

(d) Original image histogram

(e) Trim and equalize histogram

(f) Trim and stretch histogram

Figure 7: Effect of the operators *equalize*, *linear-stretch*, and *trim* in the radiographs.

We tested training the two better performing networks in Experiment II using the two pre-processings shown in Figure 7. These pre-processings, however, did not provide any improvement in the results. The results obtained are shown in Table 5, all of which are worse than the results without pre-processing, showing that the models are capable of ignoring irrelevant areas of the image and correctly interpreting the gray tones on its own.

Table 5: Performance of each network on the validation data trained and tested using a pre-processed dataset.

		Accuracy	Balanced Accuracy	AUC ROC	Kappa
DenseNet-161	**Original**	0.8374	0.8293	0.8892	0.6671
	Equalize	0.8315	0.8226	0.8840	0.6546
	Linear Stretch	0.8265	0.8169	0.8788	0.6438
VGG-16	**Original**	0.8357	0.8254	0.8877	0.6621
	Equalize	0.8265	0.8188	0.8818	0.6452
	Linear Stretch	0.8299	0.8210	0.8787	0.6512

3.2.5 Experiment V: Isolate Samples with Implants

Metal implants – such as plates, screws, and pins – used to assist the healing of a bone, are a distinct and notable feature in an image that contains them. Since most of these cases are labeled as abnormal, the model might be induced to classify all the images containing these devices as abnormal. To test this hypothesis, we manually isolated the images containing implants in one of the classes, the class humerus, as it has the smallest number of images.

By creating this additional information in the training set, we could train the network to classify each of the samples into "normal without implant," "abnormal without implant," "normal with implant," and "abnormal with implant." We trained this model using the DenseNet-161 network, the best performer in Experiment II. To measure the performance, we merge the results of the two normal classes and the two abnormal classes into one. The full confusion matrix is shown in Table 6, while the merged one in Table 7. For comparison, Table 8 shows the results of the same network trained without the metal implants annotations. The performance metrics obtained in both datasets are shown in Table 9.

Table 6: Confusion matrix of the model trained using a dataset containing annotations of whether implants are present in the patient.

			Prediction			
			Without Implant		With Implant	
			Normal	Abnormal	Normal	Abnormal
Input	Without Implant	Normal	129	5	0	0
		Abnormal	22	35	0	0
	With Implant	Normal	1	0	0	13
		Abnormal	0	1	0	82

Table 7: Confusion matrix of the model shown in Table 6 merged into two classes.

		Prediction	
		Normal	Abnormal
Input	Normal	130	18
	Abnormal	22	118

Table 8: Confusion matrix of the model trained using the regular dataset, without any extra annotations.

		Prediction	
		Normal	Abnormal
Input	Normal	126	22
	Abnormal	18	122

Table 9: Performance of the two models on the validation data.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
Without Implants Annotations	0.8611	0.8614	0.9022	0.7222
With Implants Annotations	0.8611	0.8601	0.9026	0.7218

By comparing Table 7 and Table 8, we can notice they have almost equal distribution. The extra labeling in the images causes virtually no change to the results. The performance may be possibly explained due to the imbalance of the set and lack of samples for some classes, with only 2.2% of the training data contained in “normal with implant,” leading to none of the inputs getting predicted as this class (Table 6). Attempting to balance the dataset by removing samples from other classes would create another problem: a minuscule dataset. This approach was, therefore, abandoned.

3.2.6 Experiment VI: Data Augmentation

Data augmentation increases the model capacity to generalize by exposing it to an extensive number of scenarios. In this experiment, we applied two data augmentation techniques: cropping and horizontal flip. To verify the effect of these augmentations, we trained the DenseNet-161 network once using the original, five random crops and its flips, resulting in a 12-fold increase in the size, similar to the augmentation proposed by Krizhevsky et al. [42]; and once using only the horizontal flip augmentation, doubling the number of samples. Table 10 compares the results of the two experiments with the original dataset.

Table 10: Comparison of performance on the validation data. *Augmented Dataset A* is the dataset using only horizontal flip, and *Augmented Dataset B* is the dataset using both horizontal flip and crop.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
Original Dataset	0.8374	0.8293	0.8892	0.6671
Augmented Dataset A	0.8465	0.8370	0.8929	0.6848
Augmented Dataset B	0.8432	0.8355	0.8922	0.6792

From these results, we could conclude that, despite having more data, Dataset B performs worse than Dataset A, but still better than the original. Moreover, since training time is approximately linear in the number of samples, using Dataset A results in longer training time. Therefore, we trained the other networks using Dataset B. These networks were chosen based on the results of Experiment II while avoiding more than one network from the same "family."

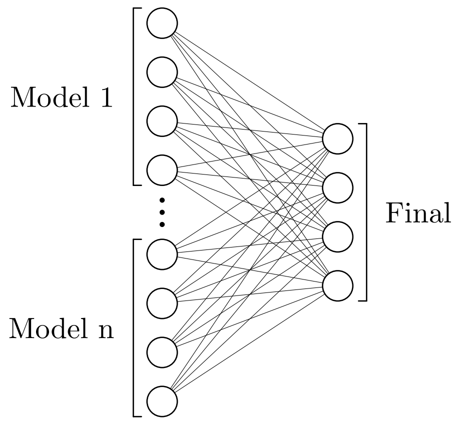
Table 11: Performance of each network on the validation data for the networks trained using a dataset augmented using flip.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
DenseNet-161	0.8465	0.8370	0.8929	0.6848
EfficientNet-B7	0.8399	0.8314	0.8913	0.6719
Inception-ResNet-v2	0.8457	0.8369	0.8892	0.6836
VGG-16	0.8374	0.8302	0.8967	0.6676
ResNet-152	0.8349	0.8243	0.8931	0.6602

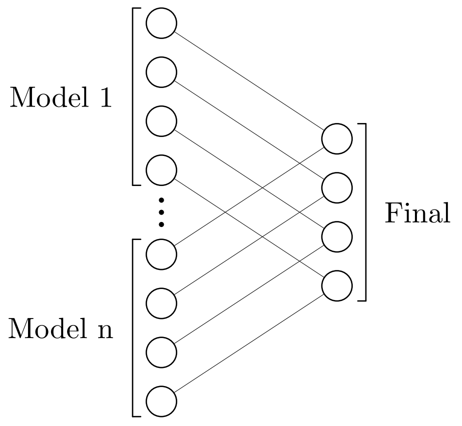
3.2.7 Experiment VII: Ensemble Model

An ensemble model combines multiple models to make a final prediction, similar to consulting multiple opinions. This will supposedly improve the model as if a single model performs poorly for a sample, its prediction may be overridden by other models, improving the model stability.

In this experiment, we will use the five models trained in Experiment VI and combine its predictions using a set of techniques. First, we made a final prediction based on consensus, i.e., the mode of the five predictions, ignoring the probabilistic distribution. Next, we combined the results using a weighted average, using one weight for each of the five models. Then we tested two techniques using neural networks: a sparsely connected and a fully connected layer to make a final prediction. The architecture of these layers is shown in Figure 8. While these last two techniques are also ultimately a weighted average, each element of the output array for each model is now independently weighted. Also, while the mere five weights in the simple weighted were obtained using an exhaustive search, the weights in these two experiments were obtained using gradient descent. Finally, we used a support vector machine (SVM) with a radial basis function (RBF) kernel to ensemble the models. The SVM is implemented by scikit-learn [29].



(a) Fully connected neural network architecture



(b) Sparsely connected neural network architecture

Figure 8: Schematic representation of the two neural network architectures used in this experiment. The number of classes shown here is 4 for simplification, while the real number of classes is 14. The number of models n is 5 in the experiment.

The results obtained are described in Table 12, comparing it with the average results obtained by the models individually. All ensemble models were able to perform better than the average performance of the single models. The models in Experiment VI present a hard to avoid overfitting problem, leading to near 100% accuracy in the training set for all the models, therefore making it impossible to find the best way to combine these predictions using the training set since practically any combination will lead to near 100% accuracy.

We must then tune the parameters in the ensemble layer using the validation set, which should inevitably lead to a higher drop in performance when transitioning to the test set.

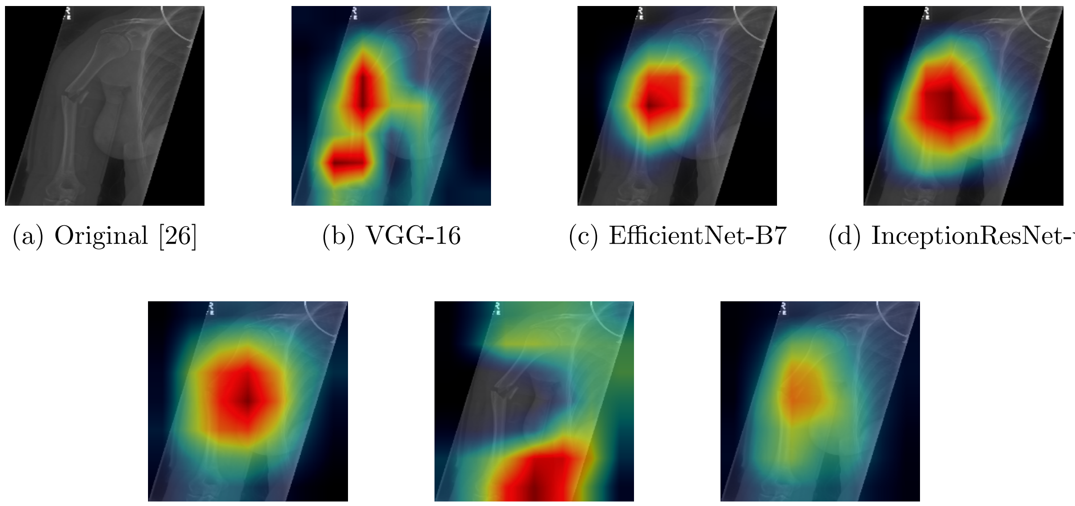
Table 12: Comparison of performance on the validation data. *Single Model (Average)* is the average performance of the models in Experiment VI for reference. The *Consensus* model does not output a probability distribution, only the final prediction, penalizing its AUC ROC score.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
Single Model (Average)	0.8409	0.8319	0.8926	0.6736
Consensus	0.8449	0.8349	0.8753	0.6811
Weighted Average	0.8641	0.8572	0.9086	0.7222
Sparsely Connected	0.8590	0.8504	0.9039	0.7110
Fully Connected	0.8540	0.8467	0.9102	0.7015
SVM (RBF)	0.8699	0.8638	0.9208	0.7345

The Gradient-weighted Class Activation Mapping (Grad-CAM) [43, 44] is a method to visualize areas of the input image that are important to reaching the outputted prediction. Grad-CAM works by processing the input through the network and obtaining a prediction. The neuron's gradients are set to zero except for the predicted class, and backpropagating until the last convolutional layer (i.e., the feature layer) as to obtain the regions of this layer that most influenced the prediction. Considering convolutional layers preserve spatial information, these regions can be extrapolated to the input layer and converted to a heatmap.

Figure 9 shows the heatmaps generated by Grad-CAM for each individual model and the ensemble model, demonstrating the improvement caused by ensembling the models. In this example VGG-16, EfficientNet-B7 and InceptionResNet-v2 predict "abnormal humerus" while DenseNet-161 and ResNet-152 predict "normal humerus." The Ensemble SVM model prediction is, correctly, "abnormal humerus." The incorrect models focus on a different part of the image but are overridden by other models.

Since the Grad-CAM uses the information from the feature layer to build the visualization and our ensemble models combine the individual models at the output layer level, we have multiple feature layers, which would preclude the use of Grad-CAM. To work around this, we built the ensemble visualization by averaging the Grad-CAM outputted matrix (used to create the heatmap) for the five models. Hence, the heatmap generated for the ensemble model is an approximation and not specific to any of the models.



(a) Original [26] (b) VGG-16 (c) EfficientNet-B7 (d) InceptionResNet-v2

(e) DenseNet-161 (f) ResNet-152 (g) Ensemble

Figure 9: Grad-CAM heatmaps for each individual model and the ensemble model for an abnormal humerus sample. Models VGG-16, EfficientNet-B7 and InceptionResNet-v2 predict correctly, while DenseNet-161 and ResNet-152 predict incorrectly. The Ensemble SVM model takes all models into consideration and outputs a correct final prediction.

4 Results and Discussion

Figure 10 summarizes the Kappa coefficients for all the experiments in Subsection 3.2, except Experiment V, which uses a subset of the dataset. Comparing to our baseline experiment, Experiments III, IV, and V did not improve the classification, while Experiments VI and VII did. Experiment II had an average Kappa of 0.6542; the SVM (RBF) model from Experiment VII improved it to 0.7345, which is the best result achieved.

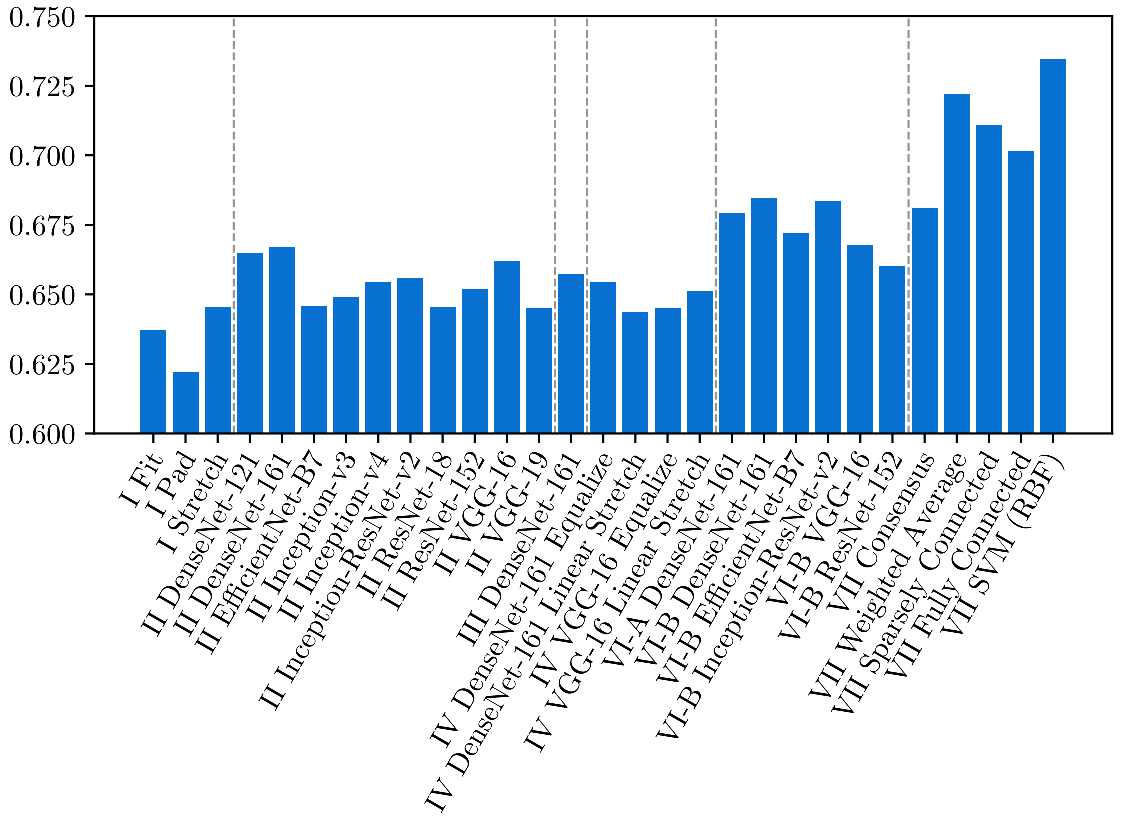


Figure 10: Summary of the Kappa coefficient for the experiments I, II, III, IV, VI, and VII. The maximum value was 0.7345.

To evaluate our model in a scenario closer to the real world, we will now use our test set, defined in Subsection 3.1, which has no overlap with the other two sets. Table 13 shows the performance metrics achieved using the Ensemble SVM model from Experiment VII, which was the best performing model on the validation data. The model's performance expectedly decreased due to overfitting. Our model performed worst on hands, with a Kappa of 0.4717, and best on elbows, with a Kappa of 0.7921. The overall Kappa was 0.6724. In contrast, human radiologists performed worst on fingers and best on wrists [26]. Our model was able to outperform two of the three radiologists evaluated on the elbow classification task, and all of them in the finger classification, but falls behind on other body parts. The MURA competition has ended, so we cannot evaluate our model on the official test set. As a consequence, the human radiologists were evaluated on the official test set, and our model, using our test set, so the performance comparison might not be completely accurate.

Table 13: Performance of the model Ensemble SVM on the test data, broken down by inputted body part.

	Accuracy	Balanced Accuracy	AUC ROC	Kappa
Shoulder	0.8266	0.8270	0.8778	0.6535
Humerus	0.8077	0.7976	0.8274	0.6061
Elbow	0.9038	0.8895	0.9023	0.7921
Forearm	0.7922	0.7755	0.8142	0.5578
Wrist	0.8835	0.8633	0.9142	0.7393
Hand	0.8090	0.7225	0.7759	0.4717
Finger	0.8563	0.8558	0.8901	0.6817
Overall	0.8484	0.8319	0.8791	0.6724

The performance difference between the test and validation sets is likely explained due to the fitting of ensemble parameters using the validation set, as described in Experiment VII. We can verify this by running the test set on the Ensemble Consensus model, which does not have any extra parameters. Table 14 shows the results of this test. These results are similar to the results obtained in the validation set (by some metrics, even better), and despite being the worst performer on Experiment VII, it performed better than the Ensemble SVM model, the best performer in Experiment VII. Therefore, we could assume that if we were able to train the models of Experiment VI in such a way to reduce overfitting, we could train Experiment VII using the train set and reduce the gap between validation and test results. Alternatively, splitting the data into four sets instead of three to fit the ensemble parameters using a separated set could also be beneficial, but would further reduce an already limited dataset.

Table 14: Performance of the model Ensemble Consensus on the test data. The model has essentially the same performance on the validation set.

Accuracy	Balanced Accuracy	AUC ROC	Kappa
0.8593	0.8339	0.8652	0.6899

5 Conclusion

The development of machine learning models for medical diagnosis presents many pitfalls. Our work proposed to explore this machine learning classification problem using convolutional neural networks and related methods. It indicated that many of the techniques supposed to improve the classification ended up hurting it. The best setting found was to stretch the input to a square, apply horizontal flip data augmentation, and ensemble a variety of architectures, reaching an AUC ROC of 0.8791 and Kappa of 0.6724. Transfer learning from a similar task, pre-processing images, and manually annotating subclasses did not improve our tests.

While the overall result was inferior to human radiologists, we were still able to achieve promising results in some scenarios. However, a transition to the clinical setting is challenging. Besides accuracy improvement under a controlled scenario, the algorithm would need, for example, to handle unexpected inputs, provide translation and rotation invariance, and include explainability to mitigate automation bias.

The greatest hindrance in this work was the models' overfitting, to which further experimentation with other methods is needed to adequately address it, such as early stopping and regularization. The development of larger datasets would also provide a big leap on this matter.

References

[1] Global Burden of Disease Collaborative Network. Global Burden of Disease Study 2017 (GBD 2017) Results. 2018.

[2] World Health Organisation. Musculoskeletal conditions fact sheet. `https://www.who.int/news-room/fact-sheets/detail/musculoskeletal-conditions`, 2019.

[3] Anthony D Woolf and Bruce Pfleger. Burden of major musculoskeletal conditions. *Bulletin of the World Health Organization*, 81:646–656, 2003.

[4] Hrushikesh Garud, Sri Phani Krishna Karri, Debdoot Sheet, Jyotirmoy Chatterjee, Manjunatha Mahadevappa, Ajoy K Ray, Arindam Ghosh, and Ashok K Maity. High-magnification Multi-views Based Classification of Breast Fine Needle Aspiration Cytology Cell Samples using Fusion of Decisions from Deep Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 76–81, 2017.

[5] Haifeng Wang, Bichen Zheng, Sang Won Yoon, and Hoo Sang Ko. A support vector machine-based ensemble algorithm for breast cancer diagnosis. *European Journal of Operational Research*, 267(2):687–699, 2018.

[6] Mizuho Nishio, Osamu Sugiyama, Masahiro Yakami, Syoko Ueno, Takeshi Kubo, Tomohiro Kuroda, and Kaori Togashi. Computer-aided diagnosis of lung nodule classification between benign nodule, primary lung cancer, and metastatic lung cancer at different image size using deep convolutional neural network with transfer learning. *PLOS ONE*, 13(7), 2018.

[7] Yiming Ding, Jae Ho Sohn, Michael G Kawczynski, Hari Trivedi, Roy Harnish, Nathaniel W Jenkins, Dmytro Lituiev, Timothy P Copeland, Mariam S Aboian, Carina Mari Aparici, Spencer C Behr, Robert R Flavell, Shih-Ying Huang, Kelly A Zalocusky, Lorenzo Nardo, Youngho Seo, Randall A Hawkins, Miguel Hernandez Pampaloni, Dexter Hadley, and Benjamin L Franc. A Deep Learning Model to Predict a Diagnosis of Alzheimer Disease by Using 18F-FDG PET of the Brain. *Radiology*, 290(2):456–464, 2019.

[8] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2097–2106, 2017.

[9] Yulia Arzhaeva, Dadong Wang, Liton Devnath, Saeed Amirgholipour, Rhiannon McBean, James Hillhouse, Suhuai Luo, David Meredith, Katrina Newbigin, and Deborah Yates. Development of Automated Diagnostic Tools for Pneumoconiosis Detection from Chest X-Ray Radiographs. 2019.

[10] Qingji Guan, Yaping Huang, Zhun Zhong, Zhedong Zheng, Liang Zheng, and Yi Yang. Diagnose like a Radiologist: Attention Guided Convolutional Neural Network for Thorax Disease Classification. *arXiv:1801.09927*, 2018.

[11] Madhuri Panwar, Amit Acharyya, Rishad A Shafik, and Dwaipayan Biswas. K-Nearest Neighbor Based Methodology for Accurate Diagnosis of Diabetes Mellitus. In *Sixth International Symposium on Embedded Computing and System Design (ISED)*, pages 132–136. IEEE, 2016.

[12] Qingbo Li, Wenjie Li, Jialin Zhang, and Zhi Xu. An improved k-nearest neighbour method to diagnose breast cancer. *Analyst*, 143(12):2807–2811, 2018.

[13] Azam Davari Dolatabadi, Siamak Esmael Zadeh Khadem, and Babak Mohammadzadeh Asl. Automated diagnosis of coronary artery disease (CAD) patients using optimized SVM. *Computer Methods and Programs in Biomedicine*, 138:117–126, 2017.

[14] Emina Alickovic and Abdulhamit Subasi. Medical Decision Support System for Diagnosis of Heart Arrhythmia using DWT and Random Forests Classifier. *Journal of Medical Systems*, 40 (4):108, 2016.

[15] Xiaoming Liu, Tianyu Fu, Zhifang Pan, Dong Liu, Wei Hu, Jun Liu, and Kai Zhang. Automated Layer Segmentation of Retinal Optical Coherence Tomography Images Using a Deep Feature Enhanced Structured Random Forests Classifier. *IEEE Journal of Biomedical and Health Informatics*, 23(4):1404–1416, 2018.

[16] Philipp Tschandl, Cliff Rosendahl, Bengu Nisa Akay, Giuseppe Argenziano, Andreas Blum, Ralph P Braun, Horacio Cabo, Jean-Yves Gourhant, Jürgen Kreusch, Aimilios Lallas, Jan Lapins, Ashfaq Marghoob, Scott Menzies, Nina Maria Neuber, John Paoli, Harold S Rabinovitz, Christoph Rinner, Alon Scope, H Peter Soyer, Christoph Sinz, Luc Thomas, Iris Zalaudek, and Harald Kittler. Expert-Level Diagnosis of Nonpigmented Skin Cancer by Combined Convolutional Neural Networks. *JAMA Dermatology*, 155(1):58–65, 2019.

[17] Jinlian Ma, Fa Wu, Jiang Zhu, Dong Xu, and Dexing Kong. A pre-trained convolutional neural network based method for thyroid nodule diagnosis. *Ultrasonics*, 73:221–230, 2017.

[18] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. DeepID3: Face Recognition with Very Deep Neural Networks. *arXiv:1502.00873*, 2015.

[19] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art Speech Recognition With Sequence-to-Sequence Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.

[20] Christopher M Bishop. *Pattern recognition and machine learning*, chapter 5. springer, 2006.

[21] The MathWorks, Inc. Convolutional Neural Network. `https://nl.mathworks.com/solutions/deep-learning/convolutional-neural-network.html`.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press Cambridge, 2016.

[23] Linda Studer, Michele Alberti, Vinaychandran Pondenkandath, Pinar Goktepe, Thomas Kolonko, Andreas Fischer, Marcus Liwicki, and Rolf Ingold. A Comprehensive Study of ImageNet Pre-Training for Historical Document Image Analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 720–725. IEEE, 2019.

[24] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[25] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[26] Pranav Rajpurkar, Jeremy Irvin, Aarti Bagul, Daisy Ding, Tony Duan, Hershel Mehta, Brandon Yang, Kaylie Zhu, Dillon Laird, Robyn L Ball, Curtis Langlotz, Katie Shpanskaya, Matthew P Lungren, and Andrew Y Ng. MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs. *arXiv:1712.06957*, 2017.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[30] Mehdi Habibzadeh, Mahboobeh Jannesari, Zahra Rezaei, Hossein Baharvand, and Mehdi Totonchi. Automatic white blood cell classification using pre-trained deep learning models: ResNet and inception. In *10th International Conference on Machine Vision (ICMV)*, volume 10696, page 1069612. International Society for Optics and Photonics, 2017.

[31] Yiting Xie and David Richmond. Pre-training on grayscale ImageNet improves medical image classification. In *European Conference on Computer Vision (ECCV)*, 2018.

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[33] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv:1905.11946*, 2019.

[34] Luke Melas-Kyriazi. EfficientNet PyTorch. `https://github.com/lukemelas/EfficientNet-PyTorch`, 2019.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[36] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *31st AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.

[37] Remi Cadene. Pretrained models for Pytorch. `https://github.com/Cadene/pretrained-models.pytorch`, 2017.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[39] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556*, 2014.

[40] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn L Ball, Katie S Shpanskaya, Jayne Seekins, David A Mong, Safwan S Halabi, Jesse K Sandberg, Ricky Jones, David B Larson, Curtis P Langlotz, Bhavik N Patel, Matthew P Lungren, and Andrew Y Ng. CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. In *33rd AAAI Conference on Artificial Intelligence*, pages 590–597, 2019.

[41] The ImageMagick Development Team. ImageMagick. `https://imagemagick.org`, 2020.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Meural Networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[43] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[44] Kazuto Nakashima. Grad-CAM with PyTorch. `https://github.com/kazuto1011/grad-cam-pytorch`, 2017.